Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks

Srikant Sharma Kartik Gopalan Susanta Nanda Tzi-cker Chiueh
{srikant,kartik,susanta,chiueh}@cs.sunysb.edu
Department of Computer Science
Stony Brook University, Stony Brook, NY-11794

***Abstract*— Simplicity, cost effectiveness, scalability, and the economies of scale make Ethernet a popular choice for local area networks, as well as for storage area networks and increasingly metropolitan-area networks. These applications of Ethernet elevate it from a LAN technology to a ubiquitous networking technology, thus prompting a rethinking of some of its architectural features. One weakness of existing Ethernet architecture is its use of single spanning tree, which, while useful at avoiding routing loops, leads to low link utilization and long failure recovery time. To apply Ethernet to cluster networks and MANs, these problems need to be addressed. In this paper we propose a Multi-Spanning-Tree Ethernet architecture, called *Viking*, that improves both aggregate throughput and fault tolerance by exploiting standard virtual LAN technology in a novel way. By supporting multiple spanning trees through VLAN, Viking makes the most of the inherent redundancies in most mesh-like networks and delivers a multi-fold throughput gain over single-spanning-tree Ethernet with the same physical network topology. It also provides much faster failure recovery, reducing the down-time to a sub-second range from that of multiple seconds in single-spanning-tree Ethernet architecture. Finally, based only on standard mechanisms, Viking is readily implementable on commodity Ethernet switches without any firmware modifications.**

Keywords: System design, Simulations, Experimentation with real networks/Testbeds, Ethernet, Spanning Tree *

I. Introduction

Metropolitan Area Networks (MAN) is a class of networks where the geographical span is extended to the boundaries of metropolitan cities. Typically, MAN are a set of interconnected LANs that work together in order to provide access and services within a metro region.

Ethernet is turning out to be the preferred networking technology for recent deployments of Metropolitan Area Networks and Cluster networks. Economies of scale, ease of service provisioning, high bandwidth, ease of interconnection with LANs, and scalability are some of the prominent reasons for this preferential status of Ethernets.

*To appear in Proceedings of IEEE INFOCOM, 2004

Though Ethernet is a preferred technology for metro and cluster networks, it has certain serious shortcomings. Primarily, the spanning tree based switching mechanism in Ethernets utilizes atmost $N-1$ links in a network of N nodes. This limited utilization produces an imbalance of load which is impractical in MAN and cluster networks from a performance perspective. Next, the use of single spanning tree means that failure of any single active link would disconnect the spanning tree resulting in disruption of network traffic. Ethernets respond to this scenario by rebuilding the spanning tree. The rebuilding can be a local process of activating previously blocked links connecting the disconnected segments or recomputation of a new spanning tree. Activation of blocked links requires multiple seconds and typical convergence period of new spanning tree construction is around 30 to 60 seconds. This is not acceptable in metro and cluster networks, since high availability is one of the primary requirements.

In this paper, we propose a Multi-Spanning-Tree architecture for MAN and Cluster Networks to improve their performance and failure resiliency. To achieve this goal, we propose a novel use of the Virtual LAN technology which is traditionally used only for segregation in networks.

This paper is organized as follows. Section II describes the Spanning-Tree Protocol and elucidates the limitations and problems that arise due to the use of a single spanning tree in the context of MAN and Cluster Networks and how these issues can be addressed by using multiple spanning trees in these environments. Section III describes related work to this research. Section IV presents an architectural overview of the proposed Viking system. Section V explains the implementation details of the Viking system. In Section VI we present the performance evaluation of the Viking system. We conclude the discussion by summarizing the work in Section VII.

II. Problem Statement

The Spanning-Tree Protocol (STP), specified by IEEE 802.1d standard [1], is a simple link layer protocol which

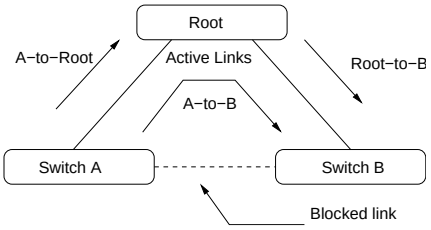

Fig. 1. Load imbalance scenario in single spanning tree. Three different flows, A-to-B, A-to-Root, and Root-to-B share same set of links despite the presence of a link between switch A and B. If somehow link A-B is made active, the overall network throughput can be improved significantly.

runs on bridges and switches in Local Area Networks (LANs). The primary purpose of STP is to provide a *loop free* switching path between all pairs of nodes in a LAN. Elimination of loops is important in switched LANs because the packet switching is carried out by flooding and reverse path learning. Broadcast and unknown destination packets are flooded over the entire network. Knowledge about paths to specific hosts is gained by snooping on the source addresses of packets received on switch ports. This information is stored in local MAC-forwarding-tables on switches. Subsequent packets are blindly forwarded to the associated ports after a simple lookup on the forwarding tables. This switching methodology presents a fast and elegant switching solution in LANs. Presence of loops in this kind of switched LANs poses problems like *broadcast storm* and *forwarding table instability*. The STP addresses these problems by reducing the topology of a switched network to a tree topology where redundant links are pruned such that there exists exactly one switching path between any pair of nodes.

Topology changes and failures in LANs are dealt by activating previously blocked links or by rerunning the spanning tree construction process. The convergence period of the STP is generally around 30 to 60 seconds. Activation of blocked links may take upto several seconds. In MAN, with increased size, the frequency of topology reconfigurations also increases. Thus, a direct application of single spanning tree switching topology is not desirable in larger environments because of its high convergence period.

