Programmable Ethernet Switches and Their Applications

A DISSERTATION PRESENTED

BY

SRIKANT SHARMA

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

August 2008

Stony Brook University
The Graduate School

Srikant Sharma

We, the dissertation committee for the above candidate for
the degree of Doctor of Philosophy,
hereby recommend acceptance of this dissertation.

Professor Tzi-cker Chiueh, Dissertation Advisor,
Department of Computer Science, Stony Brook University

Professor Yuanyuan Yang, Chairperson of Defense,
Department of Electrical and Computer Engineering, Stony Brook University

Professor Erez Zadok,
Department of Computer Science, Stony Brook University

Professor Jennifer Wong,
Department of Computer Science, Stony Brook University

Dr. Bruce Montague,
Symantec Research Labs, Mountain View, CA

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Programmable Ethernet Switches and Their Applications

by

Srikant Sharma

Doctor of Philosophy

in

Computer Science

Stony Brook University

2008

Simplicity, cost effectiveness, scalability, and economies of scale make Ethernet a popular choice for (a) local area networks (LAN), as well as for (b) storage area networks (SAN), and increasingly (c) metropolitan-area networks (MAN). Applications of Ethernet in the SAN and MAN arena elevate it from a LAN technology to a ubiquitous networking technology. With the expanded applicability of Ethernet there are certain adaptability issues which prompt rethinking of some of its architectural features. The Spanning-Tree based switching mechanism, considered to be very efficient at avoiding loops in LAN environments, is a performance bottleneck in the metro network context. Absence of an explicit switching path selection mechanism prohibits traffic engineering in metro networks. Prolonged spanning tree reconstruction periods after failures make Ethernet unsuitable to support critical applications. Lack of usage regulation mechanisms leads to insufficient isolation between different users, resulting in QoS problems.

Modern Ethernet switches support many advanced features which can be controlled through programmable interfaces. These features are VLAN tagging, rate limiting, and status monitoring. Conventionally, these features are mostly used to statically configure an Ethernet switch. This research proposes to use these features as dynamic control mechanisms in the context of metro and cluster networks to:

1. Maximize physical network link resources by enabling MPLS like traffic engineering,
2. Minimize failure recovery time, and
3. Enforce QoS requirements.

With these programmatic configurable control mechanisms, standard Ethernet switches can be used as effective building blocks for metropolitan-area Ethernet networks (MEN), storage-area networks (SAN), and computation cluster interconnects. We validated the usefulness of this new level of control over Ethernet switches with a MEN architecture that features multi-fold throughput gains and sub-second failure recovery time.

We discuss how a comprehensive resource management system can be devised using these mechanisms that can result in high performance and fault tolerant metro Ethernet, Storage, and Cluster networks. We also discuss how a network topology can be efficiently evolved by correlating the traffic profile characteristics of the end users and the traffic engineering required in the network. We describe a design methodology for Ethernet-based SAN fabrics utilizing this network evolution technique.

To this effect, we develop a network topology planning tool to minimize network infrastructure deployment cost. This topology planning work is specifically targeted towards providing automated tools to design Ethernet storage area network and cluster interconnect topologies with redundancy and fault-tolerance support.

To
Shachi, Ma, Nana,
and
My Family

Contents

List of Figures

Acknowledgments

At the onset, I would like to express my gratitude to my advisor, Professor Tzicker Chiueh for showing me by example how to do systems research. He directed me to interesting problems to work on and provided me with stimulating research environment at Experimental Computer Systems Lab (ECSL). He taught me how to develop research aptitude and always strove to make me conscious about quality of my work. He instilled a sense of pride in me and my colleagues about our research. He was always present for me when I needed his advice in technical as well as personal matters. I will always be grateful to him for the support he provided me at every juncture in the course of my doctoral studies at Stony Brook.

My dissertation committee members, Professor Yuanyuan Yang, Professor Erez Zadok, and Professor Jennifer Wong from Stony Brook University and Dr. Bruce Montague from Symantec Research Labs have helped me with my dissertation at every possible stage. Dr. Montague's advice has been extremely important to me in improving my dissertation quality. Advice from Professor Yang and Professor Zadok has been very beneficial for me while presenting my research to research community outside Stony Brook. I would like to thank my thesis proposal committee member, Professor R. Sekar for his guidance throughout my studies at Stony Brook University.

Special thanks go to my collaborators Kartik Gopalan and Susanta Nanda, who helped me fortify my dissertation research. ECSL has provided me with a great technical and personal environment for my research. This environment was created by my friends and colleagues: Pradipta De, Ningning Zhu, Kartik Gopalan, Gang Peng, Manish Prasad, Ashish Raniwala, Fanglu Guo, Jiawu Chen, Fu-Hau Hsu, and others.

There are other people in Stony Brook and outside whose invaluable company has directly or indirectly helped me with my work at Stony Brook. They are Venkatkrishnan VN, Pravin Nair, Prem Uppuluri, Arnab Ray, Ajay Gupta, Swarnima, Miti and others.

I am also thankful to Shardul Divatia, my supervisor at Symantec, for helping me balance my academic life and work life effectively. He provided me with support whenever I needed it most. He helped me in balancing my work priorities and focus on important things at hand. My colleagues at Symantec Taher Vohra and Rahul Fiske have been very helpful to me by acting as sounding boards for my various research approaches. They provided me with much needed second person perspective about my research in SAN design. My other colleagues at Symantec: Grizel, Abhay, Rohit, Arun, and Deepak provided me with a very encouraging and cheerful work environment.

I owe most this achievement to my parents and my family who encouraged me when I needed encouragement, cheered for me when I was happy, and gave me advice when I needed it most. They stood by me throughout my life and this dissertation is possible mainly because of my family.

Last but not least, this dissertation is dedicated to my wife Shachi, who gave me inspiration, strength, energy, endurance, and solace to pursue my dreams.

Chapter 1

Introduction

In late 1972 Robert Metcalfe and David Boggs, researchers at Xerox PARC, developed an experimental networking system to interconnect a few Xerox Altos personal workstations with laser printers, servers, and each other. This experimental network derived its signal clock from the Altos system clock and supported a data transmission rate of 2.94 Mbps. It was based on the AlohaNet system developed at the University of Hawaii. Robert Metcalfe named this experimental network the *Alto Aloha Network.*

By late 1973 the Alto Aloha network was capable of supporting connectivity between any computers, not just Altos workstations. The network mechanism had also evolved well beyond the Aloha system. The network was capable of providing connectivity over different physical media, such as radio waves, telephone lines, and co-axial cables. The physical medium was dubbed the *"Ether"* that transferred data to all stations in the network, the same way as the luminiferous ether that was once assumed to be the medium for propagation of light. To highlight the fact that this new network was capable of functioning without Altos workstations and was an advancement over the AlohaNet, it was renamed the *"Ethernet"*.

1.1 Ethernet Evolution: Beyond Local Networks

In 1976 Robert Metcalfe published a paper titled, "Ethernet: Distributed Packet-Switching For Local Computer Networks [MB76]." This was the beginning for a computer networking technology that has dominated not just Local Area Networks, but also increasingly important in Wide Area Networks, Cluster Interconnects, Storage Area Networks, and recently, even Super Computing Interconnects.

In the last three decades, Ethernet technology has come a long way, from its initial shared-media 3 Mbps capability, to today's switched-media form providing throughput up to several Gbps. Its simplicity, cost effectiveness, and economies of scale have enabled it to make inroads into practically all scales of networks, such as storage area networks (SAN), and Metropolitan Area Networks (MAN). Recent technological advances, such as Ethernet in the First Mile, which enables subscribers to connect to an Ethernet-based core network over a wide variety of media, ranging from voice grade copper to multi-mode fiber, further reinforce the case for metropolitan Ethernet network (MEN) architecture [IEE04b].

Availability of bandwidth up to 10 Gbps and micro-second level message latencies also make Ethernet a low-cost alternative to widely-used cluster interconnects (CI) such as Myrinet [BCF$^+$95], Quadrics [PFH$^+$02], and Infiniband [inf01]. Today (in June 2008), 284 out of the top 500 super computers in the world use Gigabit Ethernet as the interconnect of choice [Top].

Finally, with increasing growth of IP storage, iSCSI [isc], and Fibre Channel over Ethernet [Fib07] - Ethernet-based storage area networks are becoming serious alternatives to against Fibre Channel-based SAN because of lower cost and extended flexibility. Such new applications of Ethernet elevate it from a simple LAN technology to a *ubiquitous networking technology*.

At the heart of the success of Ethernet lies the fact that Ethernet standards have always kept pace with continuously changing demands of the networking world. The growth in network traffic was always accompanied by increased line speeds in Ethernet technology. The initial standardized line speed of 10 Mbps in 1980 was improved by Fast Ethernet, providing 100 Mbps in 1995. At the same time the shared bus architecture was replaced by a switching architecture with point-to-multipoint connectivity of nodes. This addressed both performance and scalability requirements of networks deployed at that time. As the demand for line speed in enterprises increased because of bandwidth intensive applications and networked storage. Gigabit Ethernet debuted in 1998 in a timely manner. The current push for 10 Gbps Ethernet and future offerings of 40 Gbps and 100 Gbps Ethernet pave the way for increasing use of Ethernet in cluster interconnects, storage networks, and super computing interconnects. The evolution of Ethernet has always been gradual and backward compatible. As a result, the migration path for deployments has always been smooth and never required disruptive upgrades.

The evolution has not been just with regard to speed. There have been several advances in Ethernet manageability and extensibility. The virtual LAN feature simplifies management of large deployments by dividing them into smaller broadcast domains. Ethernet switches have grown to be more intelligent and provide several advanced features, such as Quality of Service, redundancy, bandwidth optimization, access control, remote monitoring, and remote management.

The attractiveness of Ethernet comes from the following factors:

- **Performance Scalability**: Ethernet is suitable for deployments where bandwidth requirements range from just a few Mbps to 100s of Gbps.
- **Distance Scalability**: Ethernet can be deployed for short range Local Area Networks to long range Metro Area Networks.

- **Economies of Scale**: Ethernet is the most widely deployed networking technology in the world. A technology being used on such a scale benefits from economies of scale. The cost of Ethernet equipment is significantly cheaper than other networking technologies.

- **Flexibility and Interoperability**: Ethernet deployments are incredibly flexible. An Ethernet network can be grown and shrunk by simply adding or removing Ethernet switches. There are no tedious reconfiguration procedures to be followed and no provisioning efforts to be carried out. Ethernet networks are true plug and play networks. The Ethernet standards are so robust that, if followed properly, *an Ethernet device is an Ethernet device*. Interoperability problems between equipment from different vendors are practically never seen.

- **Administrative Ease**: Ethernet equipment is simple and easy to manage. Moreover, large number of deployments produce a large knowledge base. Familiarity and experience witwith technology increases with its wide spread penetration in enterprises. This directly reflects in ease of administration of deployments.

1.2 Ethernet Challenges

Though Ethernet has managed to gain a foothold at all scales of network technology, the primary reason has always been cost advantage rather than performance. Even though Ethernet is capable of satisfying high performance requirements, it is always seen as a low cost alternative to the established technologies. When performance comes into the picture, Ethernet deployment is always treated with skepticism.

For instance, in metropolitan networks Ethernet has proved to be an attractive service. Increasing numbers of enterprise campuses spread over metropolitan geographic spans are being interconnected with metro Ethernet services. Despite increasing availability of metro Ethernet, these services today are actually built on circuit switched technologies such as SDH/SONET [BC89] and ATM or packet switching technologies like RPR [IEE04c] and MPLS [RVC01] etc. These services are provided by means of Ethernet tunnels set up over carrier technologies. Deployment of standard Ethernet as the carrier network in the core is still rare. Though inertia in abandoning legacy deployment of conventional circuit switched technologies is one of the reasons for this, it is not the primary reason. Deployment of Ethernet in the metro core has yet to take off because there remain some architectural deficiencies with switching in Ethernet that do not justify forklift network upgrades to core networks that will extensively depend on Ethernet technology [Cor03, For02].

A similar attitude is observed when Ethernet is used to build cluster interconnects (CIs) or storage area networks (SANs). Ethernet is treated as a low cost alternative to SAN technology, such as Fibre Channel, or interconnects like Myrinet and Infiniband. Because of capacity planning complexities with Ethernet, the general practice is to deploy Ethernet CIs and SANs using a small number of switches with high port density. Clusters are aggregated over a small number of large switches. This kind of deployment removes all network planning complexity, but primarily utilizes only the crossbar backplane of Ethernet switches, rather than the Ethernet network. The aggregation cost of switching networks increases rapidly with the size of the cluster, which in turn raises the per-port cost factor. With increased port density, the ratio between per-port bandwidth and total backplane processing bandwidth rapidly decreases, as the underlying crossbar switching technology remains unchanged. Apart from cost and performance issues, aggregation also poses a reliability issue. Aggregation of nodes on a single switch or a limited number of switches presents a small set of points for complete failure. This reduces reliability and increases chances of complete failure in the event of a switch failure. All these issues limit Ethernet SAN and CI to small deployment sizes. Ethernet is seldom a choice of interconnect for large and high performance SANs and CIs.

There are several reasons why carrier Ethernet deployments are rare and why Ethernet cluster interconnects are not considered for large high performance interconnects.

Ethernet networks use a spanning tree protocol (IEEE 802.1d) to establish a loop-free path between every pair of nodes [IEE90]. The spanning tree approach fails to exploit all physical network resources, because in any network of N nodes there are at most $N-1$ links actively forwarding traffic. This produces an imbalance of load in the network. This is impractical in large scale metro networks. Further, switch and link failures require rebuilding the spanning tree, which is a lengthy process. IEEE 802.1w, the rapid spanning tree configuration protocol (RSTP), mitigates this problem by providing mechanisms to detect failures and quickly reconfigure the spanning tree [IEE00b]. The recovery period, however, can still range from an optimistic 10 milliseconds to more realistic multiple seconds *after* failure detection, which is not adequate for many applications.

Ethernet does not support any mechanism akin to MPLS, which allows users to route packets/flows along a particular path [RVC01]. As a result, it is impossible to apply any traffic engineering techniques to balance traffic load across the network [Ash01]. Traffic engineering may not be critical in small local area networks, but it is very important in the context of metro networks. In particular, the ability to route traffic on a given route can greatly help enforce QoS by leveraging a traffic prioritization scheme (IEEE 802.1p), which prioritizes certain classes of traffic over others [IEE98a].

Ethernet does not support a redundancy mechanism such as multipathing for load balancing and fault-tolerance in cluster interconnects and SANs. Multipathing is an integral feature of SAN deployments. The per-port-cost factor, scalability of port and backplane bandwidth, and reliability issues provide grounds for moving away from an aggregation approach in clusters. This requires a solution which takes into account the issues that arise out of segregation of cluster nodes across multiple switches, specifically, capacity planning and fault tolerance in Ethernet.

1.3 Dissertation Overview

In this dissertation we address two important issues that prevent wide acceptance of Ethernet in metro area networks and storage area networks. (1) The challenge faced by Ethernet in the metro arena is that of traffic engineering. Ethernet lacks traffic engineering support, which prevents it from being used as a carrier network technology in the metropolitan network core. (2) In the context of storage area networks, in addition to the traffic engineering issue, Ethernet faces the challenge of capacity planning with redundancy and fault-tolerance support.

The key insight of this research is that modern Ethernet switches incorporate advanced network control mechanisms that are programmatically configurable and can be used to improve aggregate throughput, availability, and QoS in the network. Virtual LAN (VLAN) technology provides a mechanism to tag packets with different VLAN identifiers and logically divide a physical network into multiple broadcast domains on the basis of VLAN tags [IEE98b]. This mechanism aids the security and performance of LANs by limiting the size of broadcast domains. Multiple spanning tree (MST) protocol makes it possible to program multiple spanning tree instances on a network, each associated with a distinct VLAN, isolating traffic from one another [IEE02]. Finally, most Ethernet switches can limit the rate of incoming or outgoing flows over their physical interfaces using built-in rate-limiting features.

Most of these structuring mechanisms in modern Ethernet switches are accessible through SNMP, HTTP, or command-line interfaces. It is possible to programmatically configure VLANs and their associated spanning trees, as well as interface rate limits using management protocols. It is also possible to remotely monitor switches for failures and different activities can be triggered in reaction to these failure events. These features are thus referred to as the configurable features of modern Ethernet switches.

In the first part of this dissertation, we show that the programmatically configurable control mechanisms of modern Ethernet switches can be used to build the following high-level features that are critical to MEN, SAN, and CI applications:

- Traffic engineering that routes packet traffic to balance the load on physical network links. Routes obtained from switching path selection are enforced by means of VLAN tags in a fashion similar to MPLS labels.

- Proactive switch and link disjoint backup path provisioning to provide a high degree of tolerance for switch or link failures.

- Use of rate limiting features in Ethernet switches to regulate the bandwidth consumption by end nodes in order to isolate different traffic flows from one another.

In the second part of this dissertation, we address the issue of capacity planning in storage area networks. The goal is to identify the issues faced by Ethernet in SAN environments and address those issues.

- To this effect, we develop a network topology planning tool to minimize network infrastructure deployment cost. This topology planning work is specifically targeted towards providing automated tools to design Ethernet storage area network and cluster interconnect topologies that provide redundancy and fault-tolerance support.

1.4 Dissertation Outline

The rest of the dissertation is organized as follows. Chapter 2 presents an overview of Ethernet relevant to this dissertation. Chapter 3 reviews the related research in the area of traffic engineering and network planning. Chapter 4 gives an overview of metro area networks and metro Ethernet technology and we describe a multi-spanning-tree Ethernet architecture called *Viking* aimed at solving the traffic engineering problem in metropolitan Ethernet deployments. In Chapter 5, we give an overview of storage area networks and related technologies. In this chapter, we describe and evaluate *Cassini*, a tool developed for Ethernet SAN capacity planning In Chapter 6, we present a summary of research and conclude the dissertation with future research directions.

Chapter 2

Ethernet Preliminaries

In this chapter we provide a brief overview of Ethernet. The discussion is limited to technology aspects of Ethernet which are directly relevant to the dissertation. We specifically look into (1) Switched Ethernet architecture, (2) the Spanning Tree Protocol, (3) Link Aggregation, and (4) Virtual LANs.

2.1 Switched Ethernet

In its early days, Ethernet used coaxial cable as a shared bus. Nodes[1] connected to the shared bus accessed it using the CSMA/CD media access protocol. The nodes communicated with each other by sending data frames over the shared bus. The electrical integrity of the coaxial cable was of utmost importance for proper functioning of the network. Any minor fault or improper electrical termination of the coaxial cable would result in network disruption. Figure 2.1 shows an example of a cable fault on a shared bus Ethernet network. Addition or removal of network

[1]We use the terms *Devices* and *Nodes* interchangeably to refer to the end nodes connected to the network.

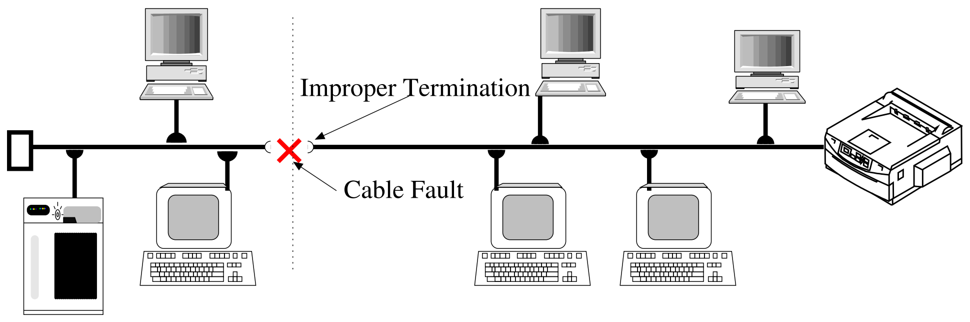


Figure 2.1: *Effect of a cable fault on a shared bus Ethernet network. The fault partitions the network, isolating multiple nodes. The improper termination caused by the fault affects all transmissions and the network cannot function.*

nodes also had a disruptive impact on network operation. At any point in time, a node could either only transmit or receive frames over the shared bus. This was half-duplex operation. The shared architecture also enforced that at any point in time there could be only one node pair communicating with each other. The shared bus architecture limited the scalability of Ethernet networks.

The solution to the bus topology problem was a *star* topology. In star topology, the shared bus was replaced with an active device called a *hub*. The hub was responsible for providing the entire functionality provided by the bus. Additionally, it was responsible for isolating faulty nodes/cables from the network. Nodes were connected to the hub using low-cost unshielded twisted pair (UTP) cable. Node connection points on the hub were called *ports*. The hub was also responsible for terminating open ports. Multiple hubs could be connected to each other to form an extended shared bus. Figure 2.2 shows a typical star topology utilizing a hub as the central connection point.

Although hubs replaced the shared bus, they were still collectively a resource shared among all the connected nodes. At any point in time only one node could transmit data. Data transmission still remained half-duplex. Thus, hubbed Ethernet

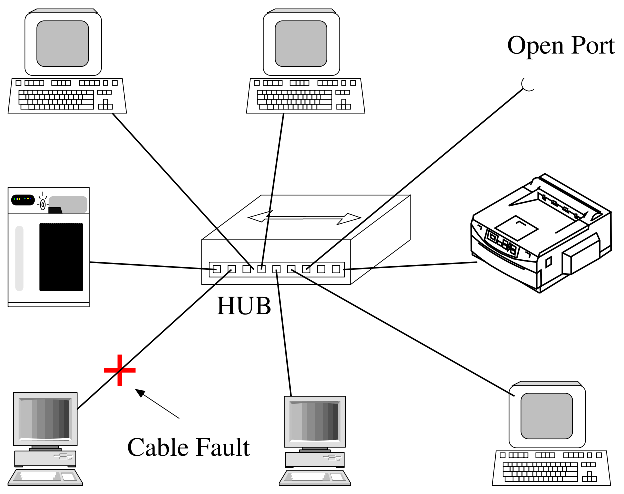


Figure 2.2: *Ethernet star topology using a hub as the central connection point. Nodes are connected to the hub using a low-cost unshielded twisted pair (UTP) cable. The hub is responsible for detecting and isolating faults in the network. It is also responsible for terminating open ports.*

did not provide scalability solution. The respite came from Ethernet bridges which connected multiple hubs together and isolated the Ethernet data link layer from the physical layer. This isolation allowed a network to be broken down into multiple segments where each hub, together with connected nodes, formed an independent segment. The bridges forwarded data packets from one segment to another only when required. This allowed multiple segments to carry out data transmission in parallel. Figure 2.3 shows an example of multi-segment Ethernet connected with a bridge.

In early bridges the bridging logic was implemented in software. Bridging was a CPU intensive and slow process. With hardware advances, it became possible to implement the entire bridging logic in hardware. Such *all-hardware* bridges were capable of forwarding data at line[2] speed. Also, with mass production, cost

[2]Line speed means the maximum possible speed on the connected medium.

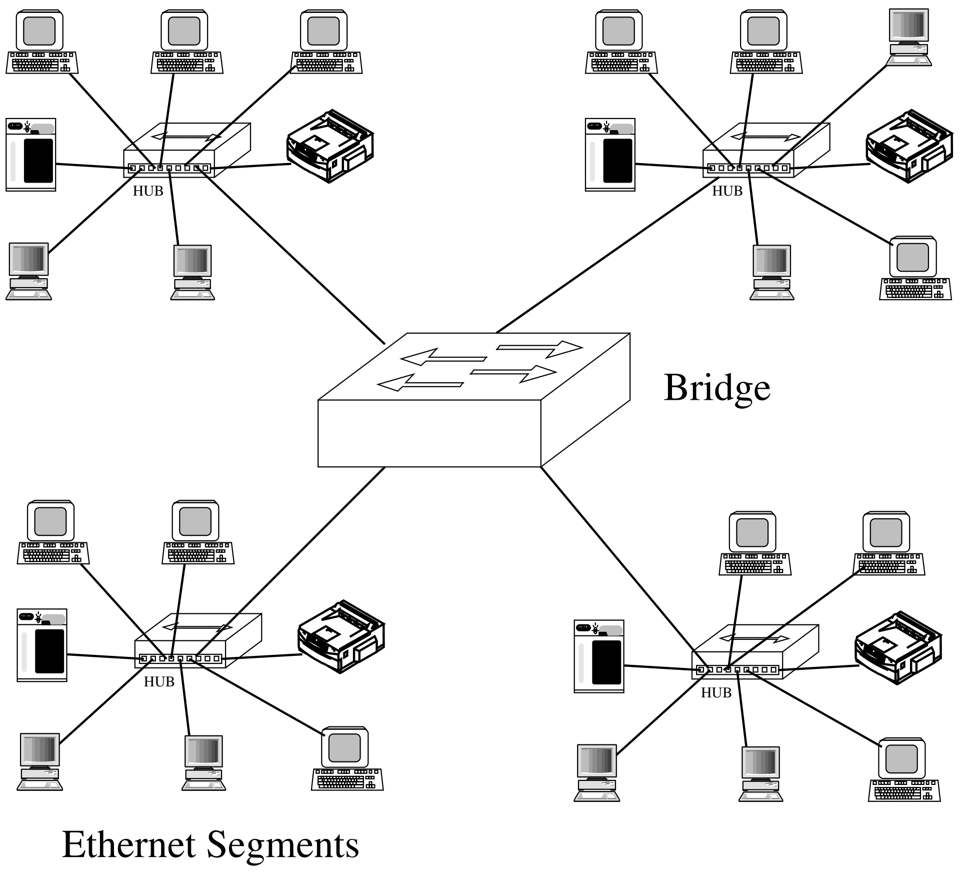


Figure 2.3: *Multi-Segment Ethernet connected together using a bridge. The bridge forwards data packets from one segment to another only when required. Each segment acts as a separate collision domain. Data transmission can proceed in each segment independent of the other segments.*

of all-hardware bridges was significantly reduced and it was possible to manufacture bridges with a large number of ports. This cost reduction allowed elimination of hubs. Nodes could be directly connected to bridges. Elimination of hubs resulted in elimination of collision domains and allowed connected nodes to operate in full-duplex mode. This was a significant development for Ethernet. All-hardware bridges were commercially called *Ethernet Switches* and the network mode with no collision domains, where nodes could operate in full-duplex mode, came to be known as the *Switched Ethernet*.

Switched Ethernet eliminated the shared bus and shared hubs. This elimination resulted in increased network throughput, by allowing multiple nodes to communicate in parallel. Further, full-duplex mode allowed nodes to transmit and receive data frames at the same time. Full-duplex mode, in effect, doubled the possible network throughput. Switched Ethernet is currently the default mode of deployment for Ethernet networks. Throughout this dissertation, we loosely use the term Ethernet to refer to switched Ethernet. Also, *bridges* and *switches* are used interchangeably.

2.2 Spanning Tree Protocol (STP)

In switched Ethernet, switches are responsible for forwarding data packets to appropriate ports. Switches keep track of node-to-port mapping by using a technique called reverse path forwarding. Each node connected to the network is assigned a unique MAC address by its manufacturer. Each data packet bears destination and source MAC addresses. These addresses identify the recipient and the originator node of the packet. Whenever a node transmits a packet, the receiving switch makes a note of the port on which the packet is received. It associates the source MAC address with the port in its internal forwarding table. Forwarding decisions for each packet are made by looking up the destination address in the forwarding table and finding the associated port. If the lookup fails, the packet is broadcast over all the ports on the switch. Eventually switches learn about port connectivity of all the connected nodes and populates the forwarding table with complete information. Occasionally new nodes are added or the ports of attachment change for old nodes. Switches respond to this situation by updating the forwarding tables.

If multiple switches are connected together, each switch associates the MAC addresses of all the nodes on every other switch with the port connecting the corresponding switch.

Reverse path forwarding works well as long as there are no loops in the network topology. Reverse path forwarding involves broadcasting of packets when the destination lookup fails. Packets broadcast by each switch are broadcast again by all the other connected switches. If there are loops in the network topology, the broadcast packets are eventually received back by the originating switch — only to be broadcast again. This results in a *broadcast storm* in the network.

To ensure that there are no broadcast storms in a network, a loop-free network topology has to be guaranteed. Ethernet uses the Spanning Tree Protocol (STP) to ensure a loop free network topology [Per85]. The STP was included in IEEE Standard 802.1D for Ethernet bridges [IEE90].

2.2.1 Tree Construction Process

Given a network topology as a graph, it is easy to compute a spanning tree for the topology. If the spanning tree construction has to be done without overall topology knowledge and in a distributed manner, it becomes an involved process. Ethernet switches using STP participate in the following distributed spanning tree construction process.

2.2.1.1 Root Election

Every Ethernet switch is assigned a unique 48 bit MAC address and a 16 bit priority. The MAC address is always fixed and the priority can be configured by users. The default priority is usually `0x8000`. During spanning tree construction all switches exchange information with each other using special packets called Bridge Protocol Data Units (BPDU). The switch with the lowest priority value is elected as the root of the spanning tree. If there is a tie, the switch with the lowest MAC address and the lowest priority is elected as the root.

2.2.1.2 Computing Least Cost Paths

After the root of the spanning tree is elected, switches start computing costs associated with all the paths that connect them to the root switch. The final spanning tree is such that the path from every node to the root switch is always the path with the least possible cost. The cost of a path is computed by summing up the cost associated with each link contained in the path. The path cost for the root switch is always 0. The path cost for all the neighboring switches is the cost associated with all the links connecting them to the root switch. The link cost depends on the data rate of the link. Table 2.1 shows the costs associated with different data rates. Once the path with the least cost to the root switch is identified for a switch, all links that would cause loops are disabled and a loop-free topology is achieved. In the active topology, ports that lead paths to the root switch are termed *root ports* and all other active ports are termed *designated ports*.

2.2.1.3 Port States

Every port on a switch can be in one of the following states:

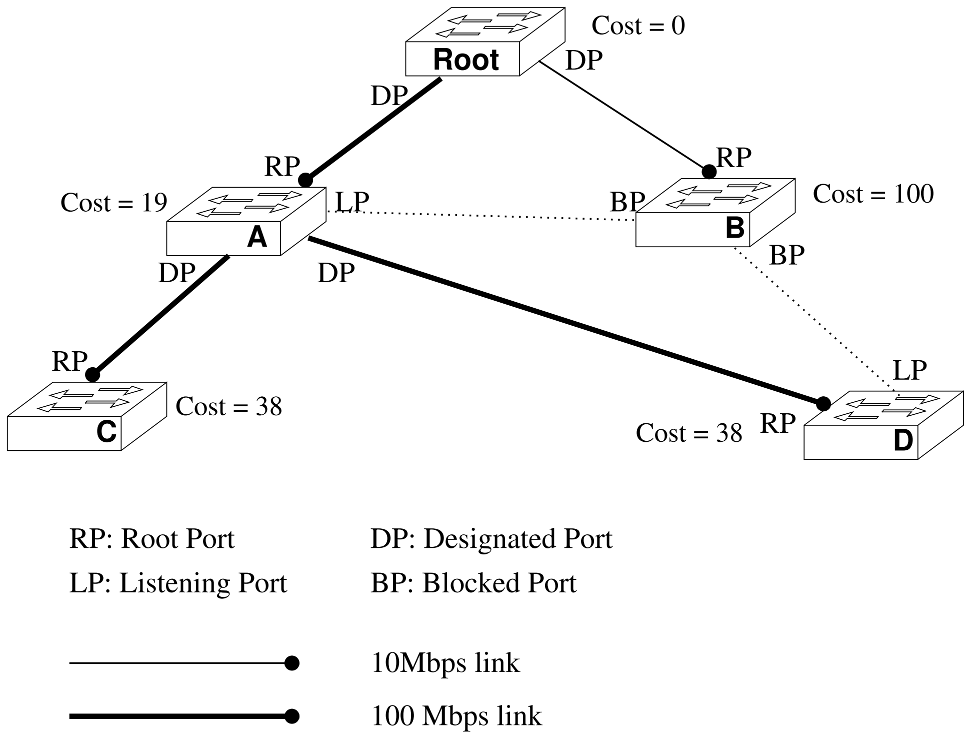


Figure 2.4: *Spanning tree creation in an Ethernet LAN. The Root node is elected based on switch priority and MAC address. All switches advertise their path cost to neighboring switches. Switches compare each other's path cost and determine the status of each port. In this figure switches* A *and* B *can reach directly to the root with path cost 0. Switch* A *advertises a path cost of 19 to switch* B*. Switch* B *advertises a path cost of 100 to switch* A*. Switch* B *blocks its port to switch* A *because it can reach the root switch with a lower cost. Switch* A *keeps its port to switch* B *in listening mode because a lower cost path may become available through it. Switch* D *chooses connectivity through* A *rather than through* B *because of lower cost.*

Data Rate	**Link Cost**
4 Mbps	250
10 Mbps	100
16 Mbps	62
45 Mbps	39
100 Mbps	19
155 Mbps	14
200 Mbps	12
1 Gbps	4
2 Gbps	3
10 Gbps	2

Table 2.1: *Link costs for different data rates*

- **Blocking**: This port is identified as a port that would cause a loop in the network. It does not participate in forwarding data packets. It still receives and forwards spanning tree construction-related BPDUs.

- **Listening**: A blocked port is changed to a listening port if the switch determines that an alternate path with lower cost to the root may be available through it. The listening port may become a forwarding port or may get blocked again, based on information received from other switches.

- **Learning**: A listening port is changed to a learning port before it starts to forward packets. A port stays in this state till a *forward delay timer* (usually 15 seconds) expires. In this state the switch learns about the nodes attached to the port.

- **Forwarding**: This is the active state for a port. The port participates in all switch activities while in this state.

- **Disabled**: This state corresponds to a completely non-functional port. In this state no packets are received by the port.

Figure 2.4 shows an example of spanning tree creation in an Ethernet LAN.

2.2.2 Rapid Spanning Tree Protocol (RSTP)

STP had certain drawbacks. The first was long convergence time. It took almost 30 to 60 seconds for a spanning tree to be constructed. The second problem was response to topology changes. Any topology change triggered by the addition and deletion of switches or switch failures was dealt with by another round of tree construction. This reduced the availability of the network.

The Rapid Spanning Tree Protocol (RSTP) was introduced as IEEE Standard 802.1w to provide faster recovery from topology changes [IEE00b]. RSTP is based on STP and is backward compatible.

In RSTP, each switch actively monitors the link status of each port and in the event of status change triggers a recovery process. It improves on recovery time by adding a new port designation, an *alternate port*. During spanning tree construction, information about alternate paths to the root switch is retained. Instead of blocking out alternate ports to the root, these are kept active. In the event that connectivity to the root port is lost, an alternate port is quickly converted to the root port.

RSTP changes the possible states that a port can be in. The blocking and listening states specified in STP are no longer used. There are only three possible states. These are *discarding*, *learning*, and *forwarding*.

RSTP aims to reduce recovery time for topology changes from minutes to a few seconds.

2.3 Link Aggregation

Spanning trees impose a restriction on Ethernet networks. At any point in time, there will be exactly one active link between any two switches. This restriction limits the maximum traffic between the two switches to the speed of the link connecting them.

Link Aggregation or Trunking is a way of combining multiple physical network links into a single logical link [IEE00a]. The logical link increases the capacity and availability of bandwidth between the connected switches.

Link aggregation is very useful in setting up high-speed backbone or core networks, where data transfer over the backbone is much more than what a single node can transmit.

Many LAN deployments usually a fat-tree-architecture where, the capacity requirement in the vicinity of root switches is significantly more than the capacity required elsewhere in the network. This increased capacity is provided by means of link aggregation.

IEEE Standard 802.3ad defines the specifics of link aggregation in Ethernet. Some of the goals of the 802.3ad standard are as follows:

- Increased bandwidth and availability, along with load sharing.
- Complete transparency to the end-nodes and higher-layer protocols.
- Backward compatibility with aggregation-unaware switches.
- Deterministic behavior of the link aggregation mechanism.
- Aggregation should be point-to-point. No multi-point support is possible.
- Only full-duplex operation is supported.

- Link aggregation between dissimilar links (with different data rates) is not supported.

2.4 Virtual LANs

The concept of VLANs is open to several interpretations. Each networking equipment vendor has different implementation strategies and hence there are different features of VLANs. Broadly speaking, VLAN is an abstraction over physical LANs which defines a limited broadcast domain independent of physical location. Essentially VLAN is a technology to group certain end-hosts together and segment large LANs into smaller broadcast domains. The grouping can be done according to some high-level membership policy. VLANs offer several advantages over traditional LANs. Some of these advantages are improved performance, ability to form virtual workgroups, simplified network administration, reduced cost of segregation, and improved security.

It is possible to define VLAN membership of end-hosts in several different ways. VLANs can be categorized by the way VLAN membership is defined.

2.4.1 VLAN Membership

Port-Based VLANs are defined by grouping physical ports of switches together. This approach was used in some initial implementations of VLANs. In this scheme, network switches are explicitly programmed to form VLAN port membership associations. This mode of VLAN membership is supported even today. The limitation of this method is that network administrators have to explicitly reconfigure the association with every physical topology change.

MAC Address Based VLANs use the physical addresses of the network interfaces to carry out membership assignment. This approach allows for physical topology changes, while retaining VLAN membership without the need of explicit reconfiguration. One of the limitations of this approach is the tediousness involved in keeping track of MAC addresses and their VLAN associations.

Protocol-Based VLANs concentrate on defining broadcast domains based on the traffic generated by end-hosts, rather than the end-hosts themselves. This scheme allows for segregating traffic based on its network-layer or transport-layer protocol.

Unlike other schemes, this scheme allows end-hosts to be members of multiple VLANs. This scheme is less popular, as the usual tendency is to group based on hosts rather than protocols. Further, most of the traffic in networks today is IP-based making the distinction on protocol alone less advantageous.

IP Address Based VLANs determine membership from layer-3 IP addresses, rather than layer-2 fields. These VLANs need advanced switches which can process layer-3 information. There are other variations of this scheme, such as using layer-3 addresses in conjunction with the higher-layer protocol of the traffic.

Explicit Tag Based VLANs use explicit VLAN identifiers in the frames transmitted by the end-hosts. These identifiers, termed VLAN IDs or tags, specify the membership of transmitting end-hosts. This way a particular end-host can be a member of multiple VLANs. IEEE 802.1q standard provides the specification for tag-based VLANs on switched Ethernet [IEE98b]. Though membership can be explicitly specified by the end-hosts transmitting the frames, the membership of end-hosts can be restricted by specifying a list of allowed VLAN membership for each switch port. Frames transmitted with tags other than the allowed VLAN membership are silently discarded by the switches.

The IEEE 802.1q standard specifies the VLAN identifier to be a 12-bit field embedded in the Ethernet frame header. The maximum number of possible 802.1q VLANs in an Ethernet network is limited to 4096.

2.4.2 Per-VLAN Spanning Trees

When there are multiple VLANs configured in a network, it is possible to increase network utilization by allowing construction of independent spanning trees for each VLAN. A spanning tree in a network of N switches enables only $N - 1$ links. All other links are blocked. If there are multiple spanning trees in a network, the overall utilization of the network can be increased by enabling more links and reducing the number of blocked links.

