

NETWORK FAULT TOLERANCE SYSTEM

by

John Sullivan

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical and Computer Engineering

by

May 2000

APPROVED:

Professor David Cyganski, Major Advisor

Professor John A. Orr

Professor Nathaniel Whitmal

Abstract

The world of computers experienced an explosive period of growth toward the end of the 20th century with the widespread availability of the Internet and the development of the World Wide Web. As people began using computer networks for everything from research and communication to banking and commerce, network failures became a greater concern because of the potential to interrupt critical applications. Fault tolerance systems were developed to detect and correct network failures within minutes and eventually within seconds of the failure, but time-critical applications such as military communications, video conferencing, and Web-based sales require better response time than any previous systems could provide.

The goal of this thesis was the development and implementation of a Network Fault Tolerance (NFT) system that can detect and recover from failures of network interface cards, network cables, switches, and routers in much less than one second from the time of failure. The problem was divided into two parts: fault tolerance within a single local area network (LAN), and fault tolerance across many local area networks. The first part involves the network interface cards, network cables, and switches within a LAN, which the second part involves the routers that connect LANs into larger internetworks. Both parts of the NFT solution were implemented on Windows NT 4.0 PC's connected by a switched Fast Ethernet network. The NFT system was found to correct system failures within 300 milliseconds of the failure.

Acknowledgements

For my family, Mom, Dad, and Julie, who supported and guided me with superhuman patience.

For Professor David Cyganski, who gave me the opportunity to pursue a graduate degree, and guided me through the rough times during this and other projects.

For my roommates, Joe Alba, Ben Clark, and Jurg Zwahlen, who lived through this experience with me. Joe spent several nights in the lab helping to debug this project when things just wouldn't work right.

For all of my friends, who listened over food and coffee while I talked about this thesis.

For my labmates, Joe Alba, Mike Andrews, Mike Driscoll, Mike Roberts, Carleton Jillson, Jeremy Johnstone, Jim Kilian, Matt Lug, Sean Price, and Nandan Sinha, for making the lab a fun place to be, and for Brian Hazzard, who worked on the Fault Tolerance project for Lockheed-Martin for the first couple of months.

For my thesis committee, Professor David Cyganski, Professor John A. Orr, and Professor Nathaniel Whitmal, who helped make the final document what it is.

For Lockheed-Martin Government Electronic Systems Division, who funded this project.

Thank you...this thesis would have never reached this point without all of you.

Contents

List of Figures	vi
List of Tables	ix
1 Introduction	1
2 Network Fault Tolerance Schemes, Past And Present	4
2.1 Hot Standby Routing Protocol	5
2.2 IP Standby Protocol	6
2.3 Virtual Router Redundancy Protocol	7
3 Development of Router Fault Tolerance System	8
3.1 Initial Concept	9
3.2 Improved Scaling	11
3.2.1 Client/Server Design	12
3.2.2 Fault Tolerance of Client/Server	13
3.3 Cross-Subnet Fault Tolerance	14
3.4 Modifying Route Table in Windows NT 4.0	17
3.4.1 Introduction to the Simple Network Management Protocol	17
3.4.2 Using SNMP to Modify NT's IP Route Table	19
4 Current Implementation of Router Fault Tolerance System	24
5 Results of Router Fault Tolerance System Tests and Benchmarks	27
6 Development of Switch Fault Tolerance System	34
6.1 Initial Concept	34
6.2 DHCP-Based Switched Fault Tolerance System	36
6.2.1 Introduction to DHCP	37
6.2.2 Description of DHCP-Based Switch Fault Tolerance	39
6.2.3 Failure Points of DHCP-Based Switch Fault Tolerance	40
6.3 SNMP-Based Switch Fault Tolerance	42
6.3.1 Description of SNMP-Based Switch Fault Tolerance	42
6.3.2 Issues with SNMP Implementations in Commercial Routers	43

6.3.3	NIC Tester Mechanism	46
6.3.4	Switch Interconnect Cable Failure	49
6.3.5	Advantages and Disadvantages of SNMP-Based SFTS	49
7	Current Implementation of Switch Fault Tolerance System	53
8	Results of Switch Fault Tolerance Tests and Benchmarks	56
8.1	NIC Tester Tests	56
8.2	SFTS Tests Before Implementation of Group NIC Change Packets	57
8.3	SNFT Tests After Implementation of Group NIC Change Packets	57
8.4	SNFT Tests Across Routers	58
8.5	SFTS Recovery from Total Switch Failure	59
9	Conclusions	62
9.1	Current State of the Network Fault Tolerance System	62
9.2	Integrated Network Fault Tolerance System	63
	Bibliography	65

List of Figures

1.1	Example Local Area Network. Each end host and router is connected to the LAN through a network interface card and a network cable.	2
2.1	Virtual Router Group.	5
2.2	HSRP routers on a network.	6
2.3	HSRP routers on a network, sharing load.	6
3.1	IP internetwork in which routers are single points of failure.	9
3.2	Cable failure stopping traffic between subnets.	10
3.3	Client/Server Router Fault Tolerance System.	12
3.4	Multiple RFTS servers exist on a subnet, each listens for broadcast message for a pseudo-randomly generated time period, after which a broadcast message is sent.	14
3.5	Server 1 transmits broadcast message after time period expires and becomes the RFTS server, while Server 2 hears broadcast message and becomes an RFTS client.	15
3.6	Servers continually interrogate all routers to obtain information on router and router connection status. Dotted lines indicate ICMP interrogation and response packet flow.	15
3.7	Server 1 detects failure of route from subnet 1 to router 1.	16
3.8	Server informs clients and other servers of required routing change.	16
3.9	Full internetwork recovery from router cable failure.	17
3.10	Example execution of the “route.exe” program, outputting the current route table to the screen.	18
3.11	Add a route to the Windows NT route table using direct SNMP calls.	19
3.12	Windows NT route table with route to subnet 192.168.4.0 added.	20
3.13	Remove a route from the Windows NT route table using direct SNMP calls.	20
3.14	Windows NT route table with route to subnet 192.168.4.0 removed.	21
4.1	Router Fault Tolerance System Test Network	26
5.1	Test network with four test cables clearly marked.	28
5.2	Router Fault Tolerance System Benchmark, No Server-Server Communication.	29
5.3	One sided network failure.	30

5.4	Router Fault Tolerance System Benchmark, SSC, No SNMP.	30
5.5	Router Fault Tolerance System Benchmark, SSC, SNMP, Part 1.	31
5.6	Router Fault Tolerance System Benchmark, SSC, SNMP, Part 2.	31
5.7	Router Fault Tolerance System Benchmark, SSC, SNMP, Part 3.	32
5.8	Router Fault Tolerance System, Router Failure Benchmark, Router Failure.	32
5.9	Router Fault Tolerance System, Router Failure Benchmark, Router Revival.	33
6.1	Switched Fault Tolerance System network cell.	35
6.2	Host 3 on an improperly configured switched network transmits a packet to Switch 3.	35
6.3	Switch 3 transmits the packet out all of its ports except for the one on which the packet was received.	36
6.4	Switches 1 and 2 receive the packet, and again transmit it out all ports except for the one on which it was received. Note that the two switches are transmitting the packet to each other, as well as to Switch 3.	37
6.5	All switches are now continuously transmitting the original packet out of all ports.	38
6.6	DHCP client broadcasts DHCPDISCOVER message to all hosts on subnet.	38
6.7	DHCP servers respond with DHCPOFFER message containing configuration information.	39
6.8	DHCP client chooses one server, broadcasting DHCPREQUEST message to notify all servers of its choice.	39
6.9	Chosen DHCP server acknowledges the client and finalizes configuration information with DHCPACK message.	40
6.10	DHCP client relinquishes IP address and configuration information by sending DHCPRELEASE message to server.	40
6.11	Using DHCP to Change IP Addresses in the Switched Network Fault Tolerance System, Part 1.	41
6.12	Using DHCP to Change IP Addresses in the Switched Network Fault Tolerance System, Part 2.	41
6.13	Host using network card with address 192.168.1.10 to transmit packets to the local subnet.	42
6.14	Host using network card with address 192.168.1.11 to transmit packets to the local subnet.	43
6.15	SFTS Host 1 starts up and transmits SFTS Startup packet to existing hosts.	43
6.16	Existing SFTS hosts process SFTS Startup packet and respond with SFTS Switch packet.	44
6.17	Cable failure detected, Host 1 changes local route table.	44
6.18	Host 1 broadcasts SFTS Switch message, which Host 2 uses to update its route table and correct for the cable failure.	45
6.19	SFTS network cell. Every host and every switch belongs to the main SFTS subnet, which is 192.168.1.0 in this example.	46
6.20	NIC Tester Subnet B in a basic SFTS cell contains one NIC in each SFTS host.	46

6.21	NIC Tester Subnet C in the SFTS cell contains the other NIC in each SFTS host.	47
6.22	Network Interface Card tester cell.	48
6.23	The NIC Tester server on Host 4 is down because of a cable failure, hosts 2, 3, and 5 enter into the server negotiation mode. Hosts 3 and 5 transmit NEWSERVER packets to announce their eligibility to become servers. . . .	49
6.24	Since there are plenty of hosts eligible to become servers, Host 2 remains a client. Host 3 transmits a COMMITSERVER packet to announce it is becoming a server.	50
6.25	Host 5 remains a client, since there are now two active servers on the test subnet.	51
6.26	If the active NICs of two SFTS-enabled hosts are connected to different switches and the switch interconnect cable fails, the SFTS will not detect the failure, and the two hosts will be unable to communicate.	51
6.27	An SFTS-enabled host detects a failure and changes its active NIC.	52
6.28	All other hosts on the subnet change their active NICs in response to the NIC change packet, so all hosts are now communicating through the same switch.	52
7.1	Switch Fault Tolerance System Test Network.	54
7.2	Pictures of routers and switches that were used during the testing of the Switched Fault Tolerance System.	55
8.1	SNFT Tests Before Implementation of Group NIC Change Packets, Failures in Cables 1 through 6.	58
8.2	SNFT Tests Before Implementation of Group NIC Change Packets, Failure in Cable 7.	59
8.3	SNFT Tests After Implementation of Group NIC Change Packets, Failures in Cables 1 through 6.	60
8.4	SNFT Tests After Implementation of SNMP Communication with Routers, Router Cable Failure.	60
8.5	SFTS Tests After Implementation of NIC Change Packets and SNMP Communication with Routers, Total Switch Failure.	61
9.1	The failure of a switch in the Switch Fault Tolerance System also causes the apparent failure of a router.	63
9.2	The Network Fault Tolerance System, combining the RFTS and the SFTS to provide fault tolerance at the router, switch, NIC, and network cable levels. 64	64

List of Tables

3.1	Route Tables for Network Nodes in Figure 3.1.	10
4.1	Test Network Node Descriptions and IP Address Information.	25
4.2	RFTS Configuration File Information for Polaris, Bastion, and Legacy.	25
4.3	RFTS Configuration File Information for Onslaught and Exodus.	25
7.1	Test Network Node Descriptions and IP Address Information.	54

Chapter 1

Introduction

The world of computers underwent an explosive period of growth during the 1980's and 1990's with the development of the Internet and the World Wide Web. During these decades, educators, students, businesses, government agencies, and home users began to use networks of computers to distribute resource intensive tasks across several systems, communicate through email and "chat rooms," buy and sell products, and make available information about companies, projects, classes, hobbies, or personal lives.

Unfortunately, as computer users began to rely on networks for critical tasks such as military or financial applications, the risk of network failures became more of a concern. Network fault tolerance schemes were developed that detected and corrected for system failures within minutes, and eventually within seconds, of the failure. However, these failover times are not acceptable in some network applications that are being used today; for example, military communications, video conferencing, and World Wide Web-based sales. The purpose of this thesis is to develop a network fault tolerance system that provides subsecond failover times.

A basic local area network (LAN) (Figure 1.1) can be thought of as having three major parts: LAN hardware (in Ethernet, for example, switches or hubs), one or more routers, and several end hosts, which may be PCs, data terminals, servers, or printers. The LAN hardware connects all parts of the network together, transmitting all information packets between the end hosts and routers. Routers are used to connect a LAN to other networks; for example, a corporation would use routers for communication between networks in different buildings or campuses, or between the company network and the Internet.

The basic LAN described above contains several devices which act as single points of

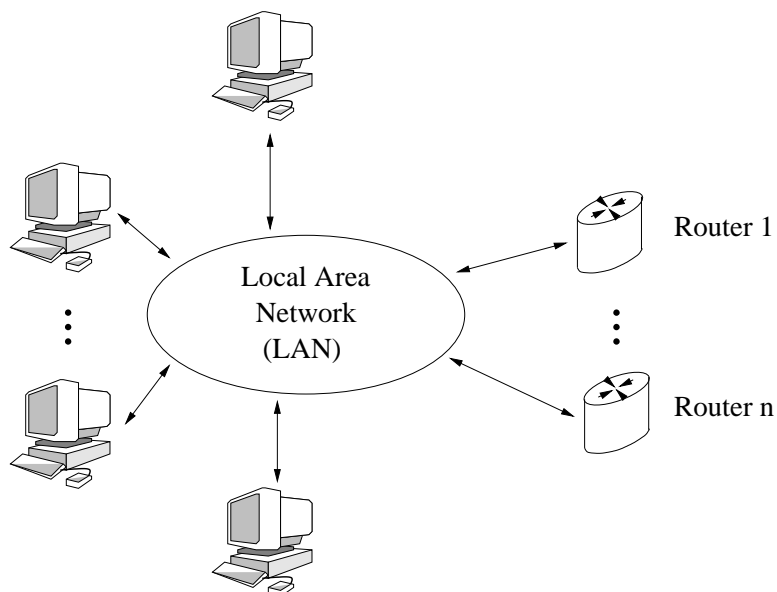


Figure 1.1: Example Local Area Network. Each end host and router is connected to the LAN through a network interface card and a network cable.

failure; if one of these devices were to fail, communication across the entire network would be affected. In Figure 1.1, the network interface card (NIC) in each end host, the switches and hubs, the routers, and each network cable presents a single point of failure for the entire system. A successful fault tolerance scheme eliminates all single points of failure in the network, making it resistant to individual failures.

The goal of this thesis was to design and implement on the Windows NT 4.0 operating system a software-based fault tolerance system for Internet Protocol (IP) network communications that provides subsecond failover times for network interface cards, routers, and switches. This system should work with existing application software, such as web browsers or email readers, and should require no special networking hardware.

The problem of achieving subsecond fault tolerance in a local area network was divided into two parts: fault tolerance within the network, which involves the NICs, routers, and switches, and fault tolerance across several interconnected networks, which involves routers. Both components of this problem were solved in the context of switched Fast Ethernet LANs, because of the dominance of Ethernet equipment in commercial networks and because of the increased network performance that switched networks provide over shared networks. The Switch Fault Tolerance System (SFTS) is designed to handle internal network cell failures,

while the Router Fault Tolerance System (RFTS) is designed to handle router failures. Both systems detect and recover from network component failures within 320 milliseconds of the failure.

Chapter 2

Network Fault Tolerance Schemes, Past And Present

In order to better understand fault tolerance systems and the current state of the art in the field of networks, several of these systems were investigated as part of the background research for this project. The three systems that will be described below, the Hot Standby Routing Protocol, the IP Standby Protocol, and the Virtual Router Redundancy Protocol, are router fault tolerance systems that are based on the concept of virtual routers.

In a virtual router system such as that shown in Figure 2.1, routers are grouped together and assigned two IP addresses, one that is unique to each router, and one “virtual” router IP address that is shared by all routers in a group. Each router in a group is assigned a priority; the highest priority router acts as the active router, and all lower priority routers act as standby routers. The routers monitor each other’s health through the transmission of “hello” packets. When an active router stops transmitting packets, the standby router with the highest priority becomes active.

Virtual router-based fault tolerance systems depend on the routers, not the end hosts, to determine when a failure has occurred and what must be done to maintain communications. No processing power is required of the end hosts; they simply establish static routes to other subnets through the IP addresses of the virtual routers.

Virtual router systems also have several disadvantages. First, if a port fails on a router or if a cable fails in such a way that the router does not detect the failure, traffic from one LAN will never reach its destination on the other LAN. Also, each of the three systems

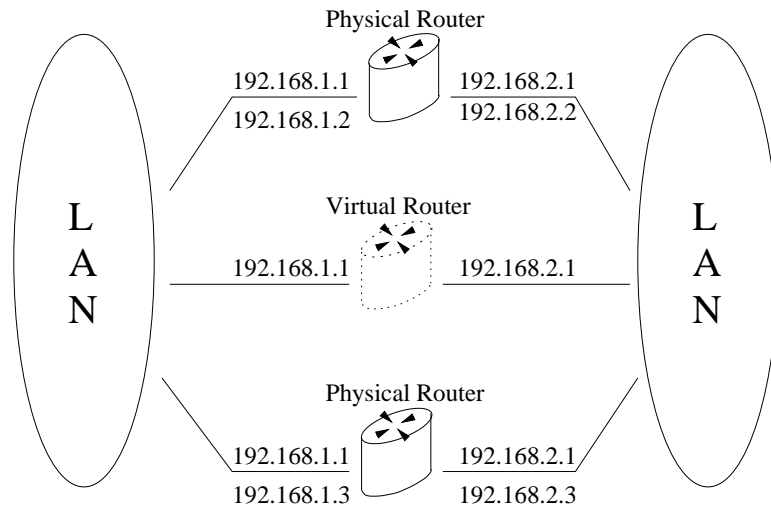


Figure 2.1: Virtual Router Group.

researched for this project is hardware dependent, requiring routers that support that specific virtual routing scheme. Therefore, fault tolerant network sub-systems cannot contain several brands of router, and existing networks cannot be made fault tolerant without the costly replacement of routers already in the networks. Also, because of the design parameters of each protocol, each of these systems requires at least one second to detect a failed router and then elect a new one to take its place: the Hot Standby Routing Protocol detects failures in at least one second, the IP Standby Protocol corrects for failures in at least five seconds, and the Virtual Router Redundancy Protocol detects failures in at least three seconds. These time limitations are not acceptable in systems that require sub-second fault tolerance.