The use of single spanning tree results in a single switching path between any pair of end-hosts. Further, the links constituting this path may be shared by multiple nodes. Thus, this single path is prone to communication failures and overload due to lack of alternate switching paths and absence of load balancing. Figure 1 depicts a typical load imbalance scenario in single spanning tree networks

Usually, the root of the spanning tree network faces most of the concentration of network traffic compared to other switches. To address this non-homogeneous traffic distribution, typical network deployments follow a *fat-tree hierarchy*. In fat-tree hierarchy, the switches at the leaf of the networks have limited switching bandwidth. These leaf switches are connected to faster intermediate switches through faster uplink ports, which in turn, are connected to even higher bandwidth switches. While in Local Area Networks, it is possible to use a fat tree topology, this is becoming less feasible in metro Ethernets for two reasons. First, the geographical spread makes pure mesh network topology more desirable. Second, as Ethernet technology advances to Gigabit and beyond, it is difficult to have intermediate links fatter than the leaf links. Thus, the performance of Metro Ethernets is clearly limited by the backplane processing bandwidth of the root switches.

In essence, the performance and the failure recovery issues with single spanning tree architecture of conventional Ethernets pose significant challenges in the deployment of Metropolitan Ethernets.

While the large span of Metro Ethernets makes it undesirable to deploy single spanning tree topology, the Cluster Networks, like storage networks, present the other extreme of the single spanning tree issue. The primary network requirement of these cluster networks is the high switching bandwidth between different nodes. In switched Ethernets, the peak bandwidth between any two segments is limited by the bandwidth of the link connecting them. Addition of multiple links does not improve the situation because, in spanning tree topology, there can be only one active link forwarding traffic to a segment. Thus, the general tendency is to aggregate the cluster nodes on a single switch with high port density. Aggregation cost of switching networks increases rapidly with the size of the cluster which in turn raises the per port cost factor. Further, increased port density does not necessarily reflect into increased backplane bandwidth as the underlying crossbar switching technology remains unchanged. Thus, the ratio of port bandwidth with backplane bandwidth progressively decreases with the increase in port density of a switch.

The performance gap between the backplane bandwidth and the port bandwidth does not increase proportionally when one uses higher bandwidth switches. For example, Fast Ethernet switches usually have several Gigabits of backplane switching bandwidth, whereas, Gigabit Ethernet switches do not possess the backplane switching bandwidth in a similar proportion to the port bandwidth. Thus the crossbar switching speeds have not kept up with the port bandwidth speeds. This trend is expected to continue which would make the argument in favor of aggregation less strong.

Apart from cost and performance issues, aggregation also presents a reliability factor. Aggregation of nodes on single or a limited number of switches presents a small set of points of complete failure. This reduces the reliability and increases the chances of complete failure in the event of complete switch failure.

The per-port-cost factor, scalability of port and

backplane bandwidth, and the reliability issues provide sufficient ground for moving away from 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, typically, the performance and fault tolerance issues because of spanning tree protocol of Ethernets.

The root cause of performance bottleneck and lower fault-tolerance threshold is the single spanning tree switching used in these networks. Multiple switching paths between a pair of end-hosts can be provisioned by simply using multiple spanning tree instances. Presence of multiple paths provides an automatic scope for load balancing and fault-tolerance. In the event of link overloads and link/switch failures, the communication between the affected end-hosts can be diverted to paths that do not include the failed network elements. If the alternate spanning trees are precomputed, the diversion can be carried out fairly quickly in a transparent fashion so that the long convergence period of spanning tree protocol no longer remains an issue and the end-hosts can communicate without interruptions. The spanning trees can be constructed intelligently so that the root switches can be distributed across the network. This distribution, and the use of possible load balancing therein, eliminates the need for adopting the fat-tree topology. Since the backplane processing load is also distributed across various roots, the need for faster links in the vicinity of roots is also eliminated.

In Cluster Networks, segregation while retaining the required high bandwidth is possible by use of multiple active links across distributed segments. Segregation also presents a favorable case of increased total available backplane bandwidth because of increase in per-port backplane bandwidth. One can also exploit the economies of scale by deploying widely available low port density switches interconnected with each other using multiple active links. Another advantage of this approach is the increased availability of the cluster. Complete failure of any switch leaves only a part of the segregated cluster useless, whereas, rest of the cluster can still function, though at a reduced performance.

The use of multiple spanning trees in networks is not a trivial task because of following reasons. Conventional switched Ethernets do not have an inherent support for multiple spanning trees. Further, the end-hosts do not have control over switching path selection. Rather, the switching path is selected by switches in the network. In addition, the switches themselves need a mechanism to distinguish between various spanning trees for effectively maintaining the MAC address forwarding database.

Viking addresses these problems by coupling the use of Virtual LAN (VLAN) technology [2] with the multiple spanning tree approach. Upcoming IEEE 802.1s standard [3] provides a provision for maintaining multiple spanning tree instances on individual VLAN basis. Viking leverages on this facility to come up with load balanced switching paths which can be 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.