The Multiple Spanning Tree Protocol (MSTP) defined in IEEE 802.1s defines an extension to the 802.1d spanning tree protocol to enable multiple spanning trees in an Ethernet LAN [IEE02]. Each VLAN has its own spanning tree, with its own set of active and blocked links. Blocked links may be active in spanning trees corresponding to some other VLAN.

The MSTP protocol is totally backward compatible with the RSTP. This allows intermixing switches supporting MSTP and switches that do not support MSTP.

2.5 Managed Ethernet Switches

With technological advances, the complexity of Ethernet switches increased. Bridging functionality was enriched with several features, such as priority management for STP, VLAN management, port monitoring and mirroring, MAC filtering, switch health monitoring, statistics gathering, etc. It was possible to statically configure Ethernet switches and deploy them for enhanced operation.

It later became possible to manage Ethernet switch configurations without disrupting their operation. These were *managed Ethernet switches* which allow access to their configuration parameters.

The managed Ethernet switches allow access to their configuration through SNMP, HTTP, or command-line interfaces. It is possible to programmatically configure VLANs and their associated spanning trees, as well as interface rate limits using management protocols. It is also possible to remotely monitor switches for failures and different activities can be triggered in reaction to these failure events. These features are thus referred to as the configurable features of modern Ethernet switches.

We rely on these programmable features of managed Ethernet switches to enable application of Ethernet in Metro Area Networks and Storage Area Networks.

Chapter 3

Related Work

In this chapter we review prior research in the area of traffic engineering in networks and network topology design.

3.1 Traffic Engineering

Traffic engineering in networks deals with performance optimization in terms of capacity utilization. The main focus of such an optimization is to minimize over-utilization of certain capacity when other capacity is available in the network. Traffic engineering includes traffic measurement, modeling, characterization, control, and application of techniques to achieve performance objectives. It also includes capacity management through network design [Ash01].

Traffic control can be carried out at various granularities. For example, in IP networks each packet is routed independently. Routing decisions are taken at intermediate routers depending on the conditions prevailing at the time of packet arrival. Routing protocols deduce the utilization of network capacity by computing the cost associated with each available routing path. The cost function can take into account several factors, such as congestion levels on certain routes, delays involved, etc. While this approach is good for Layer-3 protocols such as IP, it is not practical to implement a traffic control solution of packet granularity at Layer-2, for performance and complexity reasons. Another approach suitable for Layer-2 protocols can be continuous monitoring, measurement, and periodic route configuration. With this approach, a dynamic reconfiguration can be carried out to balance the load among different network components. Though this approach cannot deal with instantaneous overload, long term control can be carried out effectively by simple reconfigurations after periodic monitoring and measurement.

3.1.1 Traffic Engineering in Ethernet

Traffic engineering in Ethernet is a complicated problem. In Ethernet, packets are always switched along the spanning tree of the network. The sending nodes do not have any control over the switching path. In large scale networks, such as MAN, the number of network elements (links and switches) involved in a path between distant end-hosts is usually high. Failure of any of these elements can cause a complete communication breakdown between the given pair of end-hosts. Further, the intermediate links are shared by multiple switching paths. This sharing may lead to an overload situation and the absence of alternate switching paths precludes load balancing by offloading traffic to other links. Figure 3.1 depicts a typical load imbalance scenario when switching is done along a spanning tree.

The IEEE 802.1s Multiple Spanning Tree Protocol extends the original Ethernet architecture by allowing multiple spanning trees to co-exist in the same physical network [IEE02]. Each spanning tree is associated with a unique virtual LAN (VLAN) as defined by IEEE 802.1q tag based VLAN mechanism [IEE98b]. We enable traffic engineering in Ethernet networks by leveraging on tagged VLANs to

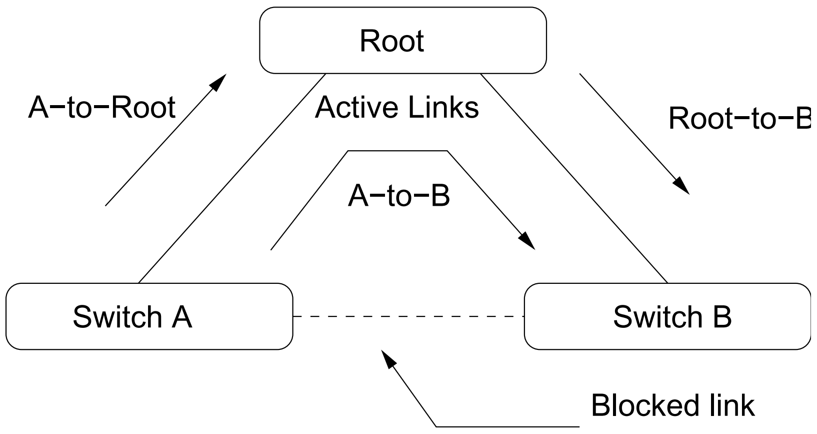


Figure 3.1: *A load imbalance scenario in spanning tree based switching. Three different flows, A-to-B, A-to-Root, and Root-to-B share the same set of links, despite the presence of a link between switch A and B. If somehow flow A-to-B can be switched along link A-B, the overall network throughput can be significantly improved.*

implement load-balanced switching paths [SGNcC04]. These paths are explicitly selected by specifying the VLAN tags associated with the corresponding spanning trees. This selection can be carried out by the end-hosts explicitly rather than by the network switches.

Another important facet of traffic engineering is network capacity design and planning. This requires a priori knowledge about the traffic in the network. Since we aim to provide traffic engineering solution for metro Ethernet, where this kind of a priori knowledge is hard to come by, we focus on traffic provisioning independent of the network design. We provides a mechanism to identify the critical portions of networks which need to be strengthened in terms of bandwidth or backplane processing. We defer our discussion about network planning and tuning in depth to Section 3.2 in the context of Cassini SAN designer.

Network traffic engineering is a widely researched topic for LAN and WAN. Further, the performance problems because of the single spanning tree in Ethernet is a well known issue. We now look at some prior work which tries to address deficiencies in Ethernet switching and improve performance. Prior research on improving network scalability and performance can be classified in three areas. One approach addresses the fundamental spanning tree issue by suggesting alterations to Ethernet switching. Another approach takes a holistic view of the network and focuses on proper planning and utilization of the network. Yet another approach addresses the Ethernet standard itself and attempts to improve the standard as a whole.

Research work in all three of these areas is now described.

3.1.2 Altering The Switching Mechanism

3.1.2.1 EtheReal

EtheReal is a real-time Ethernet switch aimed at providing connection-oriented bandwidth guarantees to multi-media/real-time applications [VC99]. This is a significant departure from the *best-effort* operation of Ethernet. To achieve real-time capabilities, EtheReal has stringent fault-tolerance requirements. EtheReal uses propagation-order spanning tree based fault recovery to improve failure detection. To do failure-recovery, EtheReal uses blocked links in one spanning tree while constructing another tree to recover from link failures.

Our traffic engineering approach in Viking is similar to EtheReal, but instead of using blocked links during spanning tree construction, Viking provisions multiple spanning trees in advance. This enables Viking to handle possible failures at most of the links (wherever possible) by using redundant links to construct backup paths. This saves time in recovering from failures. In EtheReal, all connections going through the failed link are terminated and are re-established after a new spanning tree is rebuilt. Whereas, Viking uses pre-calculated backup paths to route traffic in the link-failure cases, without causing any harm to existing connections.

EtheReal also suggests using multiple spanning tree based routing model to maximize bandwidth utilization. The goal of multiple spanning trees in EtheReal is to maximize the number of active links in the network. EtheReal was developed when the concept of VLANs was just taking shape and Ethernet switches did not have VLAN and multiple spanning tree support. With VLAN and multiple spanning tree support in commercial switches, the need for specialized switches like EtheReal was reduced. In many ways EtheReal can be considered a precursor to the Viking architecture.

3.1.2.2 AutoNet

AutoNet is an Ethernet-like self-configuring LAN architecture [RS91]. Unlike Ethernet, the switching/forwarding paths in AutoNet are not learned on demand but are precomputed. To support configurability, AutoNet uses a dedicated processor to monitor the network's physical configuration. In response to network topology changes, the forwarding tables in each switch are directly computed and updated.

AutoNet demonstrates the feasibility of reconfiguring network settings to pick proper paths to the destination by directly updating the forwarding table in switches. It operates by employing a monitor at the link level to detect error rates and link failures which triggers the recovery process. The recovery process initiates a distributed algorithm in all of the switches for topology acquisition and forwarding table recalculation. Once the recovery process is complete, newly populated forwarding tables identify the links that forward packets to their destination. AutoNet requires the switches in the network to be AutoNet compliant.

Ethernet in its earlier form relied on missing BPDUs to detect failure and invoke configuration changes. Fault detection in Ethernet was reactive rather than pro-active. Fault detection was a delayed process because missing BPDUs could be detected only after several seconds. This delayed failure detection and recovery affected the availability of the network. Today the situation is much different. Ethernet switches provide active link monitoring and fault detection mechanism, although the reconfiguration mechanism in Ethernet is still a lengthy process.

3.1.2.3 SmartBridge

SmartBridge is a bridge architecture proposed to address the scalability problems associated with spanning trees in LANs [RTA00]. In large LANs the root switches become traffic bottlenecks because the majority of traffic has to pass through them.

SmartBridges try to bring the scalability advantages of IP routing to LANs by staying within the confines of spanning-tree based switching but still somehow finding the shortest paths between end hosts.

SmartBridges require complete topology knowledge. This complete topology knowledge is acquired by inventory construction and topology acquisition logic built into every SmartBridge. SmartBridges also acquire knowledge about points of connectivity for every host in the network. Packet forwarding in the SmartBridge architecture is done along the shortest paths. Shortest paths are computed using the topology knowledge acquired during the acquisition phase.

Although shortest path switching may provide a low latency path, it does not address load balancing in the network. Reconfiguration times in the event of topology changes are remarkably low, around 10 ms to 20 ms. This, however, requires all bridges in the network to be SmartBridge compliant.

3.1.2.4 Spanning Tree Alternate Route

K-S. Lui et al. propose Spanning Tree Alternate Route (STAR) to find and forward frames over alternate paths that are probably shorter than their corresponding spanning-tree paths [LLN02]. This forwarding is done by making use of links that are traditionally blocked by the IEEE 802.1D standard.

In STAR bridging each switch maintains a *distance vector* which keeps track of its distance from other STAR switches. Each bridge further maintains a *host location* table which keeps track of the association between different hosts and the STAR bridges to which the hosts are connected. Using the distance vector and the host location, STAR bridges compute the shortest paths and forward packets along these shortest paths.

Although this approach reduces latency between most of the source and destination pairs, it risks overloading critical links, as the paths that it selects always remain static for a given network topology. Moreover, it is not obvious how it fares in recovering from a link or switch failure, as several records in all the tables then need to be recalculated.

3.1.2.5 Transparent Bridging

R. Garcia et al. describe a transparent bridge protocol for local area networking that accommodates topologies with active loops [GDS98]. The traditional IEEE 802.1D bridges are based on the spanning tree algorithm and thus disallow any presence of active loops in the topology resulting in wastage of bandwidth. This work, however, uses an improved routing scheme to allow highly-connected regular topologies, such as meshes, to be used without blocking any of the links. This protocol does not consider load on individual links while making routing decisions, The protocol involves substantial modification of frames, making it more complex and transferring additional data.

3.1.3 Network Planning and Load Balancing

3.1.3.1 Planning and Tuning

Network planning and tuning research by W. Qiao et al. addresses load balancing in networks by configuring an appropriate inexpensive topology based on source-and-destination pair load statistics [QN98]. They use block design principles to come up with the initial topology and a method for fine tuning to optimize parameters such as routing path length, traffic locality, inter-switch traffic, and channel utilization. In contrast, Viking tries to balance load at the link level, without any assumption about network topology, and develops an appropriate forwarding table, which is more useful in a practical scenario.

3.1.3.2 Valiant Load Balancing

Zhang-Shen and McKeown argue that traditional wide area networks and metro networks are grossly over-provisioned because of inherent issues with traditional routing mechanisms [ZSM08]. They suggest the use of a novel routing technique called Valiant Load Balancing (VLB) which deals with fault-tolerance and utilization. They demonstrate the utility of VLB in the context of Internet backbone networks.

In the Valiant Load Balancing architecture, the backbone network consists of a complete mesh of backbone routers. Each router acts as a connection point for an access network to the backbone. For a backbone network of N nodes with routing capacity of r, the links have capacity $2r/N$. The traffic entering the backbone is balanced equally by the ingress router to all N routers, regardless of the destination. All other routers forward the traffic to the egress router. Thus every packet in a traffic flow is forwarded twice in the backbone network. Every traffic flow is split into N sub-flows to be merged back at the egress router. The network is over-provisioned to only twice the required capacity. This is a very resilient network as it can respond quickly to any failures by simply not using the failed portion of the network.

Valiant Load Balancing tries to address the network utilization problem by suggesting the use of a mesh topology and load balanced routing. In the context of Ethernet networks, however, a mesh topology is not possible because of spanning tree issues.

Metro Ethernet providers can use Valiant Load Balancing in conjunction with VLAN based switching described in this dissertation to achieve a high utilization and fault tolerance in metro core networks.

3.1.4 Standard Support and Enhancements

3.1.4.1 MPLS

Multi-protocol label switching (MPLS) provides a framework for efficient designation, forwarding, routing, and switching of packets that flow through the network [RVC01]. It provides a means to map addresses to simple, fixed-length labels that can be used by switching and forwarding technologies to route packets. It is independent of the layer-2 or layer-3 protocols and manages traffic flows of various granularity, starting from hardware to applications. Viking can be thought of employing a similar idea where each packet contains a tag, the VLAN identifier (similar to MPLS labels) and the packet is routed to the destination host through the path in the corresponding spanning tree.

Our traffic control approach in Viking can be viewed as analogous to the establishment of virtual circuits in ATM networks or labeling in MPLS networks. Once Viking selects a switching path for any pair of end-hosts, the path is identified by the VLAN tag associated with the corresponding spanning tree. The VLAN tag can be considered similar to the labels in MPLS schema. Like ATM virtual circuits and MPLS labels, Viking tag assignment is also long-term persistent.

3.1.4.2 Rapid Spanning Tree Protocol

The IEEE standard 802.1w added changes to MAC bridge operation to provide a rapid reconfiguration capability [IEE00b]. These changes aim at a substantial reduction in the time taken to reconfigure and restore service in the existing spanning tree protocol upon a link failure or restoration. To achieve this, significant changes are introduced in the way the spanning tree algorithm works but backward compatibility with the 802.1D version is still maintained. Many metro Ethernet service vendors employ this approach to provide a superior level of fault tolerance compared to conventional Ethernet networks.

On a comparative note, Viking saves convergence time in building up the new spanning tree by maintaining a pre-computed VLAN tree, even before link failure occurs. Thus, once link failure is detected, switching to the new VLAN takes effect immediately unlike 802.1w. Viking assumes the use of IEEE standard 802.1s, a supplement to 802.1q This adds the facility for VLAN bridges to use multiple spanning trees that lets traffic for different VLANs to flow over potentially different paths in the virtually bridged LAN. This extension provides both rapid convergence and load balancing in a VLAN environment.

Cisco's Per-VLAN spanning tree (PVST) is a simple VLAN-sensitive implementation that relies on unique Bridge Protocol Data Units (BPDUs) transmitted by each switch for every VLAN, on a separate spanning tree process (STP) running on every switch for every VLAN. An enhanced implementation that extends the PVST, known as Multiple-VLAN Spanning Tree (MVST), allows similarly connected VLANs to be grouped into a single STP, making it more scalable without compromising any advantages. Viking relies on PVST implementation of Cisco.

3.2 Storage Area Network Design

Network topology design has been studied since the advent of computer networks. Research in optimal topology design is as old as networking technology itself. M. Gerla and L. Kleinrock present a survey of network design research during the initial deployment of the ARPANET [GK77]. In this survey they argue that many early computer networks were designed to provide access to a centralized computer service by a large number of remote users. Such centralized networks required a tree topology. A significant body of research focused on optimizing such tree topologies [FFCS71]. Gerla and Kleinrock pointed out that a tree structure is not appropriate for distributed resource sharing networks where traffic demands can arise between any two nodes of the network.

The SAN design problem can be considered a distributed network design problem, since there is no single traffic source or sink, but instead there can be communication between several storage clients and storage servers.

For distributed networks, the minimum-cost topology-design problem is a complicated problem with no efficient and exact solution. Certain heuristics were explored, namely, the Branch and X-Change (BXC) method [FFC70, SWK69] and the Concave Branch Elimination (CBE) method [Ger73, Yag71]. These methods were evaluated and several improvements, such as cut-saturation [Net74] and the Concave Link Elimination (CLE) procedure [SEA00] were suggested. All these methods tried to improve network connectivity while minimizing the overall cost of setting up network links. Although these methods are relevant to SAN topology design, the solutions cannot be directly applied. The reason is that the emphasis of these methods is on overall *link cost optimization* while interconnecting a fixed number of nodes. Whereas in SAN topology design, the SAN fabric needs to be designed to minimize the overall cost of both SAN switches and SAN links. SAN switches contribute to a large portion of SAN costs. The above mentioned methods do not address minimizing the number of switches and hence these methods are not directly applicable to SAN topology design.

We studied different approaches to network design. Although there is a significant body of research in generic network design, there is little research in SAN design. We review some generic design research. We also review some SAN-design research.

3.2.1 Generic Network Design

Khalil and Spencer present a LAN topology design mechanism which exploits the locality of traffic that occurs in any distributed network [KS91]. In this mechanism, nodes are combined into clusters so that the *traffic locality index* is minimized. Clusters are then assigned to different LAN segments with appropriate routing paths such that the overall *traffic balance index* is maximized. The idea is to split the entire topology design problem into two sub-problems and solve each one of them separately. Unfortunately, these two sub-problems are not independent of each other. Hence, it is not possible to find the local optimization point of each sub-problem such that the overall topology design problem is optimized.

Elbaum and Sidi stress that there is little advantage in solving the topology design problem by dividing it into sub-problems. For an efficient solution the design problem needs to be tackled as a complete problem [ES95]. To this effect they present a solution based on genetic algorithms. They improve upon the solutions presented by Khalil and Spencer with Ethernet networks in mind. They derive spanning tree topology for local area networks. Their optimization criteria is to minimize the average network delays. With decreasing communication latencies and increasing speeds in modern networks, average communication delays are less of concern than the overall network costs.

Kershenbaum et al. present a fast design algorithm for mesh and distributed networks called MENTOR [KKG91]. The inputs to the algorithm are the cost of the links between all pairs of nodes and the internode traffic requirements. The emphasis of the algorithm is on finding a good design, rather than an optimal design, which can serve as a starting point for further optimization. The cost functions are assumed to be concave functions depending on distance and capacity. A concave function suggests that the link costs do not increase linearly but sub-linearly, exploiting the economies of scale while adding extra link capacity. This algorithm can be embedded within other design procedures which subsequently refine the initial topology to seek alternative efficient topologies.

Selecting an appropriate initial topology is a very important activity in the overall topology design problem. Since topology refinement is based on heuristics and there is no single solution for appropriate final topology, the topology evolution is clearly influenced by the initial topology. There is significant advantage in using MENTOR algorithm to derive an initial topology for creating WAN topologies. The applicability of MENTOR in SAN topology design, however, is limited. The MENTOR algorithm is designed specifically for large networks where link costs dominate, rather than switching costs. There is an implicit assumption in MENTOR design that the end hosts can act as switching nodes. While this assumption is valid for networks with large geographical span, it is not valid for storage area network end-hosts. SAN end-hosts have limited amount of ports which can act only as sources and sinks for SAN traffic. Also, link costs do not have a significant bearing on network costs in SAN fabrics.

Qiao et al. address the topology design problem by configuring an appropriate inexpensive topology based on the source and destination pair load statistics [QN98]. Their approach is somewhat similar to the approach by Khalil and Spencer. They use block design principles to derive the initial topology and a method of fine tuning to optimize various parameters such as routing path length, traffic locality, inter-switch traffic, and channel utilization. Their emphasis is on minimizing the maximum and average path length between communication node pairs. With decreasing average path length, the number of network elements required for each flow is minimized and thus network cost is also reduced. It is network resource sharing between different flows, however, that brings the network cost further down. The implicit assumption is that the overall network cost is minimized in the end. Minimization of path length, however, results in clustering of network nodes together and the optimizations are effective only in pockets of the network, resulting in load imbalance in the network.

Most of the above mentioned work emphasized network costs due to links alone, average delay, and/or path-length optimizations. The number of nodes and the traffic within them is an invariable factor for all the above algorithms. This assumption is not valid while designing a SAN fabric. With increasing link speeds and decreasing latencies, average delay is not much of a concern for SAN traffic. Further, SAN fabric incurs network costs primarily due to network switches rather than network links. To address these issues, these existing techniques need to be adapted to optimize both link costs and the cost of network switches.

3.2.2 Automatic SAN Design

There are a limited number of tools specifically targeted towards SAN design. Further, we have not come across any tool which aids Ethernet0based SAN design. This makes Cassini, the Ethernet SAN design tool described in this dissertation, as perhaps the only existing tool focusing on Ethernet SANs.

3.2.2.1 Appia SAN designer

Appia is one of the few tools specifically targeted towards the design of storage area network fabrics using Fibre Channel [WOSW02]. Appia was developed to work in concert with other storage area networking design tools, which focus on workload and device performance information to configure storage devices and determine data placement. Appia also uses workload data-flow information to come up with a SAN topology design. Appia consists of two independent algorithms, namely *QuickBuilder* and *FlowMerge*. These algorithms differ from each other in terms of their applicability and strengths. FlowMerge is a computationally intensive algorithm used to design SAN topologies which require sparse connectivity. QuickBuilder is useful in designing densely connected networks. The primary requirement of these algorithms is that the nodes form a bipartite set of *hosts* and *devices*. The data-flow information is always about flows between hosts and devices and never across hosts or devices.

Appia formulates the SAN design problem as an integer programming problem. Heuristics are developed to substitute the computationally expensive integer programming solution.

The topologies generated by Appia are layered topologies that do not conform to any reference topology. The topmost layer consists of all hosts in the network and the bottom layer is all the storage devices in the network. The middle layers consist of switches and hubs. The network elements at a particular layer connect to network elements in preceding and following layers. This forces switching paths to consist of links from different layers. Though this kind of switching is symmetric with short switching paths, capacity in one layer cannot be utilized in other layers, resulting in load imbalance and wasted capacity.

3.2.2.2 SANTK: SAN Tool Kit

SANTK (Storage Area Network Toolkit) is a tool developed to aid SAN designers design Core-Edge topologies [Str01]. This tool is very similar to Cassini. Major differences, however, exists in capacity planning within the network and the comprehensiveness of connectivity planning.

SANTK uses pre-designed network topologies as building blocks and puts them together in a pre-defined pattern, so as to satisfy the port requirements imposed by the SAN design input. The SAN design process focuses only on port requirements and no capacity planning is carried out to satisfy flow requirements. Further, only the connectivity of Fibre Channel switches is provided and no planning for interconnecting hosts and devices to the fabric is provided.

The emphasis of this tool is on visualization of SAN topology to aid SAN designers to manually tune a generated SAN topology. It also performs scalability analysis on designed SANs by identifying the over-provisioning of ports. SANTK acts as a design and visualization aid, rather than a full-fledged automated SAN designer.

3.2.2.3 Integer Programming based SAN Designer

Thompson presents a formulation of integer program developed in Appia tailored in the context of Core-Edge topology [Tho04].

This research focuses on finding an optimal Core-Edge SAN connecting all hosts and devices together such that the cost of the SAN is minimized. This is perhaps the only automation solution that focuses on Core-Edge topology for SAN design. The SAN design, however, is only for small sized Fibre Channel fabrics. This is a computationally expensive approach of solving the SAN design problem and does not scale with SAN size. Further, the focus is entirely on connecting individual hosts with individual storage devices.

In contrast, Cassini focuses on the clustered architecture of hosts and devices in modern data centers and carries out automated SAN topology design. Cassini focuses on Ethernet specific issues such as spanning tree problems. It also focuses on providing features, such as zoning and multipathing in Ethernet environment.

Chapter 4

Viking: A Multi-Spanning Tree Ethernet Architecture

4.1 Metropolitan Area Networks

Metropolitan Area Networks (MAN) are networks where the geographical span extends to the boundaries of metropolitan cities. Typically, a MAN is a set of interconnected LANs that work together to provide access and services within a metro region. Similar to the Internet, MAN are not owned by a single organization. The network resources used in MAN are usually owned by either a consortium of users or by MAN Service Providers who charge users for MAN services. The level of service provided to different users depends on the service level agreement (SLA) between the provider and the users. Since metro services are provided on monetary returns basis, service providers strive to maximize revenues by maximizing the utilization of their network resources.

The users of metro networks are enterprises seeking to interconnect their branch offices in a city, academic institutions trying to interconnect various campus LANs, or organizations connecting to Internet service providers located within the city.

Figure 4.1 shows how a Metro Area Network connects multiple enterprise campuses.

4.1.1 MAN Infrastructure

The geographic span of MAN extends to the metropolitan limits. This span ranges from 3 to 30 miles. The bandwidth requirement in MAN is usually several 100 Mbps to a few Gbps. For such distances and bandwidth requirements, copper-based cabling is not a suitable option. Copper-based cabling can give high performance when used for short distances, up to 100 to 200 meters. Optical networks are more suitable for medium-haul metro networks.

The network infrastructure in MAN is predominantly optical-networking based protocols and equipment supporting those protocols. The technologies widely used to deploy metropolitan networks include, but are not limited to, SONET/SDH, ATM, etc. Recently there is a growing shift from these circuit-switched technologies toward packet-switched technologies such as RPR, MPLS, etc. The reason behind this shift is that conventional MAN technologies are primarily tuned for voice traffic, whereas packet switching is more appropriate for data, voice, and video convergence. Today, primary users of Metropolitan networks are enterprises seeking Internet connectivity or other services, such as Virtual Private Networks, between geographically separated sites. Thus, the bulk of the transmission over Metro networks is not voice but data, which can be handled more effectively by packet-switched transport mechanisms rather than circuit-switched technologies.

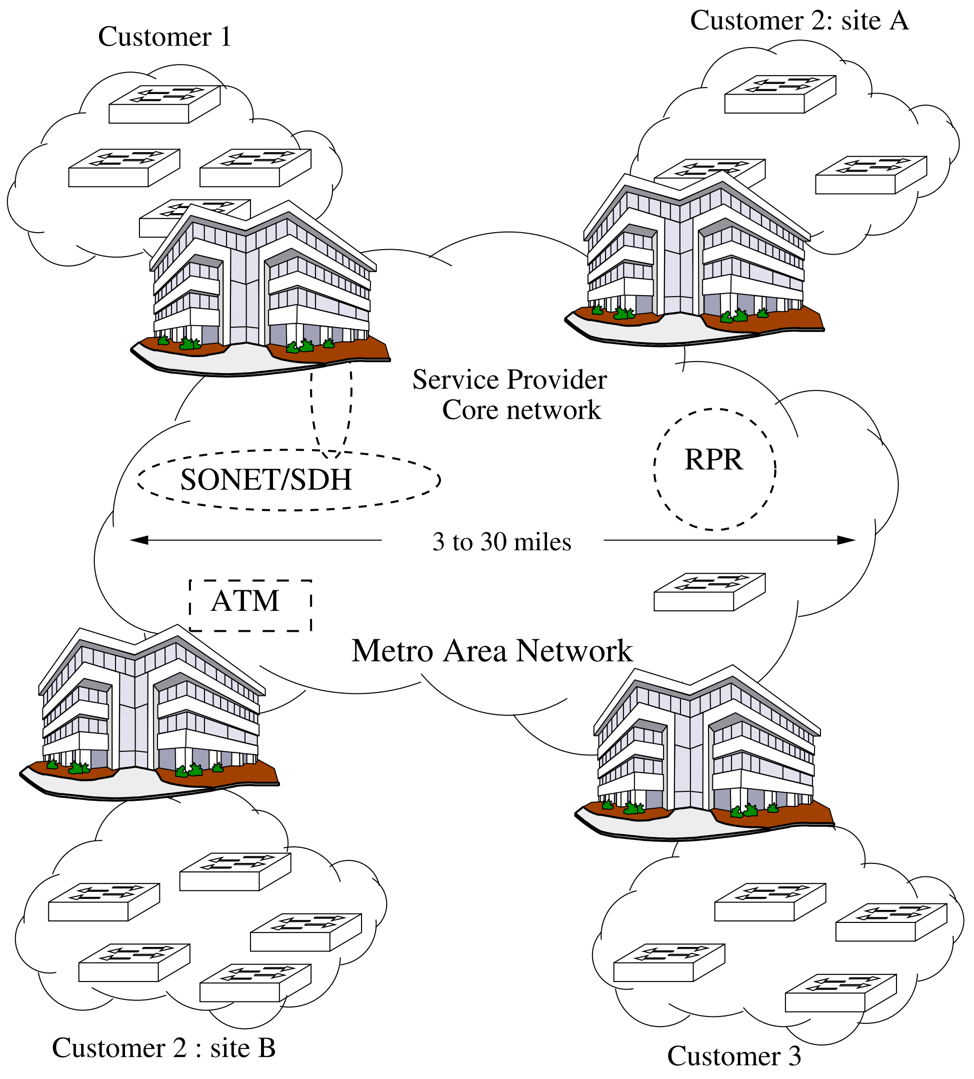


Figure 4.1: *An example of a Metro Area Network. The geographic span of MAN is usually 3 to 30 miles. Service providers maintain the core of metro networks. Metro networks provide connectivity to multiple enterprises to transparently interconnect their campuses. Service provider core networks use technologies such as SONET/SDH, ATM, MPLS, RPR, and Ethernet.*

Name	**Distance**	**Standard**	**Remarks**
100BASE-BX10	10 km	802.3	100Mbps over single-mode–fiber (SMF)
100BASE-LX10	10 km	802.3	100Mbps over a pair of SMF
1000BASE-LX	2 km	802.3	1Gbps over SMF
1000BASE-LX10	10 km	802.3	1Gbps over a pair of SMF
10GBASE-LR	10 km	802.3ae	10Gbps over SMF
10GBASE-ER	40 km	802.3ae	10Gbps over SMF

Table 4.1: *Different Ethernet physical-layer technologies and supported distances*

4.1.1.1 Metro Ethernet Technology

Ethernet is an emerging technology in the long-range optical-networking arena. Recent advances in Ethernet physical-layer technology and standardization of Gigabit and 10 Gigabit Ethernet for long distance have made Ethernet a serious contender for metropolitan core switching. The IEEE standards 10G-BASE-LR and 10G-BASE-ER define long-range and extended-range operation of Ethernet for distances up to 10 km and 40 km respectively [IEE04a]. There is also work in progress to make 40 Gbps and 100 Gbps Ethernet for operation over these distances. Table 4.1 lists different Ethernet physical-layer specifications for long-haul networks and their relative operating distances.

4.1.2 Metro Ethernet

Ethernet is the dominant Local Area Networking technology. The end-user networks of MAN clients are almost invariably Ethernet LANs. Almost 98% of data traffic in corporate LANs is over Ethernet networks [For02]. The most convenient, cost effective, and transparent way of interconnecting these LANs is to use Ethernet bridging to bring them together. One of the most popular services in metropolitan networks is *Metro Ethernet Service*.

4.1.2.1 Metro Ethernet Services

There is a subtle distinction between the terms *Metro Ethernet Services* and *Metro Ethernet Technology*. Metro Ethernet *services* are typically provided by Metro Ethernet Network providers to their end users. These services form the access services in the network. Whereas, metro Ethernet *technology* is the carrier technology that is deployed by service providers in core metro networks to provision various metropolitan services, including metro Ethernet services. With the advent of Gigabit and 10 Gigabit capabilities, Ethernet has become an attractive metropolitan carrier technology. There are multiple reasons behind the quick acceptance of Ethernet in metro networks. Some of these reasons are cost effectiveness, ability to provision connectivity and bandwidth on demand, ease of inter-working, and a large existing user base which enables ubiquitous adoption [For02].

Metro Ethernet Forum (MEF) is a consortium of network service providers and other organizations with interest in metro Ethernet [mef]. MEF is responsible for standardizing metro Ethernet service specifications [For03]. According to MEF, end-users can avail themselves of metro Ethernet services in two basic forms. Namely, the Ethernet Line (E-Line) service and the Ethernet LAN (E-LAN) service. End-users are provided with User-Network Interfaces (UNI), which are the points of presence for service providers at the end-user premises. The E-Line service provides a point-to-point Ethernet virtual connection between any two UNIs and thus it provides point-to-point Ethernet connectivity between end-users. E-LAN service, on the other hand, provides multipoint-to-multipoint Ethernet connectivity between multiple UNIs. Ethernet frames sent from one UNI may be received at one or more of the other UNIs. From an end-user perspective, E-LAN services present a LAN abstraction over a metropolitan network core. These services do not depend on the

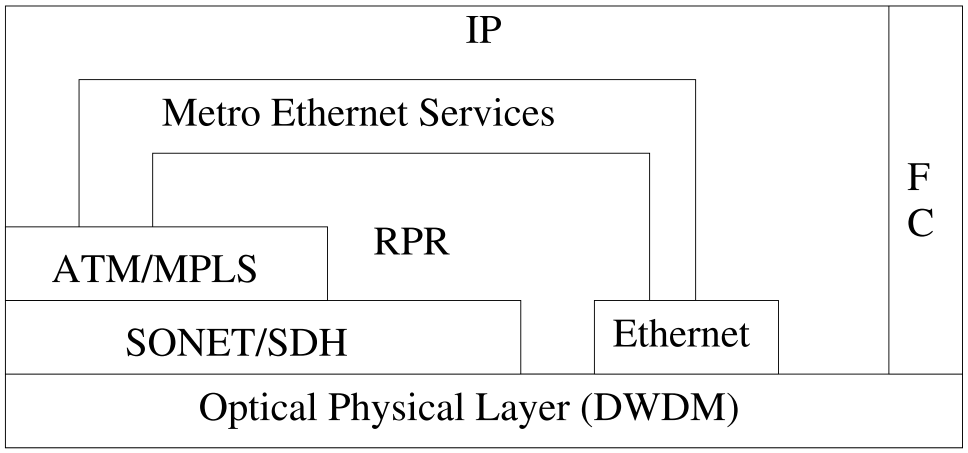


Figure 4.2: *Metropolitan Area Networks technological landscape. Optical fiber technology forms the basis of medium-haul communication in the metropolitan networks. The majority of services are provisioned using circuit-switched data-link technologies, such as SONET/SDH. Other packet-switching technologies, such as MPLS and RPR are actually built on top of SONET/SDH. Metro Ethernet services are typically provisioned over these established technologies. Service provisioning using Ethernet technology in the core is still rare.*

underlying metro transport network and can be provisioned over any possible metro transport protocols, such as SONET, RPR, MPLS etc.

It is natural to expect that the majority of Ethernet services would be provisioned using Ethernet technology as the underlying transport technology. This is, however, not the case. Although metro service providers have been quick to accept Ethernet as one of the carrier technologies, they have been hesitant to completely depend on it for provisioning major services for reasons mentioned below. The majority of services are still provisioned using legacy technologies, such as SONET/SDH. The current technology and service landscape in metro networks is shown in Figure 4.2.

4.1.2.2 Limitations of Ethernet Technology in Metro Ethernet

The Metro Ethernet Forum has identified key issues faced by incumbent and new-entrant service providers in metro networks that use Ethernet as the carrier technology [mef]. Some of these issues are quality of service guarantees, in-service operation and maintenance, protection mechanisms, and network utilization [For02].

In terms of QoS, Ethernet lacks support for any sort of admission control mechanism. When new flows are added to existing traffic, there is no way to ensure that the existing flows will have their requirements honored. There is no way of ensuring an optimal path for a traffic flow, because the spanning tree of the network dictates the switching paths for all flows. In terms of protection mechanisms, the glaring problem is that of recovery after network failures. The resource utilization problem again arises out of the spanning tree protocol, wherein Ethernet networks fail to utilize all the links in the network because not all get included in the spanning tree of the network. We recognize all these issues as offshoots of the *lack of traffic engineering capability* in Ethernet.

Several other operational and maintenance issues also need to be addressed. Two of the issues are in-service troubleshooting capabilities that do not disrupt network operation and fine-grained packet level scheduling and packet coloring, etc.

4.1.3 Motivation behind Viking

One of the goals of this dissertation is to address these traffic engineering issues in Ethernet, making it a viable option as a carrier technology in the metro core. The goal is to provide a technological solution that can enable Ethernet technology in metro networks. Our research aims to do this, while staying within the confines of existing standards and relying on existing capabilities of current Ethernet equipment. We develop a multiple spanning tree based architecture called *Viking* to address the traffic engineering problem in Ethernet.

In Viking we propose a performance and fault-tolerant solution for Metro Ethernet. The root cause of the performance bottle-neck and lower fault-tolerance threshold is the single spanning tree switching methodology used by these networks. A single spanning tree also precludes the possibility of multiple switching paths between a pair of end-hosts.

One simple way to provision multiple switching paths between a pair of end-hosts is to use multiple spanning tree instances. Since each spanning tree is responsible for providing a switching path between any pair of end-hosts, availability of multiple paths directly ensues from the presence of multiple spanning trees. The overall performance of the network can be improved by increasing the number of active links and reducing the number of blocked links in the network. The overall throughput can also be increased by providing multiple redundant links, all of which are active. This can be achieved by constructing the spanning trees in a careful fashion that do not overlap with each other.

If spanning tree instances are constructed in an intelligent fashion, it is possible to distribute the respective root switches across the network topology. Since the majority of switching load is also distributed across various roots, the need for faster links in the vicinity of roots is eliminated by making use of multiple redundant links.

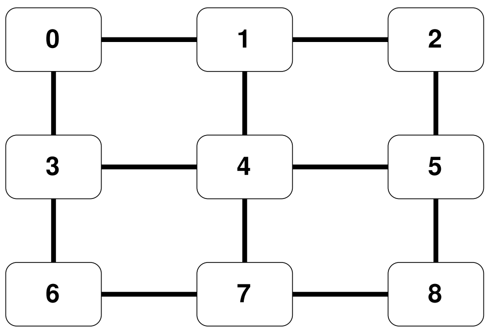
Viking addresses the traffic engineering problem by coupling the use Virtual LAN (VLAN) technology with the multiple spanning tree approach. The IEEE 802.1s standard provides a provision for maintaining multiple spanning tree instances on individual VLAN basis. Viking leverages this facility to derive load balanced switching paths that can be explicitly selected by specifying the VLAN tags associated with the corresponding spanning trees. This selection can be performed by the end-hosts, rather than the network switches.

In this chapter we describe and evaluate in detail how Viking addresses the traffic engineering problem.