2.1 Hot Standby Routing Protocol

The Hot Standby Routing Protocol (HSRP) is a Cisco Systems proprietary protocol designed to allow Cisco routers to create virtual routers[43]: each router can assume either active or standby status, and one standby router becomes active when an active router fails. Only one physical router assumes the duties of the virtual router at any given time, handling the entire load of the network. Figure 2.2 illustrates an example of a network with HSRP-configured routers.

An administrator can use HSRP to share the network load among routers by establishing

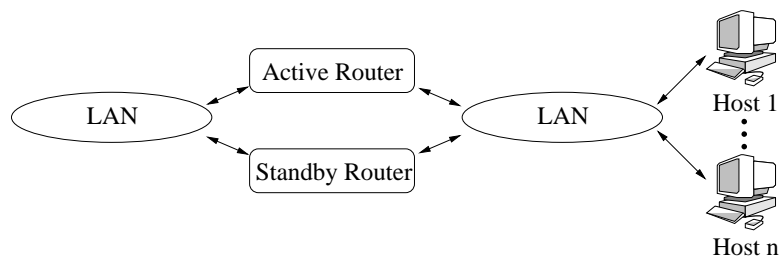


Figure 2.2: HSRP routers on a network.

more than one virtual router and configuring each physical router to assume active status for one virtual router system while acting as the standby router for the other system, as shown in Figure 2.3. The network load is divided among the routers until a failure occurs, in which case one router handles the entire load. A router must be able to assign multiple Ethernet hardware addresses, or Media Access Control (MAC) addresses, to each port in order to participate in this kind of load sharing.

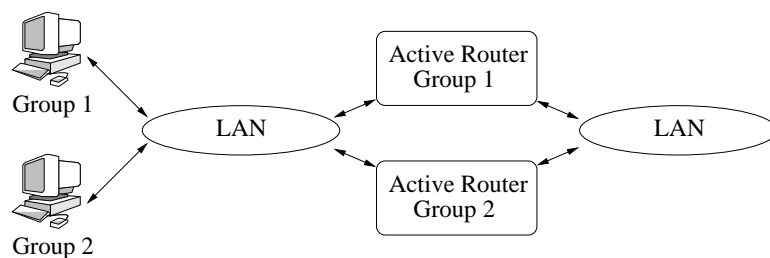


Figure 2.3: HSRP routers on a network, sharing load.

Cisco routers that can be configured for HSRP do not require any special hardware to participate in a basic virtual router system, and even older and fairly inexpensive routers from Cisco support the protocol. However, HSRP is not capable of subsecond fault tolerance for two reasons: first, the minimum time between the transmission of “hello” packets is one second, and the minimum time before an active router can be declared as having failed is one second[12, p.442].

2.2 IP Standby Protocol

The IP Standby Protocol[19], published in 1997 as a proprietary protocol of Digital Equipment Corporation, makes use of multicast “hello” packets, supports load shar-

ing, and optimizes routing of network traffic through the use of Internet Control Message Protocol(ICMP)[34] redirect packets. Unfortunately, the protocol was designed to provide a five second loss of service time for a single failure in an IP network. Routers configured for the IP Standby Protocol broadcast “hello” packets every second, and are therefore unable to detect failures in less than that period of time.

2.3 Virtual Router Redundancy Protocol

The Virtual Router Redundancy Protocol (VRRP)[20] is an Internet standard which is similar to HSRP, and supports load sharing and multicast “hello” packets (called advertisements). In VRRP, only the primary router, called the Master Router, transmits VRRP messages. When a Master Router fails, a Master is chosen in much less than one second[20, p.7], but the minimum amount of time between advertisements is one second, and the detection of a failed Master Router takes approximately three seconds[20, p.14].

Again, the response time of this protocol to router failures does not satisfy the subsecond recovery requirements specified for this project. The Router Fault Tolerance System designed for this thesis uses a new approach to the fault tolerance problem by detecting and correcting for network failures from the end hosts, not the routers, thereby eliminating the need for equipment that supports specific fault tolerance protocols. Running on the end hosts of both networks to which a router is connected allows the Router Fault Tolerance System to test a router’s network ports and cables by communication with the router’s remote network. Because of this enhanced testing, the System can provide a better fault tolerance solution than router-based schemes.

Chapter 3

Development of Router Fault Tolerance System

IP internetworks such as the Internet are generally composed of smaller networks, or subnets, which are assigned a subset of the IP addresses allocated for the internetwork. Each of these subnets may consist of other subnets, or a single LAN. Breaking down large networks through this method of subnetting establishes a hierarchical routing system for the transmission of packets across networks. Figure 3.1 is an illustration of an internetwork that has been divided into four subnets which are connected by three routers.

The hosts and routers in this example network use route tables to determine where packets must be transmitted in order to arrive at their proper destinations on other subnets. The route tables of each host and router in Figure 3.1 are shown in Table 3.1. Each entry in a route table contains three important pieces of information: a destination address, a “next hop” address, and a “network” or “subnet” mask. The destination address is the IP address of the network or host to which this route will lead, the next hop address is the router or host to which the packet will be transmitted in order to reach its destination, and the network mask is used to separate the network address from the host address in a packet’s destination IP address[13]. The network mask is used to determine if a packet will be taking the route described by this route table entry to its destination.

The routers in an internetwork such as that illustrated in Figure 3.1 are single points of failure, since communications across the network are crippled if any of these routers fails. The purpose of the Router Fault Tolerance System (RFTS) is to eliminate these single

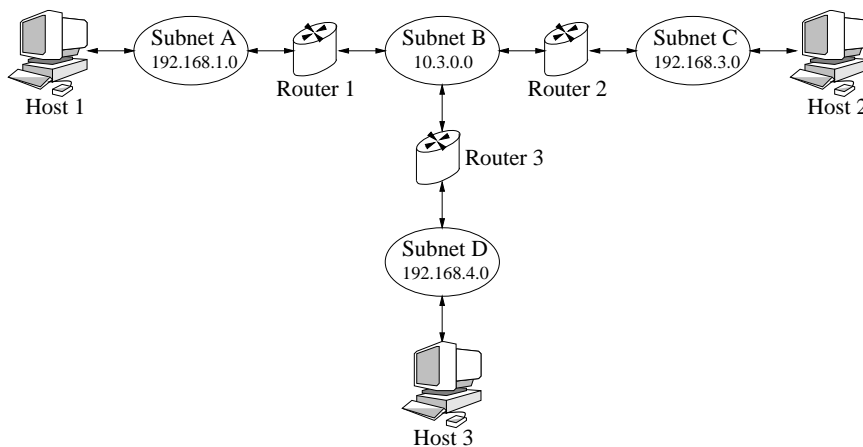


Figure 3.1: IP internetwork in which routers are single points of failure.

points of failure by providing redundancy in the routers that connect the subnets of an internetwork. An initial approach to this system was outlined in the statement of work of a research grant awarded to WPI's Convergent Technology Center by Lockheed Martin Government Electronic Systems Division, and underwent many changes before a design that was capable of subsecond fault tolerance was realized.

3.1 Initial Concept

The RFTS design originally proposed was to use continuous transmission of ICMP echo request packets from hosts running Windows NT to all routers in the subnet. Routers that transmitted echo replies back to the host were determined to be routing packets properly across subnets, while routers that failed to respond were determined to have failed. If a route in the host's route table used a failed router as a path to another subnet, the RFTS would remove that entry in the table and establish a path through a properly functioning router. A configuration file was used to list the IP addresses of routers in descending priority.

When implemented, this initial concept suffered from several problems that prevented it from achieving sub-second fault tolerance on a subnet of Windows NT hosts. First, the system scaled poorly since every host transmitted ICMP echo requests to every router; after the system was started on only a few hosts, the network became bogged down with traffic from the Router Fault Tolerance System.

Second, the initial RFTS concept was not capable of detecting all types of router failures.

Node	Destination	Destination Address	Next Hop	Subnet Mask
Host 1	Subnet B	10.3.0.0	Router 1	255.255.0.0
	Subnet C	192.168.3.0	Router 1	255.255.255.0
	Subnet D	192.168.4.0	Router 1	255.255.255.0
Host 2	Subnet A	192.168.1.0	Router 2	255.255.255.0
	Subnet B	10.3.0.0	Router 2	255.255.0.0
	Subnet D	192.168.4.0	Router 2	255.255.255.0
Host 3	Subnet A	192.168.1.0	Router 3	255.255.255.0
	Subnet B	10.3.0.0	Router 3	255.255.0.0
	Subnet C	192.168.3.0	Router 3	255.255.255.0
Router 1	Subnet C	192.168.3.0	Router 2	255.255.255.0
	Subnet D	192.168.4.0	Router 3	255.255.255.0
Router 2	Subnet A	192.168.1.0	Router 1	255.255.255.0
	Subnet D	192.168.4.0	Router 3	255.255.255.0
Router 3	Subnet A	192.168.1.0	Router 1	255.255.255.0
	Subnet C	192.168.3.0	Router 2	255.255.255.0

Table 3.1: Route Tables for Network Nodes in Figure 3.1.

Since the failure of a network cable connected to a router stops that router from properly routing packets, it can be considered a type of router failure. The initial RFTS design had no method of monitoring the cable connecting a router to a remote subnet. In the event of a cable failure such as that illustrated in Figure 3.2, the Router Fault Tolerance System on LAN 1 would detect the failure and inform all the hosts on LAN 1 to use router 2 to pass packets to LAN 2, while the RFTS on LAN 2 would not see the failure and keep router 1 as the active router for all hosts on LAN 2.

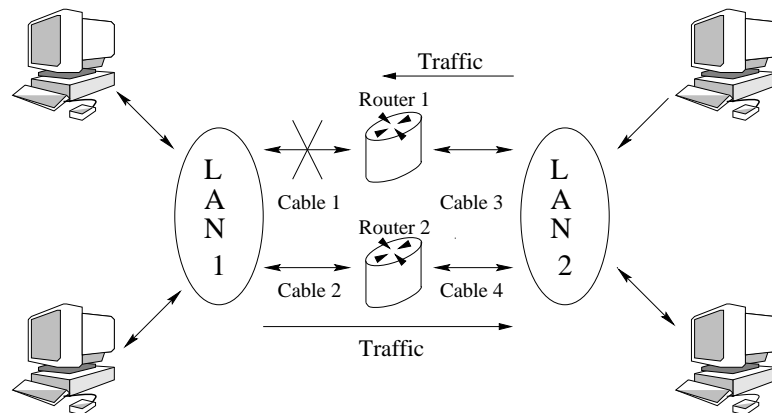


Figure 3.2: Cable failure stopping traffic between subnets.

The final problem with the initial concept of the Router Fault Tolerance System was a lack of means for efficient modification of the route table in Windows NT hosts. The initial implementation of the RFTS executed Windows NT's "route.exe" command to make changes to the route table, which contributed to the system's initial slow response to network failures. The solutions to these problems that were implemented in the RFTS design are described in the following sections.

3.2 Improved Scaling

ICMP echo request packets are useful tests of a router's functionality, since the proper reception of an echo request and transmission of an echo response require a good physical connection to the device as well as a properly configured and operating TCP/IP stack. Unfortunately, a network can quickly become congested if every host transmits echo requests to every router on that subnet. Each ICMP echo request and response is composed of the following information:

- Ethernet Header: 22 bytes
- Ethernet Trailer: 4 bytes
- IPv4 Header: 20 bytes (without options)
- ICMP Header: 8 bytes
- Optional ICMP Payload: 4 bytes
- Total: 58 bytes = 464 bits

Therefore, a network with ten hosts that constantly transmit echo requests to and receive echo responses from two routers at a rate of 100Hz will generate

$$R = (464\text{bits}) * (100\text{Hz}) * 2 * (10\text{hosts}) * (2\text{routers}) \approx 1.856\text{Mbps}$$

of network traffic to maintain fault tolerance.

3.2.1 Client/Server Design

The new Router Fault Tolerance System helps to eliminate this congestion by electing an “RFTS server” to keep track of the health of routers on a subnet and to inform all “RFTS clients” of router failures or restorations. The first host to run the Router Fault Tolerance System on a subnet becomes the RFTS server for that subnet. This host will use ICMP echo requests and responses to track router health as described above, as well as broadcast the IP address of the first properly functioning router listed in the RFTS configuration file to all hosts on the local subnet, as shown in Figure 3.3. The broadcast packet consists of the following information:

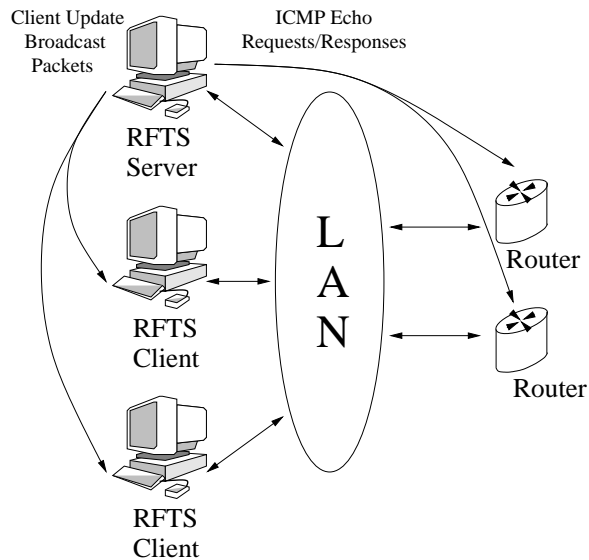


Figure 3.3: Client/Server Router Fault Tolerance System.

- Ethernet Header: 22 bytes
- Ethernet Trailer: 4 bytes
- IPv4 Header: 20 bytes (without options)
- Payload: 232 bytes
- Total: 278 bytes = 2224 bits

When transmitting ICMP echo requests, receiving ICMP echo responses, and transmitting client update broadcast packets at a rate of 100HZ, this scheme generates

$$R = (464\text{bits}) * (100\text{Hz}) * 2 * (2\text{routers}) + (2224\text{bits}) * (100\text{Hz}) \approx 0.408\text{Mbps}$$

of network traffic to provide fault tolerance, independent of the number of hosts that are utilizing the system.

The transmission rate of the ICMP echo request/reply packets must be high enough so as to allow for subsecond fault tolerance, and yet not so high as to overload the RFTS server. Higher rates provide for lower latency times when failures occur, since a RFTS server that interrogates routers more often will discover failures faster and can correct for them sooner. However, since the Windows NT 4.0 system timer has a granularity of ten milliseconds, the ICMP packets cannot be sent at rates higher than 100Hz. Furthermore, experimental programs showed that NT 4.0's TCP/IP stack becomes overloaded when transmitting several packets every ten milliseconds; the optimal router interrogation frequency for the Router Fault Tolerance System was found to be 50Hz, or one ICMP packet to each of two routers every 20 milliseconds.

3.2.2 Fault Tolerance of Client/Server

Although the client/server concept solved the problem of poor scaling, it introduced a new consideration: how to detect and recover from the failure of the RFTS server. In the Router Fault Tolerance System, the RFTS server is a single point of failure that must be eliminated in order for the system to be considered reliable.

The Router Fault Tolerance System uses the client update broadcast packets as "heartbeats" that inform the clients on a subnet that a server exists on the network and is properly monitoring the health of the routers on that network. If the number of skipped heartbeat packets exceeds a threshold (whose default is five packets), the clients on the subnet assume that the current server has failed, and the first client to time out assumes server status so that router monitoring is resumed.

Another problem that must be addressed in this design is partitioning of the network. When a network of hosts is divided into two or more groups that are isolated from each other, the host that was acting as the RFTS server for that subnet cannot reach clients on partitions other than its own. In this case, the hosts on partitions with no server will elect new servers and the Router Fault Tolerance System will resume normal operations.

A network that can be partitioned, however, can also be joined back together. This case will result in more than one RFTS server on a subnet or partition, as is shown in Figure 3.4. Therefore, each RFTS server listens for heartbeat packets that originate from a host other than itself. If another server is heard, the servers will enter an election mode in which each server waits a pseudo-randomly generated time period for one server to broadcast a message to claim server status for the subnet. If a server hears this message, it will relinquish server status and become a client (as happens to server 2 in Figure 3.5). Otherwise, it will win the election and remain the RFTS server for that subnet after waiting the full time period and broadcasting its message to all the other servers on that subnet (server 1 in Figure 3.5).

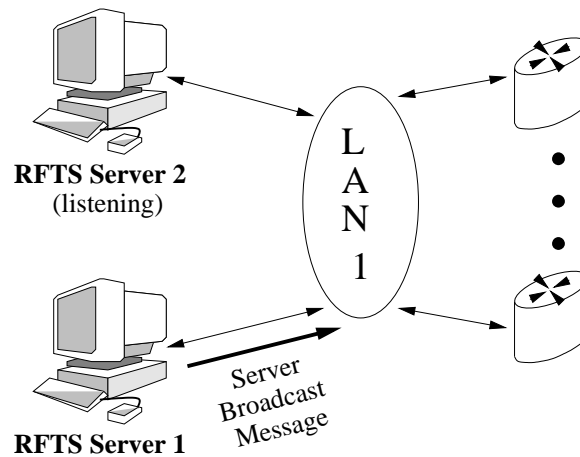


Figure 3.4: Multiple RFTS servers exist on a subnet, each listens for broadcast message for a pseudo-randomly generated time period, after which a broadcast message is sent.