III. Related Work

Network traffic engineering is a widely researched topic for Local Area Networks and Wide Area Networks. Further, the performance problems because of single spanning tree in Ethernets is a well known issue. The idea of using blocked links in one spanning tree while constructing another in order to recover from a link failure is nicely established in EtheReal [4]. Viking approach is similar to that of EtheReal, but instead of using blocked links during spanning tree construction, Viking computes multiple spanning trees in advance. This enables Viking to take care of possible failures at most of the links (wherever possible) by using redundant links to construct backup paths. This process saves time in recovering from failures apart from reducing the delay. In EtheReal, all the connections going through the failed link are terminated and are reestablished after a new spanning tree is rebuilt. Whereas, Viking uses the pre-calculated backup paths to route traffic in the link-failure cases without causing any harm to the existing connection. The switching to the new route, though, reduces the bandwidth by a small factor for a few milliseconds.

Autonet [5] demonstrates the feasibility of reconfiguring the network settings for picking proper path to the destination by updating the forwarding table in a switch. It operates by employing a monitor at the link level to detect error rates and link failures to trigger the recovery process. The recovery process uses a distributed algorithm in each of the switches for topology acquisition and forwarding table recalculation. From the next packet onwards, the newly appointed link is used to forward the packets to the destination. In contrast, Viking maintains a set of pre-calculated alternate paths to be used in the event of a failures, and thus makes the recovery process smooth and fast. Autonet requires the switches in network to be Autonet compliant, whereas, Viking can be used with commodity switches which are compliant with 802.1q standard and support multiple spanning tree instances.

SmartBridge [6] is a new bridge architecture proposed to address the problems associated with spanning trees in LANs. Packet forwarding in smartbridge architecture is done along the shortest paths. This is achieved by carrying out a topology acquisition The topology acquisition logic is built into the network switches. Although shortest path switching may provide a low latency path, it does not address the load balancing issue in the network. The reconfiguration times are remarkably low,

around 10 ms to 20 ms. However, this requires all bridges in the network to be smartbridge compliant. In contrast, Viking relies on modifying the end hosts to use VLAN technology. All modern operating systems provide a support for VLANs which can be readily used to implement Viking functionality. Further, Viking adopts a load balancing approach to efficiently use the network.

King-Shan Lui et. al [7] discuss a novel approach to find and forward frames over alternate paths that are probably shorter than their corresponding tree paths. This makes use of the links that are traditionally blocked by the IEEE 802.1D standard. Although the approach reduces latency between most of the source and destination pair, it risks overloading of critical links, as the paths that it comes out with always remain static for a given network topology. Moreover, it is not very obvious how it fares in recovering from a link or switch failure, as several records in all the tables need to be recalculated in such a scenario. Viking, however, tries to combine the two aspects, reduction of latency as well as the load over each of the links to improve the overall performance. And apart from that, it also takes care of link and switch failures to improve the robustness of the network.

A description of a new transparent bridge protocol for local area networking that allows topologies with active loops can be found in [8]. 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. However, this work uses improved routing schemes to allow highly connected regular topologies, like meshes, to be used without blocking any of the links at all. Unlike Viking, it does not consider loads on individual links while taking the routing decision, and the protocol involves substantial modification of the frames, making it more complex and transfer extra amount of data.

Load balancing in network by coming up with an appropriate inexpensive topology, given the loads for source and destination pair is well documented in the Network Planning and Tuning work [9]. It uses the block design principles to come up with the initial topology and a method for fine tuning to optimize parameters like 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 coming up with an appropriate forwarding table for the packets, which is more useful in a practical scenario. In addition, it also has methods to evolve itself by coming up with better paths as the network topology changes with time (by regularly studying the traffic statistics between host pairs). Whereas, in the Network Planning work, everything has to be done apriori, to achieve appropriate balance in the load which may not be feasible in many practical cases.

Multi-protocol label switching technique (MPLS) provides a framework for efficient designation, forwarding, routing, and switching of the packets that flow through the network. It provides 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 granularities, starting from hardware to applications. Viking could 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.

The IEEE standard 802.1w [10] added some changes necessary to the MAC bridge operation that provide a rapid reconfiguration capability. This assumes a substantial reduction in the time taken to reconfigure and restore the service in the existing spanning tree protocol on a link failure or restoration. To achieve this, it introduces significant changes in the way the spanning tree algorithm works but still maintains the backward compatibility with the 802.1D version. Many metro Ethernet service vendors employ this approach to provide a superior level of fault tolerance compared to conventional Ethernet networks. However, Viking saves the convergence time in building up the new spanning tree by keeping a pre-computed VLAN tree even before the link failure occurs. Thus once the link failure is detected, switching to the new VLAN immediately takes effect 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 BPDUs transmitted by each switch for every VLAN, on a separate spanning tree process (STP) running on every switch for every VLAN. The next implementation that extends the PVST, known as Multiple-VLAN Spanning Tree (MVST) allows similarly connected VLANs to be grouped into a single STP, thus making it more scalable and without compromising on the advantages. Viking relies on PVST implementation of Cisco.

IV. The Viking System

A. Architectural Overview

The core idea of Viking system is to use multiple spanning trees in conjunction with VLAN technology to maximize the overall throughput performance of the network by utilizing multiple redundant links. Further, Viking provides fault-tolerant features by providing a mechanism to divert the affected communication over to alternate paths after detecting failures. In effect, Viking

strives to provide a *fault-tolerant traffic engineering solution for Metro Ethernets and Cluster Networks.*

Viking relies on Virtual LAN technology for selection of appropriate switching paths. VLANs are conventionally used to simplify network administration, reduce cost of segregation, and improve security. Viking deviates from this conventional paradigm and uses tag based VLANs [2] to select the desired switching path between a pair of end-hosts. All paths which can possibly be used as switching paths are absorbed in different spanning trees. Since each spanning tree instance corresponds to a particular VLAN, explicit selection of a VLAN results in an implicit selection of the switching path associated with the corresponding spanning tree. In case of failures, the end-hosts merely need to change the VLAN id in subsequent frames to select an alternate switching path.