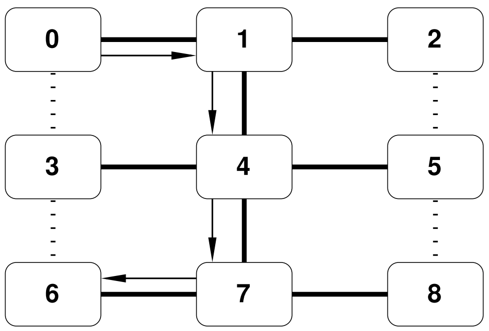
4.2 VLAN-based switching

The IEEE 802.1s MST protocol allows for the existence of multiple spanning trees in an Ethernet network, where each spanning tree corresponds to a different VLAN. These multiple spanning trees can be used to provide load balanced and fault-tolerant switching paths for different communicating nodes. The path selection can be done in two ways. In the top-down approach, one can configure a large number of spanning trees such that the combination of all these trees encompasses all the links in the network. The switching paths can be selected at run time based on utilization of different links so that the overall network throughput is maximized. Alternatively, in the bottom-up approach, given the traffic profile or traffic requirements of any network, one can come up with different load balanced switching paths such that the overall network utilization is efficient [Gop03]. These load balanced switching paths can be aggregated together such that there are no loops in the aggregation. Such a constrained aggregation would yield multiple spanning trees. Each spanning tree can further be associated with a unique VLAN tag and every packet can be switched along the corresponding spanning tree, based on the VLAN tag.

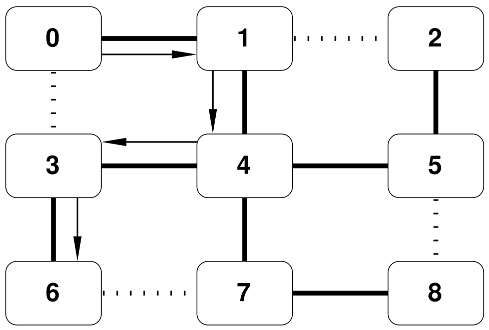
The IEEE 802.1q VLAN specification can accommodate a maximum of 4096 VLANs. The number of spanning trees required in a top-down approach increases drastically with the network size and hence the top-down approach cannot be used effectively in large scale networks. In a bottom-up approach, the number of spanning trees, and hence the number of required VLANs, depends on the number of switching paths. Based on this restriction we opted to use the bottom-up approach



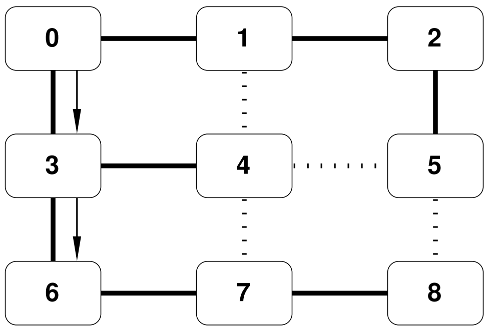
(a) Network Topology



(b) VLAN X Spanning Tree



(c) VLAN Y Spanning Tree



(d) VLAN Z Spanning Tree

Figure 4.3: *Different possible spanning trees for a given topology. Each spanning tree can be associated with a unique VLAN tag. Switching paths from different spanning trees can be selected by appropriate VLAN tags. (a) represents an example mesh network topology, (b), (c), and (d) represent spanning trees corresponding to VLAN X, VLAN Y, and VLAN Z. A communication between hosts connected to switch 0 and switch 6 would take paths 0-1-4-7-6 with VLAN X, 0-1-4-3-6 with VLAN Y, and 0-3-6 with VLAN Z. Availability of alternate switching paths enables load balancing, which can form a basis for efficient traffic engineering. Further, different spanning trees load different links to different extents. An intelligent spanning tree configuration approach can provide uniform load distribution across all links, so that the overall end-to-end network throughput is maximized.*

in VLAN-based switching.

For every incoming packet, Ethernet switches analyze the packet header for

VLAN tags. If a VLAN tag is found, the packet is switched along the corresponding spanning tree. Thus, any specific switching path can be selected by simply inserting the appropriate VLAN tag in the packet header. A host can insert different VLAN tags while communicating with different destination nodes and thus can select different load balanced switching paths. Figure 4.3 shows an example scenario where different spanning trees yield different switching paths which can be used to balance the network load. This mechanism is very much analogous to MPLS, where the packet switching paths are selected based on the labels present in packet headers. The difference is that, in MPLS, the labels are inserted by ingress routers, whereas here the VLAN tags need to be inserted by the end-hosts.

This VLAN-based switching is feasible only if desired VLAN spanning trees can be imposed (configured) on any network. This is where the configurable features of Ethernet switches come into the picture. Almost all switches, which provide support for the 802.1s MST protocol, facilitate configurability of links in terms of associated VLANs. Usually, whenever a VLAN is associated with different switches (links) in a switched network, the switches participate in a distributed spanning tree setup process and build a packet forwarding spanning tree for that VLAN. If the links with which a particular VLAN is associated already form a spanning tree, this spanning tree becomes the default switching spanning tree for that VLAN. Thus, with the availability of VLAN configurable features in switches, it is possible to realize controlled VLAN-based switching, which enables efficient traffic engineering.

To provide an effective traffic engineering solution, Viking needs to address the following issues :

Topology Knowledge

Since Viking can be used independent of network design and planning, it needs to acquire knowledge about network topology by external means. This knowledge can be acquired in an automated fashion by using available network topology discovery tools. Some of the works that facilitate topology discovery of Ethernet are Topology-d [OG98], IDMaps [JJJ$^+$00], and Remos [BGM$^+$99]. Alternatively, topology information can be manually provided by network administrators. This does not pose any problem since, for large scale and planned networks such as metro Ethernet, topology information has to be maintained in some configuration database for efficient management. This information can readily be provided to Viking after appropriate transformations.

Load Characterization

Viking needs to measure and monitor the network load continuously, so as to carry out spanning tree reconfigurations for efficient load balancing. It addresses this by implementing a statistics gathering mechanism. Statistics are continuously collected at end-hosts during network activity. The distributed traffic information is periodically integrated to obtain periodic pair-wise network utilization information. This information is used by Viking traffic management logic to determine switching paths, which utilize the network resources in a load balanced fashion.

Resource Provisioning

Resource provisioning is perhaps the most important part of Viking. Once the information about network topology and the pair-wise load statistics are obtained, Viking needs to derive appropriate switching paths between given node pairs. These switching paths further need to be combined to form spanning trees which can be associated with different VLANs. This task is divided into three primary sub-tasks, namely Path Selection, Path Aggregation into spanning trees, and Spanning tree configuration. Path information also needs to be percolated to the end-hosts for final path selection. This is because ultimately the end-hosts are responsible for transmitting frames with appropriate VLAN tags, so that the frames are switched along the selected paths. Further, in the event of reconfiguration, the end-hosts need to be informed about changes in switching paths in a transparent fashion. Details of path selection, path aggregation, spanning tree construction, and information percolation are provided later in this section.

Fault-Tolerance

Apart from load balancing and efficient resource utilization, Viking also needs to address the issue of switch and link failures in the network. To effectively tackle failures, Viking pre-computes backup switching paths for each pair of end-hosts. The backup paths need to be node-and-edge disjoint paths, when compared to the primary paths. Viking strives to provide backup paths in a fashion that minimally impacts the load balance scenario. In the event of failures, the end-hosts can be simply informed to use the backup paths by inserting into the frames the VLAN tags corresponding to the backup path spanning tree. Viking needs an effective failure detection mechanism so that the end-hosts can be informed about alternate path selection within a short duration of failure occurrence. For this purpose, Viking relies on the failure detection support provided by the network switches.

4.3 Resource Provisioning

As described earlier, traffic management in Viking consists of three primary tasks, Path Selection, Path Aggregation into spanning trees, and Spanning tree construction.

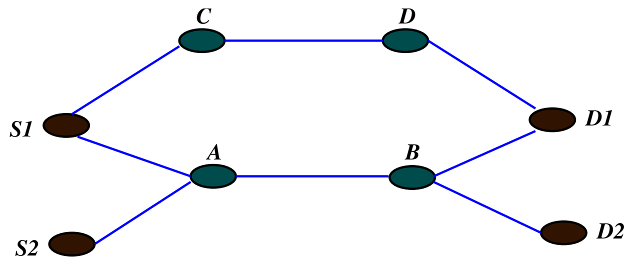


Figure 4.4: *A simple problem of selecting a route from S_1 to D_1. Selecting the route (S_1, C, D, D_1) leaves the critical link (A, B) free for future virtual connections between S_2 and D_2.*

4.3.1 Path Selection

The goal of the path selection algorithm is to maximize network resource utilization by supporting the expected traffic between as many source and destination nodes as possible. To achieve this goal we use a primary-backup path selection algorithm called Link Criticality Based Routing (LCBR) [Gop03]. The main intuition behind this route selection algorithm is to find routes that balance the loads across different parts of the network and, to the maximum extent possible, avoid critical links in the network that will carry significant load.

Consider the network topology shown in Figure 4.4. We need to select a route between nodes S_1 and D_1. There are two candidate routes: (S_1, A, B, D_1) and (S_1, C, D, D_1). Which of these two routes is better from the perspective of network usage efficiency? Say we also expect traffic between nodes S_2 and D_2. Then the best route to select between S_1 and D_1 would be (S_1, C, D, D_1), because it leaves the resources along the link (A, B) free for traffic between S_2 and D_2. The challenge here is to identify that link (A, B) is a critical link.

The path selection algorithm selects a primary route X and a backup route Y that can support an expected bandwidth requirement of $B(s, d)$ between a given source s and destination d. The backup route Y provides the guarantee that, if at most one network element (link or node) fails and the network element happens to lie on the primary route X, then the corresponding source-destination traffic will be diverted to Y. Thus we are guarding against the possibility of a single network element failure. Note that the basic requirement for being able to tolerate single-element failures is that the primary and backup routes must be completely disjoint with respect to all intermediate network elements. This is to ensure switch-over to the backup route if any one element fails along the primary route.

We say that two primary routes ***intersect*** with each other at network element e if both the primary routes pass through element e. Whenever a network element e fails, we need to activate backup routes for all the primary routes that intersect at e.

Every link l has one ***primary set*** $Prim(l)$ that contains the IDs of all the primary routes that pass through link l. In addition, each link has a total of $(m + n)$ ***backup sets*** of reservations, where each set corresponds to one network element; m is the number of links and n is the number of nodes in the entire network. Each backup set at link l is represented by $Bkp(l, e)$, $1 \leq e \leq (m + n)$, where each backup set corresponds to one network element e. The backup set $Bkp(l, e)$ contains IDs of those primary routes whose backup routes traverse link l and whose primary routes intersect at the network element e. In other words, $Bkp(l, e)$ represents the set of backup reservations at link l that need to be activated in the event of failure of network element e.

Recovery from failure of a network element occurs as follows. During normal operations, each link carries traffic for its primary set $Prim(l)$. Whenever a network element e fails, its corresponding backup sets $Bkp(l, e)$ are activated at all the links l of the network. Activating a backup set at a link l implies that, in addition to the traffic for routes in primary set $Prim(l)$, the link carries the traffic for reservations in its backup set $Bkp(l, e)$.

Assuming that at most one network element can fail at a time, the residual capacity R_l of a link l is calculated as follows.

$$R_l = C_l - \sum_{(s,d) \in Prim(l)} B(s,d) - \max_{All\ e} \sum_{(s,d) \in Bkp(l,e)} B(s,d) \tag{4.1}$$

In other words, the residual capacity R_l is obtained by deducting the sum of primary reservations and the maximum of the sum of the backup reservations, in each backup set, from the total link capacity C_l.

First we define a network-wide metric $cost(G)$ that measures the extent of load on resources in network G. An important factor in computing $cost(G)$ is the notion of *expected load* ϕ_l on link l. The expected load ϕ_l indicates the importance of a link l in terms of how critically all the source-destination pairs in the network need the link for carrying their traffic. Assume that a total of x network routes are possible between a source-destination pair (s, d). Out of these, say y routes pass through link l. Then the criticality of the link l with respect to source-destination pair (s, d) is defined as the fraction of routes between s and d that pass through link l, i.e. $\phi_l(s, d) = y/x$.

Assume we have the knowledge of expected traffic demand $B(s, d)$, which represents the total bandwidth demand expected between the source-destination pair (s, d). The expected load ϕ_l on link l is defined as the sum of fractional expected demands on the link from all possible source-destination pairs in the network, i.e, $\phi_l = \sum_{(s,d)} \phi_l(s, d)B(s, d)$. Link criticality $\phi_l(s, d)$ is largely static since it is completely determined by network topology and changes only when the topology changes. Also, $B(s, d)$ changes relatively infrequently, such as on a daily basis. Thus the values of ϕ_l can be periodically pre-computed offline and kept ready for use in the online route selection phase.

Let C_l be the total capacity and R_l be the residual capacity of the link l at any instant. The cost metric $cost(l)$ of link l is defined as $cost(l) = \frac{\phi_l}{R_l}$. The metric

$cost(l)$ represents the expected load per unit of available capacity on the link. A link l with more residual capacity R_l is less critical whereas one with more expected load ϕ_l is more critical. Note that the most dynamic component of the link cost is the residual link capacity R_l. The minimum value of link cost $cost(l)$ is ϕ_l/C_l when the residual capacity is maximum at $R_l = C_l$.

The metric $cost(G)$ for the entire network G is defined as follows.

$$cost(G) = \sum_{l \in G} \left(cost(l) - \frac{\phi_l}{C_l} \right)^2 \tag{4.2}$$

The metric $cost(G)$ represents the squared magnitude of the distance vector between the current link costs and the minimum link costs in the network G. Ideally, we would like the state of the network to be as close to the idle-state operating point $(\phi_1/C_1, \ldots, \phi_m/C_m)$. The squared sum has the advantage that it captures the impact of both the magnitude of individual link costs and the variations among them.

The LCBR algorithm for selecting both primary and backup routes is presented in Figure 4.5. As input to the algorithm, we supply a list of pre-computed route pairs (X, Y) of potential primary and backup routes between source s and destination d. These candidate route pairs are pre-computed by first finding the k shortest primary routes between s and d. Next, for each primary path X, we find the k shortest backup routes from the residual graph that excludes links and nodes along X. Efficient algorithms for finding the k shortest routes have been proposed in [Fox75,Epp98]. The final route pair is selected from among the k^2 shortest route pairs that are pre-computed, since these are most likely candidates to yield low values of metric $cost(G)$. The parameter k can be tuned to increase or decrease the accuracy of the algorithm in minimizing $cost(G)$.

```
Input: Network topology $G$.
    New route request between nodes $s$ and $d$
    Average bandwidth requirement $B(s, d)$
    For all links $l$: $\phi_l$, $R_l$ and $C_l$
    List $\mathcal{L}$ of candidate primary-backup route pairs $(X, Y)$
Output : Primary route $X(s, d)$ and backup route $Y(s, d)$

$cost_{min} = \infty$; $X(s, d) = Y(s, d) = nil$
For each route pair $(X, Y)$ in the list $\mathcal{L}$.
    If $B(s, d)$ cannot be satisfied along $X$ or $Y$
        then skip to next route.
    Recompute the residual capacities $R'_l$ for each link $l \in X \cup Y$.
    Recompute the $cost(l) = \frac{\phi_l}{R'_l}$ for each link $l \in X \cup Y$.
    Recompute the $cost(G) = \sum_{l \in G} \left( cost(l) - \frac{\phi_l}{C_l} \right)^2$.
    If $cost(G) < cost_{min}$ then
        $cost_{min} = cost(G)$
        $X(s, d) = X$ and $Y(s, d) = Y$.

If $(cost_{min} > cost\_threshold)$ then
    Reject the route request
else
    Select route-pair $(X(s, d), Y(s, d))$ as
    primary-backup route-pair between $s$ and $d$
```

Figure 4.5: *Link Criticality Based Route Selection Algorithm* to select both the primary and backup routes between source s and destination d with bandwidth of $B(s, d)$.

For each candidate primary-backup pair (X, Y) between s and d, the LCBR algorithm checks if the bandwidth requirement $B(s, d)$ can be satisfied by the available residual capacities along both the routes X and Y. If there are sufficient resources, then the LCBR algorithm recomputes the projected residual capacities R'_l at each link l along routes X and Y. The projected R'_l values are then used to compute the projected cost $cost(G)$. Request for a route between s and d is rejected if either (a) no route has sufficient resources to satisfy the bandwidth demand between s and d or (b) the minimum value of $cost(G)$ is greater than a pre-defined threshold. The latter case indicates that selecting a route between s and d would take the network to a highly critical state that may not be conducive for admitting future paths.

The LCBR algorithm is executed for every potential source-destination pair to determine their primary and backup routes. Nodes are admitted in increasing order of min-hop distance. This ensures that traffic between nodes that are topologically close to each other can indeed traverse over short primary and backup routes. Having more short paths as primary and backup routes in turn helps later in the path aggregation phase, by reducing the number of VLANs required to support all the routes.

4.3.2 Path Aggregation and Spanning Tree Construction

Once the primary and backup switching paths are determined, these paths need to be grouped together to form spanning trees. Each distinct spanning tree corresponds to a distinct VLAN. The maximum number of spanning trees, and hence the number of VLANs that can be supported in a network, is dependent on the capabilities of network switches. Since the number of VLANs is a scarce resource, it is essential to group paths together so that the number of spanning trees that need to be constructed is minimized.

Aggregating different paths into a minimal number of spanning trees can be reduced to an instance of the classical minimum set cover problem. Formally, the set cover problem is as follows:

Input:

The universal set U such that $U = \{1, \ldots, n\}$

A set of subsets $S_1, \ldots, S_m\ of\ U$ i.e. $S = \{\ S_i \subseteq U \mid i = 1, \ldots, m\}$

Goal:

Find the smallest subset of subsets $T \subset S$ such that $\cup_{\{t_i \in T\}} t_i = U$

Each instance of the path aggregation problem could have been solved by first constructing all possible spanning trees (set S) using the load balanced paths (set U) and then finding the smallest collection of these spanning trees (set T) by solving the minimum set cover problem. The minimum set cover problem, however, is known to be a computationally NP hard problem and the greedy heuristic is commonly the approach used to solve it [Sla96]. Further, computation of all the possible spanning trees is akin to the computation of the power-set of set of all paths, with the additional restriction of eliminating subsets which incorporate loops. This computation is exponential. Thus finding a minimal number of spanning trees for a given set of paths is a computationally hard problem.

Viking uses a simple heuristic-based greedy algorithm to achieve this goal without expending too much computation. Instead of constructing all the possible spanning trees and then greedily selecting trees in the order of the number of paths covered by trees, Viking carries out a greedy construction of spanning trees such that a large number of paths are grouped together to form spanning trees. Since the

```
Input:
    Network topology $G$.
    Load balanced paths $P$.
Output:
    Spanning Trees encompassing all paths.


Let the set of all load balanced paths be $P$.
Let the set of all edge pairs be $EP$.
Let the set of spanning trees be $S$.
Let $S$ = $\phi$.
Sort the members of $P$ in the descending order of path length.
While ($EP\ != \phi$ and $P\ != \phi$ )
    Sort the members of $EP$ in descending order of their
    frequency of appearance in members of $P$.
    Let $ep$ = Next element in $EP$.
    While $\exists\ p \in P$ such that $ep \subset p$
        Remove $p$ from $P$.
        Find $s \in S$ such that $p$ and $s$ do not form a loop.
            Merge $p$ with $s$.
        If no such $s$ is found, add $p$ to $S$.
```

Figure 4.6: *Path Aggregation algorithm. For the given input of selected paths P and the network topology G, the algorithm computes a set of spanning trees. Each spanning tree can be associated with a unique VLAN which can be used to select the desired switching paths.*

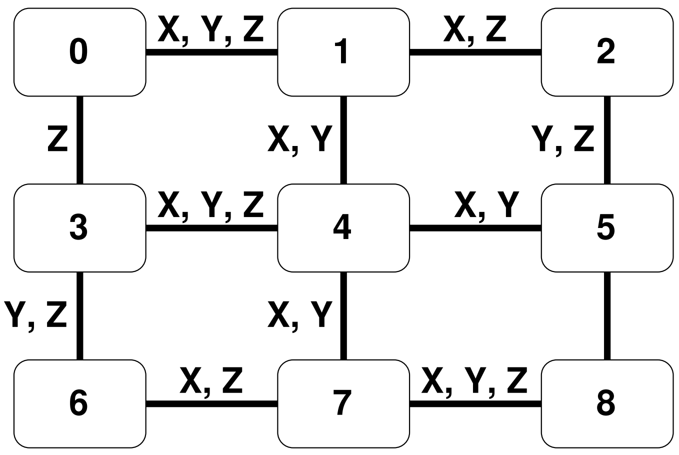


Figure 4.7: *Constructing desired spanning trees on an interconnect of Ethernet switches. By selectively enabling and disabling different VLANs on different links, the desired spanning trees can be forced on the interconnect. This figure shows the VLAN configuration on different links required to construct the spanning trees shown in Figure 4.3.*

number of overall spanning trees required is inversely proportional to the number of paths grouped together, the number of spanning trees can be reduced by grouping a larger number of paths together. To increase the number of paths per spanning tree, Viking tries to merge paths which share *common features*. Common feature based merging is accomplished by the order in which paths are selected for merger. The common features can be sub-paths, edges, or simply nodes. For computational ease, Viking limits the length of sub paths to a pair of edges. The complete algorithm for path aggregation is described in Figure 4.6. After the termination of this algorithm, the output obtained is a collection of spanning trees. These spanning trees can then be used to create independent VLANs. Specific paths can then be selected by transmitting frames over the associated VLANs.

In conventional bridged networks, the spanning tree construction process is carried out by the participating switches in a distributed manner. The final spanning tree structure depends on the bridge protocol data unit (BPDU) message latencies and the order in which the BPDUs get exchanged. Most modern Ethernet switches provide configurability options, so one can specify the relative switching priority for packets of different VLANs over different ports. Using this VLAN priority, it is possible to enable or disable association of particular VLANs to different ports. This enabling and disabling of VLANs can be extended to different links, by enabling or disabling VLAN switching on the corresponding switch ports. Thus, one can enable or disable packet switching corresponding to different VLANs on different links. Using these configurability features, one can force a spanning tree on an interconnection of switches. If the set of links that have a particular VLAN enabled on them already interconnect together in a spanning tree fashion, these links constitute the default spanning tree for that VLAN since this is the only active topology for the VLAN.

Configuration can be carried out remotely through SNMP interfaces. Viking uses the topology and spanning tree information obtained from the path aggregation phase to force the desired spanning trees on the interconnect of switches. Figure 4.7 shows how the spanning trees corresponding to Figure 4.3 can be constructed by selectively enabling different VLANs on different links.

4.4 Fault Tolerance

Most managed (configurable) switches provide status monitoring facilities where one can remotely setup traps, which can get triggered by various events, such as link failures, neighboring switch failures, link recovery, switch recovery, neighbor discovery, etc. Using these traps one can detect topology changes and failures in the network. If for every communicating host-pair, a backup path is provisioned, it is easy to deal with a failure scenario by simply switching communication to the VLAN that corresponds to the backup path. In order to deal with failure of

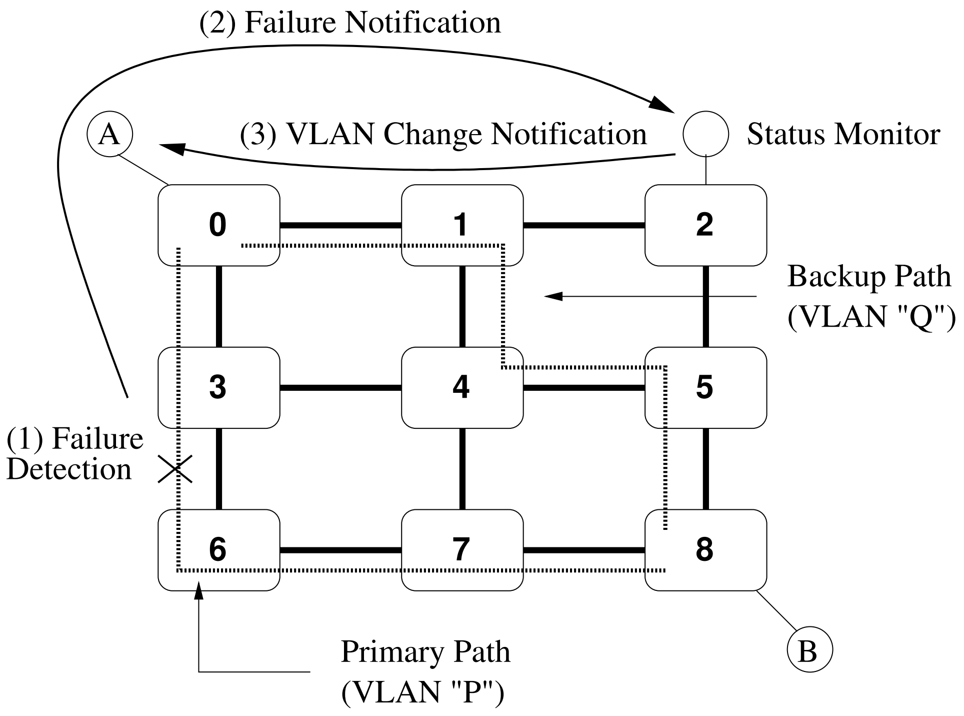


Figure 4.8: *Event monitoring based failure recovery mechanism. The Status Monitor registers with all switches (0-8) for event notifications. Node A and Node B communicate with each other using path 0-3-6-7-8 over VLAN P. The failure of link 3-6 is detected by switch 3 and the status monitor is notified of it. The status monitor determines the list of all affected sender nodes and sends them notifications to start using alternate VLANs. In this case, Node A receives a notification to use alternate VLAN Q, which provides the path 0-1-4-5-8. The entire recovery period consists of failure detection, failure notification to the status monitor, status monitor lookup of alternate VLANs, and notification to the affected nodes.*

any switch or link in the communication path, the backup path must be a link-and-switch disjoint path[1] to the primary path. Due to this disjoint nature, the primary and backup paths belong to different spanning trees (and hence different VLANs). Failures in one spanning tree do not affect other spanning trees that do not include the failed links. Since in Viking the primary and backup paths are pre-provisioned through traffic engineering, the fail-over duration is limited to failure detection delay and failure communication latencies between switches and status monitoring nodes. The failure detection latencies for commercially available low-end managed switches, such as the Cisco Catalyst 2924, range from 400 to 500 milliseconds. The fail-over period observed in this case does not exceed 500 to 600 milliseconds and is consistently in the sub-second range. This is a significant improvement over the multiple-second fail-over latency of 802.1w RSTP deployments. Note that the 802.1w failure recovery period does not take into account the failure detection period and is just the convergence period for spanning tree recovery. Further, the rapid recovery of 802.1w depends upon fast failure detection. The fault tolerance mechanism in Viking can also greatly benefit from any such fast failure detection. Thus, on a comparative note, failure recovery of Viking is dependent solely upon message latencies.

One caveat about the above mentioned detection and recovery mechanism is that it requires the presence of status monitoring node(s) in the network which is(are) aware of the entire network topology and the traffic provisioning therein. Further, it is essential that the status monitoring node is always reachable from all the switches in the network and from all nodes which need to be notified about the failure and the new VLAN tags that need to be used. If a network failure disrupts the communication path between the failure-detecting switch and the monitoring node itself, the fail-over mechanism cannot proceed. This situation, however, can be overcome by

[1]The disjoint nature of backup paths does not extend to the ingress and egress links and switches.

dispatching the failure notification over multiple communication paths to the status monitoring node. Another possibility is to have multiple status monitoring nodes strategically located in the network, such that at least one status monitoring node can communicate with the failure-detecting switch and all the affected nodes. An example failure recovery scenario is depicted in Figure 4.8.

4.5 System Implementation

Though Viking relies on configurable features of Ethernet switches to realize load-balanced VLAN-based switching, there are other tasks that the Viking system needs to perform in a distributed manner and yet other tasks in a centralized manner, outside the switch logic. For example, the final VLAN selection logic has to reside on end-hosts, as it needs to be integrated with the network stack on the end-hosts. Also, load measurement and monitoring can be done at the end-hosts which are the originating and terminating points for traffic. Other tasks, such as traffic management, etc., can be done efficiently in a centralized manner. Further, network status needs to be monitored continuously for failures. Failure recovery requires of status monitors at strategic locations in the network, so that failure notifications are delivered without getting affected by the very failures being notified.

For this purpose, Viking implementation is based on the client-server model. In Viking terminology, the clients are *Viking Proxy Clients* (VPC), the server(s) *Viking Manager(s)* (VM), and the status monitors *Viking Status Monitors* (VSM). The overall Viking architecture is shown in Figure 4.9.

4.5.1 Viking Proxy Clients

Each of the end-hosts needs to run a *Viking Proxy Client* (VPC) module which is responsible for load measurement and VLAN selection during network operation.

In the metro Ethernet service environment, however, it is not possible to modify the network stack of every end-host. The Viking client software is a thin layer in the protocol stack responsible for run-time VLAN selection and traffic characterization. This activity can be performed non-intrusively by proxy clients present at the service provider point of presence at the customer location.

Run-time VLAN Selection: In order to carry out run-time VLAN selection, the VPC needs to alter Ethernet frames before these are sent to the service provider network. VLAN selection logic is implemented as a *Viking Virtual Interface* (VVIF). Each VVIF is associated with a physical network interface connected to the network. The network stack on proxy clients uses VVIF for sending and receiving packets on the network. VVIF, on reception of the first frame for any specific destination, sends an *association query* to the Viking Manager and obtains the VLAN id for the path which was preselected in the path selection process. This VLAN identifier is cached and inserted in every subsequent frame for this destination. The cached entries are periodically invalidated, to take into account any network reconfiguration carried out by the Viking Manager. In the event of link or switch failures, the VM pro-actively informs the VPCs that are the users of affected VLANs to change their VLAN association to select the backup switching paths. In response, the VVIF refreshes the cached VLAN ids with the ids corresponding to the backup paths.

Load Measurement: The VVIF on every VPC is the sole pass-through point for the entire network traffic. This makes it an ideal choice for measuring and monitoring the traffic load. The VPC keeps track of all peer nodes and the amount of traffic exchanged with these peers through the VVIF. VPC then periodically sends updates to the Viking Manager, which uses them for future load balancing.

4.5.2 The Viking Manager and Viking Status Monitor

The Viking Manager is the central component responsible for traffic engineering and fault tolerance. Further, the VM also needs to inform the end-hosts about the VLAN information when required. Since all end-hosts update the VM with their respective traffic information, the VM has a global view of the network resource utilization. The resource utilization information, in conjunction with network topology can be used to identify the critical portions of the network and carry out appropriate network tuning.

Traffic Engineering: All VPCs periodically send traffic monitoring updates to a VM. In the long term, the VM has global knowledge about pair-wise load statistics in the network. This load characterization and the network topology information are used to select the load balanced primary and backup paths. These paths are further aggregated into different spanning trees and are associated with VLANs. The VM then reconfigures switches in the network to use the constructed spanning trees. The VM continuously monitors the load characteristics between all node pairs. Should there be a considerable change in load characteristics resulting in overload of links, the VM reconfigures the spanning trees to adapt to the changed traffic pattern. Once the spanning trees are configured, the per-node pair paths, the spanning tree information, and the VLAN information are stored in hash tables for fast lookup. The VPCs send query messages to the VM whenever they encounter packets meant for destination for which the VLAN association is not known. The VM responds to these queries after looking up the response in the hash tables.

Fault-Tolerance: For failure detection, Viking relies on switch support for SNMP traps. Each of the switches in the network is configured to send SNMP traps to the Viking Status Monitor whenever any event of interest takes place. Typically, events of interest are link failures, port failures, carrier loss, etc. Switches notify VSMs of failure using SNMP. The VSMs use this notification to find the VLANs, and hence the paths, in which the affected links are active. The source VPCs of the affected paths are then pro-actively informed to use the backup paths instead of failed primary paths. Further, the VMs initiate the reconfiguration process for the changed topology, to tackle subsequent failures. Since VMs are the central nodes responsible for overall Viking operation, the reliability of VM is crucial. To avoid making VM a single point of failure, the VM itself may be a fault-tolerant server, where a secondary VM can work as a hot-standby server ready to take over the role of primary in the event of failures.

Ethernet switches use a reverse path learning mechanism to populate their forwarding tables. In the absence of forwarding information, the default action is to send the packets over all interfaces. In the event of failover, the sending nodes are forced to use backup paths from different VLANs. In most cases, it is quite probable that the forwarding tables corresponding to the backup VLAN are not populated or may not have entries corresponding to the receivers. Thus, after failover there may be a surge of traffic on all the links constituting the spanning tree of the backup VLAN. This may result in lowering the throughput of the network and, in the worst case, may lead to congestion. Further, if the direction of traffic is unidirectional or if the reverse communication takes place over a different path covered by a different VLAN, reverse path learning may never take place successfully. This problem can be addressed by expediting the reverse path learning gratuitously. The VMs inform the VPC corresponding to the receiver nodes to send gratuitous ARP replies on the backup VLAN, so that the forwarding tables are quickly populated and traffic surges are avoided.

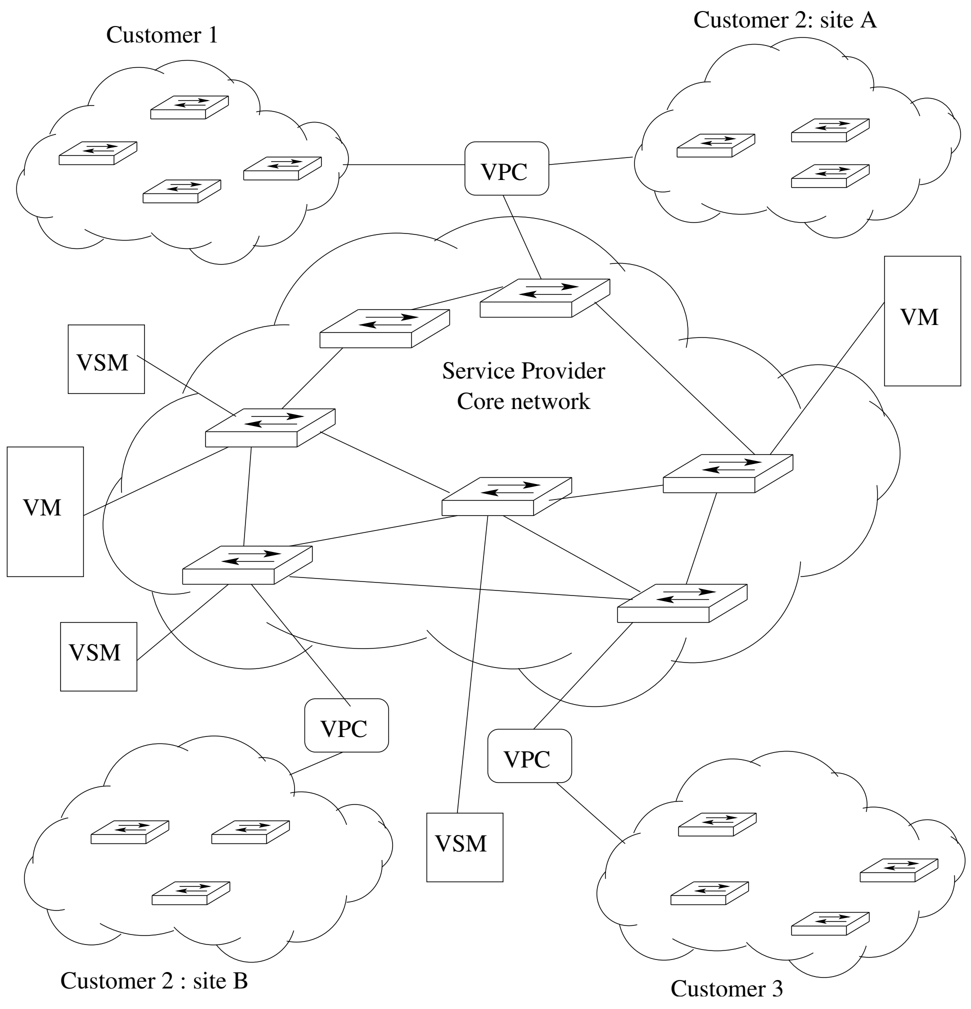


Figure 4.9: *The overall Viking system architecture. The customer sites are connected with the service provider core network through Viking Proxy Clients (VPC). The VPCs insert appropriate VLAN tags and maintain load statistics. The Viking Manager(s) (VM) interact with VPCs and carry out traffic engineering in the core network, based on load statistics. The VMs are also responsible for configuring the VLAN spanning trees in the core network. The Viking Status Monitors (VSMs) monitor the network for failures and reconfigurations. VSMs send change VLAN notification to the appropriate VPCs to switch to backup paths.*

4.6 QoS Enforcement

Traffic engineering and bandwidth provisioning alone cannot provide quality of service in a network. It requires network usage *policing*, *regulation*, and *adaptation* to provide a desired degree of QoS in the network. Without usage regulation, the best one can do is to provide QoS on the lines of the priority based DiffServ in the Internet [BBC$^+$98, FKSS98]. This kind of mechanism is inadequate to insulate different traffic flows from one another and is not an acceptable scenario in MAN, where adherence to service level agreements (SLA) is a crucial requirement. Ethernet also specifies a DiffServ-like traffic-class prioritization mechanism which is supported by almost all Ethernet switches [IEE98a]. But it is clearly inadequate, because of lack of global enforcement.

One way to regulate network usage and insulate different traffic flows from one another is to monitor and enforce usage right at the ingress points in the network. If the amount of inflow traffic does not exceed the total engineered traffic, different traffic flows cannot affect each other, unless some of the network elements have failed. If traffic engineering is performed with proper backup provisioning, it can be ensured that all traffic flows are insulated from each other even after failures.

Many mid-end and high-end Gigabit Ethernet switches support configurable rate-limiting features. For efficient wire speed, these features are usually implemented in hardware. Rate-limiting can be used to provide restricted bandwidth usage, based on a predefined profile or on per-physical port usage. Excess traffic can either be dropped or re-prioritized. Though typically supported configurable parameters for rate-limiting are quite extensive, rate-limiting can simply be specified in terms of raw bandwidth limitations and burst size limits. Using these rate-limiting features, it is possible to regulate the inflowing traffic at the ingress ports. The amount of allowed inflow traffic can be determined from the SLA, in the metro network scenario, or from the traffic characterization in the cluster interconnect scenario. Further, 802.1p traffic classification and prioritization can be used to mark as lower priority the traffic that is sent in excess of the allowed traffic, so that this traffic is serviced only if there is spare bandwidth available on every link and every switch along the path.

It is worth exploring whether all switches in a network need to support this feature. Rate-limiting is essential only when an end-host is connected to a switch. In MAN a setup, the switches can be classified as core switches and edge switches, where the edge switches serve as ingress points. The core switches need not participate in the rate limiting activity, as the edge switches ensure that the traffic reaching core switches is already rate-limited. This mechanism is analogous to ingress filtering in the Internet. An alternate viewpoint is that usually there is spare bandwidth available at the edges but the core of the network carries most traffic, so it should be the core where bandwidth regulation takes place rather than at the edges. This argument has some merit to it. The final decision, however, can be made only after traffic engineering. But unfortunately traffic engineering requires the knowledge of the topology which can be acquired only after deployment. Thus, rate-limiting at the core or at the edge can performed after careful analysis of the traffic load distribution in network.