3.3 Cross-Subnet Fault Tolerance

With the addition of the client/server functionality to its design, the Router Fault Tolerance System was able to detect and adapt to router failures on the subnet on which the system was running. However, the RFTS implementation with only the capabilities outlined so far could not detect and adapt to failures of communication between the router and the other subnet to which it was connected, the type of failure shown in Figure 3.2. This remote subnet problem was solved by providing servers with the ability to communicate with servers on other subnets, so that the basic Router Fault Tolerance System installation requires the RFTS software to be executed on both sides of the routers being tested, as

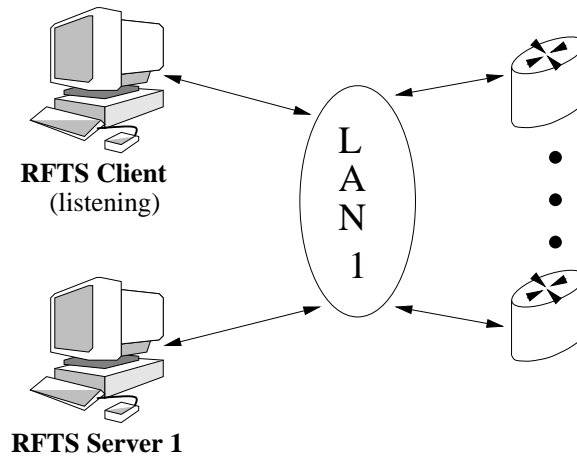


Figure 3.5: Server 1 transmits broadcast message after time period expires and becomes the RFTS server, while Server 2 hears broadcast message and becomes an RFTS client.

shown in Figure 3.6.

An RFTS server that detects a failed router first modifies its route table to establish a valid route to the remote subnet (Figure 3.7), then broadcasts a server-server communication packet to that remote subnet informing the remote server of the router failure (Figure 3.8). The remote server keeps track of each router's health on the remote subnet, and will use only routers that can communicate properly with both of the subnets to which they are connected (Figure 3.9). In order for this scheme to work properly, each server must also transmit a server-server communication packet when a router returns to an operational state.

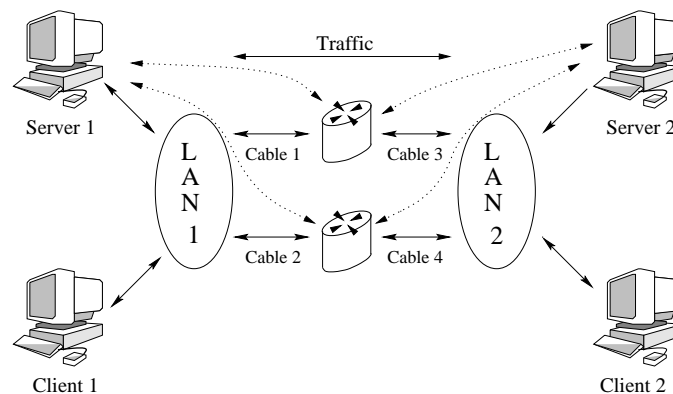


Figure 3.6: Servers continually interrogate all routers to obtain information on router and router connection status. Dotted lines indicate ICMP interrogation and response packet flow.

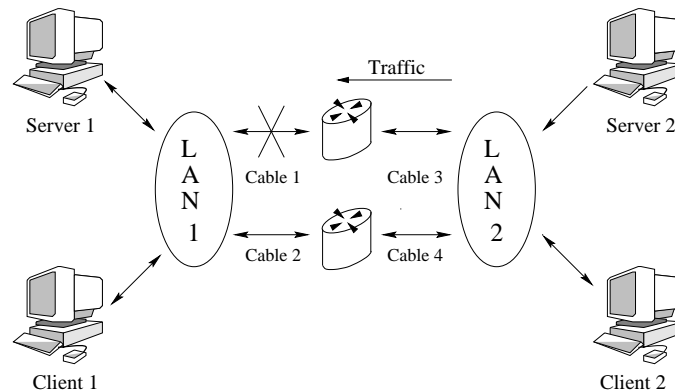


Figure 3.7: Server 1 detects failure of route from subnet 1 to router 1.

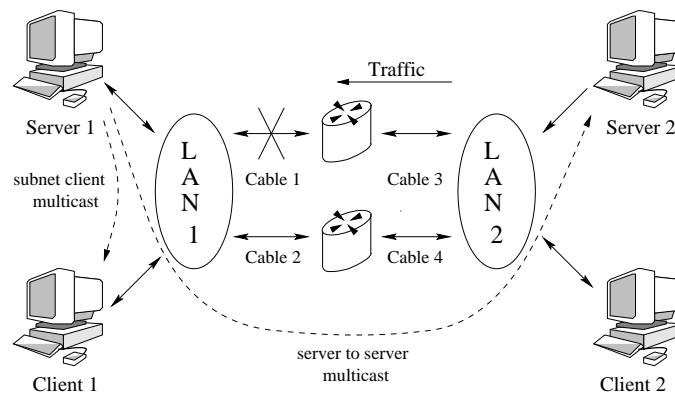


Figure 3.8: Server informs clients and other servers of required routing change.

All server-server communication packets are broadcast to their destination subnets to avoid the need to keep track of which host is acting as the RFTS server on each subnet. Subnet broadcasting is possible because each end host is assigned an IP address as well as a subnet mask. Each host on a subnet receives every packet destined for the host's IP address as well as for the broadcast address, which is the subnet address with the highest possible host address; for example, for the network with address 192.168.1.0 and subnet mask 255.255.255.0, the broadcast address is 192.168.1.255. Broadcast packets are the easiest way to handle fault tolerance of server-server communication, since the IP address of the recipient does not need to be known, and RFTS clients and other hosts will simply ignore the packets.

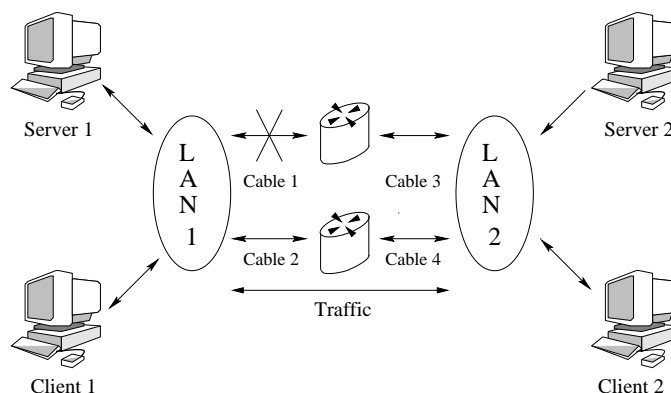


Figure 3.9: Full internetwork recovery from router cable failure.

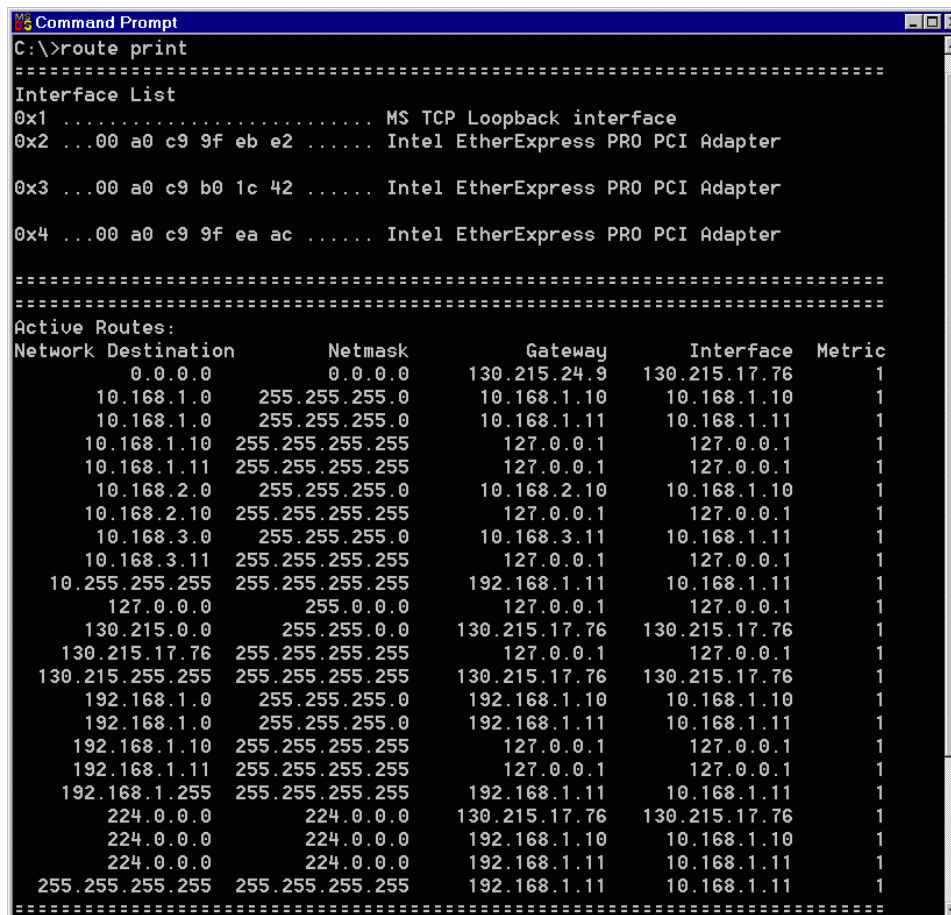
3.4 Modifying Route Table in Windows NT 4.0

The two requirements for a subsecond fault tolerance system are fast detection and correction of failures. The previous sections outline features of the Router Fault Tolerance System that allow for fast detection of failures; in order to correct for them, the RFTS must be able to quickly modify routes in a host's route table.

Windows NT 4.0 server contains a utility named "route.exe" (Figure 3.10), which uses Simple Network Management Protocol (SNMP) calls to modify the route table of the NT host. While this utility could have been used directly by the final Router Fault Tolerance System, the time to execute the utility caused too much of a delay to make this approach worthwhile: benchmarks run on a 300MHz Pentium II system showed this overhead to be approximately 120ms. Further research determined that direct SNMP calls were able to modify the route table fast enough to provide subsecond fault tolerance when used by the RFTS. The following sections describe SNMP and how it is used in the Router Fault Tolerance System.

3.4.1 Introduction to the Simple Network Management Protocol

The Simple Network Management Protocol, first proposed in February of 1988 [39, p.8], is designed to allow a workstation to gather information about an entire network as well as to control parameters on network nodes. This first version of SNMP, outlined in Request for Comment 1067, defined five messages, or Protocol Data Units (PDUs), which the management workstation can use to communicate with other network nodes: Get, Get-



```

C:\>route print
=====
Interface List
0x1 ..... MS TCP Loopback interface
0x2 .. .00 a0 c9 9f eb e2 ..... Intel EtherExpress PRO PCI Adapter

0x3 .. .00 a0 c9 b0 1c 42 ..... Intel EtherExpress PRO PCI Adapter

0x4 .. .00 a0 c9 9f ea ac ..... Intel EtherExpress PRO PCI Adapter

=====
Active Routes:
Network Destination    Netmask          Gateway         Interface    Metric
0.0.0.0                0.0.0.0          130.215.24.9    130.215.17.76  1
10.168.1.0             255.255.255.0    10.168.1.10    10.168.1.10   1
10.168.1.0             255.255.255.0    10.168.1.11    10.168.1.11   1
10.168.1.10           255.255.255.255  127.0.0.1      127.0.0.1     1
10.168.1.11           255.255.255.255  127.0.0.1      127.0.0.1     1
10.168.2.0             255.255.255.0    10.168.2.10    10.168.1.10   1
10.168.2.10           255.255.255.255  127.0.0.1      127.0.0.1     1
10.168.3.0             255.255.255.0    10.168.3.11    10.168.1.11   1
10.168.3.11           255.255.255.255  127.0.0.1      127.0.0.1     1
10.255.255.255        255.255.255.255  192.168.1.11   10.168.1.11   1
127.0.0.0             255.0.0.0        127.0.0.1      127.0.0.1     1
130.215.0.0           255.255.0.0      130.215.17.76  130.215.17.76  1
130.215.17.76         255.255.255.255  127.0.0.1      127.0.0.1     1
130.215.255.255       255.255.255.255  130.215.17.76  130.215.17.76  1
192.168.1.0           255.255.255.0    192.168.1.10   10.168.1.10   1
192.168.1.0           255.255.255.0    192.168.1.11   10.168.1.11   1
192.168.1.10          255.255.255.255  127.0.0.1      127.0.0.1     1
192.168.1.11          255.255.255.255  127.0.0.1      127.0.0.1     1
192.168.1.255         255.255.255.255  192.168.1.11   10.168.1.11   1
224.0.0.0             224.0.0.0        130.215.17.76  130.215.17.76  1
224.0.0.0             224.0.0.0        192.168.1.10   10.168.1.10   1
224.0.0.0             224.0.0.0        192.168.1.11   10.168.1.11   1
255.255.255.255       255.255.255.255  192.168.1.11   10.168.1.11   1
=====

```

Figure 3.10: Example execution of the “route.exe” program, outputting the current route table to the screen.

Next, Set, Get Response, and Trap[8].

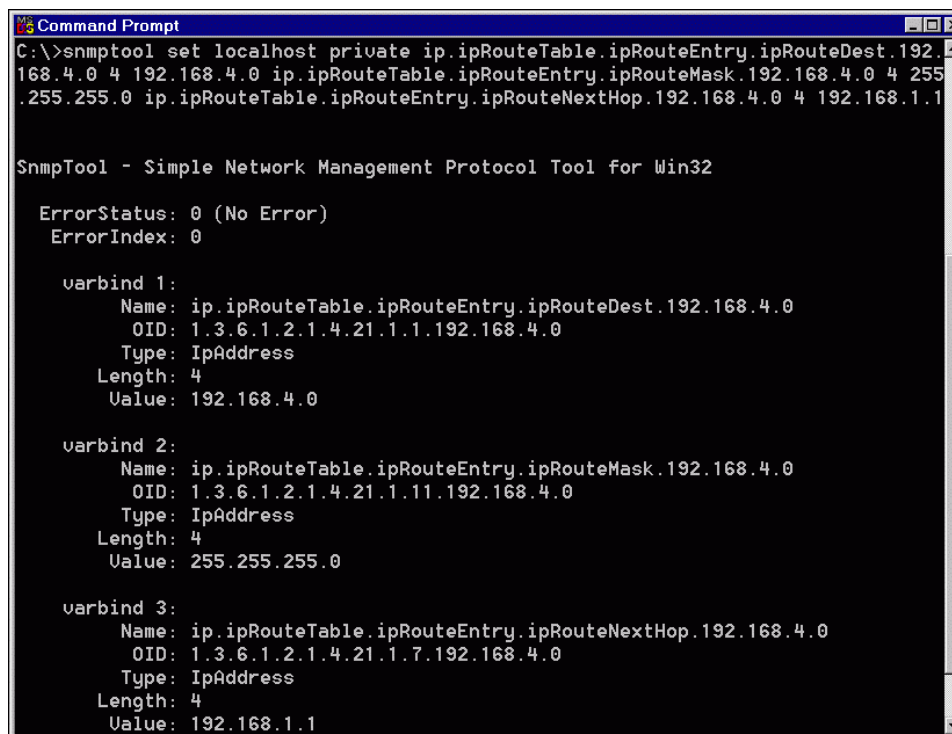
The management workstation sends requests to retrieve or set information using the Get, Get-Next, and Set commands. Network nodes transmit Get Response PDUs to the management workstation to transmit either requested information or the success or failure of a Set operation. SNMP Trap messages are “alarm messages” sent from a managed node to the management workstation to inform the network manager of certain network conditions, such as excessive transmission or reception errors.

Several other versions of SNMP exist today, including SNMPv2 and SNMPv3[39, pp.8-16]. These variations include more PDUs for more powerful management, increased number of objects to be managed, and better security. Only the basic capabilities of SNMPv1 were required to modify the route tables in this project.

3.4.2 Using SNMP to Modify NT's IP Route Table

Windows NT 4.0 includes a full implementation of SNMP and uses an application programmer interface (API) that allows applications running on an NT host to manage that host through SNMP. Since the SNMP API can read and write to the Windows NT route table, it provides a convenient way to add and remove routes. The examples shown in Figures 3.11 and 3.12 were performed using the SnmpTool, written by James D. Murray[27].

The process of adding a route to the route table requires four pieces of information: the destination subnet, the destination netmask, the gateway address, and the route metric, as shown in Figures 3.11 and 3.12. In this example, the SnmpTool is provided with the following information: the PDU to transmit (in this case, a Set PDU), the hostname to which the SNMP PDU will be transmitted (the local host), the community name (the password for the security system implemented in SNMPv1), and then the location in the management information database, data type, and data information of each object to be added.



```

C:\>snmptool set localhost private ip.ipRouteTable.ipRouteEntry.ipRouteDest.192.168.4.0 4 192.168.4.0 ip.ipRouteTable.ipRouteEntry.ipRouteMask.192.168.4.0 4 255.255.255.0 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop.192.168.4.0 4 192.168.1.1

SnmpTool - Simple Network Management Protocol Tool for Win32

ErrorStatus: 0 (No Error)
ErrorIndex: 0

varbind 1:
  Name: ip.ipRouteTable.ipRouteEntry.ipRouteDest.192.168.4.0
  OID: 1.3.6.1.2.1.4.21.1.1.192.168.4.0
  Type: IPAddress
  Length: 4
  Value: 192.168.4.0

varbind 2:
  Name: ip.ipRouteTable.ipRouteEntry.ipRouteMask.192.168.4.0
  OID: 1.3.6.1.2.1.4.21.1.11.192.168.4.0
  Type: IPAddress
  Length: 4
  Value: 255.255.255.0

varbind 3:
  Name: ip.ipRouteTable.ipRouteEntry.ipRouteNextHop.192.168.4.0
  OID: 1.3.6.1.2.1.4.21.1.7.192.168.4.0
  Type: IPAddress
  Length: 4
  Value: 192.168.1.1

```

Figure 3.11: Add a route to the Windows NT route table using direct SNMP calls.

```

Command Prompt
Active Routes:
Network Destination    Netmask          Gateway          Interface        Metric
-----
0.0.0.0                0.0.0.0          130.215.24.9    130.215.17.76    1
10.168.1.0             255.255.255.0    10.168.1.10    10.168.1.10     1
10.168.1.0             255.255.255.0    10.168.1.11    10.168.1.11     1
10.168.1.10            255.255.255.255  127.0.0.1      127.0.0.1       1
10.168.1.11            255.255.255.255  127.0.0.1      127.0.0.1       1
10.168.2.0             255.255.255.0    10.168.2.10    10.168.1.10     1
10.168.2.10            255.255.255.255  127.0.0.1      127.0.0.1       1
10.168.3.0             255.255.255.0    10.168.3.11    10.168.1.11     1
10.168.3.11            255.255.255.255  127.0.0.1      127.0.0.1       1
10.255.255.255         255.255.255.255  192.168.1.11   10.168.1.11     1
127.0.0.0              255.0.0.0        127.0.0.1      127.0.0.1       1
130.215.0.0            255.255.0.0      130.215.17.76  130.215.17.76   1
130.215.17.76          255.255.255.255  127.0.0.1      127.0.0.1       1
130.215.255.255        255.255.255.255  130.215.17.76  130.215.17.76   1
192.168.1.0            255.255.255.0    192.168.1.10   10.168.1.10     1
192.168.1.0            255.255.255.0    192.168.1.11   10.168.1.11     1
192.168.1.10           255.255.255.255  127.0.0.1      127.0.0.1       1
192.168.1.11           255.255.255.255  127.0.0.1      127.0.0.1       1
192.168.1.255          255.255.255.255  192.168.1.11   10.168.1.11     1
192.168.4.0            255.255.255.0    192.168.1.1    10.168.1.10     1
224.0.0.0              224.0.0.0        130.215.17.76  130.215.17.76   1
224.0.0.0              224.0.0.0        192.168.1.10   10.168.1.10     1
224.0.0.0              224.0.0.0        192.168.1.11   10.168.1.11     1
255.255.255.255        255.255.255.255  192.168.1.11   10.168.1.11     1

```

Figure 3.12: Windows NT route table with route to subnet 192.168.4.0 added.