In order to utilize the available network resources efficiently, following factors should be taken into consideration while constructing the spanning trees.

For fault-tolerance, there should be at least two switching paths for any node pair in two different spanning trees which do not share intermediate nodes or links. One of these paths can be the primary switching path and the other one can be the backup switching path. For effective load balancing, the path selection should maximize the utilization of marginally loaded links and minimize the use of heavily loaded links. Also, to maximize the number of overall active links, the spanning trees should minimally overlap with each other. The maximum number of VLANs, and hence the number of spanning trees possible in network, is dictated by the networking equipment used. The VLAN identifier space in 802.1q specifications is limited to 4096 entries. But the maximum number of spanning trees supported by switches is far less than this space. Typically, switches like Cisco Catalyst 5000 switches support a maximum of 1024 spanning tree instances. Other lower end switches like Cisco Catalyst 2900 support a maximum of 64 spanning trees.

Thus, it becomes imperative to minimize the number of required spanning trees while maximizing the number of active links in a load balanced manner with a scope for fail-over to a backup path in the event of failures. Viking addresses these issues by following a *traffic engineering* approach. Viking further provides a scope for better traffic segregation by using IEEE 802.1p [11] traffic prioritization mechanism in conjunction with VLANs. Typically, the traffic corresponding to backup paths for a node pair can have a lower priority than traffic corresponding to the primary path of some other node pair sharing the same set of links. This can be achieved by marking the priority bits in transmitted frames in accordance with 802.1p specification.

B. Traffic Engineering

Traffic engineering aspect of networks deals with the performance optimization in terms of capacity utilization. The main focus of optimization is to minimize over-utilization of capacity when other capacity is available in the network. Traffic engineering includes traffic measurement, modeling, characterization, control, and application of techniques to achieve the performance objectives. It also includes capacity management through the control of network design [12].

Traffic control can be carried out at various granularities. For example, in IP networks, each packet is routed independently. The routing decisions are taken at intermediate routers depending on the conditions prevailing at the time of packet arrival. The notion of utilization of network capacity is derived by the routing protocols by computing the cost associated with each available routing path. Another approach can be that of continuous monitoring, measurement, and periodic route configuration. In this approach, a dynamic reconfiguration can be carried out to balance the load amongst different network components. Though this approach cannot deal with instantaneous overload situations, the long term control can be carried out effectively by simple reconfigurations after periodic monitoring and measurement. Viking follows the second approach of long term monitoring and reconfiguration. This scheme fits well in the context of Metro Ethernet and Clusters because the path selection in the purview of Viking requires a configuration of spanning trees and Virtual LANs which cannot be carried out at a finer granularity. Further, it is not practical to perform frequent reconfigurations because of the large number of network elements that need to be configured in large scale networks like metro Ethernets. The reconfiguration tools available to Viking are usually the management interfaces provided by network equipment, typically SNMP interfaces.

The traffic control approach of Viking is analogous to establishing 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 tags associated with the corresponding spanning tree. VLAN tags are similar to labels in MPLS schema. Like ATM virtual circuits and MPLS labels, Viking tag assignment is also long term.

Another important facet of traffic engineering is the network capacity design and planning. This requires an apriori knowledge about the traffic in the network. Since Viking aims to provide traffic engineering solution to Metro Ethernets and Clusters, where this kind of apriori knowledge is hard to come by, Viking does not deal with the network design issues. Instead, Viking provides a mechanism to identify the critical portions of networks which need to be strengthened in terms of bandwidth or backplane processing. In effect, Viking provides network tuning guidelines for improving the network performance. This tuning can be performed by simple increment of resources in the critical portions of the network identified by Viking.

For providing effective traffic engineering solution,

Viking needs to address various issues like, topology acquisition, load characterization, fault-tolerance, traffic management, etc. The remainder of this section briefly discusses these issues with an emphasis on traffic management.

Topology Knowledge: Since Viking is not involved in network design and planning, it needs to obtain the knowledge about network topology by external means. This knowledge can be obtained in an automated fashion by using available network topology discovery tools. Some of the works that facilitate topology discovery of Ethernets can be cited as Topology-d [13], IDMaps [14], Remos [15]. Alternatively, the topology information can be manually provided by network administrators. This does not pose any problem since, for large scale and planned networks like Metro Ethernets, the 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 issue by implementing a statistics gathering mechanism. The statistics are continuously collected at end-hosts during the network activity. The distributed traffic information is periodically integrated to obtain a periodic pair-wise network utilization information. This information can then be used by Viking traffic management logic to determine switching paths which utilize the network resources in a load balanced fashion.

Fault-Tolerance: 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, which can be selected by simply using the VLAN tags corresponding to the backup path spanning tree. Viking also 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 network switches.