In cluster and storage networks, potentially all switches may support end-hosts. This makes it essential to support rate-limiting at all switches. Also, the traffic profile may change from time-to-time. Strict rate-limiting on a changing traffic profile fails to capture the changing requirement and hence is inappropriate. The proper way to tackle this is to adapt soft rate-filtering with re-prioritization of excess traffic, so that no traffic is dropped as long as there is network capacity available to service it. In addition, if the current traffic profile differs significantly from the profile used for bandwidth provisioning, traffic re-engineering needs to be carried out.

The approach of regulating traffic at ingress ports works well if the total provisioned bandwidth is the same as the bandwidth stipulated in the service level agreement. This kind of bandwidth provisioning, however, leads to under-utilization of network resources. Service providers cannot take advantage of possible statistical multiplexing. Viking is designed to take the load characteristics into account during the traffic engineering phase. If the load characteristics vary significantly from the peak demand stipulated in the SLA this may lead to congestion in the network. In particular, the following three scenarios are possible in the network.

- If the instantaneous traffic profile of a client does not deviate from the traffic profile used for traffic engineering, there is no congestion in the network and the SLA terms are not violated.

- If the instantaneous traffic profile of a client significantly differs from the traffic profile used for traffic engineering, congestion in the network may result. If the traffic load does not exceed the load stipulated in the SLA, it should not be regulated by using the rate limiting features of Ethernet switches. This would lead to SLA violation.

- If the instantaneous traffic profile exceeds the SLA stipulated traffic, it can be regulated by using the rate-limiting features of Ethernet switches. It is, however, imperative to avoid any congestion that may result because of under provisioning due to statistical multiplexing.

Thus, the rate-limiting features of Ethernet switches are effective in insulating different flows from different clients, when the clients misbehave or exceed the traffic limits allowed by SLA. Rate-limiting, however, is not an effective solution when SLA terms are not violated by clients and when service providers resort to statistical multiplexing for under-provisioning, in order to increase network utilization.

Rate-limiting in Ethernet is inadequate to provide a complete QoS solution because one can only police and regulate network usage using rate-limiting. The requirement of *adaptation* to a changing traffic profile cannot be satisfied with a rate-limiting mechanism. In order to adapt to changing traffic profiles, one needs to consistently monitor the traffic between different clients and carry out re-routing, if required. In addition, individual links in a network also need to be monitored to detect any signs of congestion. Changes in traffic profile are easy to detect. Since the VPCs are sole pass-through points for all the traffic in the network and are responsible for load characterization, it is trivial to detect changes in traffic profile at the VPCs directly. If the traffic profile change is beyond a certain threshold, the VMs can be notified about the changes and a traffic re-engineering can be carried out. to select new congestion-free paths for all affected clients. Further, all managed switches support remote monitoring of switch ports for relevant statistics. Port statistics include many useful parameters, such as total frames sent and received, frames dropped, collisions encountered, amount of data transferred etc. Periodic monitoring of these statistics can infer the status of the connected links. Impending network congestion can be detected by increased use of different links and an appropriate action can be taken to alleviate the traffic load in the congested region.

Superficially it may appear that monitoring the traffic profile at VPCs and triggering traffic re-engineering after a change in characteristics alone should be sufficient to address the problem of congestion. Re-engineering traffic, however, is computationally and administratively time consuming. It is not an effective solution to address minor traffic profile variations. Traffic re-engineering should be triggered only at coarse granularity of traffic profile variation. Thus, it is not just appropriate, but essential, to react to traffic profile changes only after they exceed a certain predetermined threshold. It may so happen that the traffic profile of several node pairs may vary below the threshold without triggering the re-engineering, but the variations may result in increased load at certain bottleneck links, leading to congestion. To address this problem it is essential to monitor both the traffic profile and the individual link status periodically.

Thus an effective QoS solution requires the following steps to be taken by any Viking-like system:

- The VM should configure the rate-limiting parameters at the ingress switches according to the available traffic profile. The rate-limiting should be soft, such that all excess traffic should be serviced as long as there is capacity in the network. The excess traffic should be marked as low-priority traffic, so that eventually when faced with congestion this excess traffic is dropped. This approach ensures insulation and isolation among different clients.

- VPCs should monitor traffic characteristics continuously and trigger an alarm to the VM if the traffic profile change exceeds a predetermined threshold.

- VSMs should monitor individual links using the port statistic feature of Ethernet switches. If the traffic on any link exceeds the total engineered traffic, some of the traffic should be diverted to alternate links and the corresponding VPCs should be informed pro-actively to switch to different VLANs.

- After receiving alarms from VPCs or VSMs, the VM should either tune the paths, by rerouting traffic so that the impending congestion is avoided, or should re-engineer the traffic, if required. After re-engineering, the ingress switches should be reconfigured according to the new traffic profile.

These steps ensure that the network is constantly monitored for usage and the usage is regulated. Further, path selection in the network can be adapted to traffic changes, so that a dynamic rerouting similar to IP networks is performed to utilize the spare capacity in the network, alleviating the congestion scenarios.

4.7 Evaluation of Viking

Performance evaluation of Viking was carried out through extensive empirical measurements of a prototype. To verify the usability of Viking in large-scale networks, performance evaluation was done through simulations.

4.7.1 Simulations

The Viking system was evaluated for its performance at path selection and path aggregation. Simulations were carried out to determine the maximum bandwidth that could be supported in the network. The network topology was assumed to be a grid topology, which can represent metro Ethernet and cluster networks. The simulations were carried out with both a uniform traffic pattern, with each node communicating with other nodes with an equal traffic load, and a skewed traffic load with client-server and peer-to-peer communication. The uniform traffic distribution is representative of cluster networks and the skewed traffic distribution represents a typical scenario in metro Ethernet. The simulations were run with grids of sizes 16, 25, 36, 49, and 64 nodes connected using links with a capacity of 100 Mbps. In the uniform load scenario, the traffic between nodes is 10, 8, 5, 2, and 1 Mbps for grids of size 16, 25, 36, 49, and 64 respectively. In the skewed distribution, each network is assumed to have around 10% of the nodes as servers. All other nodes are assumed to be clients communicating with all servers. The client-server bandwidth in this case is 30, 20, 15, 8, and 5 Mbps. The peer-to-peer communication bandwidth in this case is similar to the uniform load scenario. Each node has around 10% of the other nodes as peers. The effectiveness of path selection was compared for possible

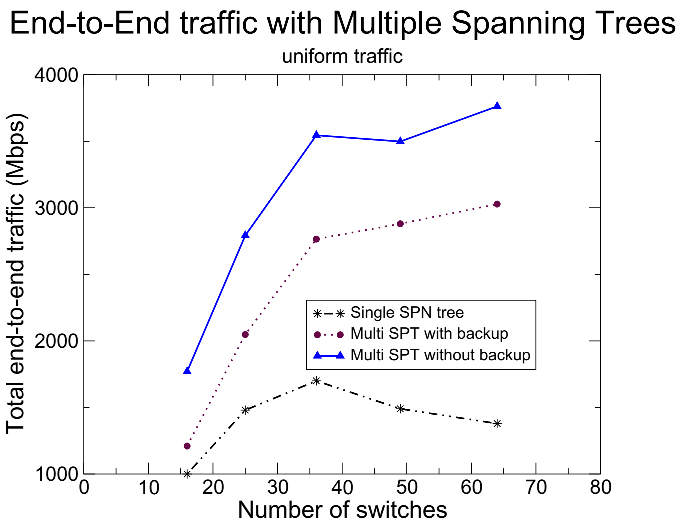


Figure 4.10: *Total end-to-end traffic in a network with a single spanning tree and traffic in multiple spanning trees with and without backup provisioning. The network follows a uniform traffic pattern while communicating between switches.*

throughput against the single spanning tree case. Path selection was carried out for both cases, with and without backup redundancy. All traffic contained at least one hop between the switches.

Figure 4.10 shows the comparative maximum throughput for a single spanning tree network versus the Viking path selection. The traffic pattern is assumed to have a uniform distribution across all node pairs. This is a representative distribution for cluster networks. It can be seen that the total aggregate end-to-end throughput is always better in Viking. As the number of nodes increases, the performance of the Viking system shows considerable advantage. This is because of availability of more active links in the topology. Thus, Viking has scalable performance, a desirable feature for growing networks.

Figure 4.11 shows similar scenario with a skewed traffic pattern. For a network of 49 nodes, there is a performance dip compared to the 36 node network. The reason for this is the positioning of servers in the network. Figure 4.12 shows the

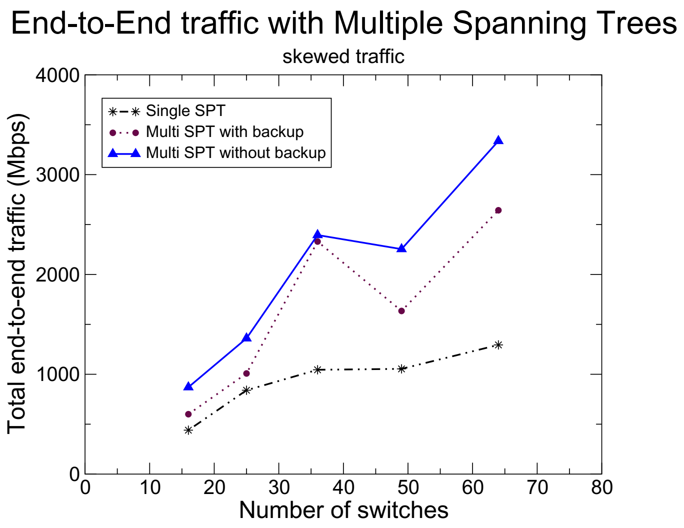


Figure 4.11: *Total end-to-end traffic in a network with a single spanning tree and traffic in multiple spanning trees with and without backup provisioning. The network follows a skewed traffic pattern while communicating between switches. The case with 49 switches shows a dip in performance because of saturation of links near servers.*

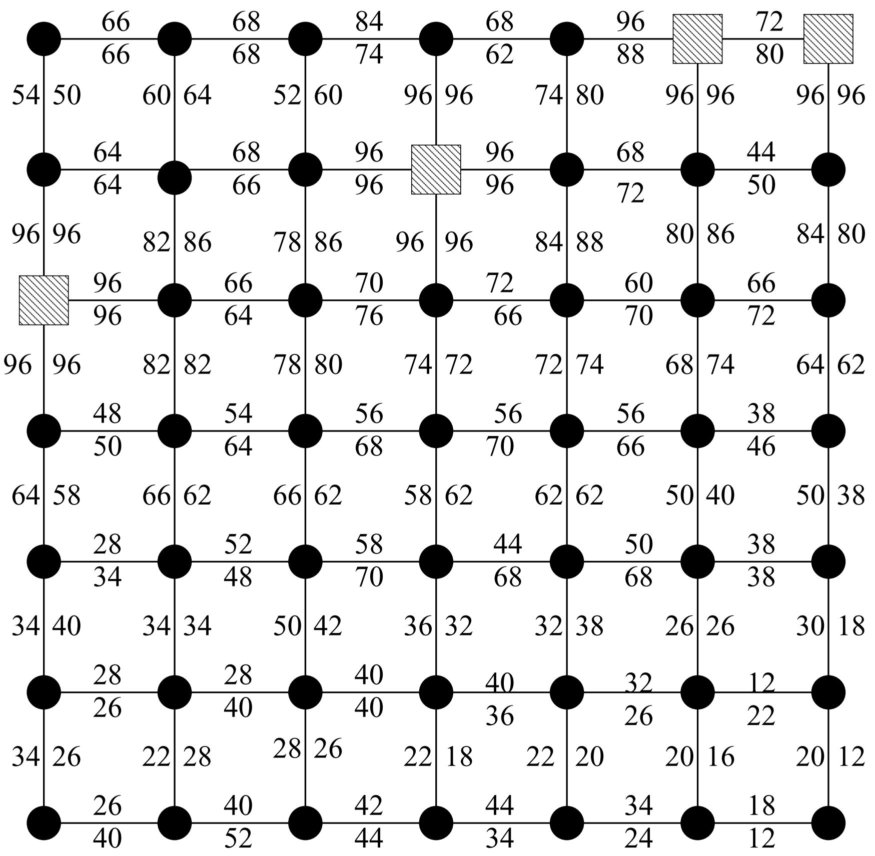


Figure 4.12: *Load distribution in a 7x7 grid network. The edges represent duplex links of 100 Mbps capacity. The links are marked with the total provisioned bandwidth. Note that certain links in the vicinity of servers (marked as squares) are completely saturated. This network can be tuned by adding new links in parallel to the existing links.*

traffic distribution across different links for this network of 49 nodes. Because of saturation of links in the vicinity of server nodes, the performance of the entire network is bottle-necked. This can be tackled by using additional links to connect the switches facing high load.

We performed a simulation study to understand the performance of the path

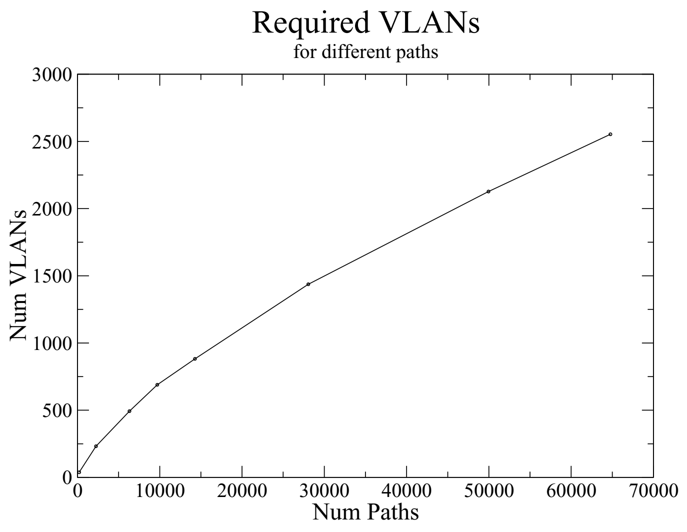


Figure 4.13: *The number of required VLANs with a different number of paths. The required number increases sub-linearly with increase in the number of paths. For a large network with around 65,000 paths, around 2,600 VLANs are enough to cover all paths.*

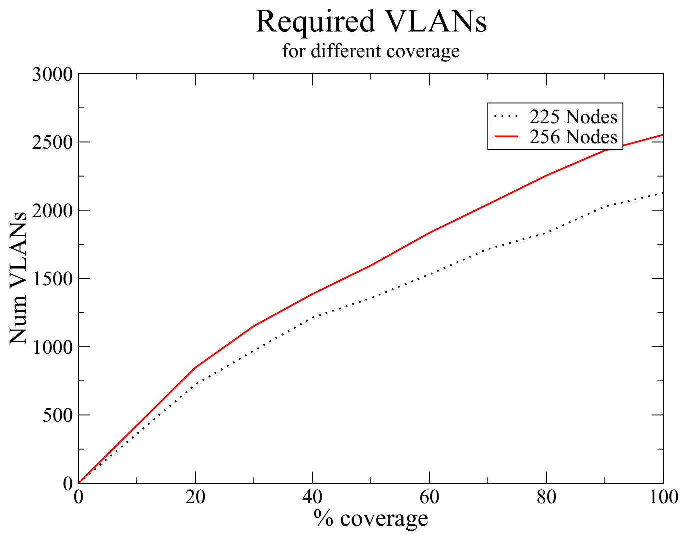


Figure 4.14: *The number of required VLANs with different proportion of network coverage. When communication is limited to only a portion of the network, the number of paths are less and hence the number of required VLANs is also less.*

aggregation algorithm and the required total number of spanning trees in large networks. Figure 4.13 shows the number VLANs required to accommodate a different number of paths. The simulations were run for different grid networks varying in size from 16 nodes to 256 nodes. The simulations were performed to consider the maximum number of required paths in every network, so that each node could communicate with every other node in the network. The maximum number of required paths in a grid network of 256 nodes is close to 65,000. The maximum number of spanning trees, and hence VLANs, required for these paths is around 2600. This requirement is well within the limit of 4096 VLANs.

In metro networks, it is highly unlikely that there will be traffic between all the switches in the network. In a more realistic scenario, traffic is always between switches, which are connected to different sites of the same customers. Thus the number of paths required is much less than the total number of distinct switch pairs in the network. Figure 4.14 shows the requirement of VLANs in large networks containing 225 and 256 nodes, with varying amount of path coverage. It can be seen that the actual number of VLANs required is much lower, because of efficient path aggregation.

4.7.2 Empirical Performance

A Viking prototype was implemented using Cisco Catalyst 2924 switches. The prototype included 3 switches connected in a triangular fashion. Each pair of switches was connected by Fast Ethernet trunks. Each switch supported 4 Pentium-4 class end-hosts with Fast Ethernet network interface cards. The prototype was evaluated for throughput performance and fault tolerance.

The Viking Virtual Interface overhead on packet latency was insignificant. Processing each packet required around an additional 15 microseconds, on average. This latency is attributed to the lookup and insertion of the VLAN ID in transmitted

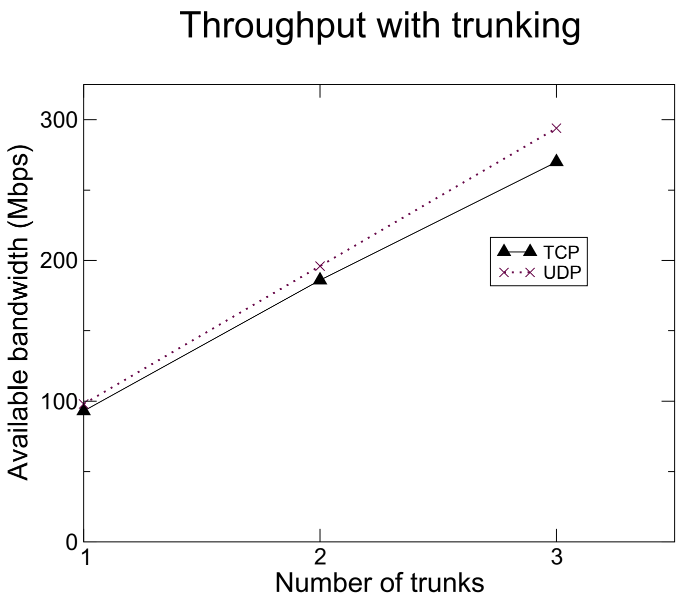


Figure 4.15: *Throughput for different numbers of active links acting as trunks. The performance is linear in the number of active links.*

packets.

An experimental verification of the advantage of additional active links was carried out. In this setup, two switches were connected using a single link, 2 active links, and 3 active links. The total throughput for UDP and TCP traffic was measured. The performance was observed to increase linearly with increasing link capacity. Figure 4.15 shows the TCP and UDP throughput for varying numbers of active links.

Figure 4.16 shows the effect of link failure on TCP throughput. The experiment was run with a setup where two switches were connected by two links. These links were then configured to belong to two different VLANs. To have a more realistic scenario, we introduced enough background traffic to keep link utilization at the maximum level in the VLAN that served the main traffic. Upon link failure, the TCP traffic passing through this link falls back onto the backup VLAN, which

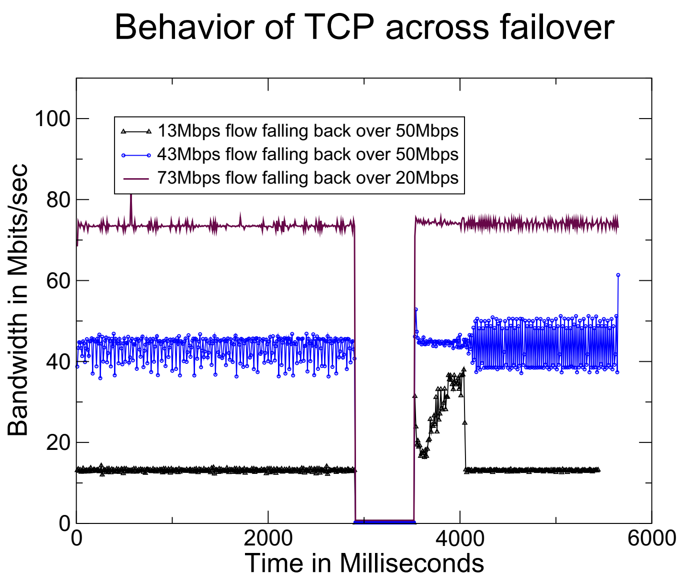


Figure 4.16: *Behavior of TCP across fail-over. After fail-over, TCP has to adapt to the existing traffic on backup links. The amount of traffic on backup links was maintained at a sufficiently high level so that the links are saturated after fail-over. This was the worst case scenario for fail-over. It was observed that TCP takes around 300 ms to 400 ms to recover from fail-over.*

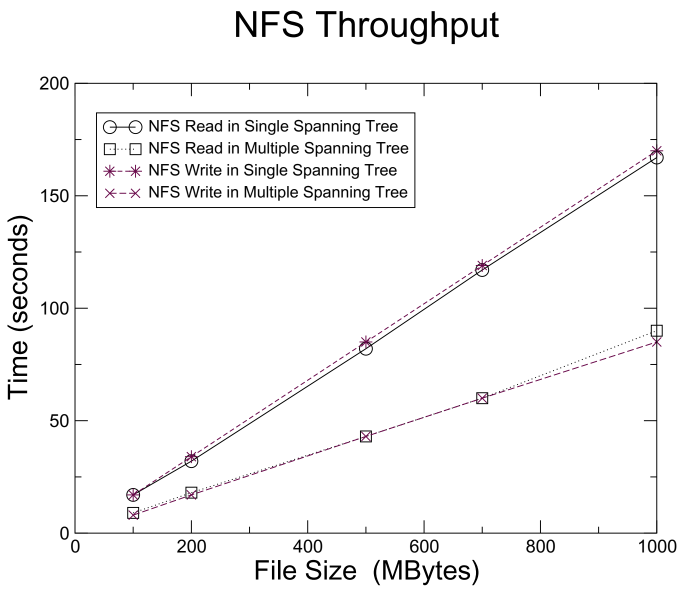


Figure 4.17: *Performance of NFS in the multiple spanning tree and single spanning tree cases. In presence of two active links, the performance is exactly double compared to the performance in single spanning tree scenario.*

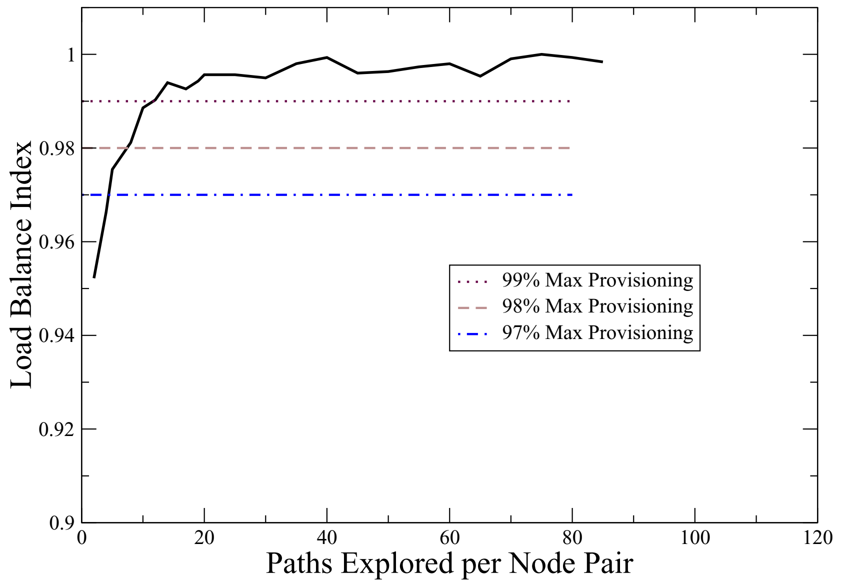
passes through the other link. There was an expected drop of bandwidth for the recovery period (around 600 milliseconds). Once the backup VLAN was in use, the TCP flow regained its momentum, hardly suffering any more delay, and attained stability, in around 300ms to 400ms.

In another set of experiments, the three switches were connected with each other using two links each. There were 6 different VLANs configured to make all the links active. The maximum end-to-end throughput in Viking was observed to be around 1.2 Gbps for single-hop communication and around 600 Mbps for two-hop communication. Whereas, in the case of a single spanning tree, it was 400 Mbps for single-hop communication and 200 Mbps for two-hop communication.

Figure 4.17 shows the performance of Viking in a typical NFS server cluster. In this scenario there are 2 NFS servers serving multiple NFS clients. The clients reside on a separate switch. In Viking's setup, these 2 switches are connected using 2 active links participating in different VLANs. The files on servers are accessed parallely by different clients. The total time for NFS read and write operations on files of different sizes is shown. The performance for the two active link scenario is observed to be double, compared to the performance of the single spanning tree case.

The fault-tolerance mechanism was tested by manually unplugging links from switches. The down-time because of failures can be broken down into 3 parts, namely, the failure detection period, the alternate VLAN lookup period, and the failure notification to end-hosts period. The failure detection period ranged from 400 milliseconds to 600 milliseconds. In comparison, the alternate VLAN lookup required only a few milliseconds and the notification time was less than a millisecond. Thus, the overall down-time was dominated by failure detection. The data loss for UDP streams at different rates was proportional to the data rate.

After failures, the Viking Manager recomputes the selected paths to brace for subsequent failure. Thus, the total time required to recompute paths and reconfigure the spanning trees is the critical period during which another failure in paths with affected nodes cannot be tolerated. Path selection is a computationally intensive process. The computations can be minimized by reducing the search space for appropriate paths. The reduced search space *may* result in a less efficient path selection. Efficiency of path selection is deduced by the number of paths that can be actually provisioned without causing a load imbalance situation in the network. Viking path selection was evaluated for total duration of computation against the search space. The evaluation was done for a grid network of 64 nodes. The traffic distribution was assumed to be uniform across all possible node pairs in the network. The search space for the primary path was varied from 1 path to 15 different paths. Each primary path search was accompanied by a search space for 5



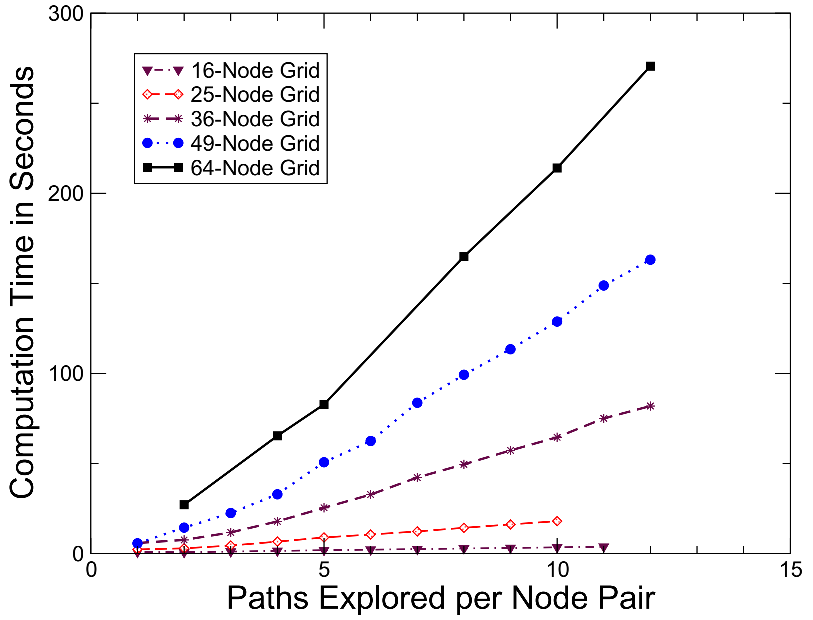


Figure 4.18: *Total time spent in recomputing path selection. With increased search space the re-computation time increases. For grid networks of sizes from 16 to 64 nodes, the re-computation is shown on the bottom. The graph on the top shows the load balance index for a limited number of searches in a 64 node grid network. Even for a limited search, the number of paths is within 97% of maximum possible load balance.*

backup paths. Figure 4.18 shows the relation of re-computation time to the size of the search space. With a larger search space, the amount of traffic that can actually be provisioned also increases while reducing the load imbalance. We define the *load balance index* as the ratio of the total traffic that can be provisioned after a limited search to the total traffic that can be provisioned after exploring a large search space, such as 5,000 alternate paths. Figure 4.18 shows the comparative load balance achieved after a limited search. It can be seen that a limited search space of a small number of paths has a load imbalance within 97% of the maximum possible load balance. Thus, limiting the search space does not impact the load balance scenario significantly. Whereas, it significantly reduces the computation time.

This path selection is followed by reconfiguration of spanning trees in the network.

Thus, the *complete* failure recovery time can be broken down into the following components: A failure detection time of around 400 to 600 milliseconds; the VLAN change notification time of a few milliseconds; and finally, the reconfiguration time which depends on the topology. The total down-time incurred is in the sub-second range, which is around 30 to 60 times less than for conventional single spanning tree architectures.

Chapter 5

Cassini: A SAN Provisioning Tool

5.1 Storage Area Networks

Storage Area Networks (SANs) are special-purpose high-speed networks that interconnect front-end servers, such as NFS servers and DBMS servers, to back-end storage devices. Front-end servers can access back-end storage devices over the SAN using the same disk access interfaces as for local disks, such as SCSI disks. SAN storage architecture eliminates the strict one-to-one association between servers and storage devices and enables sharing storage across multiple servers. Today, the preferred storage access technology in enterprise and data center environments is invariably Storage Area Network technology. Networked storage architecture has become immensely popular because it is more flexible, scalable, manageable, and robust than the conventional Direct Attached Storage (DAS) architecture.

5.1.1 Shared Storage Architecture

The appetite for storage capacity in enterprises and data centers has grown rapidly. Storage demand is fueled by increased data retention requirements as well as compliance regulations and fault tolerance needs. At the same time, the enterprise application model has shifted from centralized mainframe and mid-range systems to clustered application servers and distributed file and data servers. The traditional DAS approach of provisioning each server with dedicated persistent storage is inadequate to meet the scalability, management, and availability requirements of enterprise and data center environments.

The DAS approach fails to meet today's enterprise needs because of some inherent issues. It forces a strict one-to-one association between servers and storage devices. In order to avoid frequent storage provisioning, each server needs to be (over)provisioned with sufficient storage, without any scope for sharing. This leads to storage fragmentation. All the dedicated storage needs to be physically connected to corresponding servers. There is a limit to direct connectivity beyond which adding additional storage becomes difficult, resulting in scalability problems. A server crash or malfunction makes the associated storage unavailable unless it is physically detached and reattached to some other server reducing the overall data availability.

Conventionally, storage dedicated to a server is treated as a peripheral of the server and is managed as a storage island. There is certainly significant value in recognizing storage as a distinct resource and treating it as a *subsystem rather than a peripheral* in enterprise environments. Enterprise-wide storage consolidation offers such an opportunity and alleviates storage management problems.

The growing trend of server virtualization and physical server consolidation in data center environments exacerbates the inadequacy of DAS and provides an additional impetus to decouple dedicated storage from servers and move it to a shared

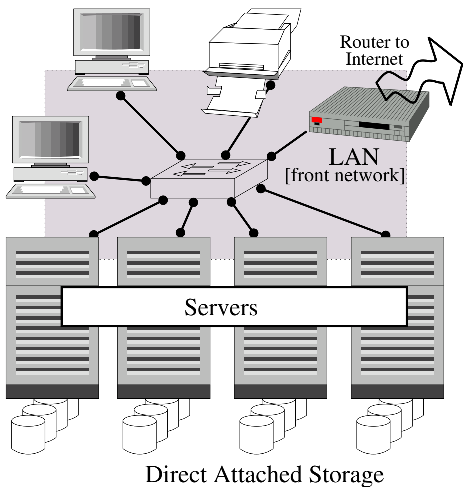


Figure 5.1: *Direct Attached Storage architecture. Each server has dedicated storage connected to it over an I/O bus and a protocol such as SCSI. Storage cannot be physically shared by multiple servers. This architecture fails to scale in enterprise environments because of fragmentation of storage capacity and management problems.*

environment for consolidation. As enterprises move to consolidate their previously separately administered data storage islands into centrally managed shared storage facilities, SAN technology will play a critical role.

Sharing of storage resources among multiple computer systems requires peer-to-peer interconnection of computers and storage devices. This interconnection can be achieved by using *Network Attached Storage (NAS)* devices or by accessing the storage through a SAN. Even though NAS and SAN technologies have significant differences, there is sufficient overlap between the two to cause confusion. In the context of the research work of this dissertation, SAN is a dedicated network infrastructure that enables shared storage and where the majority of storage traffic is block SCSI data. In contrast, NAS devices piggyback storage traffic over existing network infrastructure and the majority of storage data is accessed in terms of files and/or objects [Cla03].

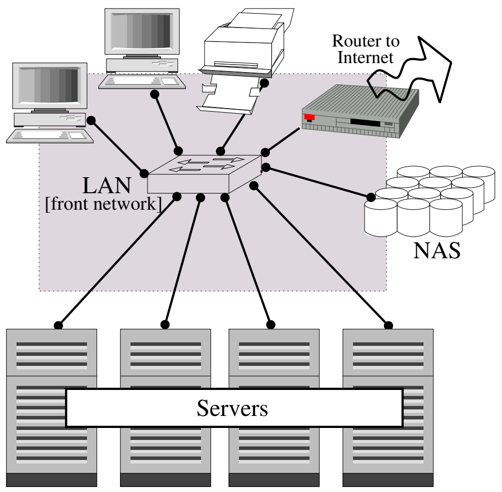


Figure 5.2: *Network Attached Storage architecture. Storage can be pooled together and shared by multiple servers that access it over the network. Storage pooling and sharing reduces fragmentation and improves manageability. The servers interact with NAS devices by using file and object semantics.*

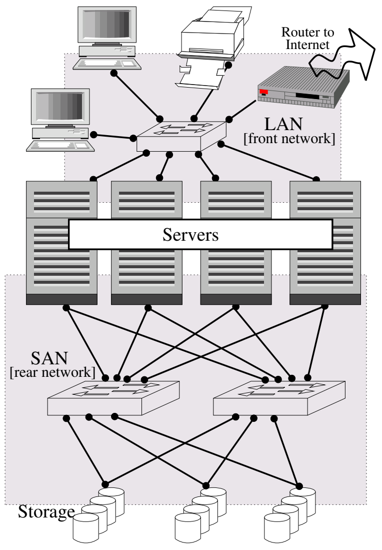


Figure 5.3: *Storage Area Network. The SAN is a dedicated network infrastructure to enable shared storage, where the majority of storage traffic is block SCSI data. The SAN complexity is hidden from the applications and the operating systems on the servers by Host Bus Adapters (HBAs) that make the shared storage appear as if it is direct attached storage.*

Figure 5.1 shows the conventional direct attached storage architecture where each server has dedicated storage connected to it over an I/O bus and a protocol such as SCSI. Figure 5.2 shows the network attached storage architecture where shared storage devices are connected to a production LAN and data movement between the storage devices and servers is in terms of files and/or objects. Figure 5.3 shows the storage area network architecture, where shared storage devices are connected to servers by means of a dedicated network.

The Storage Networking Industry Association (SNIA) formally defines a SAN as:

- A network whose primary purpose is the transfer of data between computer systems and storage elements and among storage elements. Abbreviated SAN. A SAN consists of a communication infrastructure, which provides physical connections, and a management layer, which organizes the connections, storage elements, and computer systems so that data transfer is secure and robust. The term SAN is usually (but not necessarily) identified with block I/O services rather than file access services.

- A storage system consisting of storage elements, storage devices, computer systems, and/or appliances, plus all control software, communicating over a network.

5.1.2 SAN Technologies and Protocols

SANs are often referred to as the networks *behind* servers, to distinguish them from the *front* Local Area Networks that are used to interconnect the servers with each other and with the end-user workstations [Cla03].

The SNIA definition of SAN does not refer to any specific technology or protocol. Rather, the definition focuses on the services offered by the SAN infrastructure and on the management of those services. The reason behind this refrain from referring to any specific technology is that there are different technologies available to realize different services from a SAN infrastructure. Unlike LANs, where the technology landscape is dominated by Ethernet and TCP/IP, the technology for SANs is different and crowded. Fibre Channel (FC) is the dominant SAN technology. In addition to FC, there are other SAN technologies such as Fibre Channel over Ethernet (FCoE), Internet SCSI (iSCSI), Fibre Channel over IP (FCIP), and Internet FCP (iFCP). All these technologies are competing for share in SAN space. Figure 5.4 lists some of these technologies and shows the protocol stack associated with each technology. Designing a SAN for a set of applications requires a thorough understanding of the applications and their requirements as well as the technology that is appropriate for application specific SAN deployment.

5.1.2.1 SCSI

SCSI *(Small Computer System Interconnect)* is a set of standards developed to provide an efficient way of connecting and transferring data between computers and peripheral devices. It is primarily used to connect storage devices to computers using a parallel bus. SCSI can also be used to connect printers, scanners, and other peripherals to computers. SCSI is responsible for transferring data blocks between an *initiator* device and a *target* device in a reliable fashion.

SCSI is not responsible for assembling data blocks for transfer or placement of data on storage devices. This demarcation helps in hiding the complexity of physical formats, which in turn enables a uniform way of connecting all devices to the SCSI bus.