When a route is removed from the table, only the destination subnet address is required (Figure 3.13), and the system removes all routes that contain that address (Figure 3.14). The information provided to the SnmpTool via the command is similar to that in the previous example; in this case, the Route Type is being changed to 2, or “invalid”.

```

Command Prompt
C:\>snmpset localhost private ip.ipRouteTable.ipRouteEntry.ipRouteType.192.168.4.0 1 2

SnmpTool - Simple Network Management Protocol Tool for Win32

ErrorStatus: 0 (No Error)
ErrorIndex: 0

varbind 1:
  Name: ip.ipRouteTable.ipRouteEntry.ipRouteType.192.168.4.0
  OID: 1.3.6.1.2.1.4.21.1.8.192.168.4.0
  Type: INTEGER
  Value: 2

```

Figure 3.13: Remove a route from the Windows NT route table using direct SNMP calls.

One weakness of the Microsoft SNMP API is its inability to report errors that occur while actually modifying the NT route table; the API reports errors that occur only in the transmission and reception of SNMP packets and the syntax of the PDUs contained

```

Command Prompt
Active Routes:
Network Destination    Netmask          Gateway          Interface        Metric
0.0.0.0                0.0.0.0          130.215.24.9     130.215.17.76    1
10.168.1.0             255.255.255.0    10.168.1.10     10.168.1.10     1
10.168.1.0             255.255.255.0    10.168.1.11     10.168.1.11     1
10.168.1.10            255.255.255.255  127.0.0.1       127.0.0.1       1
10.168.1.11            255.255.255.255  127.0.0.1       127.0.0.1       1
10.168.2.0             255.255.255.0    10.168.2.10     10.168.1.10     1
10.168.2.10            255.255.255.255  127.0.0.1       127.0.0.1       1
10.168.3.0             255.255.255.0    10.168.3.11     10.168.1.11     1
10.168.3.11            255.255.255.255  127.0.0.1       127.0.0.1       1
10.255.255.255         255.255.255.255  192.168.1.11    10.168.1.11     1
127.0.0.0              255.0.0.0        127.0.0.1       127.0.0.1       1
130.215.0.0            255.255.0.0      130.215.17.76   130.215.17.76   1
130.215.17.76          255.255.255.255  127.0.0.1       127.0.0.1       1
130.215.255.255        255.255.255.255  130.215.17.76   130.215.17.76   1
192.168.1.0            255.255.255.0    192.168.1.10    10.168.1.10     1
192.168.1.0            255.255.255.0    192.168.1.11    10.168.1.11     1
192.168.1.10           255.255.255.255  127.0.0.1       127.0.0.1       1
192.168.1.11           255.255.255.255  127.0.0.1       127.0.0.1       1
192.168.1.255         255.255.255.255  192.168.1.11    10.168.1.11     1
224.0.0.0              224.0.0.0        130.215.17.76   130.215.17.76   1
224.0.0.0              224.0.0.0        192.168.1.10    10.168.1.10     1
224.0.0.0              224.0.0.0        192.168.1.11    10.168.1.11     1
255.255.255.255        255.255.255.255  192.168.1.11    10.168.1.11     1

```

Figure 3.14: Windows NT route table with route to subnet 192.168.4.0 removed.

within. A condition could arise in which the SNMP API reports no errors, but the desired route is not added or removed from the route table. One way to avoid this condition is to check the status of the route table after each route addition or removal. The following C++ method is an example of a function that adds a route, and if no errors occur during the route addition, verifies that the route now actually appears in the route table:

```

// *****
// *
// * METHOD: routeMgr::RouteAdd( void )
// *
// * DESC: This method adds a route to the route table. The
// *       RouteParams structure contains the necessary routing
// *       information. All entries must be specified in the
// *       IP dotted decimal format. Thus, the configuration
// *       file should have entries in dotted decimal (of course,
// *       this restriction could be removed later).
// *
// *****
int RouteMgr::RouteAdd( struct routeParams *pParams )
{
    struct routeEntry routeEntry;
    struct routeEntry routeReturn;

```

```

bool    success = FALSE;
int     retVal  = TRUE;
int     numLoops = 0;

// Loop until the route has been successfully added.
while( (success == FALSE) && (numLoops < 5) )
{
    // Build the routeEntry structure
    routeEntry.ipRouteDest    = inet_addr(pParams->netAddr);
    routeEntry.ipRouteMask    = inet_addr(pParams->netMask);
    routeEntry.ipRouteNextHop = inet_addr(pParams->gateway);
    routeEntry.ipRouteMetric1 = 1;

    if(mib->SetRouteTable(&routeEntry) == FALSE)
    {
        // No error while adding route...check to see if it's actually there now.
        routeReturn.ipRouteDest = inet_addr(pParams->netAddr);
        if(mib->GetRouteTableEntry(&routeReturn) == FALSE)
        {
            // Verify that this route is the one we're looking for.
            if((inet_addr(pParams->netAddr) == routeReturn.ipRouteDest)
                && (inet_addr(pParams->netMask) ==
                    routeReturn.ipRouteMask) && (inet_addr(pParams->gateway) ==
                    routeReturn.ipRouteNextHop))
            {
                // We're done!
                success = TRUE;
                retVal  = FALSE;
            }
        }
    }
    else
    {
        // Error occurred while adding the route. Try adding it again.
        success = TRUE;
        retVal  = TRUE;
    }
    numLoops++;
}

return(retVal);
}

```

A benchmark program was written to determine how much time was required by the

SNMP API to add and remove a route from the route table. This program showed that the route table could be modified in under 10 milliseconds.

Chapter 4

Current Implementation of Router Fault Tolerance System

The network on which the Router Fault Tolerance System was implemented during this project was designed to test all aspects of the system. The Class A IP address block 10.x.x.x was split into two subnets, 10.168.1.0 and 10.168.4.0, for IP address assignments to the hosts and routers used during testing. One subnet consisted of two PCs connected to a Bay Networks four port 100BaseT switch, while the other subnet had three PCs connected to a Samsung SS6208 eight port 100BaseT switch. Four of the PCs in this test network were running Windows NT 4.0 Workstation, while one was running Windows NT 4.0 Server. The two subnets were connected by Cisco 2514 routers. Network loads were measured using a pair of HP-UX workstations that could be connected to either subnet (Figure 4.1). Table 4.1 lists a description of and the IP address assigned to each node in the RFTS test network, and Tables 4.2 and 4.3 show the configuration information for RFTS hosts on the 10.168.1.0 and 10.168.4.0 networks, respectively.

To simplify the cross-subnet fault tolerance scheme that was implemented in the Router Fault Tolerance System, each router was assigned the same IP address on both networks; for example, nftRouter1 was assigned 10.168.1.1 on the 10.168.1.0 network and 10.168.4.1 on the 10.168.4.0 network. This addressing scheme eliminates the need for the RFTS setup on each subnet to keep track of each router's remote IP address.

This test network provided the ability to test all aspects of the Router Fault Tolerance System: subsecond fault tolerance in the event of a failure in a router, router port, or router network cable, and robustness in the event of an RFTS server failure or subnet partitioning.

Node Name	Node Description	IP Address
Polaris	300 MHz Pentium II, NT 4.0 Server	10.168.1.10
Bastion	350 MHz Pentium II, NT 4.0 Workstation	10.168.1.20
Legacy	400 MHz Pentium II, NT 4.0 Workstation	10.168.1.30
Onslaught	350 MHz Pentium II, NT 4.0 Workstation	10.168.4.40
Exodus	350 MHz Pentium II, NT 4.0 Workstation	10.168.4.50
nftRouter1	Cisco 2514 Router	10.168.1.1 10.168.4.1
nftRouter2	Cisco 2514 Router	10.168.1.2 10.168.4.2

Table 4.1: Test Network Node Descriptions and IP Address Information.

Router Priority	Remote Subnet Address	Remote Subnet Mask	Router IP Address
1	10.168.4.0	255.255.255.0	10.168.1.1
2	10.168.4.0	255.255.255.0	10.168.1.2

Table 4.2: RFTS Configuration File Information for Polaris, Bastion, and Legacy.

Router Priority	Remote Subnet Address	Remote Subnet Mask	Router IP Address
1	10.168.1.0	255.255.255.0	10.168.4.1
2	10.168.1.0	255.255.255.0	10.168.4.2

Table 4.3: RFTS Configuration File Information for Onslaught and Exodus.

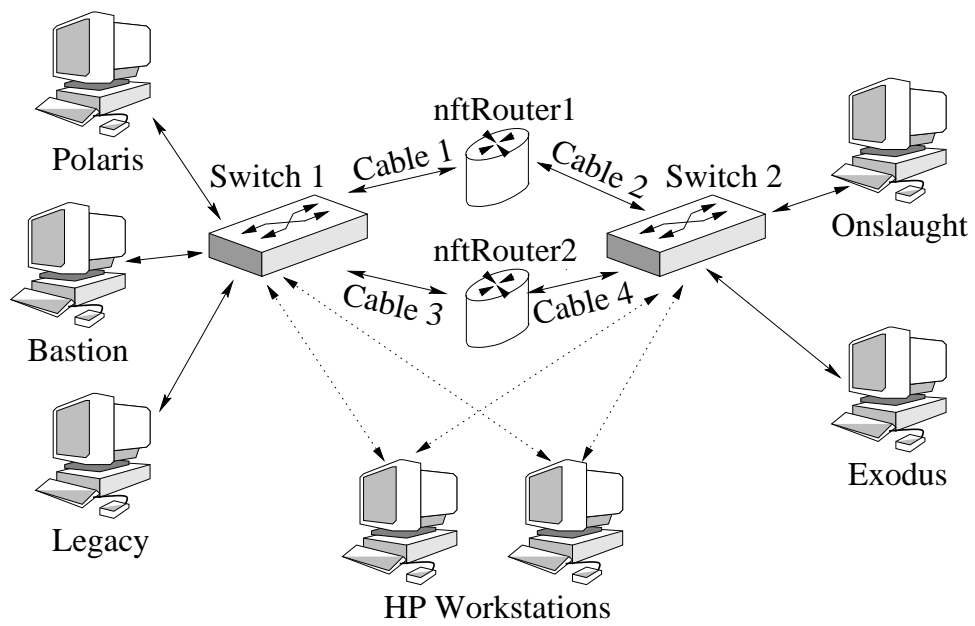


Figure 4.1: Router Fault Tolerance System Test Network

Chapter 5

Results of Router Fault Tolerance System Tests and Benchmarks

A client/server UDP benchmarking program was written to measure the subnet to subnet communication downtime that was allowed by the Router Fault Tolerance System. This benchmarking program sends timestamped UDP messages from a client on one machine to a server on a different subnet, which immediately returns the message to the client that sent it. Using this simple design, important data such as the downtime of the network and the average round trip time of packets can easily be obtained and then presented graphically.

The first benchmark test performed on the Router Fault Tolerance System consisted of removing and restoring connections between the routers and the two LANs of the test network described in Chapter 4, and observing the resulting changes in communication between the two LANs. The four cables marked in Figure 5.1 were disconnected and reconnected in a pattern that comprised up to 12 fault events.

Figure 5.2 shows the failure of a version of the system without server-server communications to preserve communications between the subnets. In this test, the benchmark packets were transmitted at 50ms intervals. Cable 1 was pulled at 5 seconds and restored at 10 seconds, cable 2 was pulled at 15 seconds and restored at 20 seconds, and cable 3 was pulled at 25 seconds. The long series of lost packets are the result of configurations like that shown in Figure 5.3, in which a router is the default gateway for one subnet, but its connection to the other subnet is broken. Because the RFTS servers on the two subnets were unable to communicate with each other, they were unable to determine if such a condition existed.

Once the server-server communication functionality was implemented, the system restored communications within 250-350ms of any connection break, as shown in Figure 5.4.

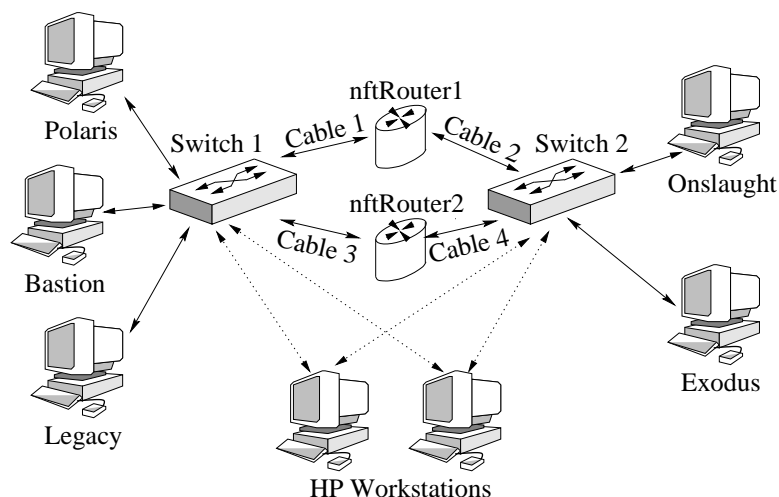


Figure 5.1: Test network with four test cables clearly marked.

The benchmark packets in this test were again transmitted at 50ms intervals, with cable 1 being pulled at 5 seconds and restored at 10 seconds, cable 2 pulled at 15 and restored at 20, cable 3 pulled at 25 and restored at 30, cable 4 pulled at 35 and restored at 40, then cable 1 again being pulled at 45 and restored at 50, and cable 2 pulled at 55 and restored at 60.

Although the version of the RFTS system tested in Figure 5.1 recovered communications within 350ms of a failure, the process of removing the default gateway and adding a new one on a particular host took approximately 120ms. Once SNMP was introduced to change a host's default gateway, the system recovered within 200ms of a cable failure, as will be shown. Figures 5.5, 5.6, and 5.7 show a benchmark test of the RFTS implementing both server-to-server communications and SNMP functionality. In these figures, test packets were sent at a frequency of 100Hz, and the network cables were pulled and restored in the same pattern as in Figure 5.4.

During the test a constant flow of traffic between the hosts was monitored, and the network cables were again pulled and restored with the same pattern as in the previous two tests. The average recovery time from a fault (time that a transmitted packet was delayed owing to a no available path condition) was found to be 88.19 milliseconds. The maximum recovery time was found to be 200 milliseconds.

The second test that the RFTS system underwent was a complete router failure test. During this test, a router was powered down, and the loss of communication time was

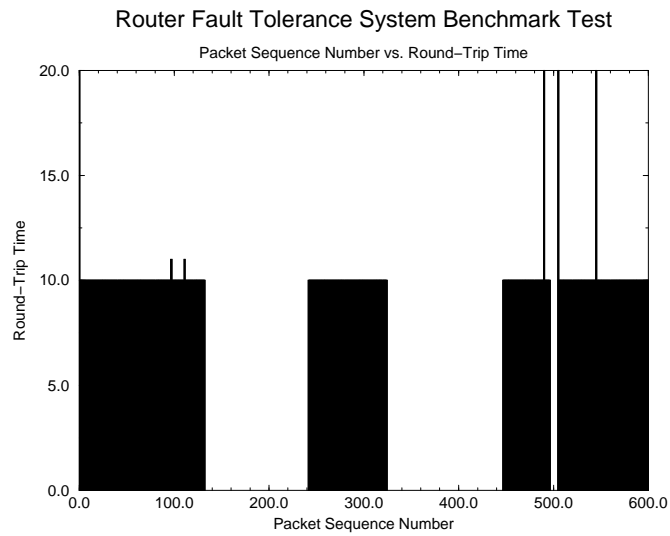


Figure 5.2: Router Fault Tolerance System Benchmark, No Server-Server Communication.

measured. In the results shown in Figures 5.8 and 5.9, test packets were sent at a frequency of 20Hz. Figure 5.8 shows the recovery time for the router failure, while Figure 5.9 shows the recovery for the router restoration. The system switches routers within 250ms of the router failure, and restores the higher priority router within 150ms of the router's revival.

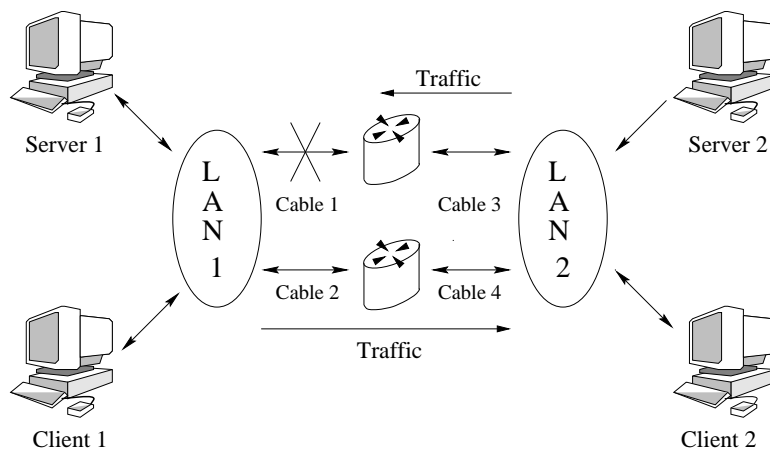


Figure 5.3: One sided network failure.