Traffic Management: Traffic Management is perhaps the most intensive part of Viking. Once the information about network topology and pair-wise load statistics is obtained, Viking needs to come up with appropriate switching paths between given node pairs. These switching paths further need to be combined together to form spanning trees which can be associated with different VLANs. This task is divided into three primary subtasks, namely, Path Selection, Path Aggregation into spanning trees, and Spanning tree configuration. The path information also needs to be percolated to the end-hosts for final path selection. This is because the end-hosts are ultimately 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.

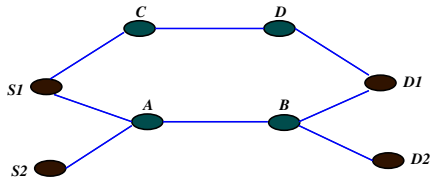

Fig. 2. A simple problem of selecting a route from S_1 to D_1. Selecting the route (S_1, C, D, D_1) leaves the critical link (A, B) free for future virtual connections between S_2 and D_2.

Path Selection: The goal of path selection algorithm is to maximize the 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) [16]. The main intuition behind 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 are expected to carry significant load.

Consider the network topology shown in Figure 2. 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? Let's say that we also expect traffic between nodes S_2 and D_2. Then the better 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.

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 would 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 w.r.t. all intermediate

network elements. This is to ensure complete switch-over to backup route if any one element fails along the primary route.

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 the different 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 w.r.t. source-destination pair (s, d) is defined as the fraction of routes between s and d that pass through link l, i.e. $\phi_l(s, d) = y/x$.

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, $\phi_l = \sum_{(s,d)} \phi_l(s, d)B(s, d)$. Where $B(s, d)$ is the traffic demand between s and d.

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.

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

$$cost(G) = \sum_{l \in G} \left(cost(l) - \frac{\phi_l}{C_l} \right)^2 \quad (1)$$

The metric $cost(G)$ represents the squared magnitude of 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 $(\phi_1/C_1, \ldots, \phi_m/C_m)$. Squared sum has the advantage that it captures the impact of both magnitude of individual link costs and the variations among them.

The LCBR algorithm for selecting both primary and backup routes is presented in Figure 3. 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 and 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 [17], [18]. The final route pair is selected from these k^2 shortest route pairs. The parameter k can be tuned to increase or decrease the accuracy of the algorithm in minimizing $cost(G)$.

For each candidate primary-backup pair (X, Y) between s and d, 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 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 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.

Input: Network topology G.
  New route request between nodes s and d
  Average bandwidth requirement $B(s, d)$
  For all links l: ϕ_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

Fig. 3. *Link Criticality Based Route Selection Algorithm* to select both the primary and backup routes between source s and destination d with bandwidth of $B(s, d)$.

Path Aggregation: 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. Since the number of VLANs is a scarce resource, it is essential to group the paths together so that the number of spanning trees that need to be constructed is minimized.

Viking uses a simple heuristic based algorithm to achieve this goal. Since the 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 including larger number of paths together. To increase the number of paths per spanning tree, Viking tries to merge paths which share *common features*. 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 algorithm for path aggregation is described in Figure 4.

After the termination of this algorithm, the set S will have a collection of spanning trees. These spanning trees can then be used to create independent VLANs. The individual paths can then be selected by transmitting frames over the associated VLANs.

Spanning Tree Construction: In conventional networks the spanning tree construction process is carried out by the participating switches in a distributed manner. It is also possible to force a spanning tree on an intercon-

```
Let the set of all paths be P
Let the set of all edge pairs be EP
Let the set of spanning trees be S
Set S = φ
Sort the members of P in the descending order of path
length
While (EP ! = φ and P ! = φ )
Sort the members of EP in descending order of their
frequency
of appearance in members of P
    Set ep = Next element in EP
    While ∃ p ∈ P such that ep ⊂ p
        Remove p from P
        Find s ∈ 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
```

Fig. 4. Path Aggregation algorithm. For the given input of selected paths P and the network topology, the algorithm computes a set of spanning trees.

nection of switches by configuring the VLAN priority of different links. For all links that are members of a particular spanning tree the associated VLAN can be configured to have a high priority and for all remaining links it can be blocked out. This, in effect, is forcing a spanning tree on the interconnect of switches since the only active topology in the context of a VLAN is a spanning tree topology. All modern Ethernet switches provide a configuration functionality through SNMP. Viking carries out the actual creation of spanning trees in the network by means of these SNMP interfaces.

V. System Implementation

The Viking system needs to perform certain tasks in a distributed manner and certain other tasks in a centralized manner. For example, the final VLAN selection logic has to reside on every end-host as it needs to be integrated with the network stack on the end-hosts. Also, the load measurement and monitoring can be effectively done at end-hosts which happen to be the originating and terminating points for traffic. Other tasks, such as, traffic management, failure discovery and recovery, network configuration, etc., can be done efficiently in a centralized manner. For this purpose, Viking implementation is based on the client-server model. In Viking terminology, the clients are called *Viking Node Controllers* (VNC) and the server is known as the *Viking Manager* (VM).

A. Viking Node Controller

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

Run-time VLAN Selection: In order to carry out the run-time VLAN selection, the VNC needs to alter Ethernet frames before these are sent out by the network interfaces. 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 end-hosts uses VVIF for sending and receiving packets on the network. The VVIF, on reception of first frame for any specific destination, sends an *association query* to the Viking Manager and obtains the VLAN id for path which is 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 VNCs that are the users of affected VLANs, to change the VLAN association in order 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 end-host 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 VNC keeps track of all peer nodes and the amount of traffic exchanged with these peers through the VVIF. VNC then periodically sends updates to the Viking Manager, which uses them for future load balancing.