Figure 5.5 shows the typical way of connecting SCSI devices to computers. A computer is equipped with *SCSI host adapters*. An adapter is connected to multiple *SCSI controllers* by means of the SCSI bus. These individual controllers control

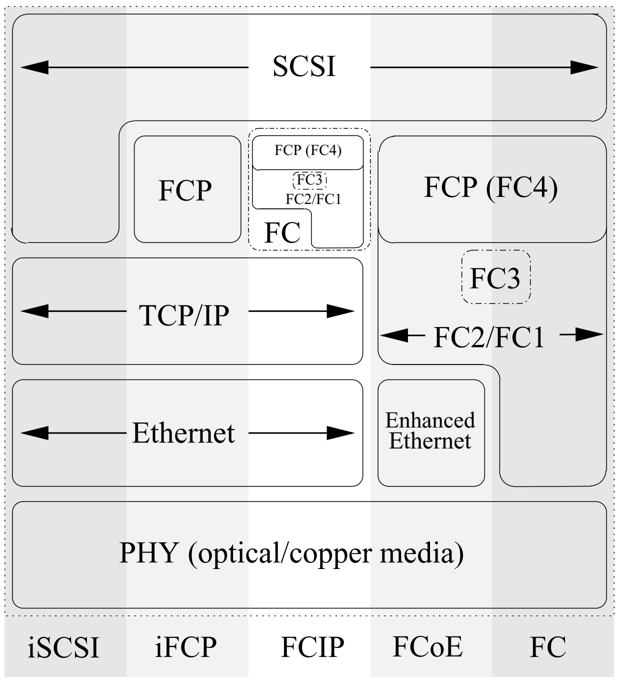


Figure 5.4: *Storage Area Network technologies and protocols. The dominant technology in SAN space is Fibre Channel (FC). Along with FC there are other technologies such as iSCSI, FCIP, iFCP, and FCoE. All these technologies have some distinction and some overlap. Despite their distinctions, one significant common feature is that all the technologies provide SCSI semantics for accessing storage devices.*

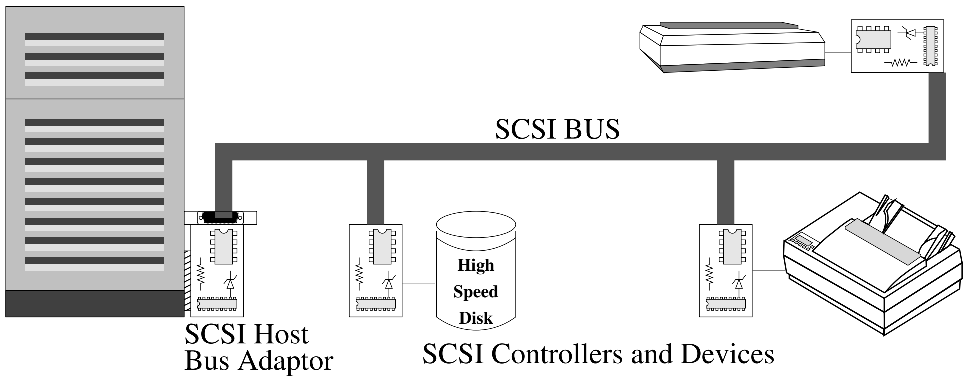


Figure 5.5: *The standard way of connecting SCSI devices to a computer. The computer is equipped with a SCSI HBA connected to multiple SCSI controllers by means of the SCSI bus. These individual controllers control the connected devices.*

the connected devices. The SCSI host adapter and device-specific SCSI controllers hide the complexity of communication over the bus from the host and the devices. Depending on the type of bus used there are different variants of SCSI. In the very beginning, there was parallel SCSI, which utilized an 8-bit parallel bus. The SCSI standards defined various aspects of communication on this parallel bus, such as voltage-signaling mechanism, connector type, frequency of operation, etc. The original parallel SCSI evolved into different variants with changing bus characteristics: Fast SCSI, Fast Wide SCSI, Ultra SCSI, and Ultra Wide SCSI just to name a few. These variants are termed SCSI interfaces.

Over time, the increasing number of SCSI interfaces gave rise to the SCSI Architecture Model to provide a common basis for the coordination of SCSI standards and to specify those aspects of SCSI I/O behavior that are independent of a particular technology and common to all implementations [Int06]. This architecture model allowed the SCSI protocol to be agnostic about the underlying physical technology. The SCSI bus could be replaced with any communication infrastructure, without impacting communication between the host and the devices. This transparency is precisely the reason that SCSI has emerged as the default higher-level storage protocol for SANs. In SAN environments the SCSI bus is replaced with the network fabric and the SCSI host adapters and device controllers take care of hiding the complexity of communication over the SAN. Figure 5.4 highlights that all SAN protocols, despite their differences, provide SCSI semantics for accessing storage devices. All the SAN protocols shown in Figure 5.4 conform to the SCSI Architecture Model and thus are part of SCSI suit of standards.

5.1.2.2 Fibre Channel

Fibre Channel (FC) is the dominant Gigabit-speed SAN technology. Despite its name, FC is not confined to communication over optical fiber. It supports communication over fibers as well as copper cabling. It currently supports data rates of 4 Gbps and distance up to 10 km. between end-points. FC is essentially a high-performance serial link supporting its own, as well as higher-level protocols, such as FDDI, ATM, HIPPI, IP, IPI, and SCSI. The SCSI interface protocol on FC is called the *Fibre Channel Protocol (FCP)*.

5.1.2.3 Fibre Channel Topologies

The communicating end-points in a Fibre Channel network are called *ports* and the network itself is termed the *fabric*. There are three primary FC topologies, depending on the way ports are connected to the fabric. These are:

- **Point-to-Point**: This is the topology in absence of any fabric. An initiator is directly connected to a target. There can be only one initiator and one target in each point-to-point topology.

- **Arbitrated Loop**: In this topology up to 126 devices can be connected to each other in the form of a physical or a logical loop. The connected devices share the entire available bandwidth. The disadvantage of loop topology is that any localized failures or changes in the loop cause the entire loop to stop working.

- **Switched Fabric**: Switched Fabric uses switches to connect hosts and devices, either directly or on arbitrated loops. Switched mode ensures an efficient use of bandwidth by enabling a fabric configured for servers and storage resources. This fabric architecture is similar to switched Ethernet networks. Fabric switches manage the state of the fabric and provide optimized interconnects.

Regardless of topology, the core protocol used in Fibre Channel SAN is the Fibre Channel Protocol (FCP), a SCSI interface that defines communication between SCSI initiators and targets (hosts and storage devices) over FC. All FC devices support FCP, which is a clear and consistent way of converting SCSI commands to serial format and transmitting them over a network.

5.1.2.4 iSCSI

iSCSI *(Internet Small Computer System Interface)* is a TCP/IP-based protocol for establishing and managing connections between IP-based storage devices, hosts, and clients. With the advent of Gigabit Ethernet, the bandwidth and latency concerns of IP networks are alleviated and IP networks meet the performance requirements of fast system interconnects and, as such, are good candidates to “carry” SCSI. The iSCSI specification describes a means of transporting SCSI packets over TCP/IP, providing for an interoperable solution that can take advantage of existing Internet infrastructure and Internet management facilities and that can address distance limitations.

5.1.2.5 FCIP

Fibre Channel over IP *(FCIP or FC/IP)* is an IP-based storage networking technology developed by the Internet Engineering Task Force (IETF). FCIP mechanisms enable the transmission of Fibre Channel (FC) information by tunneling data between two SANs over IP networks. This tunneling mechanism facilitates data sharing over a geographically distributed enterprise. One of the two main approaches to storage data transmission over IP networks, FCIP is among the key technologies expected to help bring about rapid development of the storage area network market.

5.1.2.6 iFCP

iFCP *(Internet Fibre Channel Protocol)* is a standard for extending Fibre Channel SANs across the Internet. iFCP provides a means of transferring data between Fibre Channel storage devices in a local SAN and devices on the Internet, using TCP/IP. TCP provides congestion control as well as error detection and recovery services. iFCP merges existing SCSI and Fibre Channel networks with the Internet. iFCP can either replace or be used in conjunction with existing Fibre Channel protocols, such as FCIP (Fibre Channel over IP). iFCP is a gateway-to-gateway protocol, where TCP/IP switching and routing components complement and enhance, or replace, the Fibre Channel fabric.

iFCP addresses some problems that FCIP fails to deal with. For instance, FCIP is a tunneling protocol that simply encapsulates Fibre Channel data and forwards it over a TCP/IP network, as an extension to the existing Fibre Channel network. FCIP, however, is only equipped to work within the Fibre Channel SAN environment, while the storage industry trend is increasingly towards Internet-based storage area networks. Because iFCP gateways can either replace or complement existing

Fibre Channel fabrics, iFCP can be used to facilitate migration from a Fibre Channel SAN to an IP SAN or hybrid network.

5.1.2.7 Fibre Channel over Ethernet (FCoE)

While Fibre Channel is the dominant SAN technology, Ethernet is the dominant networking technology. There is a strong market interest in a converged Ethernet-based networking protocol for SANs. Converged Ethernet-based networking offers economy of scale, IT expertise, and simplified networking.

Though iSCSI, iFCP, and FCIP offer storage networking using IP, and hence Ethernet, the proponents of Fibre Channel over Ethernet (FCoE) claim that these technologies have inherent issues that prevent their widespread acceptance.

Though iSCSI is popular in small enterprises, workgroups, and remote offices, it has not been able to gain strong foothold in large enterprise data centers. The justification cited is that large enterprises are heavily invested in FC SANs. It is not cost effective for them to replace their expensive FC SANs and management infrastructure with iSCSI and associated tools. Also, iSCSI is not trusted with mission critical applications.

FCIP and iFCP do offer an integration of FC technology and IP technology and protect existing investment in FC technology. Proponents of FCoE, however, claim that IP-based Ethernet cannot reliably carry Fibre Channel traffic at the Quality of Service levels that production environments require. The reason behind this inadequacy is that the reliability requirements of higher-level protocols are not satisfied by the underlying Ethernet and the overhead of TCP to provide reliability is unacceptable from a latency and jitter perspective.

The solution to this problem is found in developing a converged enhanced Ethernet layer that adds reliability features to the conventional Ethernet.

FCoE enables FC traffic to run over a single enhanced Ethernet segment in the data center and supports SAN management by maintaining logical FC SANs across the Ethernet. This is done without any performance degradation and changes to the Fibre Channel frames. Converged enhanced Ethernet provides reliability and congestion management. It also dispenses with IP and provides seamless connectivity between native FC devices and FCoE devices.

FCoE is still a standard/technology under development. It remains to be seen to what degree it become accepted by the market and what is the real potential of this technology.

5.1.3 Motivation behind Cassini

Ethernet has several advantages over Fibre Channel. First and foremost is the cost advantage. Ethernet equipment is more cost-effective than Fibre Channel equipment. A typical 10 Gigabit Ethernet switch with 48 ports costs just a few thousand dollars. A comparable Fibre Channel switch may cost several tens of thousands of dollars. Next advantage is in terms of complexity. Ethernet deployments are inherently less complex than Fibre Channel deployments. The additional investment required in Ethernet management knowledge and training is significantly less compared to expenditure for Fibre Channel.

While development of iSCSI was the first step in enabling SANs to use Ethernet technology, FCoE is a clear testament that there is tremendous interest in migrating to Ethernet for SAN deployments. However, for Ethernet technology to succeed as a viable alternative to Fibre Channel, it must provide support for all Fibre Channel primitives. While FCoE can deal with complete feature compatibility with Fibre Channel, there are a few FC primitives, such as multipathing and zoning, which native Ethernet infrastructure and FCoE do not support.

Another overlooked problem of SAN design is that of topology design. Usual business practice is to treat SAN design as a mere connectivity problem for initiators and targets, without appropriate capacity planning. The problem for Ethernet SANs is worse. Ethernet SAN design and deployment is carried out according to the conventional Ethernet LAN topologies, which are not really tuned for SAN operations. At times, even the boundaries between regular local area networks and SANs are blurred and the Ethernet SANs are overlaid on top of enterprise LANs. Although this practice results in significant cost saving, the trade-off comes at the cost of reduced reliability and performance.

Topology design has to be one of the first tasks that SAN designers should undertake when designing Ethernet SANs. The topology design for Ethernet SANs, however, is not similar to FC SAN topology design. There are some differences in these inter-networking technologies that make FC SAN topologies incompatible with Ethernet deployments. Fibre Channel can support loops in network topology, while Ethernet is averse to loops in a network. Fibre Channel can support multiple paths between any two nodes, while Ethernet blocks out links by virtue of its spanning tree algorithm. Fibre Channel supports zoning for access control, while Ethernet does not have any explicit zoning features.

The Viking architecture readily addresses the problem of multipathing. By using VLAN mechanism, Viking supports backup path provisioning in Ethernet. Incidentally, the VLAN mechanism is also a perfect counterpart of zoning in FC SANs. VLANs are conventionally used to simplify network administration, reduce cost of segregation, and improve security. Using VLANs, one can easily implement a zoning mechanism. If one ensures that only communicating initiator and target pairs are included in a VLAN then the zoning objective is automatically achieved.

Ethernet SAN design should be done by taking into account Ethernet-specific constrains and features. To this effect, we propose a design tool, called *Cassini*, that automates the topology design-task for Ethernet (or FCoE) SANs. Cassini

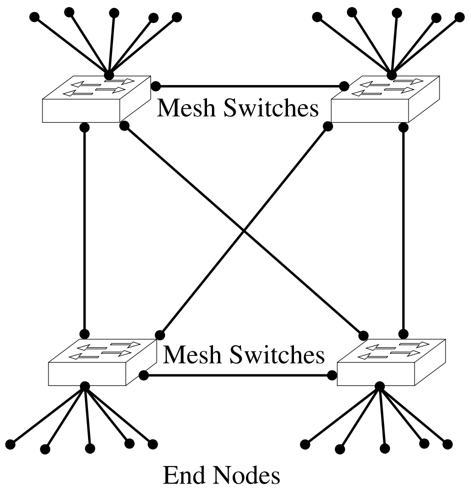


Figure 5.6: *Mesh topology. End-points are connected to a mesh of switches with small-to-moderate port density.*

specifically targets current data center and enterprise environments, where clusters of application servers and redundant storage are used for high availability. Cassini takes into account the constrains imposed by the spanning tree protocol while designing the SAN topology. It makes use of Ethernet VLAN technology to provide an Ethernet alternative to the zoning and multipathing facilities of Fibre Channel SANs. It leverages on Viking architecture while addressing these issues. It carries out SAN design by taking into account the clustered architecture in data centers.

5.2 SAN Design Considerations

5.2.1 SAN Topologies: Mesh or Core-Edge?

One of the key requirements of any automated topology generator is that, regardless of generation method, the final design must be able to support all the traffic and at the same time it should be manageable and flexible. A low-cost SAN with

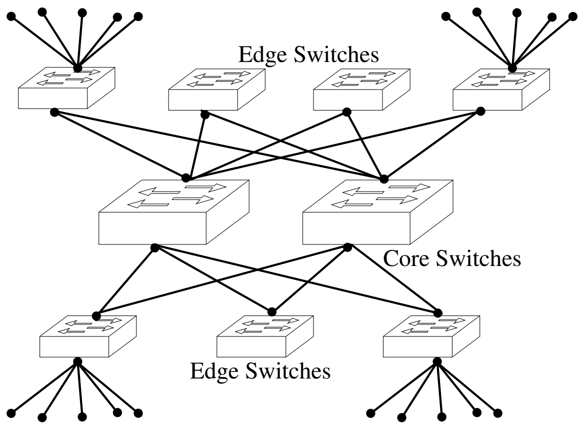


Figure 5.7: *Core-Edge topology. End-points are connected to edge-switches with small-to-moderate port density. The edge switches are in turn connected together through high port density and high performance core-switches*

highly optimized capacity planning but that is complex to visualize may not be easily manageable, whereas a symmetrical and well organized topology is easy to manage. Flexibility and manageability are qualitative terms. There is no precise measure to determine the flexibility and manageability of a SAN. Conventional wisdom, however, suggests that there are two preferred topologies for SANs. These are *Core-Edge* and *Mesh* topologies. Figure 5.6 shows a typical mesh topology and Figure 5.7 shows a typical core-edge topology.

A standard managed Gigabit Ethernet switch, such as the Cisco Catalyst 2960, provides connectivity for 8 to 48 GigE ports with a backplane switching capacity varying between 16 to 32 Gbps [cis]. A small SAN with a limited number of end-points can easily be supported using a few such switches. These switches can be connected together to form a mesh. The mesh SAN is a very simple SAN, which provides any-to-any connectivity across all the switch ports. There is no capacity planning necessary, since the backplane switching capacity of all the switches exceeds the overall requirement of the end-points.

Mesh topology, though simple to construct, cannot scale beyond four or five switches. With increasing number of switches, the number of ports required for a mesh interconnect grows exponentially and the number of available ports for end-points decreases drastically.

Core-edge topology is a scalable topology suitable for large SANs. Switches in core-edge topology are divided into two categories, edge-switches and core-switches. The servers and storage devices in the SAN are connected to edge-switches. The edge-switches are inter-connected together by means of a small number of core switches. Expansion overhead of this topology is minimal. Capacity planning for edges and core can be done independent of each other with room for expansion, such that the number of edge-switches grows only when they run out of ports to connect to cluster-nodes. The core grows when it runs out of switching capacity or ports to support the edge network(s). This kind of capacity planning enables SAN expansion in phases, without requiring a complete overhaul.

Core-Edge topology makes it easy to organize end-nodes into different edge networks according to their function. For instance, there can be multiple edge networks each dedicated to application servers, appliances, storage devices, etc. This segregation according to function enhances the manageability aspect of SANs. Keeping the manageability aspect in focus, we decided to pursue generation of core-edge topology in Cassini.

Though core-edge topology is the preferred topology for FC SANs, there are certain differences in Fibre Channel and Ethernet technologies which pose problems while deploying core-edge SANs using Ethernet. A core-edge topology allows users to provision multiple paths between source and destination. These multiple paths are used for redundancy and load balancing. Whenever there are multiple paths between a pair of nodes, there are bound to be loops in the network. This loop formation and the subsequent spanning-tree construction in Ethernet alters the active topology. Any alternations in active topology that disable active links in the network result in removal of switching capacity. This reduction in network capacity results in unused network resources affecting the capacity utilization. Cassini addresses these spanning tree issues in SAN while carrying out topology generation.

5.2.1.1 Modularity of Core-Edge Topology

There are several ways to generate a network topology. One way is to combine several building blocks together according to some predetermined rules and generate a topology in a single pass. Another way is to iteratively refine some seed topology until the final outcome meets some acceptance criterion.

An advantage of core-edge topology is its modularity, which makes it suitable for automation. The SAN design problem can be split into multiple sub-problems of designing edge networks and a core network to interconnect them. Each edge network can be designed independently, for a set of end-points with adequate connectivity and capacity to the core of the SAN. Once the design of all the edges is finalized, an appropriate core can be designed that provides sufficient connectivity and capacity for the edges to communicated with each other. This eliminates the need for end-to-end provisioning and the design process can focus on local edge and core provisioning independently. Figure 5.8 shows the different stages of SAN topology generation for a core-edge design.

5.2.2 Clustered Deployment in SAN

The first step in network planning is to begin with a network load specification. In its most basic form, the load specification is a matrix where every source and destination pair is enumerated along with the amount of network traffic between the end-points. With such a load matrix, the network topology generation problem can

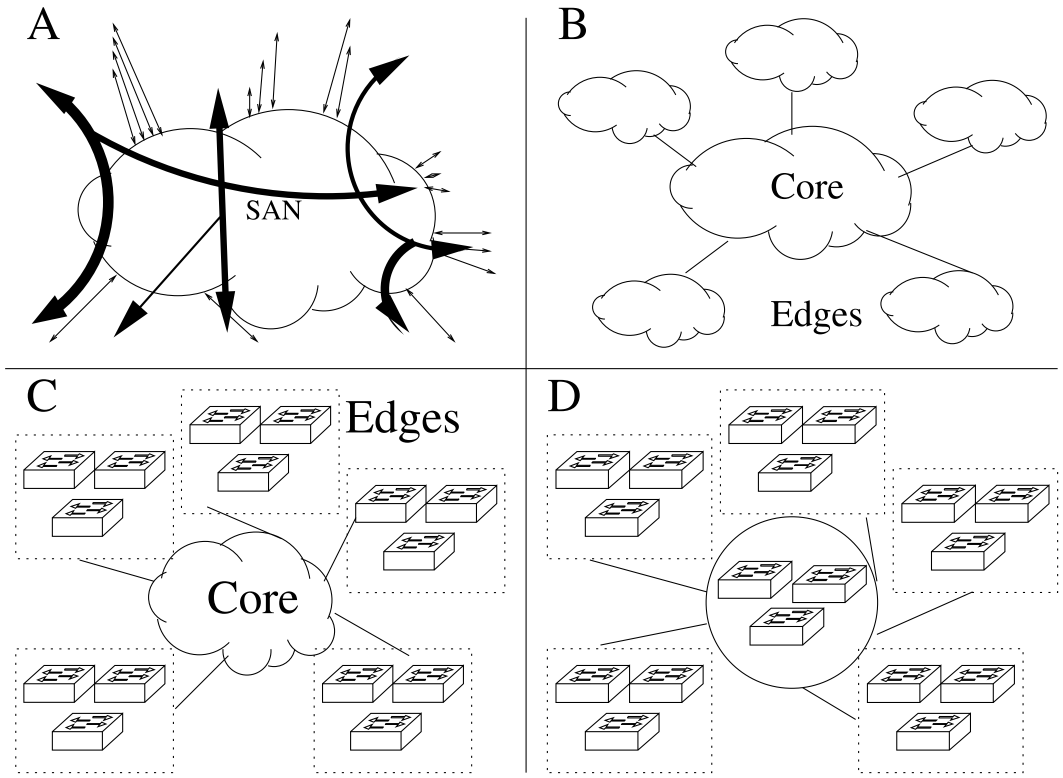


Figure 5.8: *Different stages in Core-Edge SAN design. (A) Information about different traffic flows and end-points in the SAN is compiled. (B) The overall SAN design problem is divided into simple sub-problems of Core and Edge design. (C) Each edge is designed independent of other edges. (D) Finally, a core is designed providing connectivity to all the edges, with adequate capacity.*

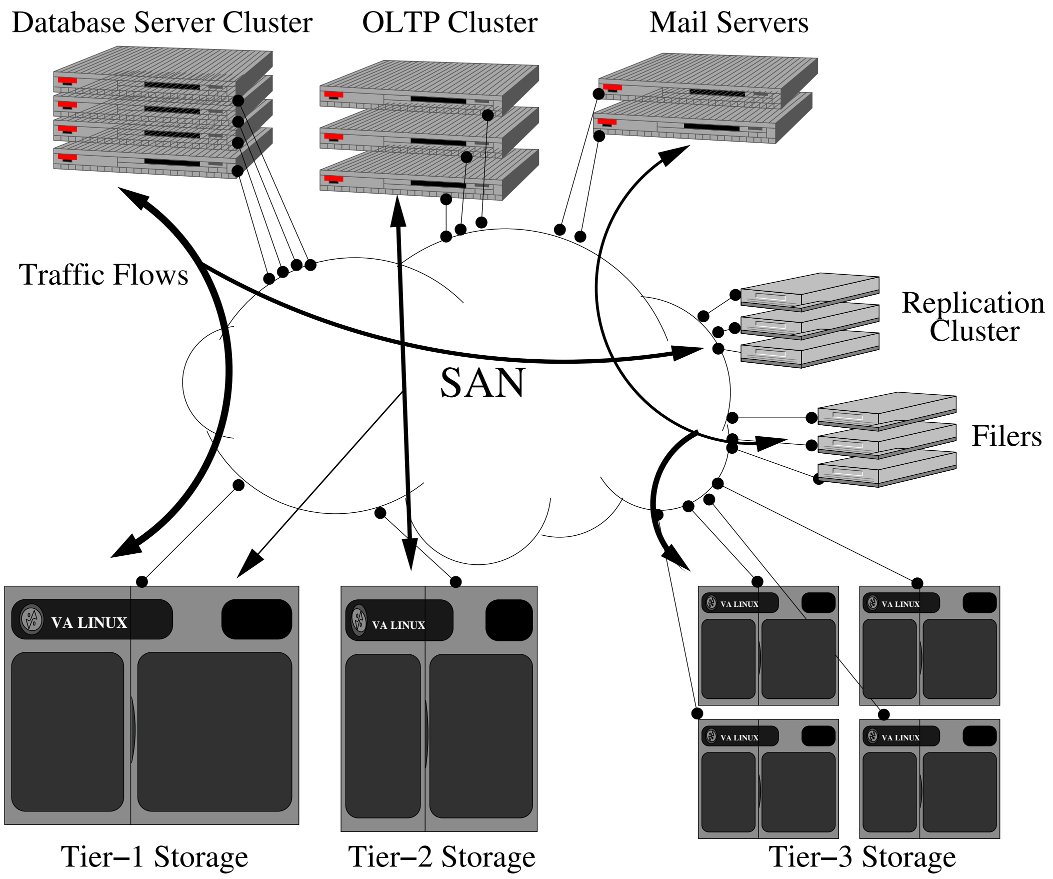


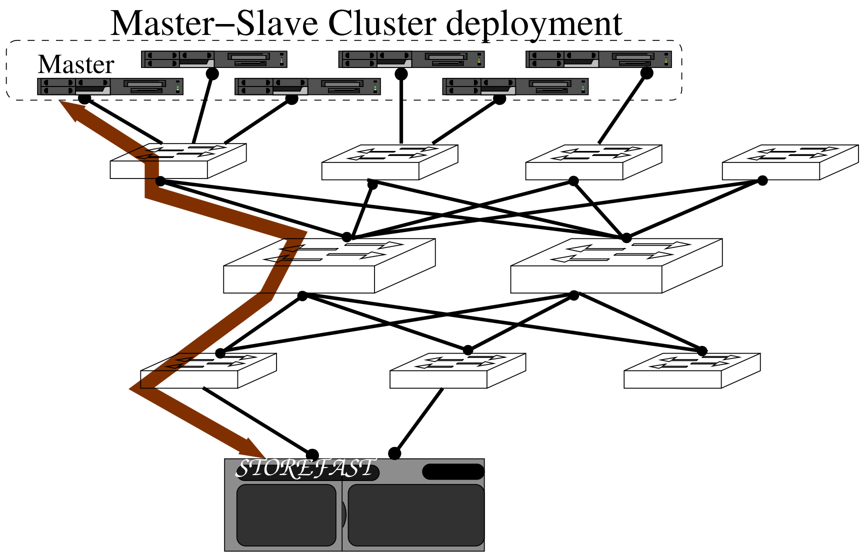
Figure 5.9: *Typical I/O traffic in data-center and enterprise SANs. The end-points in enterprise SANs are typically clusters of (a) application servers, (b) appliances, and (c) storage servers. Application users can easily specify the I/O traffic between such clusters generated by a specific application. It is, however, very difficult to attribute traffic share to individual nodes within a cluster.*

be formalized as a graph generation problem, where the generated graph accommodates all the traffic flows. This generated graph can be refined further, such that it has the lowest possible cost. The Appia FC SAN designer adopts this two step approach [WOSW02]. It generates an initial graph to accommodate all traffic flows and then uses integer programming to refine it further, to minimize the cost.

In reality, network load cannot be specified in simple terms. The end-points in SANs are seldom single nodes. Most applications are often deployed on clusters of servers. Clustering of servers provides high-availability, scalability, and reliability. Storage is often mirrored over multiple storage devices for redundancy. Application clusters are tightly integrated with other disaster recovery and data protection appliances, such as replication, backup, and filing appliances. These appliances are also clustered together for high-availability and scalability.

Based on their roles, the end-points in an enterprise SAN can be broadly categorized as (1) application clusters, (2) appliance clusters, and (3) storage clusters. The emphasis on *clusters* is to recognize the fact that it is difficult to specify I/O traffic between every node pair. It is, however, possible to predict the traffic generated by a particular application across a set (cluster) of nodes. Given an application, the administrators/users of the application can easily provide a fair estimate of I/O traffic generated by the application. Figure 5.9 shows a typical clustered environment in enterprise SANs.

Clustered deployment of servers provides high-availability, redundancy, and parallelism. There are primarily two configurations in which a cluster can be deployed. There is the *master-slave* configuration where only one server from the cluster is active and all other nodes are kept as *hot standbys*. The hot standby servers are used only in the event of failures. There is also a distributed/parallel model of cluster deployment where multiple servers in the cluster are active simultaneously, to leverage parallelism. In a master-slave deployment, it is fair to assume



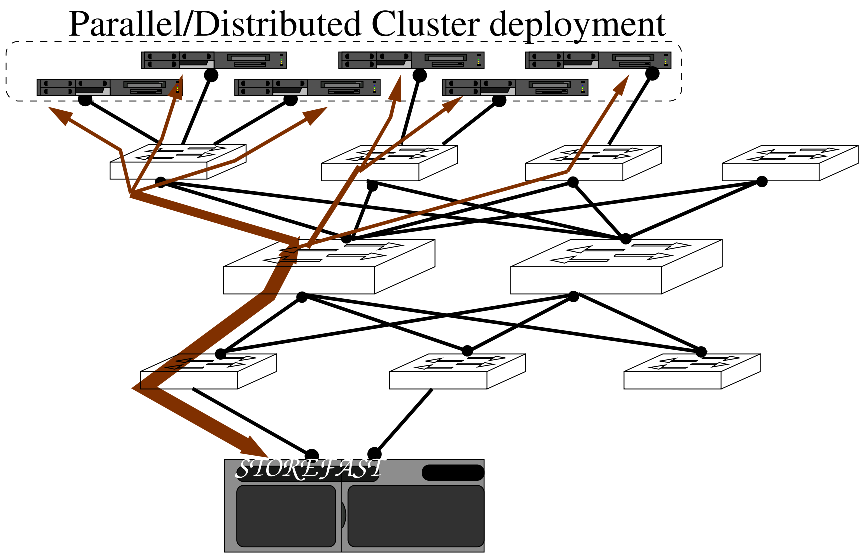


Figure 5.10: *Traffic distribution for different clustered deployments. In master-slave deployment, 100% of the traffic originates and terminates at the master node. In parallel/distributed deployment, the traffic load is almost equally shared by all the nodes. The resource provisioning requirements are different for the different types of deployment.*

that the entire, or majority of, traffic corresponding to the cluster is directed at a single master node. In such a case every node can potentially carry the entire traffic corresponding to the cluster. Whereas, in case of a parallel cluster all the nodes in the cluster can potentially share the traffic load in a balanced manner. Figure 5.10 shows how traffic can spread across multiple nodes in case of parallel/distributed and master-slave cluster deployment.

Usually, the entire traffic on an edge-switch gets aggregated over the links that connect the edge-switch to the core-switches. If a cluster is limited to a single edge-switch, then it is easy to attribute traffic between the edge-switch and the core-switch, corresponding to the cluster. The links from the edge-switch to the core can be provisioned accordingly. If the cluster has to be split across multiple edge-switches, it becomes difficult to predict the load on each edge-switch and consequently it is difficult to determine the load distribution of the cluster on edge-switches.

In order to ensure proper resource provisioning it is important to take into account the extent to which the traffic load of a cluster is carried by a single node or a group of nodes connected to a single edge-switch. In other words, it is important to know the extent of traffic load balancing across the nodes of every cluster. Since SAN administrators and users have this knowledge about cluster behavior, it is best to obtain this information as part of the traffic specification. Cassini thus relies on obtaining traffic load balancing information as part of the input specification. This information is specified in terms of the percentage of traffic load carried by a single node. For example, in the case of a master-slave cluster configuration, this would be 100% for every node in the cluster. For a distributed cluster with 5 nodes and perfect load balancing, this percentage would be 20% of the load.

In a core-edge topology end-nodes are always connected to the edge switches. Traditionally SAN designers have designed SANs with just two edges. One edge is for initiator nodes (application servers) and the other edge is for target nodes (storage servers). This restriction to two edges limited the complexity of manual design. If design automation is available, the restriction is not relevant. Further, it becomes desirable to segregate clusters into different edges based on their functionality. For instance, in data centers there are (a) application clusters, (b) storage clusters, and (c) appliance clusters; which should be separated into different edges. Cassini supports SAN design for multiple edges.

5.2.2.1 Edge Membership of Clusters

As mentioned before, traditionally core-edge SANs were designed such that there were only two edges in the network. One edge encompassed storage initiator nodes and the other storage target nodes. This convention is not relevant when storage appliances come into the picture. Storage appliances are special purpose devices that are designed to perform specific storage-related tasks, such as virtual tape libraries, backup devices, replication devices, etc. These appliances act as storage devices (targets) for application servers but act as initiators towards their own dedicated storage [HB06]. Such devices cannot be classified as pure targets or pure initiators.

The motivation behind segregating clusters into different edges is to classify them into different physical groups such that initiators and targets do not get included in the same edge. The reason for this convention is that core-switches are generally the enforcing points for zoning, masking, and other policy-related features. It is imperative that communicating nodes communicate through the core-switches. Communication through core-switches can be ensured if devices are included in different edges.

In Ethernet SANs this is an artificial restriction, because the notion of initiator and target nodes is not relevant in the context of Ethernet devices. The relationship in Ethernet is a peer-to-peer relationship. The initiator and target relationship exists only in the higher-level storage protocols. Nevertheless, it is still desirable to ensure that each communicating node pair communicates through the core-switches, conforming to the uniform structure of the SAN and enabling policy mechanisms such as rate control.

Manually determining the cluster-to-edge membership becomes a tedious task with increasing number of clusters. Cassini supports automation that partitions clusters into different edges, such that the overall number of edges is minimum and communicating clusters are always included in different edges.

A SAN design tool like Cassini should take clustering into account. The SAN design process should begin with information about cluster deployment and the traffic specification for all the applications supported by the SAN. The design process should end with a topology with appropriate capacity and connectivity for all the nodes belonging to different clusters.

5.2.3 MultiPathing

One of the major advantages of shared storage is high availability of data. Separation of storage from application servers allows storage to be shared between multiple servers. This is achieved by replacing the SCSI bus connecting servers and storage with a network. If one of the network components itself fails the storage becomes unavailable. The availability of the network itself becomes a crucial factor in a SAN configuration.

Multipathing provides a redundancy solution to SAN connectivity failures. Multiple paths between initiators and targets are provisioned to deal with failures. Every server and storage device is equipped with multiple HBAs so that the connectivity loss because of a bus adapter failure is also tolerated.

Multipathing software running on servers keeps track of the availability of storage through different paths. In the event of access failure over one path, multipathing software transparently switches over to alternate paths, without any application disruptions.

Fibre Channel uses the Fabric Shortest Path First (FSPF) protocol to determine switching paths between initiator and target nodes. FSPF supports multipath routing while providing a loop-free topology. Ethernet uses a distributed spanning tree protocol (IEEE 802.1d) to construct and impose a logical spanning tree on the physical network and it routes packets along the links of this spanning tree [IEE90]. Because of this spanning-tree architecture, some of the physical links are left unused and the effective network topology is a spanning tree of the physical topology. If the physical topology has loops in it, the active topology is only a subset of the physical topology. The VLAN based solution in Viking enables us to bridge the gap between FC SANs and Ethernet SAN by enabling multipathing in Ethernet.

5.2.3.1 Complete-Switch-Disjoint Paths

Multipathing in a SAN provides multiple paths between two end-points for redundancy and load-balancing. Figure 5.11 shows how core-edge topology facilitates multiple paths between two nodes. Multipathing is achieved by means of having a redundant path through the core to inter-connect edge-switches together. Though this multipathing provides protection against core-switch failures, it is not adequate protection against edge-switch failure. If an edge-switch fails, all the the nodes connected to the failed switch face disruption of their traffic. SAN administrators prefer to protect crucial cluster nodes by connecting them to multiple edge-switches. Such crucial nodes are equipped with multiple NICs (HBAs for FC SANs) which are connected to different edge-switches. Figure 5.12 shows an example of complete-switch-disjoint path allocation in a SAN.

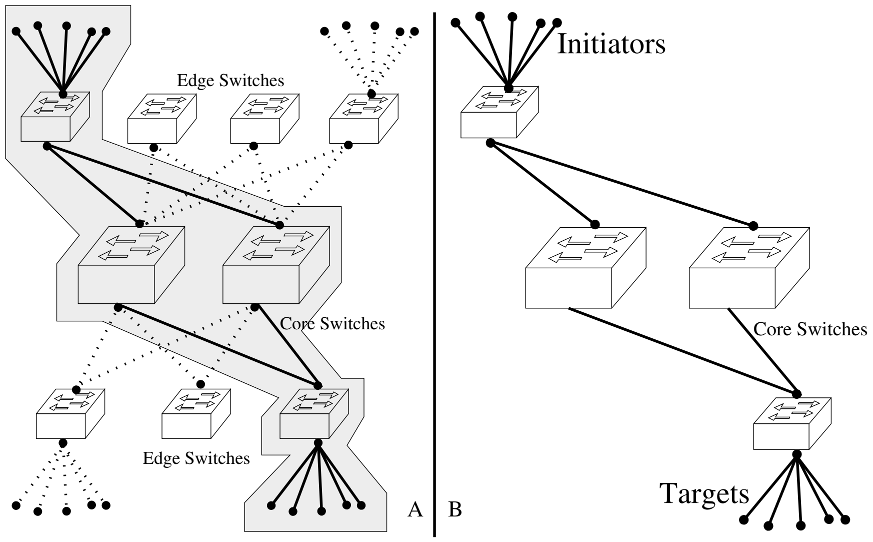


Figure 5.11: *Zoning and multipathing in SANs. Zoning provides access control within a SAN and determines which initiators and targets can communicate with each other. Multipathing provides multiple paths between two end-points for redundancy and load-balancing. Multipathing introduces loops in network topology.*

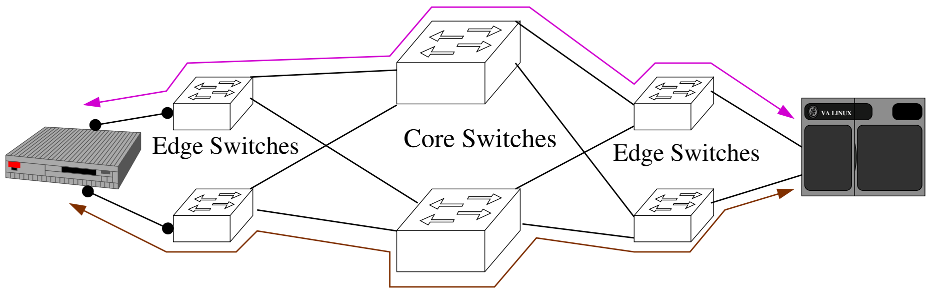


Figure 5.12: *Cassini supports complete-switch-disjoint paths between end-nodes. Each node is equipped with multiple network interfaces (ports). Cassini ensures that these ports are connected to different edge switches during topology design. These edge-switches are in turn connected to different core-switches. These switch-disjoint paths provide protection against core as well as edge-switch failures.*

Cassini supports complete-switch-disjoint multiple paths if cluster nodes are equipped with multiple ports (NICs). If the input specification indicates that each cluster node has multiple ports and a complete-node disjoint path is desired, Cassini makes sure that ports from a single node are distributed over at least two different edge-switches, to protect paths against edge-switch failure.

5.2.4 Zoning and Masking

Zoning is an essential feature for trouble-free SAN operation. Because of SAN transparency, operating systems on application servers cannot distinguish between direct attached storage and SAN storage. All storage visible to a server is assumed by operating systems to be exclusive to the server. In a SAN, every storage device is connected to every server. There are two problems with this complete visibility. First, there may be performance issues because of the large number of devices visible to an operating system. Second, there is a potential for data corruption because wrong devices may get accessed.

Zoning and Masking are access control mechanisms in SANs which limit the visibility of certain storage devices to only specific servers. Nodes (servers and storage) which are zoned together can communicate with each other. Nodes which are masked out are not even aware of the presence of other servers and devices in the SAN.

SANs use *zoning* to provide isolation between a set of initiators and targets. Zoning provides a mechanism to partition a SAN into smaller *zones* to facilitate access control. The main advantage of zoning comes from reduction of data corruption, by limiting the number of targets and initiators that can communicate with each other. For large SANs, zoning is an very essential feature. In absence of a mechanism such as zoning, all initiators and targets can potentially communicate with each other. Zoning addresses performance issues as well. Time spent in scanning the SCSI bus on an initiator is directly proportional to the number of connected devices. With a large number of connected devices, system startup and failure recovery times are much longer. Figure 5.11 shows how zoning limits the communication in SAN in small groups.

Zoning also provides rudimentary security in a SAN, controlling access to the data. It is not, however, a comprehensive data security solution in SAN.

Cassini supports zoning facility by leveraging on VLAN mechanism. Cassini ensures that only communicating initiator and target pairs are included in a VLAN.