Router Fault Tolerance System Benchmark with Server-Server Communication

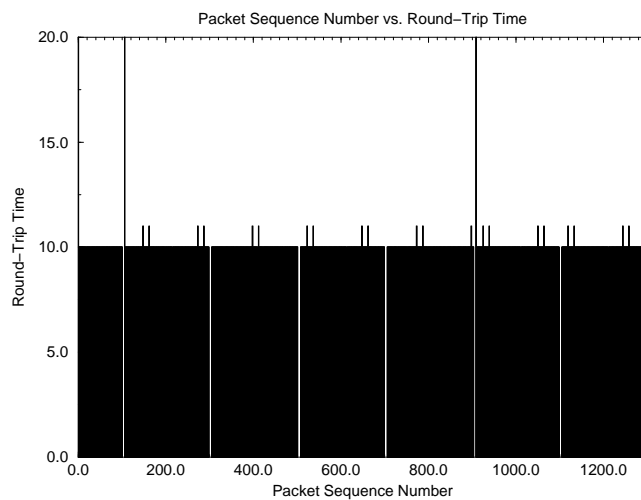


Figure 5.4: Router Fault Tolerance System Benchmark, SSC, No SNMP.

Router Fault Tolerance System Benchmark Test with SNMP and SSC

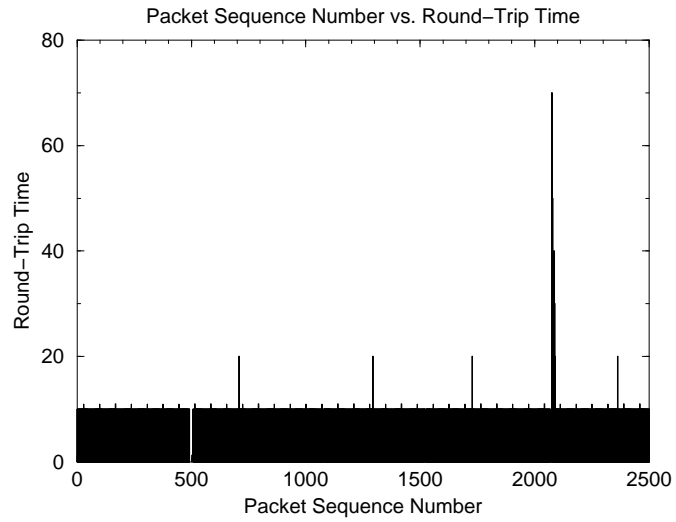


Figure 5.5: Router Fault Tolerance System Benchmark, SSC, SNMP, Part 1.

Router Fault Tolerance System Benchmark Test with SNMP and SSC

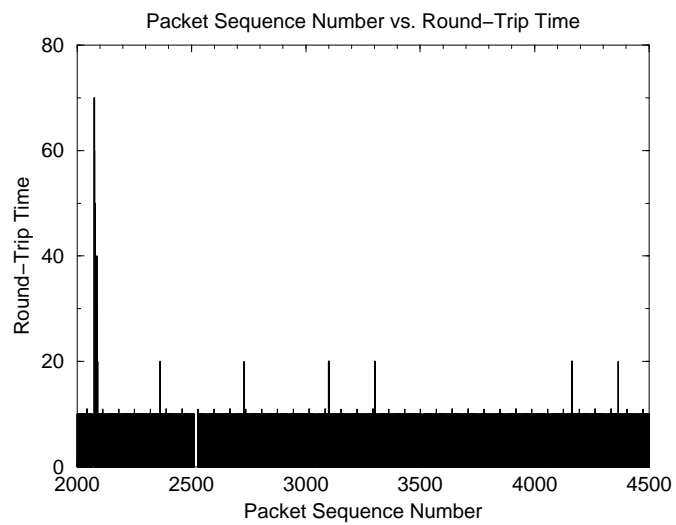


Figure 5.6: Router Fault Tolerance System Benchmark, SSC, SNMP, Part 2.

Router Fault Tolerance System Benchmark Test with SNMP and SSC

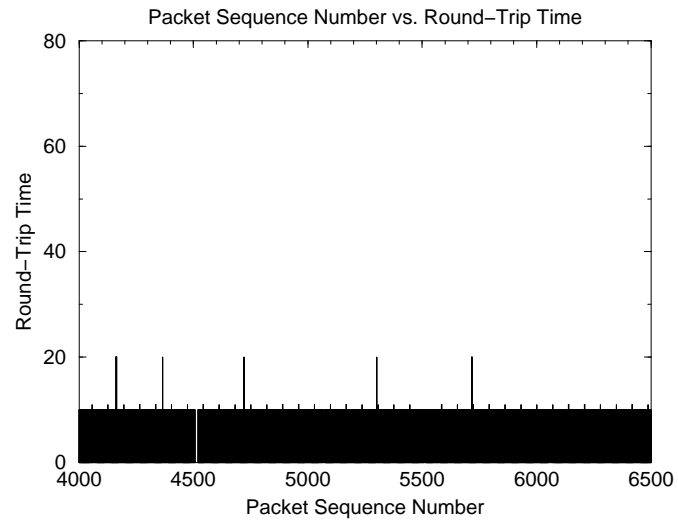


Figure 5.7: Router Fault Tolerance System Benchmark, SSC, SNMP, Part 3.

Router Fault Tolerance System Benchmark with Router Failure

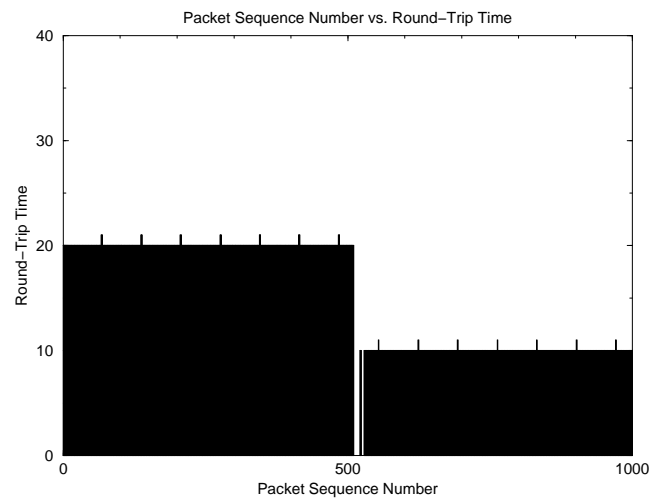


Figure 5.8: Router Fault Tolerance System, Router Failure Benchmark, Router Failure.

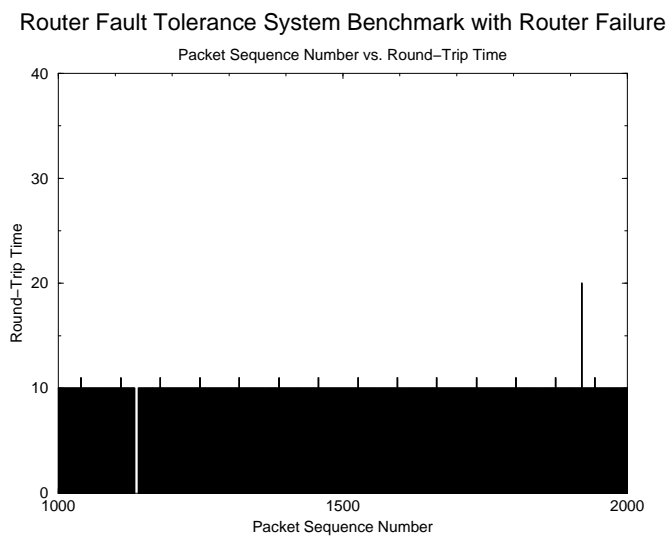


Figure 5.9: Router Fault Tolerance System, Router Failure Benchmark, Router Revival.

Chapter 6

Development of Switch Fault Tolerance System

Although the Router Fault Tolerance System provides a solution to the problem of subsecond response to network failures across subnets, it does not address the problem of equipment failures inside of a subnet. Just as the router connecting two non-fault-tolerant subnets is a single point of failure, network interface cards, cables, and switches all act as single points of failure for either a single host or for the entire subnet. The purpose of the Switch Fault Tolerance System (SFTS) is to eliminate these single points of failure and provide subsecond response to failure of these devices, ensuring virtually constant communications within a subnet while requiring no special modifications to application-layer software.

6.1 Initial Concept

Eliminating a single point of failure requires the addition of redundant systems that can assume the duties of the original when a failure occurs. In a local area network, this redundancy must apply to switches, the network cables connecting switches and hosts, and the network cards installed in the hosts. The SFTS was designed for a network in which each host is multiply homed, and each NIC in a host is connected to a unique switch, as shown in Figure 6.1.

Unfortunately, the attempt to introduce redundancy into a network such as this one introduces many problems. For example, if configured incorrectly, redundant switches will generate network floods in which one packet is transmitted from switch to switch until a physical connection is broken. An example of a topology that results in a network flood is

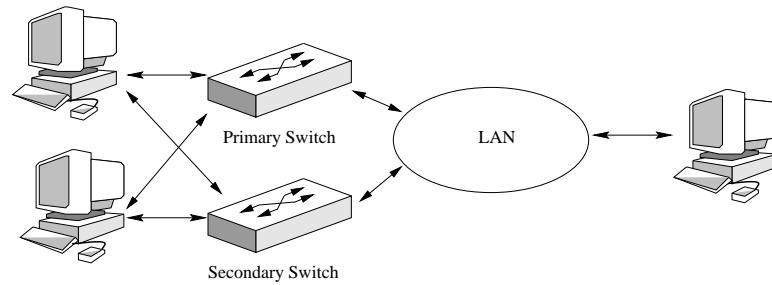


Figure 6.1: Switched Fault Tolerance System network cell.

shown in Figures 6.2, 6.3, 6.4, and 6.5, where a packet from one host propagates through the switched network until all switches are continuously transmitting the same information.

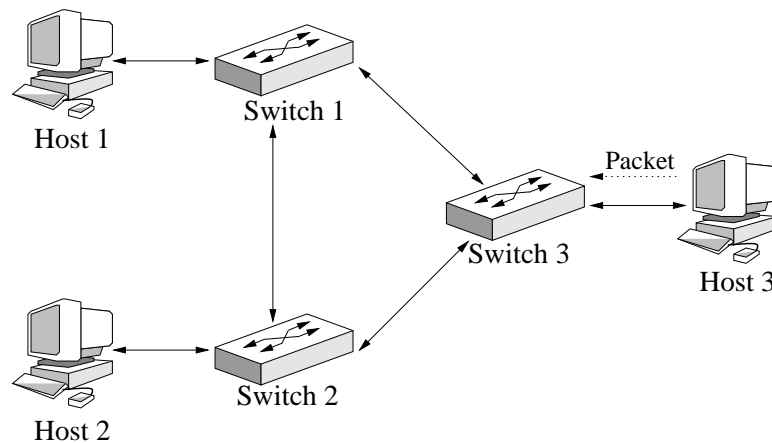


Figure 6.2: Host 3 on an improperly configured switched network transmits a packet to Switch 3.

Also, multiply homed hosts must choose only one network interface card through which packets will be transmitted and received. Although a host actually has little if any control over which NIC receives packets from the network, it must transmit packets through only one NIC, or else the other hosts on that network must be prepared to process duplicate packets. Duplicate packets unnecessarily increase the traffic on a network, as well as the load of each host that must determine whether a packet is a duplicate and should therefore be ignored.

Finally, in the event of a failure, a host must transmit and receive all packets through its secondary NIC without disturbing higher layer applications on itself or any other host on the network. Applications such as web browsers, terminal programs, or video conferencing

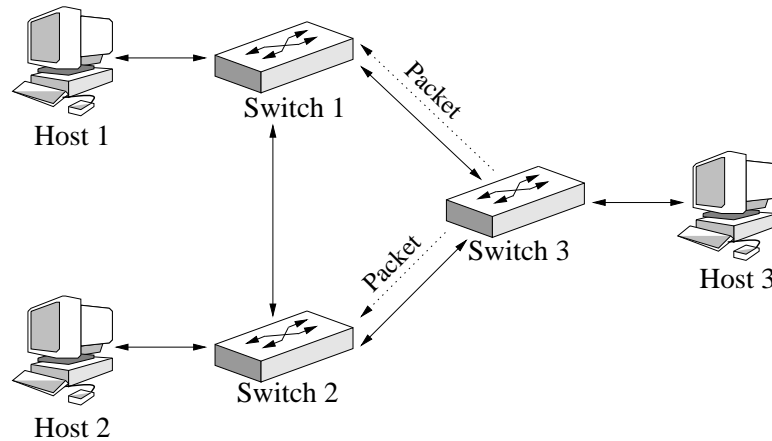


Figure 6.3: Switch 3 transmits the packet out all of its ports except for the one on which the packet was received.

software establish connections through the IP address of one NIC, and can maintain those connections through only that address. When the network card through which a host communicates with a subnet changes, the IP address through which its connections must be maintained also changes. To overcome this obstacle, either special application-layer software that can handle these changes must be used, or the fault tolerance system must be provided with the ability to modify a NIC's IP address, or at least make that address appear to have been modified.

6.2 DHCP-Based Switched Fault Tolerance System

The most difficult challenge encountered during the development of the Switched Fault Tolerance System was maintaining existing network connections while changing the NIC through which network traffic flows. The first proposed solution to this problem was to change the IP address of each NIC upon detection of a failure. This address change would make the NIC change transparent to all applications. Unfortunately, unlike UNIX systems such as Linux which can change the IP addresses of their NICs in real-time using applications such as "ifconfig," Windows NT 4.0 requires a system reboot in order to effect a change in a NIC's IP address. This solution is not acceptable in a sub-second network fault tolerance system.

NT 4.0 does support an internet standard for automatic configuration of a host's IP address, subnet mask, and gateway information, as well as other settings. The Dynamic

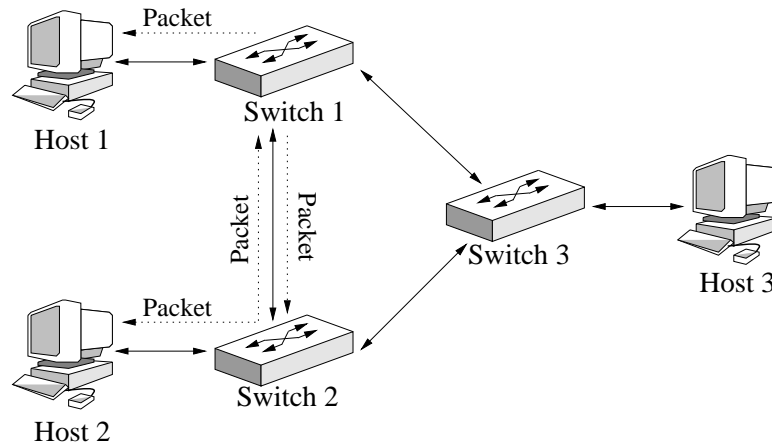


Figure 6.4: Switches 1 and 2 receive the packet, and again transmit it out all ports except for the one on which it was received. Note that the two switches are transmitting the packet to each other, as well as to Switch 3.

Host Configuration Protocol (DHCP) requires that a host be able to modify the IP addresses assigned to its NICs according to the availability of those addresses from a central server. The first implementation of the Switch Fault Tolerance System makes use of this protocol to control the NIC through which network traffic passes.

6.2.1 Introduction to DHCP

DHCP, defined in RFC2131[17], is a client/server system designed to provide an easier solution for managing IP addresses on a subnet, as well as for configuring systems on a network. Each DHCP server on a network is provided with information about available IP addresses as well as other host configuration information. Each end host that is configured to obtain its IP address through DHCP communicates with the server through broadcast packets.

An end host, or DHCP client, obtains configuration information from a DHCP server through a four step process. First, the DHCP client announces itself to all DHCP servers on the subnet through a DHCPDISCOVER message, which may contain configuration information such as a preferred network address[17] (Figure 6.6). Each DHCP server may transmit a DHCPOFFER message, which contains configuration information such as an IP address, address lease time (the amount of time that a client may hold a particular IP address), and default gateway (Figure 6.7). The client chooses one server, and notifies all servers on the subnet of its choice through a DHCPREQUEST message (Figure 6.8).

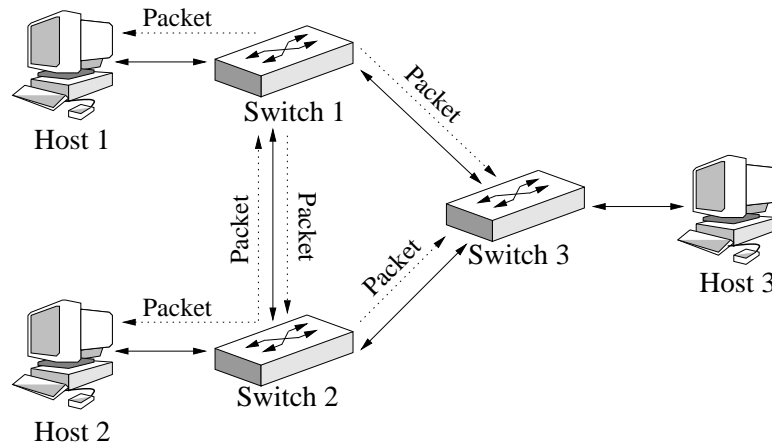


Figure 6.5: All switches are now continuously transmitting the original packet out of all ports.

The chosen server then finalizes the client's configuration by transmitting a DHCPACK message, at which point the client is considered to be configured and may participate on the network (Figure 6.9). Each host also has the option to release its address before the lease has expired by transmitting a DHCPRELEASE message to its DHCP server (Figure 6.10).