B. Viking Manager

Viking Manager is the central place responsible for traffic engineering and fault tolerance. Further, the VM also needs to inform the end-hosts about the VLAN information as and when required. Since all end-hosts update the VM with their respective traffic information, the VM has the 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 VNCs periodically send traffic monitoring updates to the central 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, the VM reconfigures the network 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 VNCs 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 hash tables.

Fault-Tolerance: One important aspect of using multiple spanning trees is easy and quick recovery in the event

of failures. To support fault tolerance, Viking needs a mechanism to detect failures in the network. After failure detection, VM pro-actively informs all affected end-hosts to change the VLAN association to switch over to backup paths. This information is used by VVIF on end-hosts to choose appropriate VLANs.

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 Manager whenever any event of interest takes place. Typically, the events of interest are link failures, port failures, carrier loss, etc. The switches notify failures to the VM using SNMP. The VM uses this notification to find out the VLANs, and hence the paths, in which the affected links are active. The source hosts of the affected paths are then pro-actively informed to use the backup paths instead of failed primary paths. Further, the VM initiates the reconfiguration process for the changed topology to tackle subsequent failures. Since, VM is the central node responsible for overall Viking operation, the reliability of VM is very crucial. To avoid making VM as 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. It is also possible that the connectivity of switches sending SNMP messages to VM may fail leading to missing failure notifications. This issue can be addressed by sending SNMP messages over mutually exclusive VLANs in a broadcast fashion.

VI. Performance

Performance evaluation of Viking was carried out through extensive empirical measurements of a prototype. For large scale networks the evaluation was done through simulations.

A. Simulations

The Viking system was evaluated for its performance at path selection and path aggregation. The 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 Ethernets and cluster networks. The simulations were carried out against a uniform traffic pattern of each node communicating with other nodes with equal traffic load and a skewed traffic load of client-server communication and peer-to-peer communication. The uniform traffic distribution is a representation of cluster networks and skewed traffic distribution represents a typical scenario in metro Ethernets. The simulations were run against grids of sizes 16, 25, 36, 49, and 64 nodes connected using links with a capacity of 100 Mbps. In uniform load scenario, the traffic between nodes is 10, 8, 5, 2, and 1 Mbps for grids of sizes 16, 25, 36, 49, and 64 respectively. In 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 other nodes as peers. The effectiveness of path selection was compared for possible throughput against the single spanning tree case. The path selection was carried out for both cases, with and without backup redundancy. All traffic comprised of at least one hop between the switches.

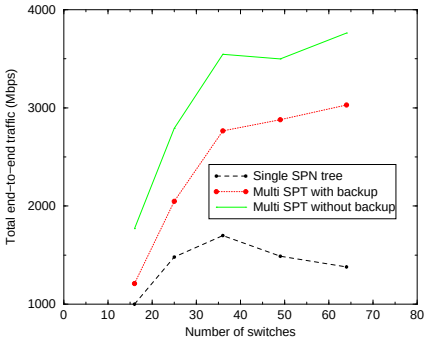

Fig. 5. Total end-to-end traffic in network with single spanning tree, and multiple spanning trees with and without backup provisioning. The network follows a uniform traffic pattern while communicating with each other.

Figure 5 shows the comparative maximum throughput for single spanning tree network against the Viking path selection. The traffic pattern is assumed to have a uniform distribution across all node pairs. This is a representative distribution of cluster networks. It can be seen that the total aggregate end-to-end throughput is always more in Viking system. As the number of nodes increases the performance of Viking system shows considerable advantage. This is because of availability of additional number of active links in topology. Thus, Viking has an upwardly scalable performance, a desired feature for growing networks.

Figure 6 shows similar scenario with a skewed traffic pattern. It can be noted that, for a network of 49 nodes, there is a performance dip compared to the 36 node network. The reason for this can be attributed to the positioning of servers in the network. Figure 7 shows the traffic distribution across different links for this network of 49 nodes. It can be seen that, because of saturation of links in the vicinity of certain server nodes the performance of the entire network is bottlenecked. This can be tackled by using additional links to connect the switches facing high load.

Figure 8 shows the number VLANs required to accommodate different number of paths. For a topology of 8x8 grid, the minimum number of VLANs is 38 for 500 distinct paths. This number increases with an increase in path count. For example, for 3500 paths the number of VLANs is 110. Usually the maximum number

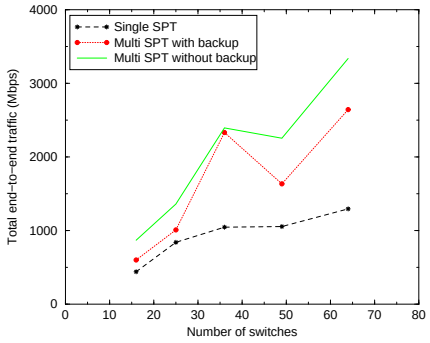

Fig. 6. Total end-to-end traffic in network with single spanning tree, and multiple spanning trees with and without backup provisioning. The network follows a skewed traffic pattern while communicating with each other. The case with 49 switches shows a dip in performance because of saturation of links near servers.