5.2.5 Capacity Planning in SAN

Most SAN designers resort to manual topology design. The usual practice is to pick a standard/stock network topology and handcraft it to resolve connectivity issues for all initiator and target devices in the SAN. This manual design process only ensures a connectivity path for all I/O traffic flows. It does not address capacity planning requirements. In manual design, visualizing and understanding interactions of different traffic flows and deducing their impact on each other is very hard. Because of this, required network capacity cannot be estimated properly and SAN designers end up either over-provisioning or under-provisioning network capacity. For small SANs this may not pose any problem. For large deployments, however, over-provisioning is clearly not desirable. The manual design process also requires a subsequent continuous tuning to achieve proper SAN utilization. This process is manual trial-and-error design method and is clearly tedious, inaccurate, and time consuming. SAN designers can greatly benefit from automated design tools like Cassini, which can:

- Analyze the interactions of different traffic flows and deduce their impact on each other.

- Estimate required physical resources and determine an efficient way to organize them.

In effect, automation tools like Cassini address the issue of capacity planning and connectivity planning. Such a tool reduces equipment cost and the time to deploy networks resulting in significant savings.

5.3 Switch Allocation and Topology Generation

The goal of Cassini is to a generate core-edge topology for a given set of clusters adhering to a given traffic specification. The actual process for topology design involves accepting user input, designing multiple edge topologies pertaining to the input, and designing a common core to interconnect all the edge topologies.

5.3.1 Input Specification

Cassini assumes the *clustered* deployment of end-points in SAN. The required information about clusters involves details like number of nodes in the cluster, number of ports per node, and multipathing requirements. The traffic specification lists the bandwidth requirement of traffic between all communicating clusters. The clusters are always connected to the edge switches. Proper capacity planning ensures that there are sufficient links handling the traffic between every cluster and the core.

Cassini generates a core-edge topology for SANs. There are two distinct network building blocks involved in topology design, edge-switches and core-switches. The edge-switches have small-to-moderate port density. The core-switches have high port density and a high performance backplane. Cassini's input specification describes details about these network elements.

The overall input to Cassini includes the following details:

- Number of Clusters.
- Description of each cluster. This description includes:
  - Number of nodes in the cluster.
  - Number of ports per node.
  - Desirability of a complete-switch-disjoint path.
  - The extent of the cluster load carried by each node.
- The characteristics of edge-switches and core-switches are:
  - Number of ports.
  - Link speed of each port.
  - Backplane switching bandwidth.
- Details about the traffic load of every application running on the different clusters. This includes traffic generated by an application between every relevant cluster. It should be noted that a cluster may appear in traffic specifications corresponding to multiple applications. For example, a single backup server may be responsible for backing-up data of multiple applications on different application servers and hence may appear in multiple application traffic specifications.

5.3.2 Edge Design

The first step of the edge design process is to partition the clusters into multiple edges, if necessary. The second step of edge design is to compute the number of edge-switches required to accommodate nodes from all the clusters that belong to an edge. The third step is to generate a connection topology for cluster nodes and the switches included in every edge. While accommodating all the nodes onto edge-switches, both connectivity and capacity issues are worked out. The switches must have enough ports to accommodate all the nodes and at the same time none of the switches should be subjected to more traffic than it can handle. Each edge-switch has to be connected to two distinct core-switches, for protection against core-switch failures. Further, if complete-switch-disjoint paths are desired for any of the clusters, the switch assignments must be such that every node from the cluster is connected to at least two edge-switches. Capacity and connectivity planning must also take into account the connectivity and capacity required to connect the edge-switches to the core-switches.

The edge design steps can be enumerated as follows:

1. Partition the clusters into multiple edges, such that initiators and targets are included in different edges.

2. Estimate the number of switches required to support all the clusters that belong to each edge.

   - Ensure that sufficient capacity and connectivity is provided to connect cluster nodes with the edge-switches and also the edge switches with core-switches.

   - Ensure that the fault-tolerance of connectivity requirements are satisfied. Protection against core-switch failures can be provided by planning connectivity of every edge-switch with at least two core switches. Edge switch redundancy can be provided by provisioning complete-switch-disjoint paths.

3. Minimize the edge-switch resource cost.

4. Generate a topology for the edges being designed.

5.3.2.1 Cluster-to-Edge Assignment

The key requirement for automated partitioning of clusters into multiple edges is that the number of edges be minimized. The advantage in keeping the number of edges low is that it increases the number of cluster members in each edge and reduces possible fragmentation of the switch resources allocated to each edge.

In a conventional core-edge SAN, the traffic pattern is such that communication is always between *hosts* (initiators) and *devices* (targets). There is no (SCSI-related) communication between a pair of hosts or between a pair of devices. The desired connectivity graph of a SAN is always a *bipartite graph*, where the end nodes are pure initiators or targets. It is easy to classify clusters into initiator and target categories by simply analyzing the connectivity graph.

The problem changes when appliances come into the picture. Application hosts communicate with appliances. The appliances in-turn communicate with their own dedicated storage. Multiple appliances may also communicate with each other. This changed communication pattern results in a connectivity graph that, instead of being bipartite, is a k-partite graph with $k >= 3$. This means that the minimum number of edges required to ensure that every pair of clusters use the core infrastructure for communication is k.

Determining the partiteness of a graph is an instance of the vertex coloring problem. The minimum number of colors required to paint the vertices with different colors such that no two adjacent vertices are painted with the same color defines the *chromatic number*, or the partiteness of a graph.

Determining the chromatic number (or the partiteness) of a graph is an NP-hard problem. Fortunately there is a well-known heuristic, the Welsh-Powell algorithm, that solves the vertex coloring problem. Cassini applies the Welsh-Powell heuristic to partition the clusters into different edges [WP67]. The cluster partitioning problem is simply converted into the vertex coloring problem.

The application of Welsh-Powell heuristic is as follows. Cassini constructs a connectivity graph G from the traffic specification. Every cluster is denoted as a vertex in the graph. Vertices in the connectivity graph are connected together if there is any communication between the corresponding clusters. The partitioning involves the following steps:

```
1.  Sort the clusters in non-increasing order of
degree of connectivity.
2.  Reset the edge membership of every cluster.
3.  Traverse the clusters in order, doing the following
for each cluster:
      Mark the cluster as a member of Edge_1 if it is not
      a member of any edge and if it does not communicate
      with any other cluster from Edge_1
4.  Repeat step 3 for Edge_2, Edge_3, ... until no cluster
is left without edge membership.
```

This procedure ensures that clusters included in each edge do not communicate with each other. Further, the number of edges required is low. After the cluster membership for each edge is determined, Cassini designs an edge topology for clusters included in each edge.

5.3.2.2 Switch Allocation

Cassini does not aim to compute the absolute minimum number of required edge-switches. Such a tight requirement would result in a complicated assignment of different nodes from same cluster to different edge-switches. Moreover, tight capacity planning would warrant mixing nodes from different clusters, so that some nodes from a low-traffic cluster offset the high traffic generated by nodes from some other

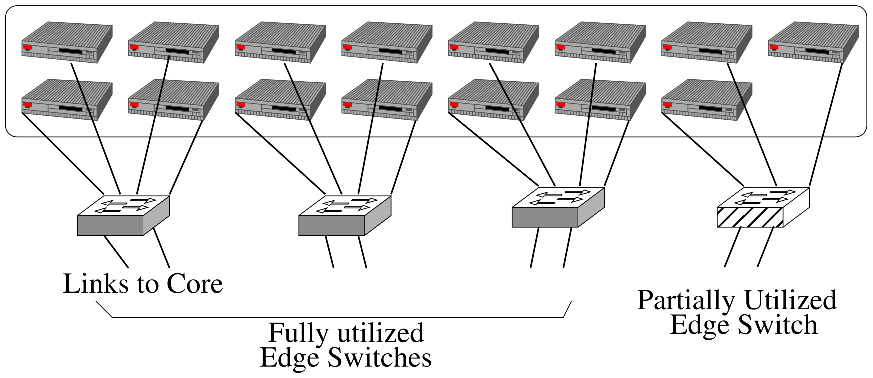


Figure 5.13: *Edge Switch allocation for a cluster. The maximum number of cluster nodes an edge-switch can handle is computed. A topology map of the edge is generated, with all the nodes connected to edge-switches. The generated map is such that all the edge-switches are fully utilized with connections from cluster nodes, with the possible exception of one partially-utilized switch.*

clusters. Such a mix-and-match of cluster nodes can easily lead to fragmentation of cluster nodes across several edge switches. This fragmentation is a manageability nightmare for network administrators. Instead, Cassini aims at simple assignments that will be more manageable and flexible. Cassini tries to minimize cluster fragmentation and tries to place nodes of a single cluster on an edge-switch, to the maximum extent possible.

Cassini follows a simple switch allocation policy for every cluster. First, the maximum number of cluster nodes that an edge-switch can accommodate is computed. This computation is performed by taking into account the traffic that needs to be sustained over the switch and the number of ports required to connect the cluster nodes, as well as the inter-switch-links required to connect the edge-switch to core-switches. Once the maximum number of cluster nodes per edge-switch is determined, a topology map of the edge is generated, with all the nodes connected to edge-switches. The generated map is such that all the edge-switches are fully utilized with connections from cluster nodes, with the possible exception of a single switch which is partially utilized. This process is repeated for every cluster that belongs to the edge being designed.

Input for edge topology design algorithm for a cluster is the information about the cluster and the edge-switches. Cluster information includes the number of nodes in the cluster — n, the number of ports per-node — p, and the traffic load — L. Edge switch information includes switch details, such as number of ports — $EdgePorts$, line speed of each port — $LinkSpeed$, and backplane switching capacity – $SwitchCapacity$.

The edge design algorithm processes the input to compute the maximum number of nodes, $NumNodes$, that an edge switch can support. It also computes the number of links, $LinksToCore$, that are required to connect each edge switch to the core. It then outputs the topology for all the nodes in the cluster such that at most $NumNodes$ nodes in the cluster are connected to every edge switch. The edge switch is provisioned with $LinksToCore$ ports for connecting links to the core and remaining ports are connected to the cluster nodes. Figure 5.13 shows the manner in which edge-switches are populated for a cluster.

The following steps describe the edge design algorithm for a single cluster:

```
Input:
        Cluster $C$ with $n$ ($N_1$ to $N_n$) nodes.
        Each node $N_i$ has $p$ ports.
        ($2p$ if a complete-switch-disjoint path is desired.)
        The cluster load $L$ and maximum traffic load $l$ between
        a single node/port and core is computed from
        the traffic specification.
        Edge-switch characteristics $EdgePorts$ each port with
        speed $LinkSpeed$ and capacity $SwitchCapacity$
```

Output:

```
    Topology $T$ with $pn$ ports connected to $e$ ($E_1$ to $E_e$)
    edge-switches.
    Details about the cluster edge-links between each
    edge-switch $E_i$ and the core.
```

Algorithm:

```
If a complete-switch-disjoint path is requested, divide $2pn$
ports into $2$ symmetrical sets of $pn$ ports each.
Solve the problem for $pn$ ports.

Compute the maximum number of nodes an edge-switch
can support using the following steps:
    Set $NumNodes = 0$
    Repeat the following steps until
          explicit BREAK is called:
       Set $i$ = $NumNodes + 1$
       Set $SwitchLoad$ = MIN($L$, $l * i$)
       Set $LinksToCore$ = 2 * CEIL($SwitchLoad/LinkSpeed$)
       If $SwitchLoad <= SwitchCapacity$
          AND $EdgePorts >= p * i + LinksToCore$
             $NumNodes = i$
       Else
             BREAK out of loop

Set $e$ = CEIL($n/NumNodes$)
Allocate $e$ edge-switches in topology $T$.
Divide the $n$ nodes into $e$ groups of $NumNodes$ each.
```

```
Connect $NumNodes$ nodes with $p$ ports each, to $e$ switches.
Reserve $LinksToCore$ ports from each edge-switch to connect
them to the core.
Mark these ports as belonging to the cluster $C$.
Associate each of the $LinksToCore$ ports with
($NumNodes/LinksToCore$) nodes.
```

The edge layout generated by the previous algorithm is shown in Figure 5.13. If a complete-switch-disjoint configuration is desired then two symmetrical edge topologies are generated, such that each node has presence in both the topologies. The topologies are such that all the edge-switches, with the possible exception of a single switch are fully utilized.

5.3.2.3 Switch Cost Reduction

Once all the clusters are accommodated on edge-switches, the switches can be categorized into two classes, namely, fully utilized and partially utilized. Here Cassini finds an opportunity to reduce the resource cost without sacrificing manageability of the network. Cassini tries to merge partially utilized switches together, such that the number of highly-utilized switches is maximum ensuring that switching capacity and switch ports are not wasted by under-utilizing them. Finding the best possible merge is a hard and computationally expensive problem. Cassini resorts to standard bin-packing heuristics to achieve this goal.

Thus, the Cassini edge design process is divided into two phases. The first phase results in each cluster being connected to some fully utilized edge-switches and at most a single partially utilized edge-switch. In the second phase, all partially utilized switches from different clusters are merged together, resulting in a reduced number of highly utilized switches.

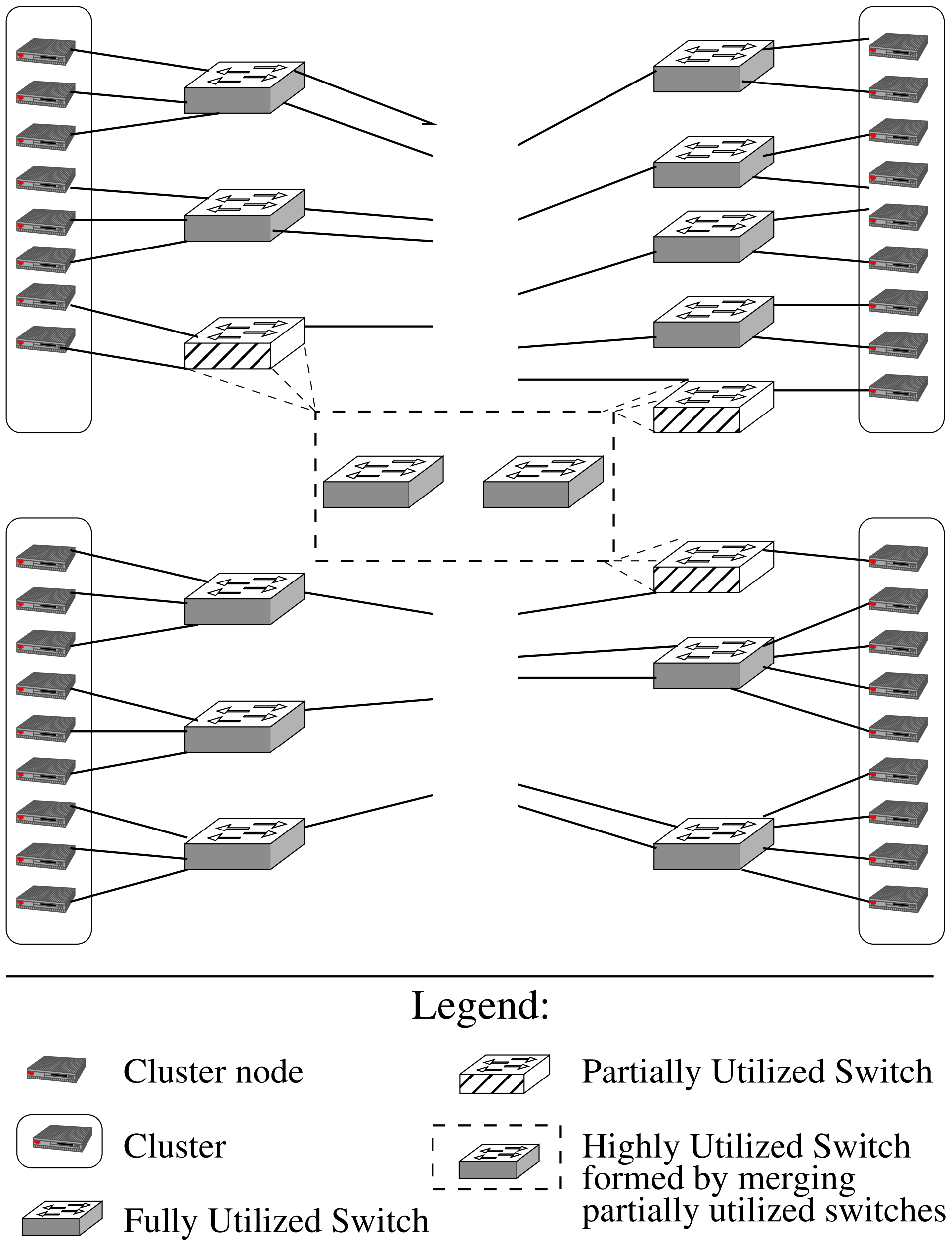

Legend:
Cluster node
Cluster
Fully Utilized Switch
Partially Utilized Switch
Highly Utilized Switch formed by merging partially utilized switches

Figure 5.14: *Edge design for multiple clusters. Fragments of clusters on partially utilized switches are merged together using a bin-packing algorithm to improve overall switch utilization.*

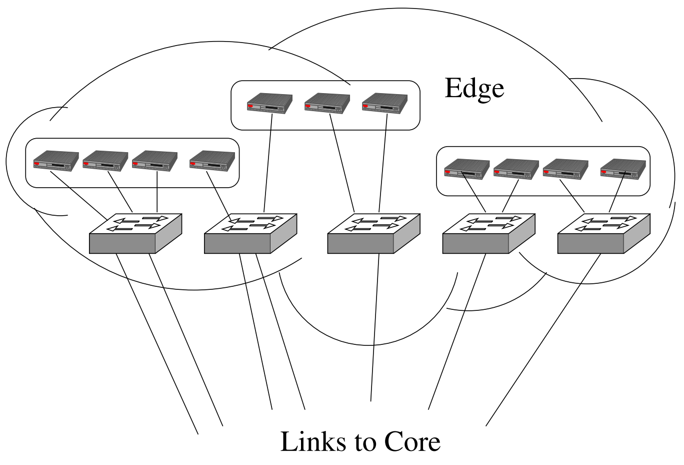


Figure 5.15: *The final topology of an edge after the merge of partially utilized switches is complete. The number of partially utilized switches is reduced and the overall utilization of existing switches is improved.*

This merging problem is a simple variation of the classical *bin-packing* problem. The bin-packing problem is a combinatorial NP-hard problem. In bin-packing, objects of different volumes must be packed into a finite number of bins of capacity V in a way that minimizes the number of bins used. The difference in switch merging is that instead of bins we have edge-switches with finite capacity. Further, instead of a single resource, the packing activity has to be performed with respect to two resources. These resources are the utilized ports and the utilized switching capacity on the edge-switch.

There are two heuristics to solve the bin-packing problems. These are *best fit decreasing* and *first fit decreasing* strategies. It has been established that these heuristics have an upper bound of $11/9 \ \ OPT \ + \ 1$ bins, where OPT is the number of optimal bins [Min91].

Cassini uses the *first fit decreasing* strategy to solve the edge-switch merging problem. Input to the edge switch merge algorithm consists of a list of all partially-utilized switches, $SwitchList$ in the edge. The details of each switch E in the

$SwitchList$ include the utilized port count EP and the backplane load on each switch EL. The edge-switch merge algorithm merges the switches together using first fit decreasing strategy and outputs the new topology after the merge. The algorithm is as follows:

Input:

```
A list $SwitchList$ of $n$ switches, $E_1$ to $E_n$.
Each switch $E_i$ with utilized port count $EP_i$ and
   switch load $EL_i$
```

Output:

```
A list of $m$ edge-switches, such that $m < n$
All connections on switches from the input switch list
   reassigned to new switches.
```

Algorithm:

```
Sort the input list of switches in decreasing
   order of utilized port count $P_i$.
For switches with the same utilized port count use
   switch load $L_i$ as a second parameter for sorting

Repeat the following steps until $SwitchList$ $! = \phi$:
   Allocate a new switch $ESwitch$ with available
      ports as $ESwitchP$ and capacity as $ESwitchL$
         While $\exists$ $E$ $\in$ $SwitchList$ such that
         $EP$ $<=$ $ESwitchP$ AND $EL$ $<=$ $ESwitchL$
            Find first such $E$ by scanning through
                  the list.
            Remove $E$ from $SwitchList$
            Merge $E$ with $ESwitch$ as follows
               Reassign $EP$ ports to $ESwitch$
```

```
Set $ESwitchP$ = $ESwitchP$ − $EP$
Set $ESwitchL$ = $ESwitchL$ − $EL$
```

The merge process is shown in Figure 5.14. The outcome of the merge is shown in Figure 5.15. The number of partially utilized switches is reduced and the overall utilization of existing switches is improved.

5.3.3 Core Design

Once all edge topologies in the SAN are completely designed, these edges have to be connected together through the common core. The core must provide connectivity to all the edges and also enough switching capacity to handle communication across all the edges. If the port and capacity requirements of all edges can be handled by a single core-switch, the core design is trivial. Links emanating out of all the edges are connected to a single core-switch and the SAN design is complete. For small SANs this may well be the case. For large SANs which require more than a single core-switch, however, the core design is somewhat more involved.

The goals of core design are similar to the edge design goals. The core has to provide connectivity and capacity to the edges. Also, resource cost has to be minimized.

Core design involves the following four steps:

1. Estimating the required number of core-switches.
2. Determining the distribution of different edge-links to the core-switches.
3. Interconnecting the core-switches together with the appropriate number of inter-switch-links.

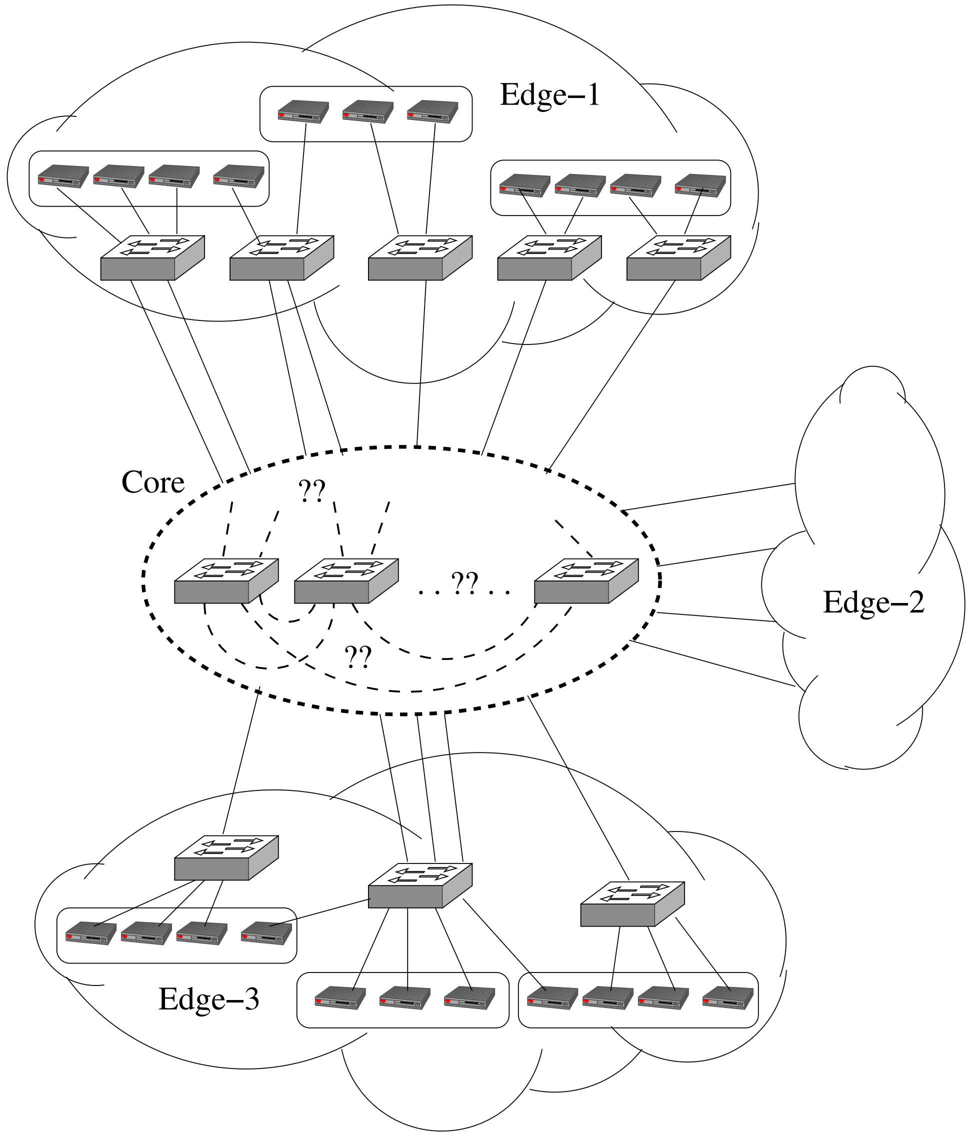


Figure 5.16: *Core design process. Once the edge design is complete, Cassini estimates the number of core-switches, determines the connectivity between edge and core-switches, and determines the connections between the core-switches.*

4. Minimizing core-switch cost.

These core design tasks are illustrated in Figure 5.16.

5.3.3.1 Differences with Edge Design

Conceptually edge-design and core-design are similar to each other. In both cases we need to determine the number of switches required and interconnect the switches together. There are, however, key differences in both design requirements that warrant totally different design processes. Unlike edge design, capacity planning and inter-switch-link provisioning in the core cannot be performed without complete knowledge of the traffic load over a switch and its neighboring switches. Edge-switches do not communicate with each other. This allows us to do capacity planning on an edge-switch independent of other edge-switches.

The input traffic specification provides traffic load information at the *cluster* granularity. Every edge-link connecting edge-switches and core-switches carries only a portion of the entire cluster traffic. Though each edge-link carries a different amount of load, depending on the number of cluster nodes associated with the link during the first phase of edge design, the aggregate load carried by all edge-links corresponding to a cluster is always the total traffic load of the cluster. From a core design process perspective, it is beneficial to deal with the aggregate load rather than fractional load, as this simplifies the task of capacity planning. If all the edge-links corresponding to a cluster are connected to a single core-switch, the task of distributing edge-links is converted into a simple task of assigning different clusters to different core-switches. This has the added benefit of avoiding fragmentation of cluster traffic. Further, estimating traffic across different core-switches becomes easy, as it is just a matter of computing traffic between clusters assigned to different switches. This simplification is shown in Figure 5.17.

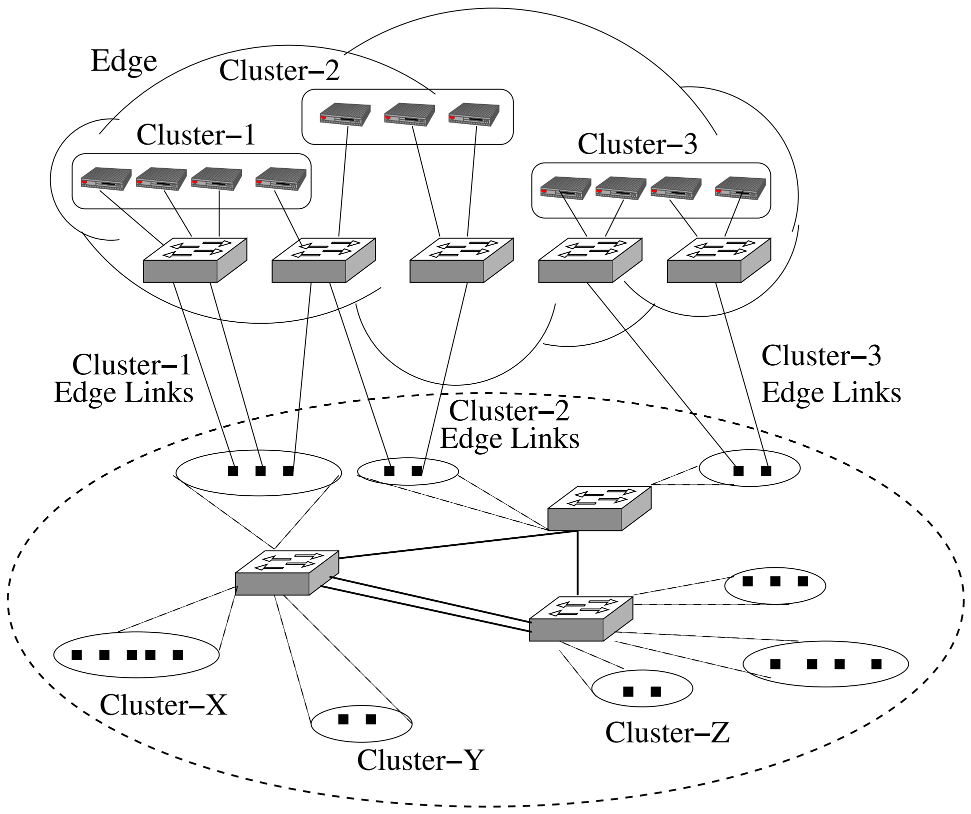


Figure 5.17: *Cluster to core-switch assignment. The task of distributing edge-links to core-switches is simplified if all the edge-links corresponding to a single cluster are treated as aggregates. Capacity provisioning becomes easy, because entire cluster traffic is localized to a single switch. Fragmentation of cluster traffic is avoided. Estimating traffic across different switches also becomes easy, as it is just a matter of computing traffic between different clusters.*

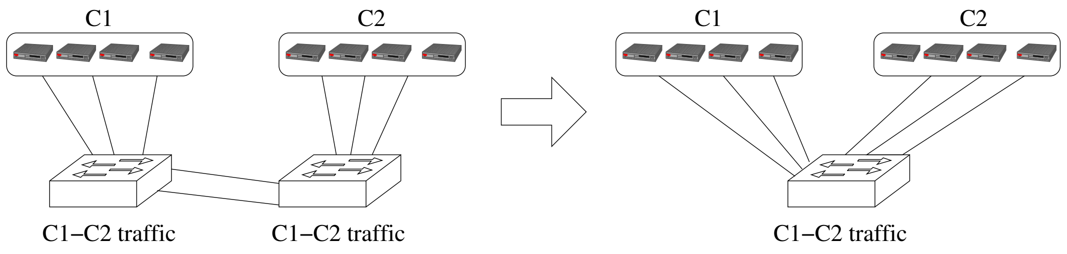


Figure 5.18: *Core switch utilization after a merge. The merging of two core-switches does not result in the sum total of the switch load and the used port count. The merged switches eliminate the traffic over the inter-switch-links and the ports used by these inter-switch-links.*

Thus, all edge-links from a single cluster are always connected to the same core switch. In edge design, however, this is not the case. The nodes from a single cluster are connected to multiple edge-switches.

Core-switches cannot be merged together using the same bin-packing technique used in the edge design phase. The key difference here is that the switch load after a merge is not the sum total of the individual switch loads. Also, the used port count after the merge is not the sum total of the used port counts before the merge. The final load after merge is actually less than the sum total of individual switch loads. The final load differs from the sum-total by the amount of traffic over the inter-switch-links. Ports used by inter-switch-links are also freed up by the merge, reducing the overall count of used ports. Figure 5.18 shows an example of how the switch load and used port count is actually reduced after the merge. Bin-packing cannot deal with such dynamic values for a merge.

These differences require a different core design process and the edge design algorithms are not applicable for core design.

5.3.3.2 Core-switch Allocation and Edge-link Assignment

Though assigning entire clusters to core-switches simplifies the task of edge-link distribution, it does not solve the problems of determining how many core-switches are required and which clusters should be assigned to which core-switches.

If the convention of connecting all the edge-links from a cluster to the same core-switch is followed, determining the number of required core-switches is trivial. The *maximum* number of core-switches required will never exceed the total number of clusters for which the SAN is being designed. The implicit assumption here is that a single core-switch can always handle an entire cluster in terms of capacity and connectivity. Given that core-switches are typically high port density and high backplane-capacity switches, this assumption is a reasonable assumption.

Thus, the core design process begins by allocating the maximum number of core-switches and assigning clusters to them. The number of inter-switch-links between each pair of core-switches is easily computed from the traffic between them. A core with the maximum number of switches is clearly an over-provisioned and under-utilized core. Almost every core-switch would have significant amount of unused ports and spare backplane switching capacity. Such under-utilized switches can be merged together to improve resource utilization and reduce the required number of core-switches.

5.3.3.3 Merging Core-Switches

Core-switches are typically high port density and high performance switches. The ratio of backplane switching capacity to port bandwidth is usually very high. If the source and destination nodes of traffic connect to different core-switches, the traffic has to be sent over the inter-switch-links connecting the two switches. Large

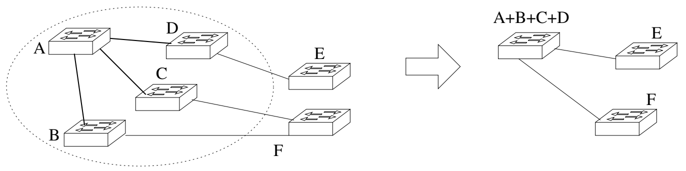


Figure 5.19: *Traffic Locality based merge. Switch A is the core-switch with the maximum number of inter-switch-links. Switches B, C, and D can be merged together with A to improve the locality of traffic and reduce the number of inter-switch-links.*

amounts of inter-switch traffic requires large number of inter-switch-links. Dedicating more switch ports to inter-switch-links reduces the number of available ports for cluster edge-links and hence increases the number of core-switches required to support the edge-links. Also, the traffic flowing through inter-switch-links adds to the backplane switching load of all the connected switches. The key to improving resource utilization in the core is to merge core-switches in such a way that the count of inter-switch-links is reduced.

Traffic Locality-based Merge

Reduction in the number of inter-switch-links is possible if the distribution of clusters over core-switches takes into account the locality of traffic. If multiple clusters with heavy traffic between each other are placed on the same core-switch, traffic gets localized on the single switch and traffic over inter-switch-links is reduced. Cassini follows this guideline to improve traffic locality while merging core-switches.

Figure 5.19 illustrates how traffic locality based merge is carried out. Consider the initial core design as described in this Figure. Switch *A* can be identified as the switch with the maximum inter-switch-traffic. Neighboring switches B, C, and D can be merged with A, if A has sufficient spare capacity and spare ports. This merge reduces the inter-switch-traffic by localizing traffic within a single core-switch. If switch A still has spare capacity and ports, a further merge can be carried out with switches E and F.

Input to the locality-based merge algorithm consists of all list of all the clusters — C, the traffic load of of each cluster — $Load_c$, and the pairwise traffic specification $Traffic_{ij}$ between every cluster pair C_i and C_j. The essence of locality-based merge is to allocate a core-switch for each cluster, connect corresponding edge-links to the core-switch and then merge these switches together. The merge process identifies a $PivotSwitch$ with the maximum number of inter-switch-links and then repeatedly tries to merge neighboring switches with this pivot switch, till the pivot switch either runs out of ports or runs out of backplane switching capacity. Whenever a $PivotSwitch$ gets saturated, a new $PivotSwitch$ is identified and the merge continues till no more merges are possible. The output of the locality-based merge is a merged topology of core-switches. The algorithm for core design using locality-based merge is as follows:

```
Input:
      A list of all the clusters.
      Edge-links associated with every cluster $C$.
      Traffic load $Load_c$ associated with every cluster $C$.
      The pairwise load specification $Traffic_{i,j}$ between every
         cluster pair $C_i$ and $C_j$.
Output:
      Topology for the Core and the interconnections to Edges
Algorithm:
      For every cluster $C$ allocate a core-switch and connect
         corresponding edge-links to the allocated switch.
      Determine the backplane load on every switch, based on
         traffic load $Load_c$ of the connected cluster.
```

```
    Determine the inter-switch-traffic and inter-switch-links
       between all the core-switches using the pairwise load
       specification $Traffic_{i,j}$.
$try\_merge$ :
    Sort the switches in non-increasing order of
       the number of inter-switch-links to each.
    Identify $PivotSwitch$ as the first non-saturated switch
       in the sorted switch list.
    Sort the neighbors of $PivotSwitch$ in decreasing order
       of their inter-switch-traffic with the $PivotSwitch$
    Traverse the neighbors in order and merge them with
       $PivotSwitch$ if possible.
          After every merge recompute the load on the
          $PivotSwitch$ and the number of inter-switch-links
             to other switches.
          If the $PivotSwitch$ cannot be merged with any of
          its neighbors, mark it as saturated.
    Repeat the process from $try\_merge$ until no $PivotSwitch$
       can be identified.
    Output the topology obtained from the merges.
```

Greedy Merge

The traffic locality-based merge focuses on a pivot switch and tries to merge it with neighboring switches until it gets saturated. Once the pivot switch is saturated, a new pivot switch is identified and the merge process continues. Merging of any two switches, however, results in a change in traffic dynamics, as shown in

Figure 5.18. Because of this change, the pivot switch may not remain the best candidate for subsequent merges after the first merge. To contend with this we use a *greedy merge* strategy. In the greedy merge process, instead of a single pivot switch a pair of switches are identified as candidates to merge. The candidates are chosen such that the inter-switch-traffic between them is higher than any other switch pair. If neither of these candidates is saturated and a merge between them is possible, the merge is carried out. If either one of the switches is saturated or they cannot be merged, the next suitable pair of candidates is chosen.

This process greedily attempts to always merge the switches with highest inter-switch-traffic. Input to the greedy merge algorithm is the same as the input to locality based merge. It consists of all list of all the clusters — C, the traffic load of of each cluster — $Load_c$, and the pairwise traffic specification $Traffic_{ij}$ between every cluster pair C_i and C_j. The essence of greedy merge is merge is to allocate a core-switch for each cluster, connect corresponding edge-links to the core-switch and then merge these switches together. The merge process identifies two switches SW_x and SW_y, which can be merged together. The switches are such that the inter-switch-traffic $SwitchTraffic_{xy}$ between them is higher compared to any other switch pair in the core. These two switches are merged together. Merge process repeats till all the core-switches are saturated and no additional merge is possible. The output of the greedy merge is a merged topology of core-switches. The algorithm for core design using greedy merge is as follows:

The algorithm for greedy merge is as follows:

Input:

```
    A list of all the clusters.
    Edge-links associated with every cluster $C$.
    Traffic load $Load_c$ associated with every cluster $C$.
    The pairwise load specification $Traffic_{i,j}$ between every
```

```
          cluster pair $C_i$ and $C_j$.
Output:
      Topology for the Core and the interconnections to Edges
Algorithm:
      For every cluster $C$ allocate a core-switch and connect
          corresponding edge-links to the allocated switch.
      Determine the backplane load on every switch, based on
          the traffic load $Load_c$ of the connected cluster.
$try\_merge$ :
      Determine the inter-switch-traffic and
          inter-switch-links between all the core-switches
          using the pairwise cluster load specification $Traffic_{i,j}$.
      For every pair of core-switches $SW_x$ and $SW_y$ and
          $SwitchTraffic_{x,y}$ between them, construct a tuple
          $< SW_x, SW_y, SwitchTraffic_{x,y} >$.
      Sort all the tuples in decreasing order of $SwitchTraffic_{x,y}$.
      Find the first tuple from the sorted list such that $SW_x$
      and $SW_y$ can be merged.
          Merge $SW_x$ and $SW_y$.
      Repeat the process from $try\_merge$ until no merge can be
          performed.
      Output the topology obtained from the merges.
```

5.3.3.4 Dual-Core Topology

The entire traffic in a SAN with a core-edge topology flows through a small number of core-switches. If any of the core-switches fails, the potential for traffic disruption is very high. For high availability of a SAN it is important to have some failover strategy to deal with such core-switch failures. Most SAN designers opt for redundant core topologies, where every edge-switch is connected to at least two or more core-switches. In the event of a single core-switch failure, traffic can be diverted through alternate switches without major disruption.