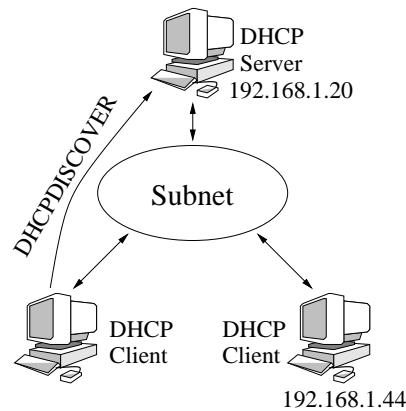


Figure 6.6: DHCP client broadcasts DHCPDISCOVER message to all hosts on subnet.

DHCP servers can be set up to associate certain IP addresses with certain hosts, using a unique identifier such as the Ethernet MAC address of the host's NIC, the manufacturer's serial number for that host, or a DNS name[17]. This association is used for networks in which each host must get the same IP address every time it is configured.

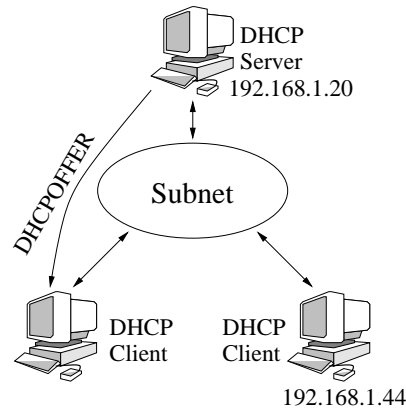


Figure 6.7: DHCP servers respond with DHCPOFFER message containing configuration information.

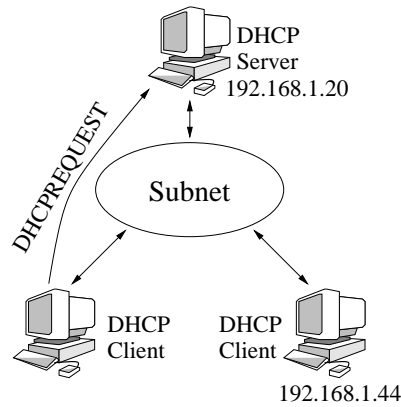


Figure 6.8: DHCP client chooses one server, broadcasting DHCPREQUEST message to notify all servers of its choice.

6.2.2 Description of DHCP-Based Switch Fault Tolerance

In the DHCP-Based version of the Switch Fault Tolerance System that was initially investigated, each mission critical host is configured to obtain IP address information through DHCP. Since a single DHCP server would introduce yet another single point of failure for a network, each mission critical host is provided with a DHCP server that responds to only those MAC addresses of the NICs installed in that host. The DHCP server is also provided with an IP address/MAC address association list which the SFTS software can change in the event of a NIC, switch, or cable failure.

When the DHCP-based Switch Fault Tolerance System is started, a healthy NIC is identified and assigned the primary IP address, and all redundant NICs are assigned redundant

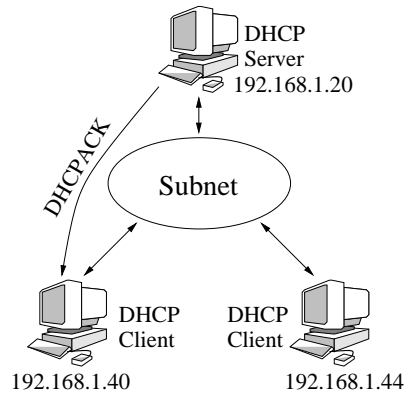


Figure 6.9: Chosen DHCP server acknowledges the client and finalizes configuration information with DHCPACK message.

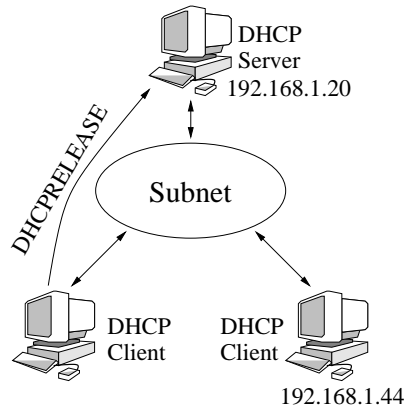


Figure 6.10: DHCP client relinquishes IP address and configuration information by sending DHCPRELEASE message to server.

IP addresses. When a failure is detected, the SFTS software identifies another healthy NIC, releases the primary IP address from the failed NIC, and reassigns it to the new active NIC, as shown in Figures 6.11 and 6.12.

6.2.3 Failure Points of DHCP-Based Switch Fault Tolerance

Although the DHCP-based switch fault tolerance scheme described above would seem to work, it suffers from a major problem which prevents it from being a candidate for use in the Switch Fault Tolerance System. The problem encountered is that Windows NT 4.0 terminates all connections through a particular IP address once that address has been released. If a switch fails during a video conferencing session, despite the action of the Switched Fault Tolerance System enabling new connections to be established from the

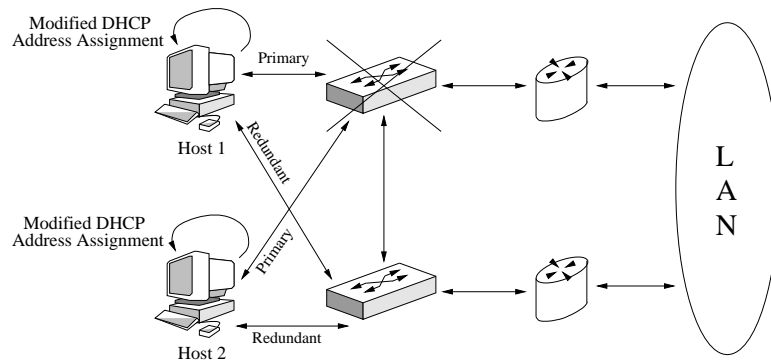


Figure 6.11: Using DHCP to Change IP Addresses in the Switched Network Fault Tolerance System, Part 1.

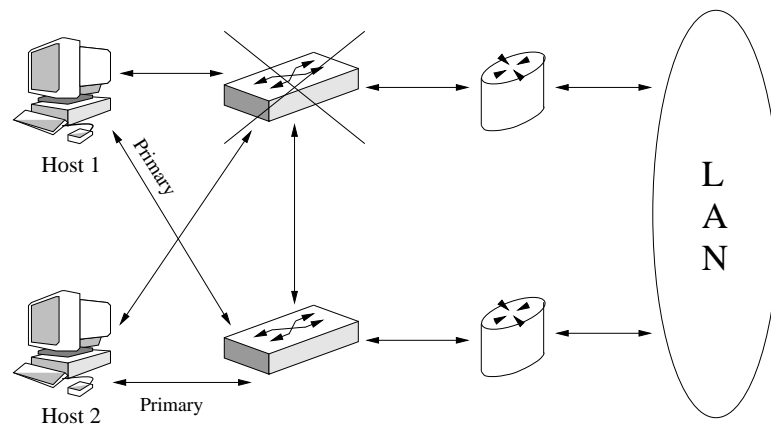


Figure 6.12: Using DHCP to Change IP Addresses in the Switched Network Fault Tolerance System, Part 2.

original IP address, any existing connections such as the conference are terminated.

An attempted resolution of this problem in the DHCP-based switch fault tolerance scheme revealed a new problem with the system. When a failure is detected, the redundant NIC can be assigned the primary IP address before the address has been released from the failed NIC. If a true failure occurs, a duplicate IP address is not detected on the network since the failed NIC has somehow been disconnected, and network connections are not terminated since the IP address through which the connection was initially established is always assigned to at least one NIC on that host. Unfortunately, false detections can and do occur, triggering duplicate IP addresses to exist on the network at the same time. When NT 4.0 detects duplicate addresses, it shuts down all NICs involved in the problem, rendering them unusable until the system is rebooted and the duplicate IP address condition

is resolved.

6.3 SNMP-Based Switch Fault Tolerance

The discovery of the DHCP-based switch fault tolerance system's fundamental flaws triggered the need for a new scheme that would control the flow of network traffic through a desired NIC while maintaining existing network connections. Further research showed that SNMP can be used to modify the route table of an end host to control through which NIC outgoing traffic passes.

6.3.1 Description of SNMP-Based Switch Fault Tolerance

The route table of every host configured to use IP networking contains an entry for the route to the host's local network, dictating through which NIC packets will be transmitted, as shown in Figures 6.13 and 6.14. An SNMP-based switch fault tolerance scheme must simply change the route to a host's local network to control which NIC is being used for packet transmission. However, receiving incoming packets is a different problem: a host cannot control to which NIC incoming packets are addressed. This problem can be resolved by informing all hosts on a network as to which NIC is available to receive packets for a given host.

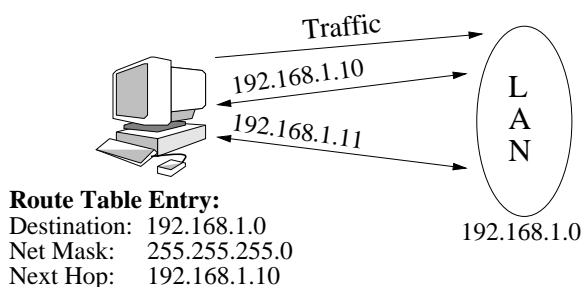


Figure 6.13: Host using network card with address 192.168.1.10 to transmit packets to the local subnet.

When the SFTS software is started on a mission critical host, it broadcasts a packet announcing all IP addresses assigned to the host, and indicating which address is being used as the primary (Figure 6.15). Every SFTS-enabled host that receives this packet adds a series of routes to its route table to redirect all traffic destined for a redundant IP address to instead be transmitted to the primary IP address, and then transmits its own IP address information to the new host (Figure 6.16). To include hosts that are not on the local subnet

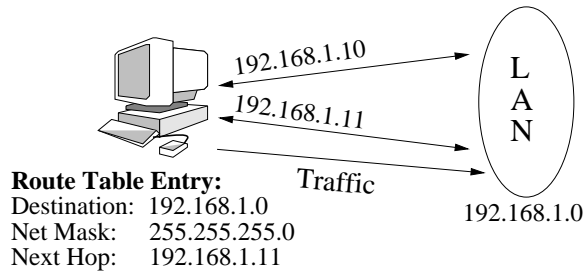


Figure 6.14: Host using network card with address 192.168.1.11 to transmit packets to the local subnet.

in the fault tolerance scheme, each SFTS host also transmits SNMP messages to each router on the subnet to establish routes to each redundant NIC through the primary NIC.

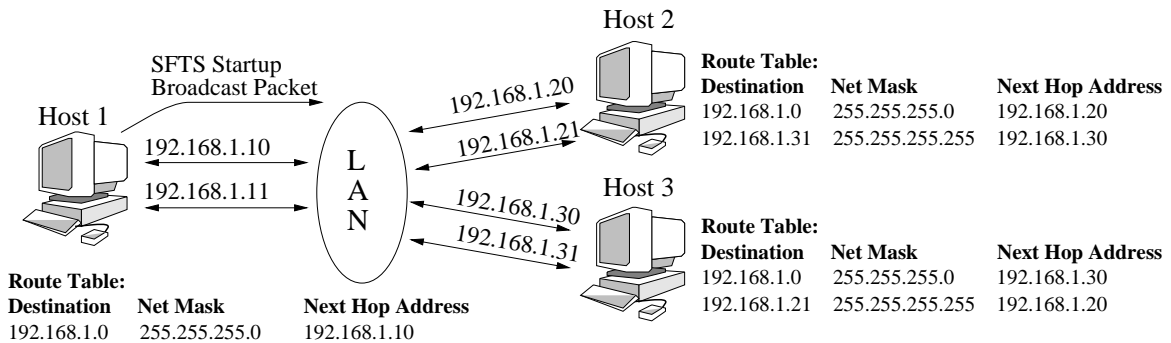


Figure 6.15: SFTS Host 1 starts up and transmits SFTS Startup packet to existing hosts.

When a failure is detected, an SFTS-enabled host modifies the route to its local subnet so that traffic is being transmitted out of a functional NIC (Figure 6.17), then broadcasts a packet to all other SFTS hosts with updated information about the primary and redundant IP addresses (Figure 6.18). The other hosts then modify all routes involving the affected IP addresses to reflect the new primary IP address information. Again, the host that suffers the failure also transmits SNMP messages to each router to modify the routes to its primary and redundant NICs that were established at startup. This route change renders the failure transparent to all hosts on remote networks.

6.3.2 Issues with SNMP Implementations in Commercial Routers

The SNMP management information database (MIB), which defines the “variables needed for monitoring and control of various components of the Internet,” [23] allows for the modification of a router’s route table using the protocol, but many router manufactur-

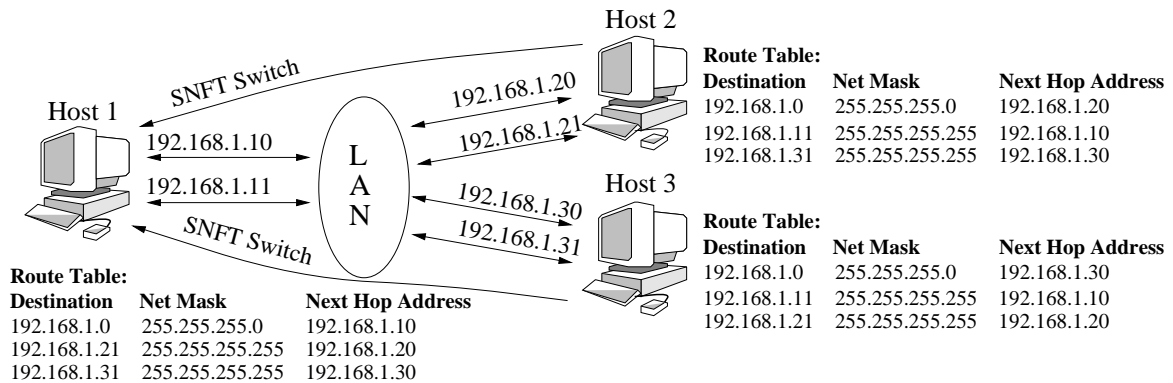


Figure 6.16: Existing SFTS hosts process SFTS Startup packet and respond with SFTS Switch packet.

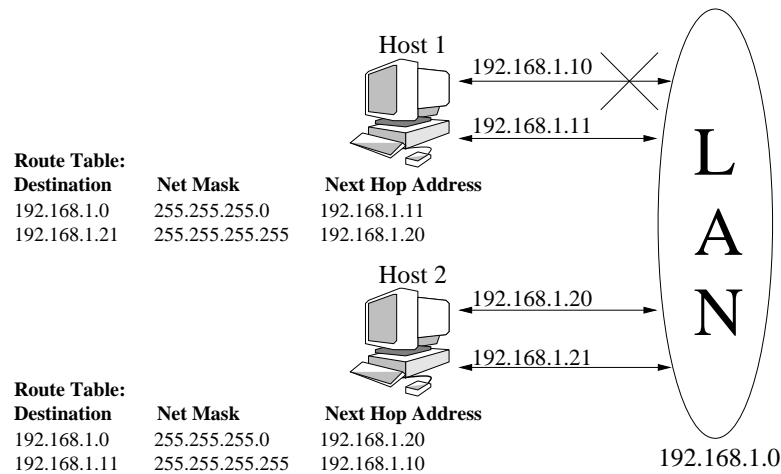


Figure 6.17: Cable failure detected, Host 1 changes local route table.

ers have chosen not to implement the standard for various reasons in spite of the IAB Policy Statement listed in section 2 of RFC1156:

“Not all groups of defined variables are mandatory for all Internet components...What IS mandatory, however, is that all variables of a group be supported if any element of the group is supported.” [23, p.2]

For example, the Cisco 2514 routers that were included as part of the RFTS test network could not be used for the Switch Fault Tolerance System because the routers do not support modification of their route tables through SNMP to avoid security holes, according to the Cisco technical support hotline. This implementation is in violation of the SNMP standard, since these routers use other variables in the IP group, of which the route table is a member.

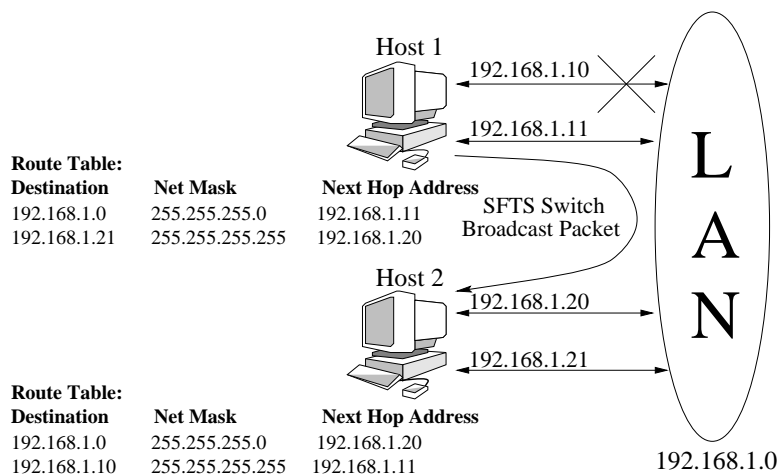


Figure 6.18: Host 1 broadcasts SFTS Switch message, which Host 2 uses to update its route table and correct for the cable failure.

A series of routers manufactured by Ascend Communications, however, only partially supports the SNMP standard definition for the IP route table. The SNMP implementation in these routers does not return an error when all writable variables of an IP route table entry were set, but fails to add the entry to the route table. Because the Ascend routers also use other variables in the IP group, this implementation is again in violation of the IAB Policy Statement in the SNMP standard.

The Netopia R9100 EN-WAN routers that were used in this project do fully support the standard in their SNMP implementations. These routers can be configured to accept SNMP messages from only certain IP addresses to prevent unauthorized users from changing the current configuration and potentially breaking into a network.

The Switch Fault Tolerance System's use of SNMPv1 may open systems without the precautions used by the Netopia R9100 routers to security holes; however, SNMPv1 contains only rudimentary security measures, and was useful for the design of a working prototype of the SFTS because of the simplicity of its implementation. In a production version, however, other versions of SNMP that include better security such as the encryption of SNMP messages would need to be considered, especially in systems such as those in financial or military institutions that may be prone to many intrusion attempts.