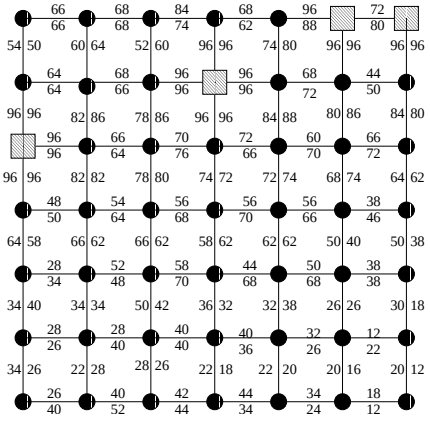

Fig. 7. Load distribution in a 7x7 grid network. The edges represent duplex links of 100 Mbps capacity. The links are marked with the total provisioned bandwidth. Note that certain links in the vicinity of servers (marked as squares) are completely saturated. This network can be tuned by adding new links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

of possible spanning trees is limited by the firmware implementation on switches. For Cisco Catalyst 5000 switches this is around 1024. It can be seen that the path aggregation VLAN requirement is well within this limit, even for large sized networks.

B. Empirical Performance

A Viking prototype was implemented using Cisco Catalyst 2924 switches. The prototype setup included 3 switches connected in a triangular fashion with each other. 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. Each packet processing required around additional 15 microseconds on an average. This

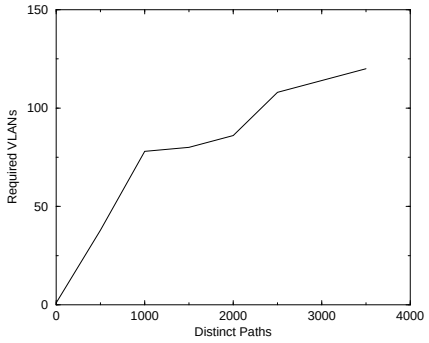

Fig. 8. The number of required VLANs with different number of paths. The required number generally increases with increase in number of paths.

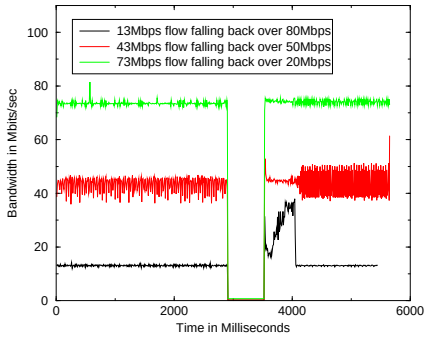

Fig. 9. Behavior of TCP across fail-over. After fail-over, TCP has to adapt to the existing traffic on backup links. The amount of traffic on backup links was maintained at a sufficiently high level so that the links are saturated after fail-over. This was the worst case scenario for fail-over. It was observed that TCP takes around 300 ms to 400 ms to recover from fail-over.

latency is attributed to the lookup and insertion of VLAN id in transmitted packets.

An experimental verification of advantage of additional active links was carried out. In this setup, two switches were connected using, single link, 2 active links, and 3 active links. The total throughput performance for UDP and TCP traffic was measured. The performance was observed to increase linearly with increasing link capacity. Figure 10 shows the TCP and UDP throughput for varying number of active links.

Figure 9 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 the link utilization at the maximum level in the VLAN that served the main traffic. Upon link failures, the TCP traffic passing through this link falls back onto the backup VLAN, which happens to pass through the other link. There was an expected drop of bandwidth for the recovery period (around 600 milliseconds) Once the backup VLAN was used, the TCP flow regained its momentum hardly suffering any

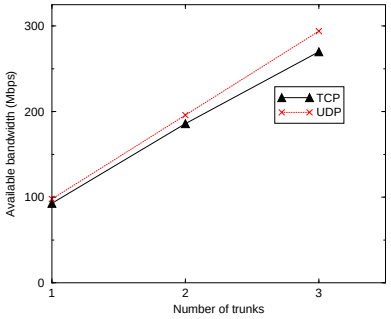

Fig. 10. Throughput for different number of active links acting as trunks. The performance is linear to the number of active links.

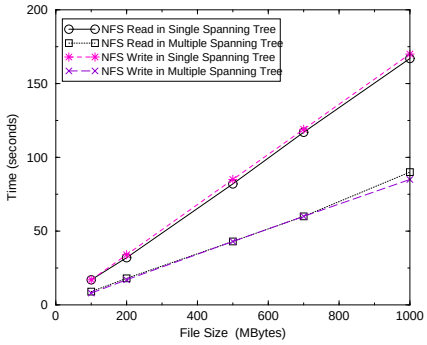

Fig. 11. Performance of NFS in multiple spanning tree and single spanning tree case. In presence of two active links, the performance is exactly double compared to the performance in single spanning tree scenario.

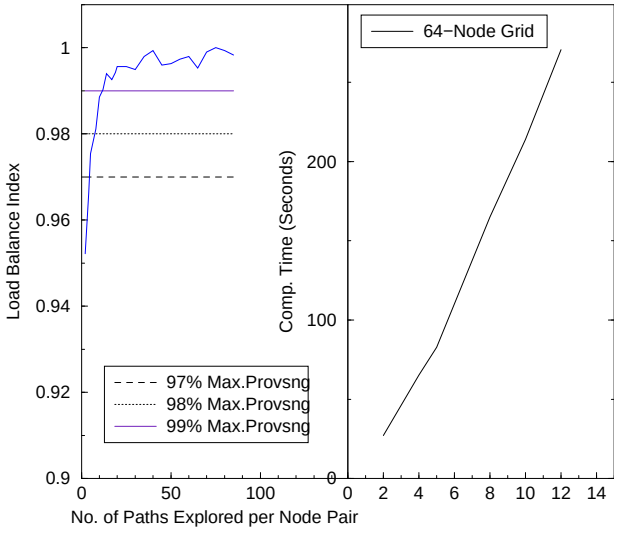

Fig. 12. Total time spent in recomputing path selection. With increased search space the recomputation time also increases. For a grid network of size 64 nodes the recomputation is shown on right side. The left side of graph shows the load balance index for a limited number of searches. Even for a limited search the number of paths is within 97% of maximum possible load balance.