Cassini also uses a dual-core topology model for fault-tolerance. During the first phase of edge design, as described in section 5.3.2.2, edge-links are provisioned such that each edge-switch is connected to two different core-switches. The edge design process always allocates edge-links as pairs, where each edge-link has an associated counterpart that connected to a different core-switch. For simplicity of the design process, it always generates two symmetrical core topologies. The edge-links are distributed across these two core topologies. Topology generated in this way is provisioned with two core-disjoint paths for every communicating node pair. The locality-based merge and greedy merge algorithms described earlier are core design algorithm for a single core. To deploy dual-core topology, two identical/isomorphic cores are designed and deployed.

5.3.4 Path Computation and VLAN Grouping

The edge and core design processes address the SAN topology design problem effectively. There is more to SAN design than just topology design. Topology design addresses connectivity and capacity provisioning in SAN. There remains an important operational issue of *routing* that must be addressed to utilize the designed topology. The core-edge topology designed by Cassini is mainly targeted towards Ethernet SANs. The underlying infrastructure is built using Ethernet switches. As illustrated in Figure 5.12, Cassini supports multiple paths between every pair of nodes in communicating clusters. There are two problems in Ethernet networks because of multipathing.

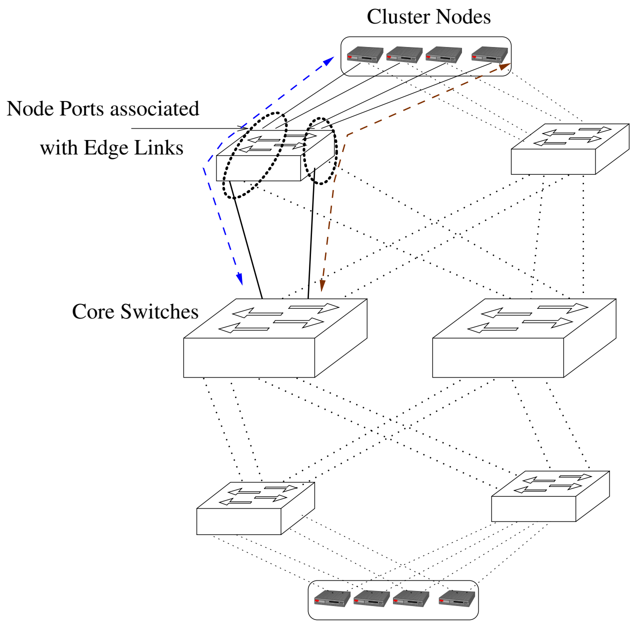


Figure 5.20: *Computation of path(s) between a node pair. During edge design each edge-link is associated with a set of nodes. This association determines the communication paths to the core for all the nodes. A path between two nodes can be formed by connecting their respective half paths to the core. When complete-switch-disjoint paths are required, two symmetrical cores are provisioned. Each node has two switch-disjoint paths to each core. Complete-switch-disjoint paths can be formed simply by connecting respective paths to the core for each core.*

The first problem is how to select the appropriate path between the nodes from several possible paths. The appropriate paths are the paths that are provisioned during edge and core topology design. The next problem is how to actually use the selected paths. Ethernet uses a distributed spanning tree protocol (IEEE 802.1d) to construct and impose a logical spanning tree on the physical network, the routes packets along the links of this spanning tree. Thus, in conventional Ethernet networks it is not possible preselect the switching path.

The path computation and VLAN grouping phase of Cassini address these issues. The edge and core design phases do not directly provision the . Rather, capacity and connectivity is provisioned in these phases, which automatically results in provisioning of paths. Path computation phase simply enumerates paths that conform to the provisioned connectivity and capacity.

Each communication path between every node pair has the following pattern.

$$\begin{tikzcd}
\texttt{Cluster1Node} \arrow[d, Leftrightarrow] \\
\texttt{EdgeSwitch} \arrow[d, Leftrightarrow] \\
\texttt{CoreSwitch [}\Leftrightarrow\texttt{ CoreSwitch]} \arrow[d, Leftrightarrow] \\
\texttt{EdgeSwitch} \arrow[d, Leftrightarrow] \\
\texttt{Cluster2Node}
\end{tikzcd}$$

Computing Half Paths

In the last step of the edge design phase, (described in Section 5.3.2.2), every edge-link is associated with a set of cluster node ports. This association determines which node will use which edge-link to communicate with other nodes. Every edge-link connects an edge-switch to a core-switch. Thus, the node to edge-link mapping automatically determines the communication path for a node to the core. The node-to-core path happens to be exactly a half portion of every path between two nodes. A complete path between two nodes can be formed simply by connecting their respective half paths to the core.

If complete-switch-disjoint paths are desired for a cluster to communicate with other clusters, every cluster node is equipped with multiple network ports connected to different switches. There are two symmetrical cores in such a case. Thus, there

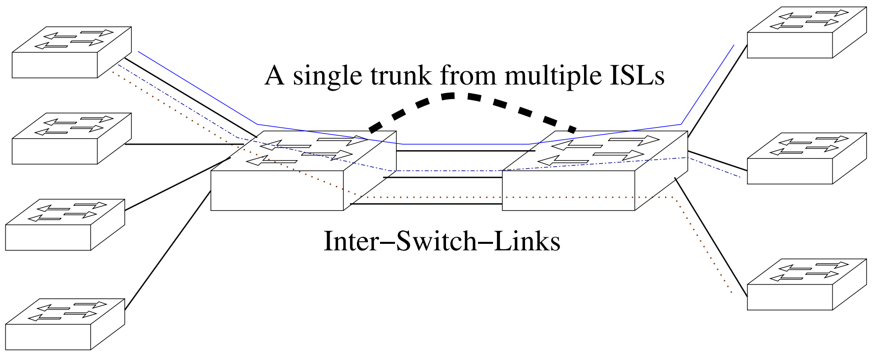


Figure 5.21: *Load balancing over inter-switch-links in the core. When there are multiple inter-switch-links between two core-switches, the load can be balanced by including all the inter-switch-links in roughly equal number of paths. Alternatively, all the inter-switch-links can be combined together into a single trunk.*

are two different paths to each of the cores, for every node. Two complete-switch-disjoint paths are formed by connecting respective paths to each core.

Figure 5.20 shows how half paths can be identified in the topology designed by Cassini.

Connecting Half Paths and Trunking

The complete path between any two nodes is formed by connecting their respective paths to the core. If both the nodes are connected to the same core-switch, then connecting the half paths is a trivial process. The common point for both the half paths is the common core-switch. The complete path is formed by simply augmenting the half paths with each other. If the nodes are connected to different core-switches, however, there is no common point for the half paths. The complete path must be formed by using the inter-switch-links provisioned during the core design phase.

If there is a single inter-switch-link between the core-switches, the complete path is formed using that link. If there are multiple inter-switch-links connecting two core-switches, the path can be formed by using any of the links. Alternatively the inter-switch-links can be combined together to form a single *trunk* that acts as a single logical inter-switch-link [IEE00a]. Trunking has another benefit, reducing loops in the network. Two inter-switch-links between the same pair of switches always form a loop. This loop can be avoided if both inter-switch-links are combined to act as a single link. Figure 5.21 shows how trunking is carried out.

VLAN Grouping

Computing paths between all communicating node pairs is just half the task. A mechanism must enable these paths and make them available for use by the end-nodes. Also, there must exist a mechanism to fail-over from one path to another switch-disjoint path in the event of failure. Cassini uses the same mechanism as Viking [SGNcC04]. The path aggregation algorithm 4.6 described in Chapter 4 is followed to come up with VLAN assignment for every link.

The failure detection and fail-over strategy is also borrowed from Viking. The fault tolerance mechanism described in Chapter 4, section 4.4 is applicable verbatim in SANs designed using Cassini.

5.4 Evaluation of Cassini

The goals of Cassini are to generate an appropriate SAN topology for a given input, devise communication paths between different nodes of communicating clusters, and determine VLAN assignments for different paths.

5.4.1 Efficiency of SAN Design

The effectiveness of the Cassini design algorithms can be measured by the appropriateness of the SAN topology produced for a given input. Appropriateness is a qualitative measure. Though resource cost minimization is one desired factor, it is not the prime focus of topology design. Along with resource cost minimization, there are other goals, such as avoiding cluster fragmentation, conforming to core-edge structure, high availability, etc. These goals aid the manageability of the SAN.

The hypothetical best case scenario for resource cost minimization occurs when there are no resources wasted in the designed SAN. Thus, a simple measure of effectiveness of a SAN design can be performed by comparing utilized and wasted resources. Wasted resources in a SAN are manifested by *over-provisioning* of resources. In an Ethernet SAN there are two primary resources, namely, switch ports and switching bandwidth. If the utilization of available switch ports and switching bandwidth is high, the efficiency of the SAN design can be deemed superior. We examined the extent of utilization and over-provisioning in SANs designed by Cassini, to get an insight into the efficiency of our design algorithms.

We define port utilization of a switch as the percentage of total number of ports actually in use out of the total available ports on the switch. Thus, for a switch with 16 ports, if only 8 of the ports are actually in use, the switch port utilization is 50%. If all of the ports are actually in use, the switch port utilization is 100%. Similarly, we define the bandwidth utilization of a switch as the percentage of total backplane switching bandwidth actually in use out of the total available backplane bandwidth of the switch. If a switch has backplane switching bandwidth of 16 Gbps and the total traffic handled by the switch is only 8 Gbps, its bandwidth utilization is only 50%.

We define the average port and bandwidth utilization of edge-switches as the average of port and bandwidth utilization of each edge-switch. Similarly, we define the average port and bandwidth utilization of core-switches as the average of port and bandwidth utilization of each core-switch.

We evaluated the effectiveness of Cassini at designing small-sized SANs as well

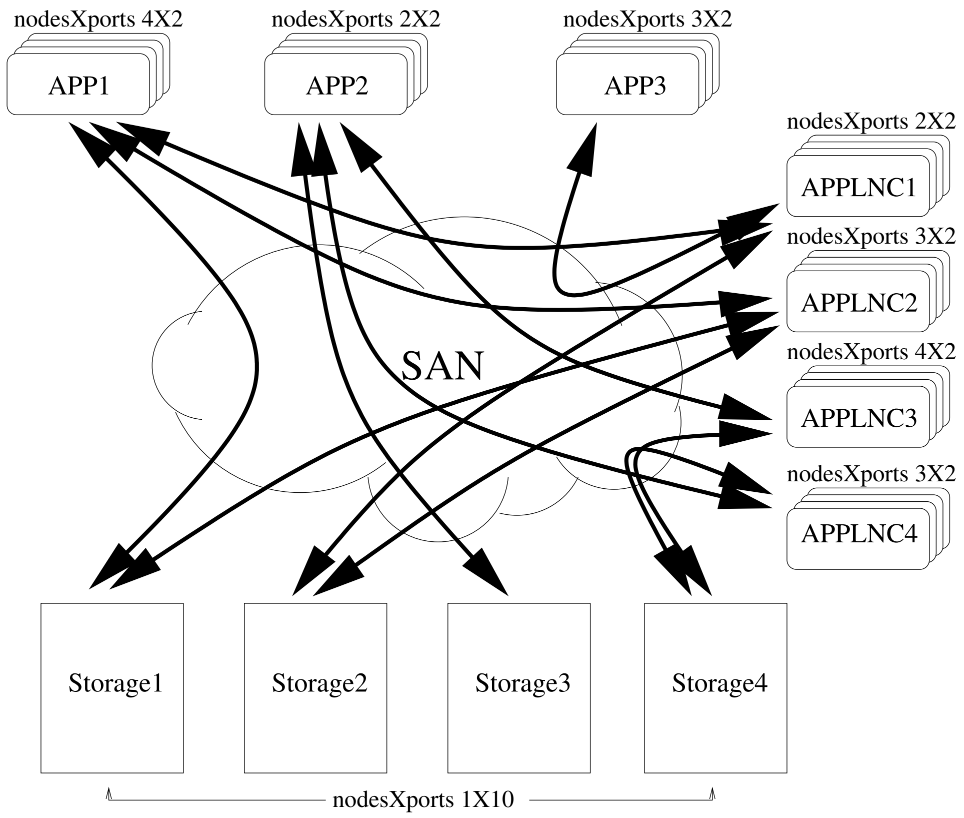


Figure 5.22: *Representation of an input for design of a small-sized SAN. The SAN was designed for 3 application clusters, 4 appliance clusters, and 4 storage nodes. Each cluster consists of multiple nodes with multiple ports. The arrows indicate traffic flows to be accommodated by the designed SAN.*

as large SANs. Small-sized SANs are usually deployed in small test and development environments or SMB (small-medium-business) environment. Small SANs provide storage connectivity to a few nodes/cluster using a small number of storage devices. In large data centers, however, SANs connect several hundred application servers with similar number of storage arrays. The small and large SAN designs we chose, are representative of these two different environments.

5.4.1.1 Designing Small SANs

Figure 5.22 shows representation of a small-sized input for which the SAN design process was carried out. The SAN was designed for a total of 11 clusters. These 11 clusters consisted of 3 application clusters, 4 appliance clusters, and 4 storage clusters. Each cluster contained multiple nodes with multiple ports. The clusters were a mix of master-slave and distributed clusters. The arrows in the Figure indicate traffic flows that were to be accommodated by the designed SAN. There were 12 traffic flows. All were equal in terms of bandwidth requirement. The SAN design was done using small switches. The edge-switches had 16 ports each and a backplane switching capacity of 16 Gbps. The core-switches were slightly larger switches, with 24 ports each and a similar backplane switching capacity of 16 Gbps. Several instances of the SAN were designed for varying traffic requirements. These various instances were analyzed for port and bandwidth utilization.

Figure 5.23 shows the number of switches required to deploy the SAN designed for input sketched in Figure 5.22. The SAN was designed with multipathing support. It required 10 edge-switches, for a low traffic requirement of 80 Mbps per flow. As the traffic flows increased, the number of required switches also increased. For a traffic requirement of 640 Mbps per flow, the number of edge-switches increased to 14 switches. The number of required core-switches ranged from a single core-switch for low traffic, to 3 core-switches for high traffic. Cassini designs a dual-core topology for supporting multipathing. Thus, two isomorphic cores were designed to support complete-switch-disjoint paths between every pair of communicating nodes.

Figure 5.24 shows the utilization of edge-switches in the designed SAN instances. In edge-switches, initial port utilization is steady, around 80%. Bandwidth utilization is low for SANs designed for low traffic and increases linearly with increasing traffic. A fairly high level of port utilization signifies that the limiting

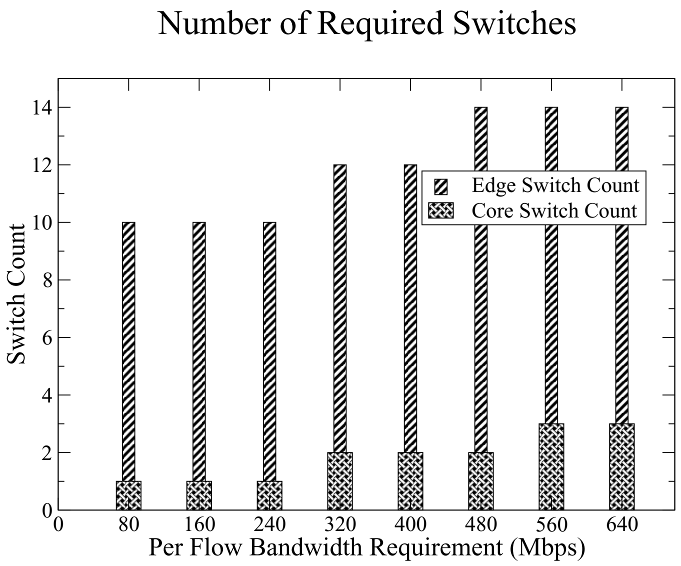


Figure 5.23: *Required number of switches to deploy for a small SAN designed for the input sketched in Figure 5.22. Different SAN topologies were designed for different traffic requirements.*

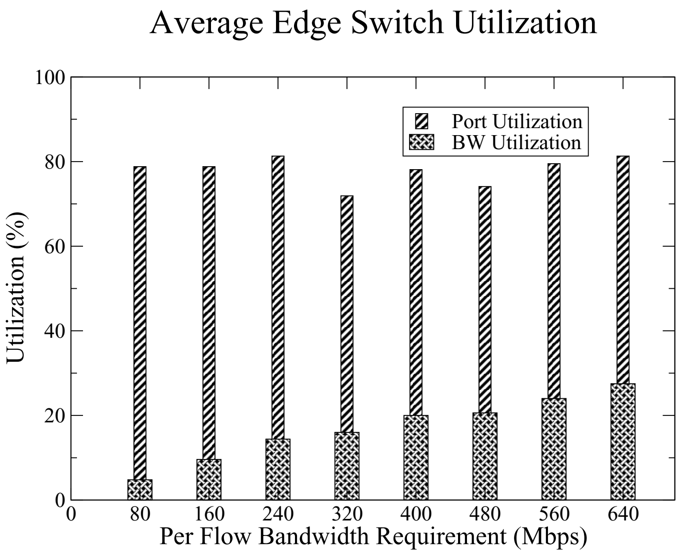


Figure 5.24: *Average edge switch utilization in a small SAN design. Port utilization is around 80%. Bandwidth utilization is low for low traffic and increases linearly with increasing traffic. Fairly high port utilization signifies that the limiting resource in this SAN is switch ports, rather than bandwidth.*

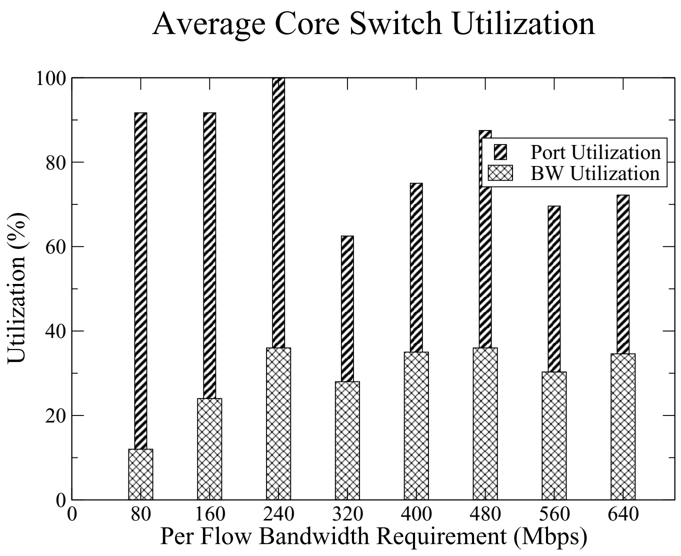


Figure 5.25: *Average core switch utilization in a small SAN design.*

resource in these SAN design instances is switch ports, rather than backplane bandwidth. With increasing traffic the number of required edge-to-core links also increases. These edge-to-core links increase port utilization by a marginal amount. If these additional links cannot be accommodated on existing edge switches, however, additional edge-switches may be required, and thus all available switch ports may not be utilized. This reduces overall switch port utilization. This scenario can be seen in Figure 5.24, as a dip between utilization occurs between 240 Mbps flows and 320 Mbps flows. Figure 5.23 shows the corresponding increase in switch count, from 10 edge-switches to 12 edge-switches.

Figure 5.25 shows port and bandwidth utilization for core-switches. Initial port utilization is fairly high, around 90%. With increasing traffic, the required number of edge-to-core links increases and the utilization goes up to 100%. Further increase in the number of edge-to-core links requires additional core-switches. Introduction of new core switches brings in a significant number of additional ports and a significant amount of additional bandwidth, which cannot be fully utilized. This reduces port utilization as well as bandwidth utilization. These utilization dips can be observed in Figure 5.25. The port utilization design corresponding to traffic flows of 320 Mbps drops to 60%. The port and traffic utilization steadily increases for further design instances and reaches 90% port utilization for the SAN design instance corresponding to 480 Mbps traffic flows. At this point, increased number of edge-to-core links requires an additional core-switch and the port utilization decreases further.

It should be noted that, in a scenario where connectivity dominates capacity, the SAN designed by Cassini maintains a fairly high level of port utilization, from 80% to 90%.

5.4.1.2 Designing Large SANs

Figure 5.26 shows representation of an input which results in a large SAN design. The SAN design was done for 50 clusters. The 50 clusters consisted of 20 application clusters, 20 appliance clusters, and 10 storage nodes. Each cluster contained multiple nodes with multiple ports. In all, there were 416 node-ports connected to the SAN. The designed SAN had to support 300 traffic flows. The first 100 flows were generated by randomly pairing application clusters with appliance clusters. The next 100 flows were generated by randomly pairing appliance clusters with storage nodes. The final 100 flows were generated by randomly pairing application clusters with storage nodes.

The SAN design was done using large switches. The edge-switches had 24 ports each and a backplane switching capacity of 16 Gbps. The core-switches were larger switches with 64 ports each and a backplane switching capacity of 48 Gbps. Multiple instances of the SAN were designed for varying traffic requirements.

Figure 5.27 shows the number of core and edge-switches required to deploy

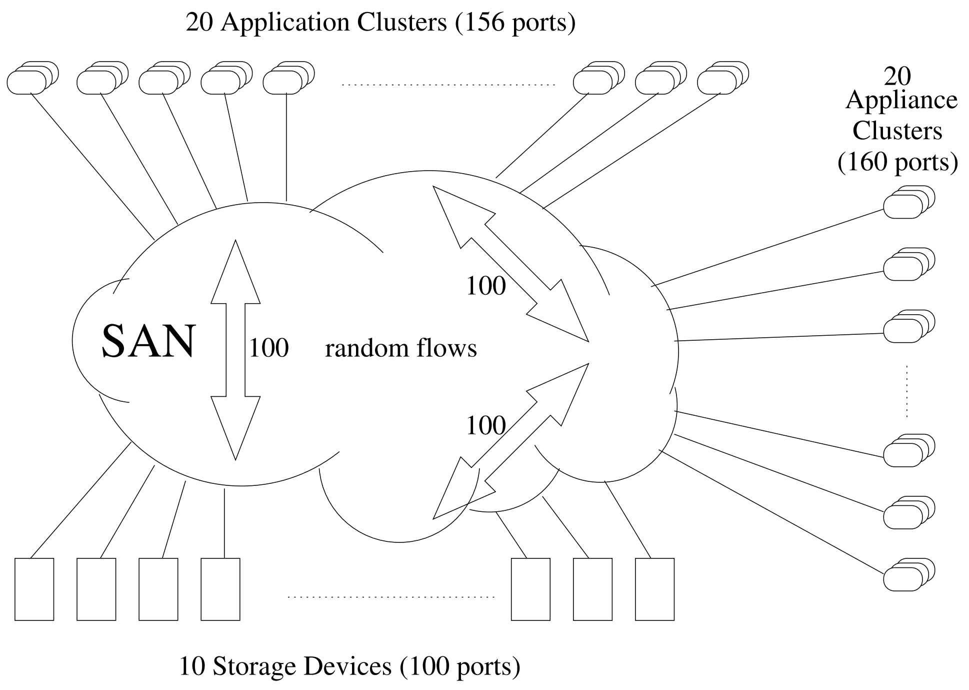

20 Application Clusters (156 ports)
20
Appliance
Clusters
(160 ports)
SAN
100
random flows
100
100
10 Storage Devices (100 ports)

Figure 5.26: *Representation of an input for a large SAN design. The SAN design was done for a large number of clusters communicating with each other. Input included 20 application clusters, 20 appliance clusters, and 10 storage clusters. Each cluster contained multiple nodes with multiple ports. In all there were 416 ports connected to the designed SAN. The designed SAN had to support 300 traffic flows. The flows were generated by randomly pairing clusters from different edges.*

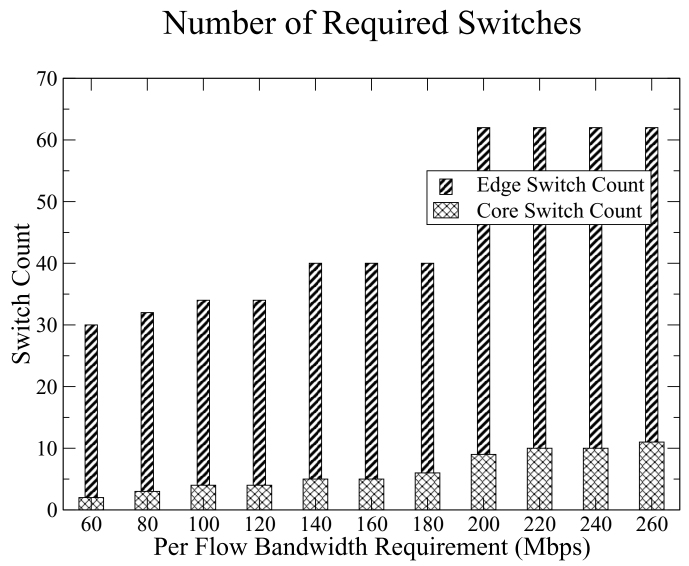


Figure 5.27: *Required number of switches to deploy a large SAN designed for the input represented by Figure 5.26.*

the large SAN designed for the input represented by Figure 5.26. Multiple SAN instances were designed by varying the traffic from 60 Mbps per flow to 260 Mbps per flow, in each direction. These SAN instances were designed with multipathing support. We examined variations in the different SANs designed to analyze the impact of an increase in traffic load on the resulting SAN design. It required 30 edge-switches and 2 core-switches to support 300 flows between 416 ports, with a traffic requirement of 60 Mbps per flow. As the traffic increased, the number of required switches also increased. For traffic of 140 Mbps per flow, the number of edge-switches is 40 and the core-switches is 5. For traffic of 200 Mbps, the number of edge-switches increased to 62. The number of core-switches also increased from 6 to 9. This amounts to a jump of 50% in the required resources.

The reason for this significant increase is explained by analyzing the edge-switch utilization shown in Figure 5.28 and Figure 5.29. Port utilization for edge-switches is consistently high, in the range of 85% to 90%. When the traffic flow

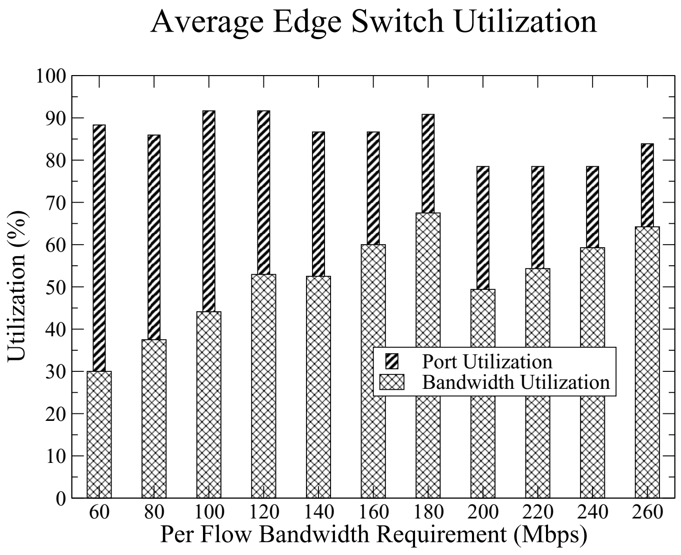


Figure 5.28: *Average edge switch utilization in a large SAN design. Port utilization ranges from 80% to 90%. The bandwidth utilization increases with traffic and ranges from 30% to 70%.*

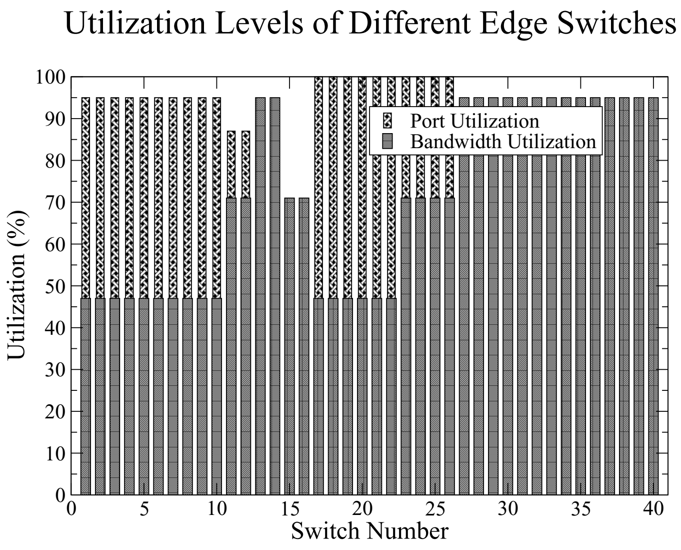


Figure 5.29: *Edge switch utilization for a SAN designed for input represented by Figure 5.26, with a traffic flow requirement of 190 Mbps. Almost all of the edge-switches in this SAN are fully saturated, with little spare ports and capacity available.*

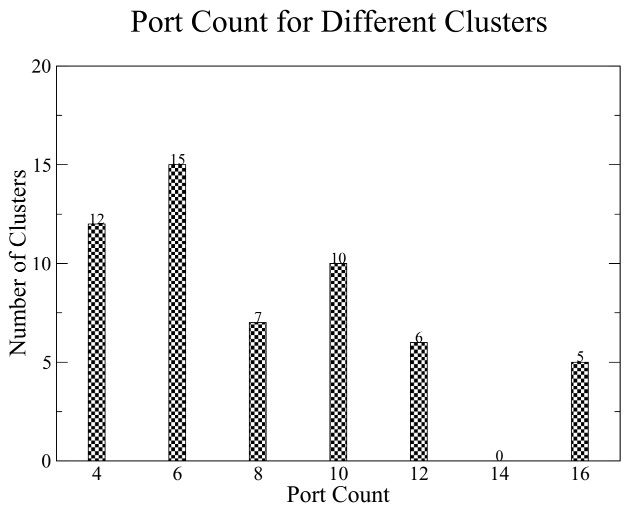


Figure 5.30: *Distribution of ports for different clusters. The 416 ports in large SAN input are not evenly distributed among all the clusters. Ports-per-cluster vary from 4 ports to 16 ports. Bandwidth utilization of edge-switches connected to clusters with low port count is high and it is low for edge-switches connected to high port count.*

requirement increases from 180 Mbps to 200 Mbps, almost all edge-switches need additional edge-to-core links. Incidentally, at this point the average bandwidth utilization is around 70%, which is also very high. This high port and bandwidth utilization signifies that almost all of the switches are saturated, with little spare ports and capacity left. Figure 5.29 illustrates bandwidth and port utilization of every edge-switch in the SAN instance designed to support a traffic flow requirement of 190 Mbps. Thus, when the traffic requirement in the SAN increases from 190 Mbps per flow to 200 Mbps per flow, a significant number of additional edge-switches are needed to support the increased bandwidth and connectivity. This increase in the number of edge-switches raises the required number of core-switches as well.

Figure 5.31 shows the port and bandwidth utilization for core-switches. Port utilization is steady, around 75% to 85%, and bandwidth utilization ranges from 45% to 65%. This level of utilization is fairly high. A high utilization is a desirable

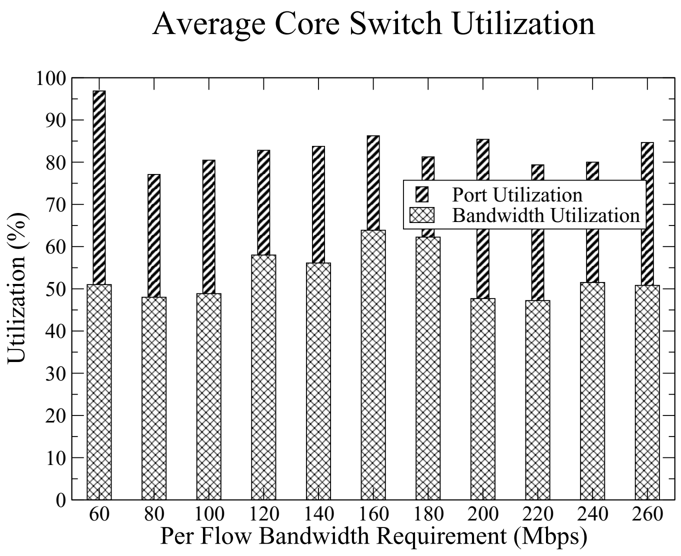


Figure 5.31: *Average core switch utilization in a large SAN design. Port utilization is steady around 75% to 85% and bandwidth utilization ranges from 50% to 65%.*

attribute of any SAN.

Figure 5.29 illustrates utilization of every edge-switch in the SAN instance designed to support a traffic flow requirement of 190 Mbps. It shows that almost every edge-switch is either port saturated or bandwidth saturated. It also shows that edge-switches are not utilized in a uniform fashion. This behavior is intriguing considering that the traffic pattern between clusters is pretty uniform. The reason for uneven utilization of switches can be explained by Figure 5.30, which shows that different clusters have different port count. Out of 50 clusters in large SAN input, 12 clusters had only 4 ports, the port-count for clusters varied from 4 ports to 16 ports each. The uniform traffic from all these clusters was distributed over different number of ports. This variation in port-count per-cluster resulted in high bandwidth utilization of edge-switches connected to clusters with low port-count. The port utilization for edge-switches connected to high port-count clusters was high and bandwidth utilization was low.

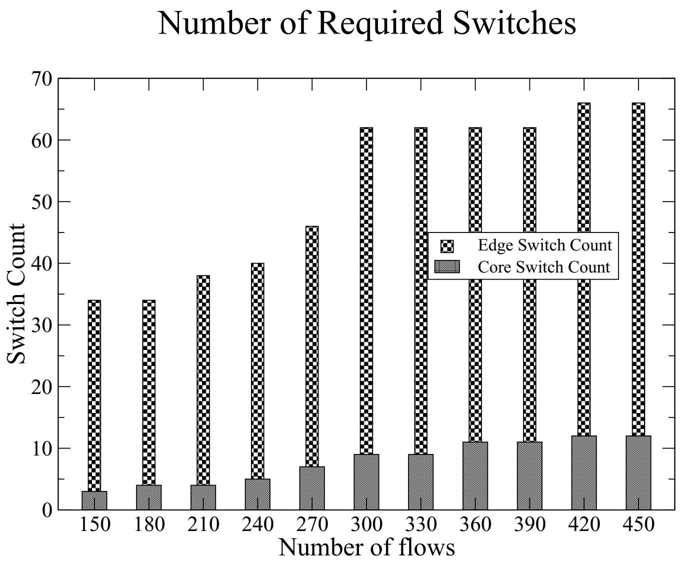


Figure 5.32: *Required number of switches to deploy a large SAN designed with varying number of flows for the input represented by Figure 5.26.*

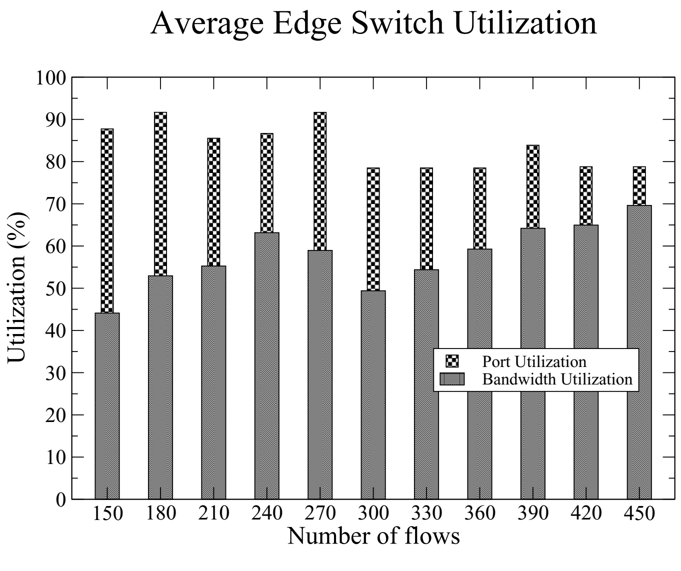


Figure 5.33: *Average edge switch utilization in a large SAN designed with varying number of flows for the input represented by Figure 5.26. Port utilization ranges from 80% to 95%. The bandwidth utilization increases with number of flows and ranges from 45% to 70%.*

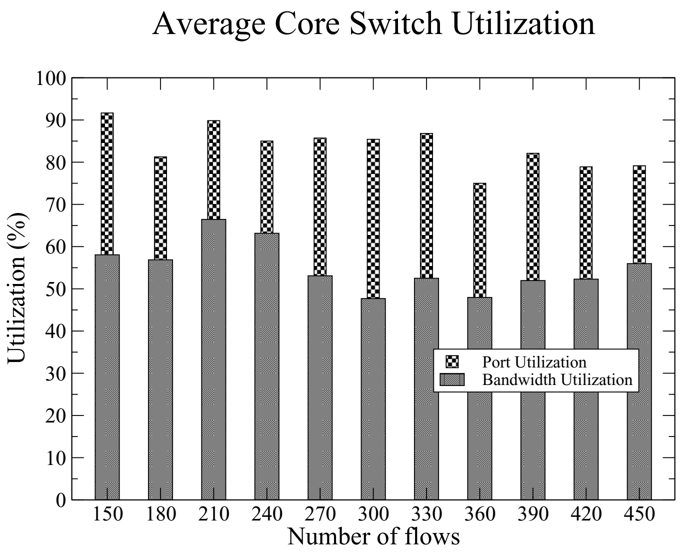


Figure 5.34: *Average core switch utilization in a large SAN designed with varying number of flows for the input represented by Figure 5.26. Port utilization is steady around 75% to 90% and bandwidth utilization ranges from 50% to 65%.*

We examined the impact of number of flows on SAN design. In previous SAN design instances, the number of flows were set at constant 300 flows and we varied the bandwidth per flow. In another set of experiments we varied the number of flows from 150 flows to 450 flows while keeping the bandwidth requirement constant at 200 Mbps per flow.

Figure 5.32 shows the required number of edge-switches and core-switches for such a SAN design. The number of switches varied from 30 edge-switches and 3 core-switches to 66 edge-switches to 12 core-switches when the number flows were varied from 150 flows to 450 flows.

Figure 5.33 shows utilization levels for edge-switches in these SAN design instances. Port utilization ranges from 80% to 95%. Bandwidth utilization increases with number of flows and ranges from 45% to 70%.

Figure 5.34 shows utilization levels for core-switches. Port utilization is steady around 75% to 90% and bandwidth utilization ranges from 50% to 65%.

This behavior is not much different from traffic variation in SAN designs.