6.3.3 NIC Tester Mechanism

In order to reliably detect failures in a NIC, network cable, or switch, the basic SFTS network cell is divided into three subnets, as shown in Figures 6.19, 6.20, and 6.21. Each NIC in each host is given two IP addresses, one belonging in the main SFTS subnet, and one belonging in a NIC tester subnet, indicated as Subnets B and C in Figures 6.20 and 6.21. The 10.168.1.x address enables the NIC to transmit and receive packets to all other hosts and routers. Subnets B and C allow the NIC tester to constantly monitor the switch and network cabling connected to each NIC.

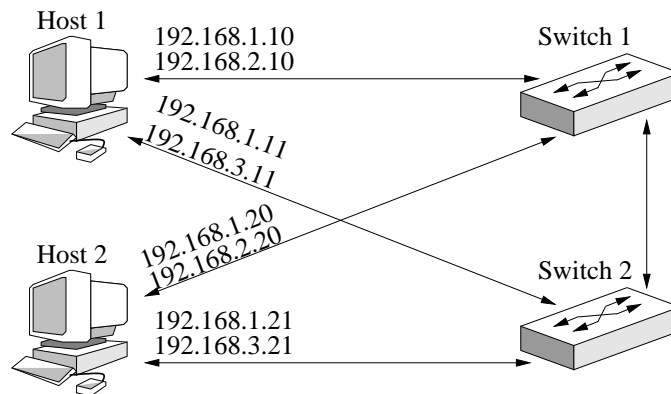


Figure 6.19: SFTS network cell. Every host and every switch belongs to the main SFTS subnet, which is 192.168.1.0 in this example.

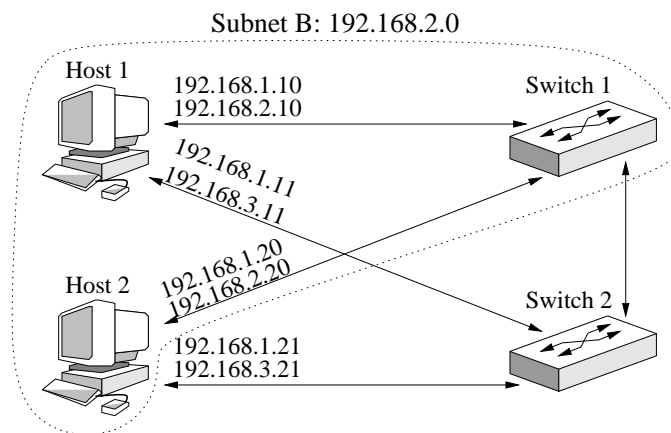


Figure 6.20: NIC Tester Subnet B in a basic SFTS cell contains one NIC in each SFTS host.

The NIC tester relies on the concept of subnet broadcasting in order to test the network

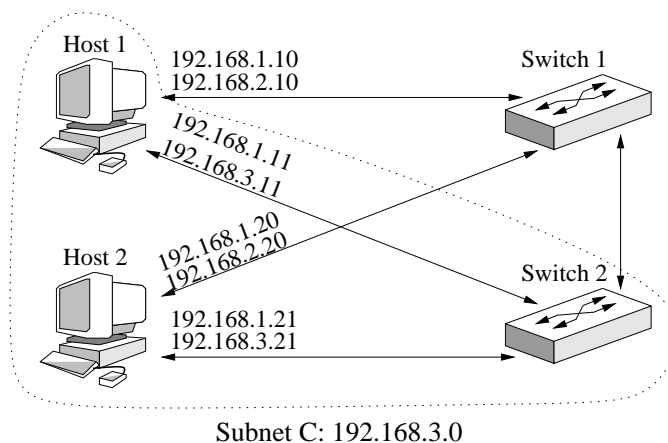


Figure 6.21: NIC Tester Subnet C in the SFTS cell contains the other NIC in each SFTS host.

cards, cables, and switches in a LAN. A subnet's broadcast address is the subnet's IP address with the host number bits set to 1. For example, to broadcast a packet to a class C network with IP address and 192.168.1.0 and subnet mask 255.255.255.0, a host would use the broadcast address 192.168.1.255. Because each host in the SFTS network cell belongs to three subnets, it will receive broadcasts from three broadcast addresses; in the example illustrated in Figure 6.19, the each host will receive packets address to 192.168.1.255, 192.168.2.255, and 192.168.3.255. However, because of the way IP addresses are assigned during the configuration of an SFTS-enabled network, any packet that is broadcast to 192.168.2.255 is transmitted through Switch 1 and received by NIC 1 on every host, and any packet that is broadcast to 192.168.3.255 is transmitted through Switch 2 and received by NIC 2.

Figure 6.22 contains an example NIC tester cell, which contains one switch and one NIC in each end host on the network. The NIC tester software is a client/server program that tests NICs through the transmission of UDP heartbeat packets, which the server broadcasts to all IP addresses in the NIC tester subnet every 20 milliseconds. Since a Windows NT 4.0 host that broadcasts a packet to the host's subnet receives the packet whether or not the NIC actually transmitted the data, two servers are required to adequately test the servers' NICs. A client or server that receives one stream of heartbeat packets on a NIC accepts that NIC as a properly functioning candidate to be the primary NIC for that host.

When the NIC tester client receives fewer than two streams of heartbeat packets, it declares the NIC in question to have failed, then enters into a negotiation with all other

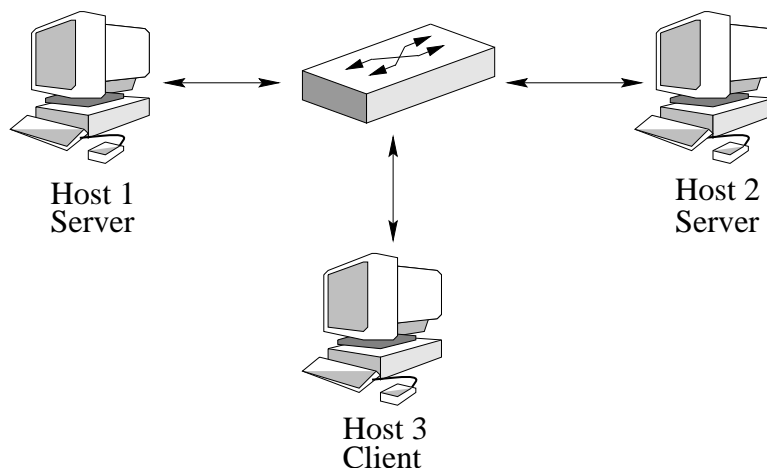


Figure 6.22: Network Interface Card tester cell.

clients on that subnet to become a server. The NIC tester implementation for this project was found to perform optimally if the heartbeat packets are transmitted at a frequency of 50Hz, and a server is considered to have failed if more than two heartbeats are missed by the clients.

The negotiation algorithm for choosing new NIC tester servers is a five-step process illustrated in Figures 6.23, 6.24, and 6.25. First, the algorithm waits a pseudo-randomly generated time for new servers to appear on the subnet (host 2 in Figure 6.23). If enough hosts have announced their eligibility to become servers, then the algorithm exits and the host remains a client (host 2 in Figure 6.24), otherwise it broadcasts a packet to all hosts on the test subnet to announce its eligibility to be a new server (hosts 3 and 5 in Figure 6.23). Another pseudo-randomly generated time is spent listening for servers to commit, or actually start serving, for the subnet (host 5 in Figure 6.24). If enough servers have committed after this time expires, then the algorithm exits and the host remains a client (host 5 in Figure 6.25), otherwise the algorithm broadcasts a packet to announce that the host has begun serving (host 3 in Figure 6.24), and the NIC tester server code is started (host 3 in Figure 6.25). The pseudo-randomly generated times for the NIC tester implementation in this project were less than or equal to 100 milliseconds, a value chosen because of good system response during testing.

Although this NIC, network cable, and switch testing scheme requires two servers, it reliably detects all failures except for switch interconnect cable failures (described in the section) while minimizing network traffic; only two servers are transmitting heartbeat packets,

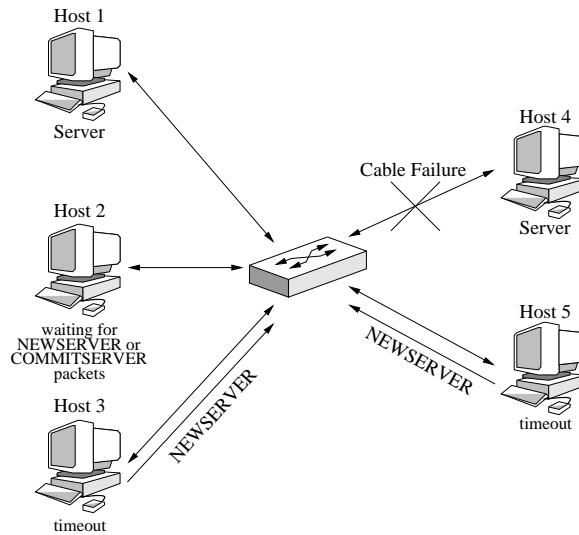


Figure 6.23: The NIC Tester server on Host 4 is down because of a cable failure, hosts 2, 3, and 5 enter into the server negotiation mode. Hosts 3 and 5 transmit NEWSERVER packets to announce their eligibility to become servers.

no matter how many other hosts are on the subnet.

6.3.4 Switch Interconnect Cable Failure

The SFTS as described here corrects for failures of switches, NICs, and the cables connecting switches to NICs, but it does not recover from certain situations in which the cable connecting the two switches in a LAN fails (Figure 6.26). To avoid this kind of failure, every host that receives a NIC change packet from another SFTS-enabled host will modify its active NIC so that every host on the subnet is communicating through the same switch (Figures 6.27 and 6.28). This solution ensures that a failure of the switch interconnect cable will not affect communications within an SFTS-enabled subnet, and leads directly to the method of integrating the Switch Fault Tolerance System with the Router Fault Tolerance System outlined in the final chapter of this report.

6.3.5 Advantages and Disadvantages of SNMP-Based SFTS

The SNMP-based Switch Fault Tolerance System has several good features:

- the switching of active NICs is transparent to every node outside the local subnet
- the SNMP functionality can be used to maintain connectivity not only to routers, but also to hosts that are not running the SFTS software

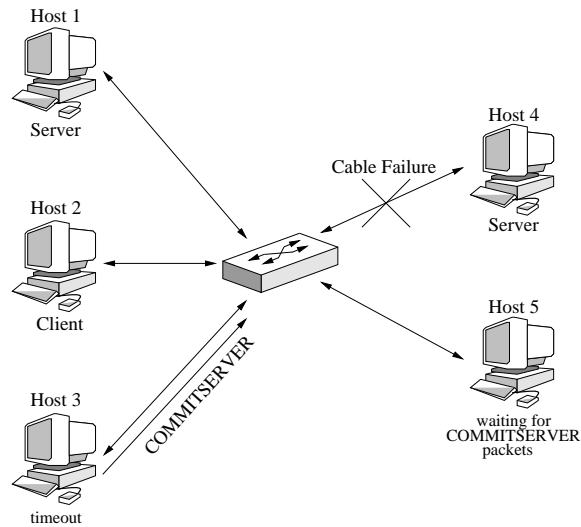


Figure 6.24: Since there are plenty of hosts eligible to become servers, Host 2 remains a client. Host 3 transmits a COMMITSERVER packet to announce it is becoming a server.

- when integrated with the RFTS, fault tolerance can be provided with respect to NICs, network cables, switches, and routers.

The concepts behind the fault tolerance schemes can also easily be ported to other operating systems, creating a platform-independent fault tolerance network.

However, the SNMP-based SFTS on Windows NT 4.0 does not provide fault tolerance for Windows file and printer sharing. Since the Windows NetBEUI protocol does not use IP networking[24], the IP route table does not affect the flow of NetBEUI packets. Unless all but one NIC has NetBEUI disabled, Windows detects itself through the multiple NICs and returns a “computer name already in use” error.

The Novell Inc., website contains product descriptions for Novell NetWare 4.2[29], NetWare 5.1[30], and NetWare Small Business Suite 5[31], which list all these products as capable of using IP for network communications, as opposed to NetWare’s more traditional use of IPX as included in NetWare version 3.2[28]. Therefore, Novell NetWare version 4.2 and higher should benefit from the Switch Fault Tolerance System, although these products were not tested for compatibility during the course of this thesis.

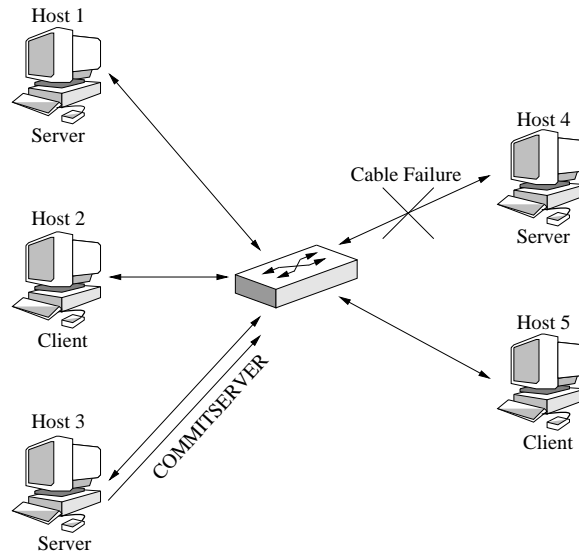


Figure 6.25: Host 5 remains a client, since there are now two active servers on the test subnet.

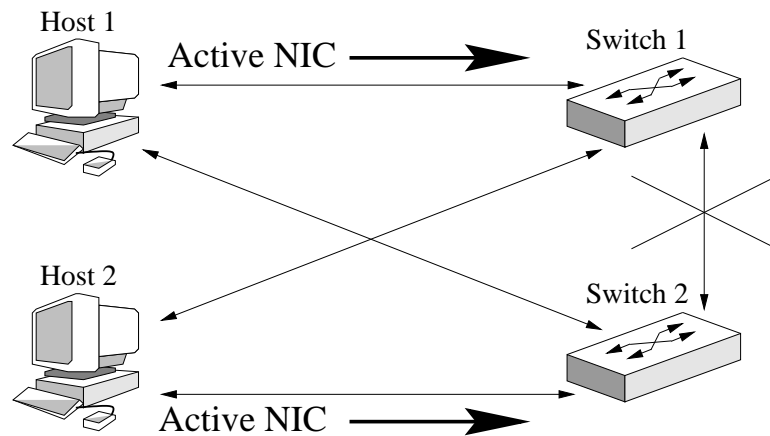


Figure 6.26: If the active NICs of two SFTS-enabled hosts are connected to different switches and the switch interconnect cable fails, the SFTS will not detect the failure, and the two hosts will be unable to communicate.

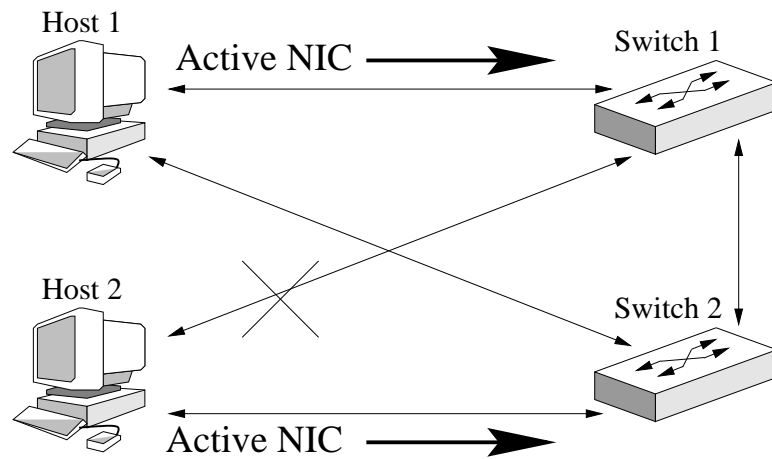


Figure 6.27: An SFTS-enabled host detects a failure and changes its active NIC.

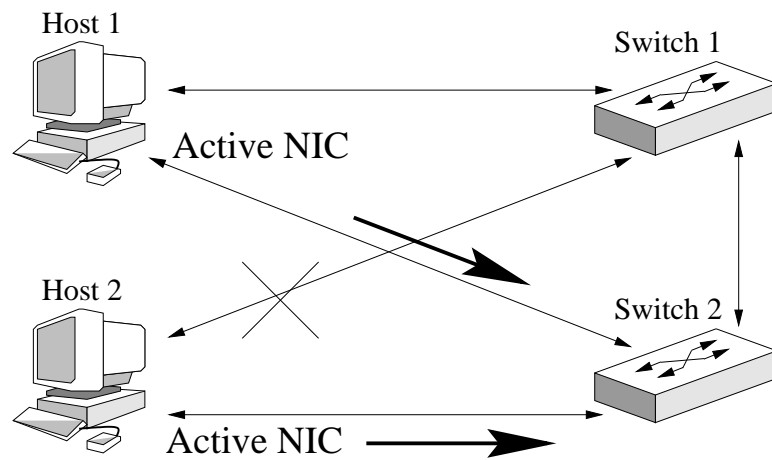


Figure 6.28: All other hosts on the subnet change their active NICs in response to the NIC change packet, so all hosts are now communicating through the same switch.

Chapter 7

Current Implementation of Switch Fault Tolerance System

The network on which the Switch Fault Tolerance System was implemented during this project was designed to test the NIC tester's robustness, as well as the loss of service time both within a switched Ethernet and across a router when a NIC, network cable, or switch failure occurs. The Class A IP address block 10.x.x.x was again used; 10.168.1.0 and 10.168.4.0 were the main subnets of both networks, while 10.168.2.0 and 10.168.3.0 were the NIC tester subnets of 10.168.1.0, and 10.168.5.0 and 10.168.6.0 were the NIC tester subnets of 10.168.4.0 (Figure 7.1). As in the Router Fault Tolerance System's test network, one switched Ethernet cell consisted of two PCs connected to a Bay Networks four port 100BaseT switch, while the other cell had three PCs connected to a Samsung SS6208 eight port 100BaseT switch, with the two cells again connected by Netopia R9100 EN-WAN routers. The PCs used for this test network are the same as those described in Table 4.1. Table 7.1 lists the configuration information for all PC hosts and routers used in the SFTS test network. Figure 7.2 is a picture of the Cisco and Netopia routers, and Bay Networks and Samsung switches that were used during the testing of the RFTS and SFTS.