more delay and attaining 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 was around 600 Mbps for two-hop communication. Whereas, in the case of single spanning tree, it was 400 Mbps for single-hop communication and 200 Mbps for two-hop communication.

Figure 11 shows the performance of Viking system 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 setup, these switches are connected using 2 active links participating in different VLANs. The files on servers are accessed parallelly by different clients. The total time for NFS read and write operations for files of different sizes is shown. The performance for two active link scenario is observed to be double compared to the performance in single spanning tree case.

The fault-tolerance mechanism was tested by manually plugging out links from switches. The down-time because of failures can be broken down into 3 parts, namely, the failure detection period, alternate VLAN lookup period, and failure notification to end-hosts. 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 failures. Thus, the total time required to recompute paths and reconfigure the spanning trees is the critical period during which another failure for paths for affected nodes cannot be tolerated. The 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 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 primary path was varied from 1 path to 15 different paths. Each primary path search was accompanied by a search space for 5 backup paths. Figure 12 shows the relation of recomputation time with the size of search space. With larger search space, the number of paths that can be actually provisioned also increases while reducing the

load imbalance. We define *load balance index* as the ratio of paths that can be provisioned after a limited search against the number of paths that can be provisioned after exploring a large search space such as 5000 alternate paths. Figure 12 also 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 load imbalance within 97% of 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 can then be followed by reconfiguration of spanning trees in the network.

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

VII. Conclusion

Traditional Ethernet architecture uses a single spanning tree to avoid routing loops. As a result, some of the links in the network are left unused, and both link utilization efficiency and failure recovery time suffer. Recognizing that it is still possible to avoid routing loops when using multiple spanning trees, we proposed a Multi-Spanning-Tree Ethernet architecture called Viking, which leverages standard virtual LAN technology in a novel way. The ability to overlay multiple spanning trees on a layer-2 network makes it possible to exploit the redundancies in the physical network and increase aggregate throughput, and to pre-provision fail-over paths and reduce failure recovery time. Because of improved performance and failure recovery characteristics, Viking makes a better building block for MAN, storage area networks, and cluster networks.

We developed a centralized multi-spanning-tree construction algorithm that aims to balance the loads of all the links in the physical network, assuming static network topology and fixed traffic workloads. Through an extensive set of simulations of the proposed algorithm, we showed that the multiple-spanning-tree Ethernet architecture and the associated spanning tree construction algorithm can indeed increase the aggregate throughput and speeds up failure recovery. To demonstrate the feasibility of the proposed architecture, we also presented the implementational details of a Viking prototype. Empirical measurements on this prototype are in line with the simulation results. In particular, the fault tolerant support of Viking provides transparent recovery in the event of link failures, with the recovery period reduced from tens of seconds in single spanning tree architecture down to 400 to 600 milliseconds.

Although Viking can be built on commodity Ethernet switches using SNMP, a more tightly integrated design could lead to even better failure recovery time. We are currently examining the feasibility of building the second Viking prototype, which will involve modification of the firmware of commercial Gigabit Ethernet switches. In addition to being more efficient in failure detection and reporting, the second Viking prototype will feature a distributed Viking manager and a dynamic reconfiguration algorithm that adapts to traffic patterns constantly.

References

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

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

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

[4] S. Varadarajan and T. Chiueh, "Automatic fault detection and recovery in real time switched ethernet networks," in *IEEE INFOCOMM*, 1999, vol. 1, pp. 161–169.

[5] T. Rodeheffer and M. Schroeder, "Automatic reconfiguration in autonet," in *ACM SIGOPS*, 1991, vol. 25 of *5*, pp. 183–197.

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

[7] K. Lui, W. Lee, and K. Nahrstedt, "Star: A transparent spanning tree bridge protocol with alternate routing," in *ACM SIGCOMM*, 2002, vol. 32 of *3*, pp. 33–46.

[8] R. Garcia, J. Duato, and J. Serrano, "A new transparent bridge protocol for lan internetworking using topologies with active loops," in *ICPP'98*, T. Lai, Ed., 1998, pp. 295–303.

[9] W. Qiao and L. Ni, "Network planning and tuning in switch-based lans," in *ICPP'98*, T. Lai, Ed., 1998, pp. 287–294.

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

[11] 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.

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

[13] K. Obraczka and G. Georghiu, "The performance of a service for network-aware appplications.," in *ACM Sigmetrics SPDT'98*, 1998.

[14] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the placement of internet instrumentation," in *IEEE INFOCOM*, 2000.

[15] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz, "A resource query interface for network-aware applications," in *Cluster Computing*, 1999, vol. 2, pp. 139–151.

[16] K. 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.

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

[18] D. Eppstein, "Finding the k shortest paths," *SIAM J. Computing*, vol. 28(2), pp. 652–673, 1998.