5.4.1.3 Bandwidth Utilization

In all of the utilization related measurements, the SAN designs were port limited rather than bandwidth limited. Figure 5.24, Figure 5.25, Figure 5.28, and Figure 5.31 all show that bandwidth utilization of switches is lower than port utilization. There are two reasons for this observed behavior. The utilization described in all the measurements *average* port and bandwidth utilization. Some switches indeed get saturated either by bandwidth or by ports. Figure 5.29 shows different levels of utilization for each switch in large SAN design. Once a switch gets saturated any additional traffic would require introduction of a new switch. Both bandwidth and port utilization for new switches are significantly low. This low utilization of the new switches affects the average port and bandwidth utilization of the entire SAN. Another reason for low bandwidth utilization is the relationship between port bandwidth, switch size, and the backplane switching capacity. For switches used in small SAN design, 16 Gbps of backplane capacity is sufficiently high for 16 ports each with 1 Gbps link speed. For these edge-switches to become bandwidth saturated the traffic on every port has to be 1Gbps. It is possible for bandwidth utilization to exceed port utilization, but only in cases where backplane switching capacity is insufficient to handle traffic from all the ports.

The bandwidth utilization in large SAN designs is higher than bandwidth utilization in small SAN designs because of switch characteristics. The switches used in designing large SANs are larger switches in terms of ports but the backplane switching capacity for these switches is same as that of small switches. Edge-switches used in designing large SANs had 24 ports but 16 Gbps backplane capacity. This is why bandwidth utilization in large SAN design was higher than small

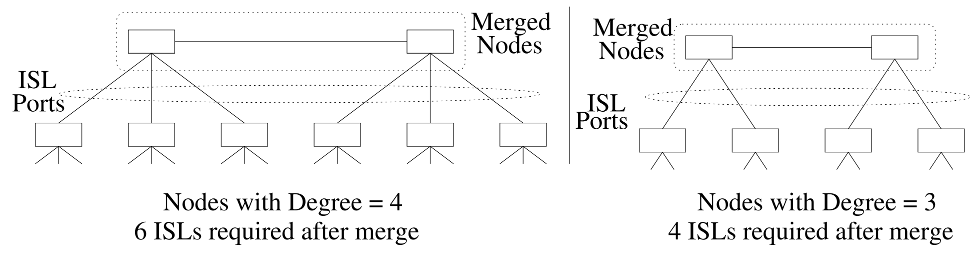


Figure 5.35: *Required inter-switch-links and the degree of cluster connectivity. Merging clusters with a higher degree of connectivity requires more ports for inter-switch-links.*

SAN design.

5.4.2 Analysis of Core Design

5.4.2.1 Impact of Communication Degree of Clusters

An important factor that determines the efficiency of core-switch merging is the average degree of connectivity of clusters. The degree of a cluster is defined as the number of other clusters with which a given cluster communicates. If the average degree of the clusters in a SAN is high, it becomes more difficult to accommodate all clusters using small core-switches. The reason behind this difficulty is simple. The number of additional ports required to merge two core-switches is directly dependent on the degree of the clusters using them. With clusters of a higher degree, a larger number of inter-switch-links are required. Figure 5.35 illustrates this situation. Two switches cannot be merged together if their inter-switch-links cannot be accommodated after the merge. For small switches, the likelihood of this situation occurring is greater with clusters of a higher degree of connectivity. When a switch cannot be merged with any other switch, an internal fragmentation of unused ports takes place and port utilization goes down. With larger switches this situation is less likely.

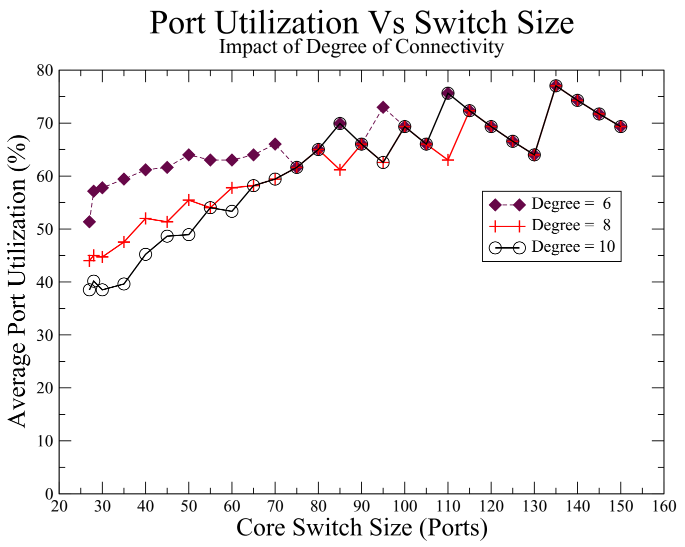


Figure 5.36: *Impact of the degree of cluster connectivity and switch size on port utilization. With a higher degree of connectivity port utilization is lower for smaller switch sizes, because of internal fragmentation. The impact is less pronounced for larger switch sizes.*

We analyzed the impact of the degree of connectivity and switch size on port utilization. We designed different SAN instances for the large SAN represented by Figure 5.26. In choosing the traffic flows, we controlled the degree of cluster connectivity. Each traffic flow was set to consume a bandwidth of 260 Mbps. SAN instances were designed with the degree of cluster connectivity set to 6, 8, and 10. The core-switch size was varied from 25 ports to 150 ports.

Figure 5.36 shows how port utilization is affected by the degree of connectivity. For SANs designed with small core-switches with 30 ports, port utilization is around 40% for an average degree of connectivity of 10. In contrast, port utilization is at 60% when the average degree of connectivity is 6. Port utilization improves with increased switch sizes. The improvement is more significant for a higher degree of connectivity than for a lower degree of connectivity. When the core-switch size reaches around 80 ports, the difference between port utilization for different

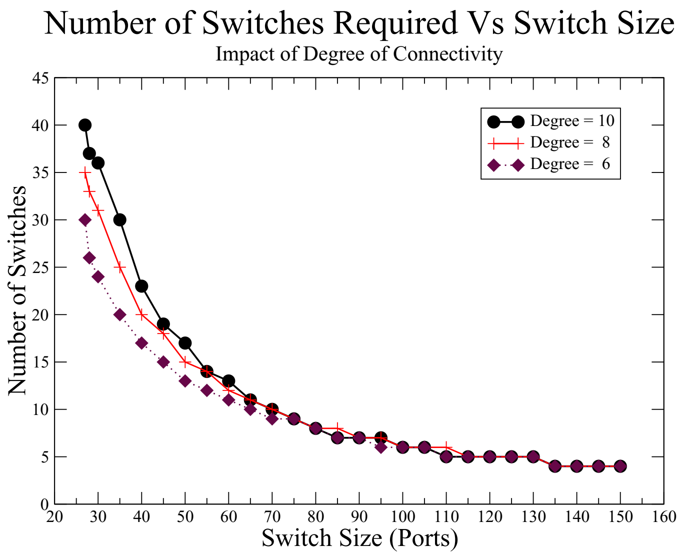


Figure 5.37: *Impact of the degree of cluster connectivity and core-switch size on the required number of switches. A higher degree of connectivity reduces the overall port utilization for small switches. Reduced port utilization results in a larger number of required switches. The impact is less pronounced for larger switches.*

degrees of connectivity becomes minimal and port utilization varies between 70% and 80%.

Figure 5.37 shows the impact of the degree of connectivity on the number of required core-switches. Since the port utilization is low when core-switches are small and the degree of connectivity is high, the number of required switches is larger. When core-switch size is around 30 ports, around 40 switches are required, corresponding to a connectivity degree of 10. The same configuration requires 35 switches, with a connectivity degree of 8, and 30 switches with a connectivity degree of 6. The number of required switches reduces significantly with an increase in core-switch size. Similar to port utilization, the reduction in the number of required switches is more significant for a higher degree of connectivity. When switch size reaches around 80 ports, the effect of the degree of connectivity largely disappears.

Figure 5.36 and Figure 5.37 show that the degree of connectivity of clusters affects the required number of core-switches. If the degree of connectivity is high, it is preferable to choose larger core-switches, to avoid fragmentation. If the degree of connectivity is very low, SAN design can be done by choosing small core-switches, keeping infrastructure costs low.

5.4.2.2 Greedy and Locality Based Core Merge

The core-switch merge process can be done in two different ways. One way is locality-based merge and the other way is greedy merge. Computationally these are similar in complexity. We analyzed efficiency of both algorithms.

Figure 5.38 shows a comparison of locality based and greedy core merge. The port utilization and bandwidth utilization for locality-based merge is higher than greedy merge, by 1% to 2%. When comparing there two algorithms, however, the higher utilization in case of locality-based merge does not mean that the performance of locality-based merge is better. On the contrary, the higher port utilization

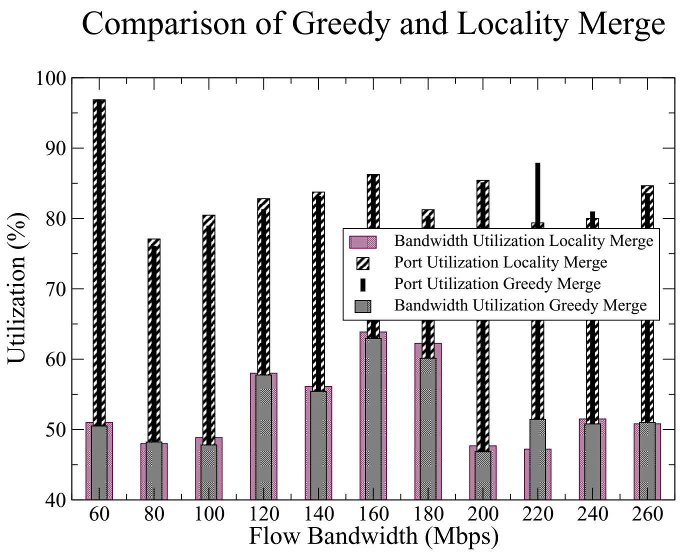


Figure 5.38: *Comparison of locality-based and greedy core merge. There is only a small difference between port and bandwidth utilization achieved by either merge algorithm.*

signifies that more inter-switch-links in the core are required when locality-based merge is performed. These inter-switch-links carry traffic from one core switch to another. Traffic over inter-switch-links increases the bandwidth used on the connected switches and increases overall bandwidth required.

The data point corresponding to 220 Mbps flows in Figure 5.38 is interesting. This is the only data point where the port and bandwidth utilization for greedy merge is higher than for locality-based merge. For this input the locality based merge resulted in 11 core-switches, while, greedy merge required 10. Because of this tighter merge and fewer switches, the port and bandwidth utilization was greater for greedy merge.

Both greedy merge and locality merge perform almost equally well. Greedy merge gives slightly better results than locality-based merge. This behavior is counter-intuitive. The reason why locality-based merge does not perform better than greedy merge can be explained by focusing on the dynamic variation of inter-switch-traffic during merge. After a core switch is identified as a pivot-switch, it is merged with one of its neighboring switches with which it has highest inter-switch-traffic. This results in localizing a significant amount of traffic on a single (merged) switch. As shown in Figure 5.18, the switch load and the used port count of the merged switch are lower than earlier. The pivot-switch does not remain an ideal candidate for subsequent merges as the advantage of inter-switch-traffic of pivot-switch reduces with every merge. On the contrary, the greedy merge always chooses candidate switches which have the highest inter-switch-traffic and can be merged together. This is like identifying a pair of new pivot-switches after every merge. The greedy algorithm is more responsive to change in measure inter-switch-traffic that takes place after every merge. Further, the greedy algorithm attempts to merge switches by considering every possible switch pair in the core. Even when there is no inter-switch-traffic between two core switches, the greedy algorithm tries to merge them. Locality based merge does not try to merge switches if they are not immediate neighbors of each other. This aggressive behavior and quick adaptation to change in inter-switch-traffic makes greedy algorithm perform better than locality-based merge.

5.4.3 Analysis of Edge Design

5.4.3.1 Multipathing Resources

Multipathing ensures that there exist a pair of complete-switch-disjoint paths between every communicating node pair. These complete-switch-disjoint paths provide protection against switch and link failures. Support for multipathing requires that every node be equipped with at least 2 ports which can be connected to different edge-switches. These edge-switches are connected to different core-switches.

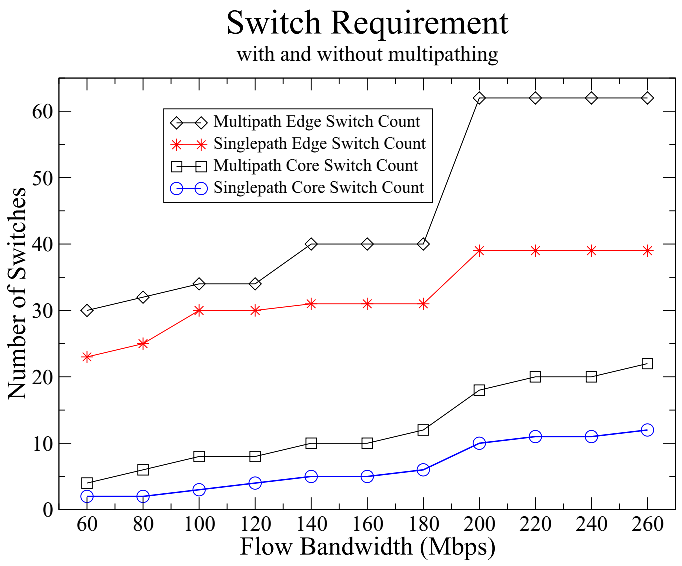


Figure 5.39: *Additional switch requirement to support multipathing. Edge switch requirement is higher for multipathing support. Core switch requirement for both the cases is almost the same. However, multipathing requires two identical/isomorphic cores. This doubles up the core-switch requirement.*

Thus complete redundancy of paths is established.

The number of edge-switches required to support multipathing is naturally higher than if the multipathing is not supported. Further, Cassini uses dual core topology to support multipathing. A dual core topology requires provisioning two identical/isomorphic cores. This doubles the required number of core-switches.

Figure 5.39 shows a comparison of the number of switches required with and without multipathing support. The input was the same as in previous experiments. There were two sets of experiments. One set of experiments was run to design SANs with multipathing and the other to design SANs without multipathing.

All connected ports on an edge-switch can be divided into two categories, namely, cluster-facing ports and core-facing ports. Ports that are connected to cluster nodes are cluster-facing and ports that carry edge-to-core links are core-facing ports. The number of cluster-facing ports for a specific input of a SAN design is always constant. The number of core-facing ports is determined by the connectivity between edge and core-switches. When the traffic load in a SAN is low, the additional number of edge-switches required for multipathing is solely governed by the additional number of core-facing ports required to connect the edge-switches to the second core. When bandwidth requirement in SAN increases, the number of required edge-to-core links also increases, while the cluster-facing ports remain unchanged in number. Thus the increase in the number of edge switches with an increase in traffic is actually due to the increased number of edge-to-core links required to support the additional traffic.

The difference in the required number of switches increases with increased traffic. For a flow bandwidth of 60 Mbps in the experiments, the number of switches for multipathing is around 30, while without multipathing it is 23. This amounts to a difference of 25%. Whereas, at a flow bandwidth of 260 Mbps these numbers are 62 and 39, a difference of around 40%.

These measurements provide an insight into the multipathing overhead of a SAN. For small SANs multipathing does not require too many resources. The resource requirement increases significantly with increase in SAN size. It is more important to have multipathing in larger SANs because the mean time between failures for larger SANs is much lower. Failure in any portion of a SAN will ripple through the entire SAN and affect the performance of every connected cluster. Multipathing ensures that failures in a SAN are contained and do not ripple through the entire network.

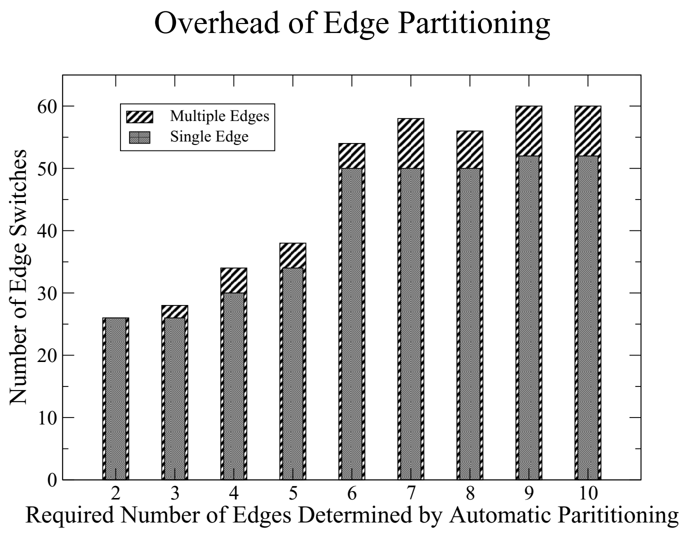


Figure 5.40: *Overhead of automatic edge partitioning. This bar graph shows the overhead of automatically partitioning clusters into multiple edges vs provisioning them into a single edge. Partitioning clusters into multiple edges increases the required number of edge-switches by 5% to 10%.*

5.4.3.2 Overhead of Edge Partitioning

Cassini provides a facility to automatically partition clusters into multiple edges. Automatic edge partitioning can be used to separate out storage initiators from storage targets.

In some cases, the number edges in a SAN are pre-determined based on SAN structure. For example, in small and large SAN design inputs the number of edges were pre-determined to be 3 based on the communication pattern of clusters in these inputs.

There can be different organizational and geographical constraints which result in multiple edges for SAN design. Cassini can deal with all these structural constraints by performing automatic cluster-to-edge assignment. We synthesized several inputs with 50 clusters and multiple traffic flows. We studied the overhead of automatic cluster-to-edge assignments when it resulted in different number of edges, varying from 2 edges to 10 edges.

Figure 5.40 shows the overhead of automatically partitioning clusters into multiple edges, as opposed to keeping them in a single edge. When automatic partitioning results in 2 or 3 edges, the required number of edge-switches does not vary much from the single edge case. At 4 partitions and beyond, however, a steady overhead of an extra 5% to 10% edge-switches is observed.

The overhead of automated partitioning is limited. Partitioning clusters into multiple edges has the advantage of reducing overall network size at one physical location. These measurements establish that partitioning can be done without increasing the network cost significantly.

5.4.4 Trunking and VLAN Grouping

Cassini uses trunking (link aggregation) to combine together all inter-switch-links between a pair of switches. This aggregation has two advantages. First, the task of load balancing traffic between two switches over multiple ISLs is delegated to the switches. Switch-level load balancing is always better than explicit load balancing by communicating nodes. Next, parallel ISLs between the same pair of switches always form a loop. This loop has to be broken by including the ISLs in different VLANs. If there are multiple parallel ISLs, there are more loops, so the number of required VLANs also increases. Ethernet switches can support a maximum of 4096 VLANs. If there are too many parallel ISLs, the VLAN requirements may become untenable.

To study the ISL and VLAN requirements and the advantage of trunking support in Cassini, we examined various SAN instances. The input for the SAN designs was the same as previous inputs described in Figure 5.26. We studied ISL and VLAN requirement with both multipathing support and lack of that support (singlepath). For

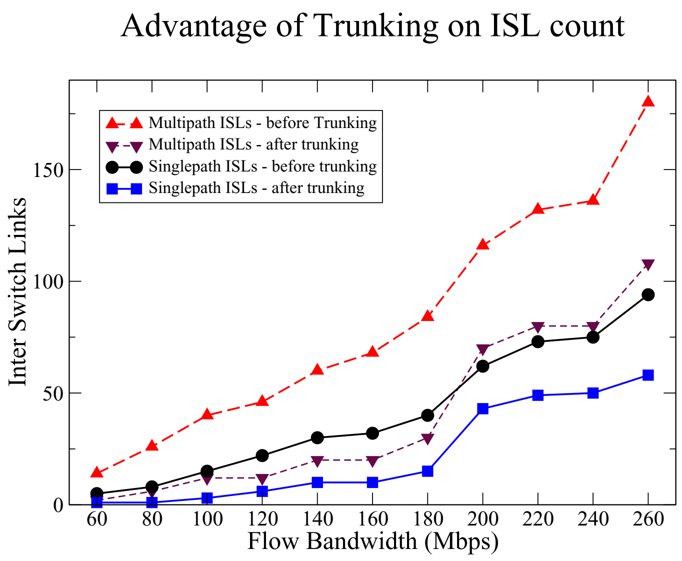


Figure 5.41: *Reduction in ISL count due to trunking. This reduction does not signify any change in the required number of physical ISLs. Rather it signifies a reduction in the number of loops in the SAN topology.*

300 input flows there were 18772 exclusive paths between different communicating nodes. With multipathing support there were 37544 paths.

The number of ISLs in a network depends on two factors, namely the number of switches and the amount of traffic across the switches. Both factors were controlled by varying the flow bandwidth between clusters. SAN instances were designed for varying bandwidths.

Figure 5.41 shows the count of ISLs in various SAN instances. For multipathing, two identical cores are required. Dual-core topology increases the number of ISLs in a SAN significantly. Comparatively, the number of ISLs in the multipathing case is almost double the ISL requirement in the singlepath case. The ISL requirement with multipathing ranges from 14 ISLs, for a low traffic of 60 Mbps per flow, to 180 ISLs, for a traffic of 260 Mbps per flow. In contrast, the ISLs required for singlepath support vary from 5 ISLs to 94 ISLs, for the same traffic

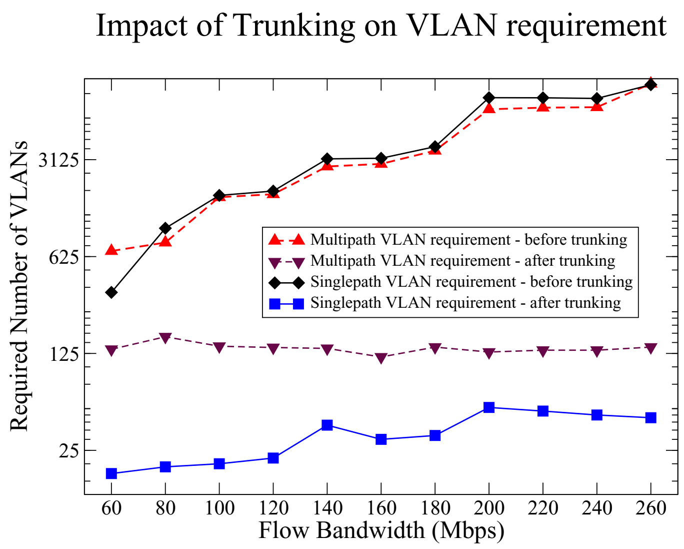


Figure 5.42: *Impact of trunking on required number of VLANs. Trunking eliminates significant number of loops in the core. This gives a better result for path aggregation into VLANs.*

input. Trunking converts all parallel ISLs into a single logical link. After trunking, the number of logical ISLs in the multipathing case varies from two logical links, for 60 Mbps traffic flows, to 108 logical links, for 260 Mbps traffic flows. For the singlepath case, the number of logical links reduces to a single logical link for low traffic and increases up to 58 logical links for high traffic.

Trunking has a desirable impact on the number of required VLANs. Figure 5.42 shows the required number of VLANs, for both the multipathing and singlepath cases. When trunking is not used, the required number of VLANs is very high. For a small SAN with only 5 ISLs, the number of required VLANs is around 350. This increases to 3881 for a traffic of 180 Mbps per flow. After this, the designed SAN size grows significantly and the number of VLANs increases beyond 8000. This is clearly untenable. When trunking is used, the number of required VLANs drops drastically. For SANs designed with multipathing support, the VLAN number varies between 118 and 140 VLANs. For SANs without multipathing support, this number ranges from 17 to 43. Such a small number of VLANs can easily be supported by all Ethernet switches.

With trunking support, the number of required VLANs is drastically reduced, such that a SAN can be deployed using a VLAN-based Viking-like multi-spanning-tree Ethernet architecture. If trunking is not used, the scalability of SANs is not possible, because the required number of VLANs becomes very large. There are two problems with a large number of VLANs. One is the limitation on number of VLANs. Ethernet supports a maximum of 4096 VLANs. A deployment of more than this is impossible. Further, each VLAN requires an independent spanning tree. Having a large number of spanning trees has its own overhead. Thus, limiting the total number of VLANs by using trunking is a significant benefit to SAN deployments.

5.4.5 Summary

We evaluated Cassini for designing small and large SANs. Our evaluation shows that Cassini SAN designs are efficient and produce limited over-provisioning of resources. The Cassini design process can scale from small SANs to large SANs. Evaluation of the Cassini SAN design process by using synthetic inputs provides an insight into the factors that influence the SAN design process. For instance, using the Cassini designer we explored the impact of the degree of connectivity of clusters on SAN costs for various switch sizes. Cassini provides a design mechanism for multipathing and lets us explore the cost of providing redundancy. We experimentally verified that Cassini does automatic partitioning of clusters into multiple edges without inducing significant overhead. Cassini also uses trunking effectively to reduce the overall requirement of VLAN space. Cassini provides details about trunking and VLAN assignment for the final deployment of a SAN using a Viking-like architecture.

Probably one of the most significant advantage of Cassini is that it allows SAN designers to efficiently answer several “What if?” questions by running a simulation. These questions could not be answered during manual design process.

Chapter 6

Conclusion

6.1 Summary of the Dissertation

The primary tenet of this dissertation is that switched Ethernet technology has reached a level of maturity that enables it to be the exclusive solution for several networking requirements other than LAN. We argue that Ethernet is no longer merely a LAN technology. It is an all-encompassing universal networking technology with applicability to metro networks, cluster networks, storage networks, and even super computing networks.

Unfortunately, the universal applicability of Ethernet is not always obvious. Ethernet does have certain drawbacks that prevent it from displacing other established technologies. For instance, lack of a fine-grained path selection mechanism and spanning tree issues in Ethernet networks are major barriers to the application of traffic engineering technology. This restriction has been a limiting factor for Ethernet deployment in core carrier networks. Though Ethernet popularity is increasing, it is from the network access perspective and not from the core service perspective.

We showed that Ethernet has enough configurable features that address its shortcomings. We can make Ethernet applicable to any networking requirement if these features are used intelligently and automatically. In this dissertation, we described how the configurable VLAN mechanisms in modern Ethernet switches can support MPLS-like load-balanced path selection, providing multi-fold network-throughput gains. When combined with status monitoring in Ethernet, this fine-grained path selection can provide sub-second level fault tolerance, which is a required feature in core metro networks. We also discussed how rate limiting features in Ethernet switches can provide QoS guarantees in Ethernet deployments.

We also extended the same VLAN-based traffic engineering mechanism to Storage Area Networks. We also identified topology design as an important but oft-ignored issue for SANs. To facilitate the applicability of Ethernet in the SAN arena, we described an approach to designing Ethernet SAN topologies in an automated manner. We developed Cassini, a tool which can aid SAN designers in topology planning and deployment. We verified the efficacy Cassini by using synthesized inputs and studying the resultant SAN topologies.

To summarize, the major research contributions from this dissertation are:

- A VLAN-based path selection mechanism for metro Ethernet networks. This path selection mechanism enables MPLS-like traffic engineering functionality in Ethernet networks by circumventing the limitations of the spanning tree based switching of Ethernet networks.

- The VLAN-based path selection enables proactive switch-and-link-disjoint backup path provisioning, providing a high degree of fault-tolerance to switch or link failures.

- We discussed how rate limiting features of Ethernet switches can be used to isolate different traffic flows from one another, in order to provide QoS in

Ethernet networks.

- We developed and analyzed an automated SAN topology design-tool which can aid SAN designers devise efficient Ethernet SAN topologies with a high degree of resource utilization.

These research contributions demonstrate that Ethernet switches have configurable features that can be controlled programmatically to make Ethernet technology a useful technology for any networking application.

6.2 Future Research

In terms of a broader research perspective, we believe there are many more high-level functions that can be composed from Ethernet's basic configurable control mechanisms. One of the future directions for this work is to explore and build high-level functionality that can enhance large-scale networks as a whole. We addressed applicability of Ethernet to MAN and SAN networks. In the context of super computing networks or specialized industrial SCADA (Supervisory Control and Data Acquisition) networks, the issues are slightly different. These issues are managing latencies, traffic prioritization, lossless packet forwarding, etc. One can identify key Ethernet issues in these specialized networks and develop Viking-like mechanism to solve pertinent problems.

Cassini is a proof-of-concept automated SAN designer. There are many issues that remain to be addressed in Cassini. Cassini designs only the initial topologies for SANs. Over time, the constituents of a SAN change. Storage devices get upgraded, the cluster composition changes, the application traffic flows change, and old devices and clusters get decommissioned. For any change in SAN usage, a SAN redesign from scratch is not warranted. One can identify stumbling blocks during

SAN upgrade process and explore efficient SAN upgrade mechanisms that are least disruptive and give maximum utilization of already deployed SAN resources. This task is harder than initial topology design and deployment Cassini.

SAN topology design when virtual machines are used is another important research direction. Virtual machines can migrate from one physical machine to another. This migration changes the traffic characteristics in a SAN. The issues involved in designing SANs, that are adaptive to traffic changes, are of great interest to data center environments. Given the trend of increasing virtualization of resources, such an automation tool would prove beneficial to SAN designers.

Network convergence in data centers is a burgeoning research area. While Fibre Channel over Ethernet tries put forward a migration path from FC SANs to Ethernet SANs, Data Center Ethernet goes one step further and tries to address the convergence of LAN, SAN, and High Performance Computing (HPC) applications. The converged data center networks require high-performance and low-latency guarantees, lossless network operation, high fault-tolerance, traffic prioritization, and cross layer optimizations with higher-layer protocols. Basic features to support all these functionalies exist in Ethernet switches even today. The research challenge is to devise configuration mechanisms that are suitable for converged data center environments. For example, low latency guaratees for HPC can be provided by shortest path switching using Viking-like VLAN-based switching. But in converged environment not all traffic is latency sensitive. Thus differentiation of traffic in data center is an important requirement. A converged environment often places conflicting requirements on the network infrastructure. Identifying these requirements of Data Center Ethernet and addressing them while staying within the contraints of existing Ethernet capabilities is an interesting and important research area.

Bibliography

[Ash01] G. Ash. Traffic Engineering & QoS Methods for IP-, ATM-, & TDM-Based Multiservice Networks. Internet Draft, Oct 2001.

[BBC$^+$98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. IETF RFC 2475, Dec 1998.

[BC89] R. Ballart and Y.-C. Ching. SONET: now it's the standard optical network. *IEEE Communications Magazine*, 27(3):8–15, Mar 1989.

[BCF$^+$95] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet: A Gigabit-per-Second Local Area Network. volume 15, pages 29–36, 1995.

[BGM$^+$99] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. A resource query interface for network-aware applications. In *Cluster Computing*, volume 2, pages 139–151, 1999.

[cis] Data Sheet: Cisco Catalyst 2960 Series Switches. URL: http://www.cisco.com last accessed Dec 21 2007.

[Cla03] Tom Clark. *Designing Storage Area Networks: A Practical Reference for Implementing Fibre Channel and IP SANs*. Addison-Wesley, 2nd edition, Jul 2003.

[Cor03] Alcatel Corporation. Enabling Profitable Metro Ethernet Services: The Next Step in Metro Networking. Alcatel Whitepaper, 2003.

[Epp98] D. Eppstein. Finding the k shortest paths. volume 28(2), pages 652–673, 1998.

[ES95] R. Elbaum and M. Sidi. Topological design of local area networks using genetic algorithms. In *IEEE INFOCOM*, 1995.

[FFC70] H. Frank, I.T. Frisch, and W. Chou. Topological Considerations in the design of the ARPA Computer Network. In *AFIPS Conference Proceedings*, volume 36, pages 581–587, 1970.

[FFCS71] H. Frank, I.T. Frisch, W. Chou, and R.V. Slyke. Optimal Design of Centralized Computer Networks. *Networks*, 1:43–57, 1971.

[Fib07] Fibre Channel Industry Association. Fibre Channel over Ethernet in the Data Center: An Introduction, 2007. URL: http://www.fibrechannel.org last accessed 19th June 2008.

[FKSS98] W. Feng, K. Kandlur, D. Saha, and K. Shin. Adaptive packet marking for providing differentiated services on the Internet. In *International Conference on Network Protocols*, Oct 1998.

[For02] Metro Ethernet Forum. Metro Ethernet Networks — A Technical Overview. Metro Ethernet Forum Whitepaper, Jul 2002.

[For03] Metro Ethernet Forum. Metro Ethernet Services — A Technical Overview. Metro Ethernet Forum Whitepaper, 2003.

[Fox75] B.L. Fox. k-th shortest paths and applications to probabilistic networks. *In ORSA/TIMS Joint National Mtg.*, 23:B263, 1975.

[GDS98] R. Garcia, J. Duato, and J. Serrano. A New Transparent Bridge Protocol for LAN Internetworking Using Topologies with Active Loops. pages 295–303, 1998.

[Ger73] Mario Gerla. The Design of Store-and-Forward Networks for Computer Communications. Ph.D. Dissertation, School of Enginnering and Applied Sciences, UCLA, Jan 1973.

[GK77] Mario Gerla and Leonard Kleinrock. On the Topological Design of Distributed Computer Networks. *In IEEE Transactions on Communications*, 25:48–60, 1977.

[Gop03] Kartik Gopalan. Efficient network resource allocation with QoS guarantees. Technical Report TR-133, Experimental Computer Systems Labs, Department of Computer Science, State University of New York at Stony Brook, March 2003.

[HB06] Dave Hitz and Akshay Bhargava. A Storage Networking Appliance. Technical report, NetApp, Feb 2006. http://media.netapp.com/documents/tr-3001.pdf last accessed Jul 2008.

[IEE90] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. Institute of Electrical and Electronics Engineers, 1990.

[IEE98a] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Supplement to Media Access Control (MAC) Bridges: Traffic Class Expediting and Multicast Filtering. Institute of Electrical and Electronics Engineers, 1998.

[IEE98b] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. Institute of Electrical and Electronics Engineers, 1998.

[IEE00a] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Link Aggregation for Parallel Links. Institute of Electrical and Electronics Engineers, 2000.

[IEE00b] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Rapid Configuration of Spanning Tree. Institute of Electrical and Electronics Engineers, 2000.

[IEE02] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Multiple Spanning Trees. Institute of Electrical and Electronics Engineers, 2002.

[IEE04a] IEEE. IEEE Standard for Local and Metropolitan Area Networks: CSMA/CD access method and physical layer specifications. Institute of Electrical and Electronics Engineers, 2004.

[IEE04b] IEEE. IEEE Standard for Local and Metropolitan Area Networks: Ethernet in the First Mile. Institute of Electrical and Electronics Engineers, 2004.

[IEE04c] IEEE. Part 17: Resilient packet ring (RPR) access method & physical layer specifications. Institute of Electrical and Electronics Engineers, 2004.

[inf01] Infiniband Specification. Infiniband Trade Association, Jun 2001.

[Int06] International Committee for Information Technology Standards (INCITS) T10 Technical Committee. Information technology SCSI Architecture Model - 4, Jan 2006. URL: http://www.t10.org/ftp/t10/drafts/sam4/sam4r05.pdf last accessed 19th Jun 2008.

[isc] iSCSI. http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-19.pdf.

[JJJ$^+$00] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In *IEEE INFOCOM*, 2000.

[KKG91] Aaron Kershenbaum, Parviz Kermani, and George A. Grover. MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing. *IEEE Transactions on Communications*, 39(4):503–513, Apr 1991.

[KS91] K.M. Khalil and P.A Spencer. A Systematic Approach for Planning, Tuning and Upgrading Local Area Networks. In *Proceedings of IEEE Globecom*, pages 658–663, 1991.

[LLN02] K. Lui, W. Lee, and K. Nahrstedt. STAR: A Transparent Spanning Tree Bridge Protocol with Alternate Routing. In *ACM SIGCOMM*, volume 32 of *3*, pages 33–46, 2002.

[MB76] Robert MetCalfe and David Boggs. Ethernet: Distributed Packet-Switching For Local Computer Networks. *Communications of the ACM*, 19(7):395–404, Jul 1976.

[mef] Metro Ethernet Forum. http://www.metroethernetforum.org/.

[Min91] Yue Minyi. A simple proof of the inequality FFD (L) 11/9 OPT (L) + 1, L for the FFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7(4):321–331, Oct 1991.

[Net74] Network Analysis Corp. Issues on Large Network Design. ARPA report, Jan 1974.

[OG98] K. Obraczka and G. Georghiu. The performance of a service for network-aware applications. In *ACM Sigmetrics SPDT'98*, 1998.

[Per85] Radia Perlman. An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN. *ACM SIGCOMM Computer Communication Review*, 15(4):44–53, Sep 1985.

[PFH$^+$02] F. Petrini, W. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network: High-Performance Clustering Technology. *IEEE Micro*, 22(1):46–57, 2002.

[QN98] W. Qiao and L. Ni. Network Planning and Tuning in Switch-Based LANs. In T. Lai, editor, *ICPP'98*, pages 287–294, 1998.

[RS91] T. Rodeheffer and M. Schroeder. Automatic reconfiguration in autonet. In *ACM SIGOPS*, volume 25 of *5*, pages 183–197, 1991.

[RTA00] T. Rodeheffer, C. Thekkat, and D. Anderson. Smartbridge: A scalable bridge architecture. In *ACM SIGCOMM*, 2000.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. IETF RFC 3031, Jan 2001.

[SEA00] C.H.E. Stacey, T. Eyers, and G.J. Anido. A Concave Link Elimination (CLE) Procedure and Lower Bound for Concave Topology, Capacity, and Flow Assignment Network Design Problems. *Telecommunication Systems*, 13:351–371, 2000.

[SGNcC04] Srikant Sharma, Kartik Gopalan, Susanta Nanda, and Tzi cker Chiueh. Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks. In *Proceedings of the IEEE INFOCOM'04*, Mar 2004.

[Sla96] P. Slavik. A Tight Analysis of The Greedy Algorithm for Set Cover. In *Annual ACM Symposium on Theory of Computing*, pages 435–441, 1996.

[Str01] Staffan Bo Strand. Storage Area Networks and SANTK. Master's thesis, University of Minnesota, Dec 2001.

[SWK69] K. Steiglitz, P. Weiner, and D.J. Keleitman. The Design of Minimum Cost Survivable Network. *IEEE Transactions on Circuit Theory*, pages 455–460, Nov 1969.

[Tho04] Timothy David Thompson. Optimal core-edge storage area network design. In *39th Annual ORSNZ Conference*, Nov 2004.

[Top] Top 500 Supercomputers project. Interconnect Family share for 06/2008. URL: http://www.top500.org/charts/list/31/connfam last accessed 19th June 2008.

[VC99] Srinidhi Varadarajan and Tzi-cker Chiueh. Automatic fault detection and recovery in real time switched ethernet networks. In *IEEE INFOCOMM*, volume 1, pages 161–169, 1999.

[WOSW02] Julie Ward, Michael O'Sullovan, Troy Shahoumian, and John Wilkes. Appia: automatic storage area network fabric design. In *Conference on File and Storage Technologies (FAST'02)*, pages 203–217, Jan 2002.

[WP67] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[Yag71] B. Yaged. Minimum Cost Routing for Static Network Models. *Networks Journal*, 1:139–172, 1971.

[ZSM08] Rui Zhang-Shen and Nick McKeown. Designing a Fault-Tolerant Network Using Valiant Load-Balancing. In *Proceedings of IEEE Infocom Mini-Conference*, APR 2008.