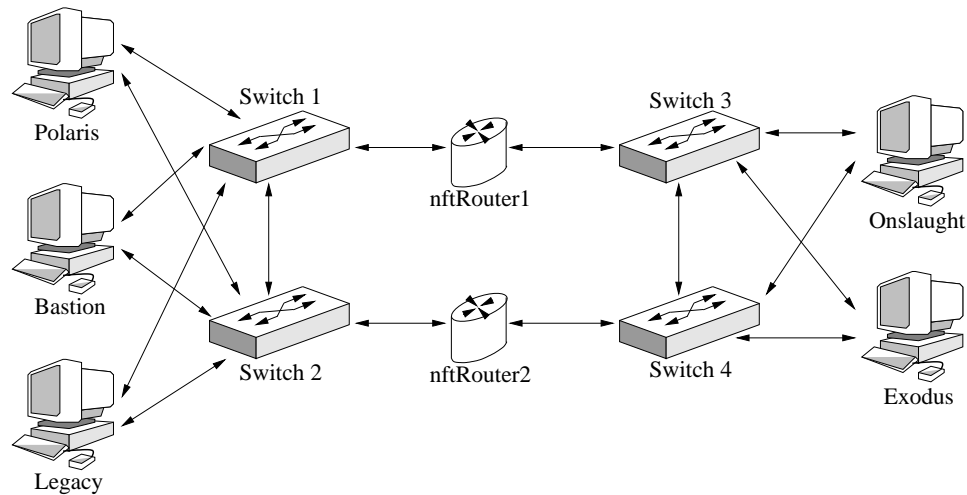


Figure 7.1: Switch Fault Tolerance System Test Network.

Node Name	Node Type	IP Addr Information		Router Addr Information		
		Main IP Addr	NIC Tester IP Addr	IP Addr	Subnet Addr	Subnet Mask
Polaris	PC Host	10.168.1.10	10.168.2.10	10.168.1.3	10.168.4.0	255.255.255.0
		10.168.1.11	10.168.3.11	10.168.1.4	10.168.4.0	255.255.255.0
Bastion	PC Host	10.168.1.20	10.168.2.20	10.168.1.3	10.168.4.0	255.255.255.0
		10.168.1.21	10.168.3.21	10.168.1.4	10.168.4.0	255.255.255.0
Legacy	PC Host	10.168.1.30	10.168.2.30	10.168.1.3	10.168.4.0	255.255.255.0
		10.168.1.31	10.168.3.31	10.168.1.4	10.168.4.0	255.255.255.0
Onslaught	PC Host	10.168.4.40	10.168.5.40	10.168.4.3	10.168.1.0	255.255.255.0
		10.168.4.41	10.168.6.41	10.168.4.4	10.168.1.0	255.255.255.0
Exodus	PC Host	10.168.4.50	10.168.5.50	10.168.4.3	10.168.1.0	255.255.255.0
		10.168.4.51	10.168.6.51	10.168.4.4	10.168.1.0	255.255.255.0
nftRouter3	Netopia R9100 Router	10.168.1.3				
nftRouter4	Netopia R9100 Router	10.168.1.4				

Table 7.1: Test Network Node Descriptions and IP Address Information.



Figure 7.2: Pictures of routers and switches that were used during the testing of the Switched Fault Tolerance System.

Chapter 8

Results of Switch Fault Tolerance Tests and Benchmarks

The testing of the Switch Fault Tolerance System involved two steps: measuring the time required by the NIC tester mechanism to detect a NIC failure, and that required by the main algorithm to complete the switch of network communications to an alternate NIC. The times required to declare a NIC as having failed and to choose a new NIC tester server were measured using Windows NT's system timer.

Several total loss of service times of the Switch Fault Tolerance System were measured using the UDP benchmark tester that was developed for testing of the Router Fault Tolerance System. The first two tests use the UDP benchmark program to demonstrate the effectiveness of group NIC change packets in providing tolerance for failures of the switch interconnect cable. The next two tests show the need for SNMP communication with the routers to establish routes to failed or redundant NICs through the primary NIC. Finally, the last test of the Switch Fault Tolerance System again uses the UDP benchmark program to show the SFTS' ability to correct from a switch failure.

8.1 NIC Tester Tests

The NIC tester has two basic tasks: the identification of a failed NIC tester server, and the choosing of a new NIC tester server for replacement of a failed one. A NIC tester client considers a NIC tester server to have failed when a preset threshold of missed heartbeat packets is exceeded. Once this condition is met, the client marks the NIC being tested as failed. The missed heartbeat threshold of the version of the NIC tester used during all of the Switch Fault Tolerance System tests described in this chapter was set at compile time

to be two packets. Therefore, a failed NIC will be detected within 40 milliseconds of the actual failure.

The NIC tester server negotiation process involves two wait periods of random length that do not exceed 100 milliseconds each. The negotiation also requires 100 milliseconds to transmit and receive packets to and from other NIC tester servers. Therefore, a new NIC tester server will be chosen between 100 and 300 milliseconds after a NIC has been detected as having failed.

8.2 SFTS Tests Before Implementation of Group NIC Change Packets

The initial version of the Switch Fault Tolerance System was able to recover from failures of each NIC in every SFTS-enabled host, of each cable connecting a switch to a NIC, and of the switches themselves. Figure 8.1 shows the successful recovery of NICs installed in the hosts and of cables connecting the switches and NICs. Network cables were removed and reconnected with the hosts' NICs in a pattern that resulted in twelve fault events, with disconnections occurring every ten seconds starting at five seconds, and reconnections occurring every ten seconds starting at ten seconds. The SFTS recovered from failures in no longer than 300 milliseconds.

This version had difficulty recovering from faults in the cable connecting the two switches. In situations where the active NICs of every host on a subnet were not synchronized, disconnecting this cable interrupted communications in a manner that the SFTS could not detect, resulting in total failure of the system, as shown in Figure 8.2.

8.3 SNFT Tests After Implementation of Group NIC Change Packets

The Switch Fault Tolerance System was then modified to include group NIC change packets, which ensured that every host on a subnet would switch its primary NIC in response to a failure detected by one host. The recovery time from failures of the switch-to-NIC cables was not affected by this change, as shown in Figure 8.3, while the new system suffered no loss of communication within the subnet when a failure occurred in the switch interconnect cable.

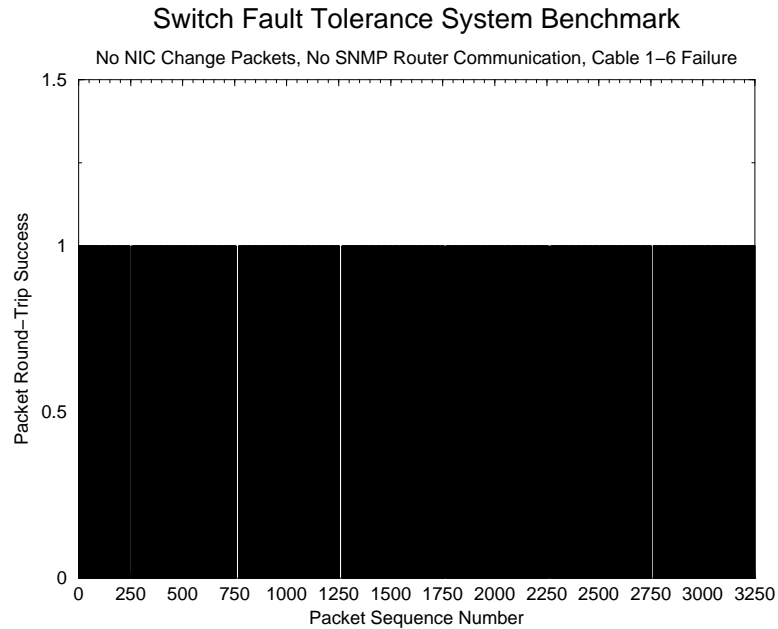


Figure 8.1: SNFT Tests Before Implementation of Group NIC Change Packets, Failures in Cables 1 through 6.

8.4 SNFT Tests Across Routers

In order to provide true fault tolerance within a subnet, the SFTS-enabled hosts have to be able to communicate with the routers in the network. Since the UDP benchmark tester does not require that the client's IP address remain constant, it returned results indicating that the system worked adequately across routers. However, the Switch Fault Tolerance System failed to preserve a connection-oriented application such as FTP, since the routers did not update routes to the failed NICs of hosts through new primary NICs.

After SNMP communication with routers was implemented into the SFTS, both connection and connectionless protocols behaved as the UDP benchmark tester shows in Figure 8.4, with all failures resulting in loss of communications times of less than 300 milliseconds. However, this implementation of the Switch Fault Tolerance System cannot detect and recover from failures in the routers themselves. An integrated network fault tolerance system that includes the fault tolerance capabilities of the RFTS and the SFTS will be discussed in the next chapter.

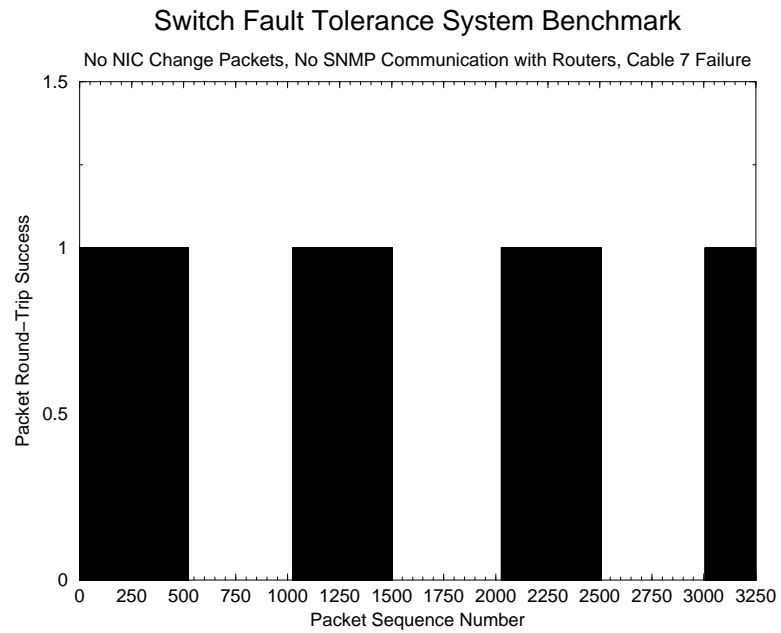


Figure 8.2: SNFT Tests Before Implementation of Group NIC Change Packets, Failure in Cable 7.

8.5 SFTS Recovery from Total Switch Failure

The final test of the Switch Fault Tolerance System involved power cycling each switch in the network and measuring the response of the entire system to the failure. As is shown in Figure 8.5, the SFTS recovers within 320 milliseconds.

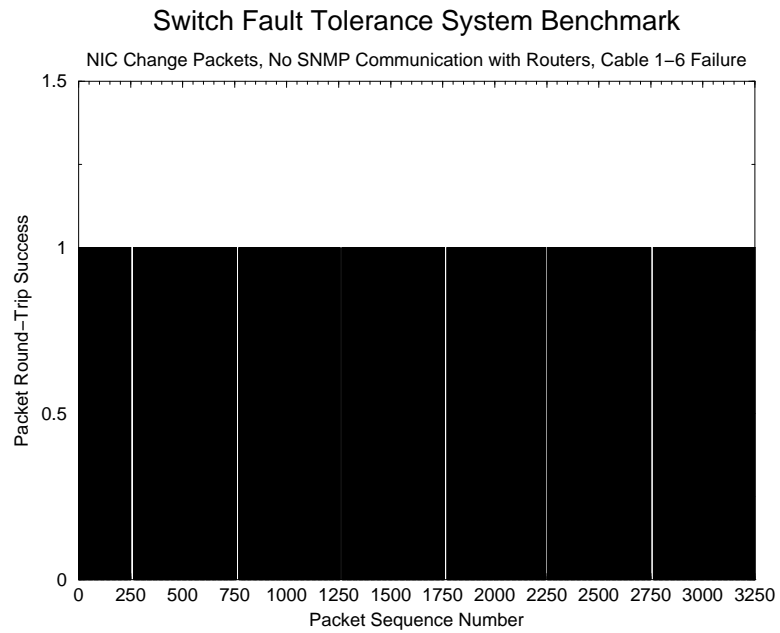


Figure 8.3: SNFT Tests After Implementation of Group NIC Change Packets, Failures in Cables 1 through 6.

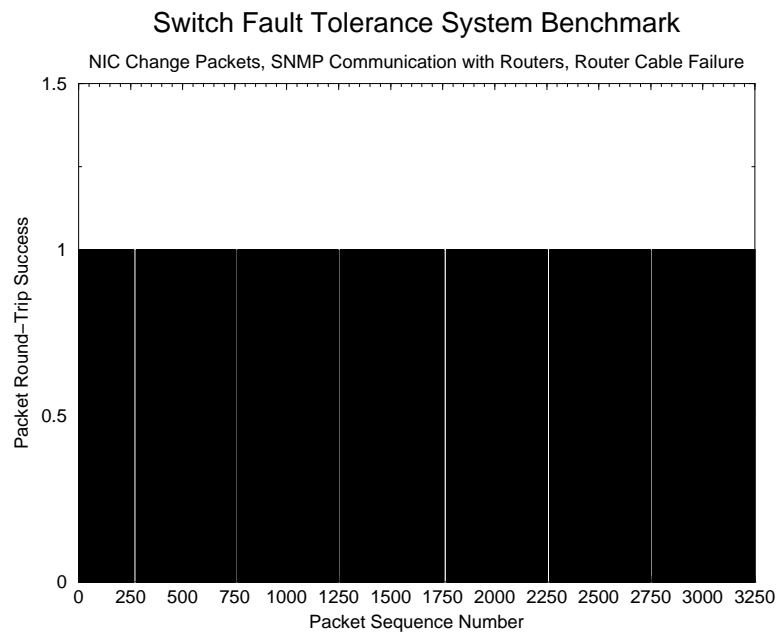


Figure 8.4: SNFT Tests After Implementation of SNMP Communication with Routers, Router Cable Failure.

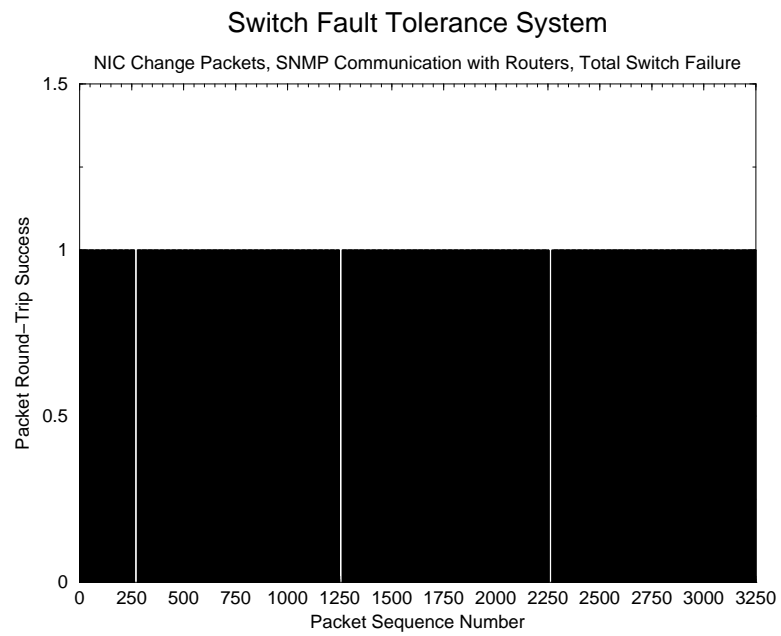


Figure 8.5: SFTS Tests After Implementation of NIC Change Packets and SNMP Communication with Routers, Total Switch Failure.

Chapter 9

Conclusions

The results of this project show that subsecond fault tolerance in switches, network cables, NICs, and routers can be achieved using an end-host, software-based system. Fast recovery times are indeed obtainable using the Router and Switch Fault Tolerance Systems, but the final versions of the Systems must be integrated in order to provide a total fault tolerance solution.

9.1 Current State of the Network Fault Tolerance System

The Router Fault Tolerance System uses SNMP to manage the IP route tables of all hosts on a subnet in response to the health of the routers which are connected to the subnet. Scalability is addressed through the client/server design of the system, which ensures that as additional hosts are added only one machine assumes the role of server. This design is resistant to server failures and network partitions; either of these events trigger an election process in which all hosts participate, resulting in at most one server per network. Communication between the servers on separate LANs allows the RFTS to recover from cable failures that isolate a router from a remote subnet.

The Switch Fault Tolerance System provides redundancy of NICs, network cables, and switches for every host in a LAN, and uses SNMP to modify the IP route table to redirect packets out of a new NIC when any one of these components fails. The system also uses SNMP to modify the routers' IP route tables so that packets from remote subnets can be directed to a functioning NIC at their destinations, but this can occur only in routers that fully support the SNMP standard as outlined in RFC 1157[10]. The SFTS' NIC tester monitors the health of all NICs, network cables, and switches through a specialized

addressing scheme for hosts and the use of broadcast heartbeat packets.

Unfortunately, each of these systems at present works independently of the other, and cannot benefit from the fault tolerance provided by the other. When a switch failure occurs, a router failure also occurs since the router can no longer be accessed, as can be seen in Figure 9.1. The two systems cannot simply be run at the same time, since a route table entry includes an interface through which outgoing packets are transmitted, and the routes established through the Router Fault Tolerance System would have to be aware of the currently active NIC in order to ensure successful communications with the other hosts on the network. However, in a network such as that in Figure 9.1 in which a router is connected to each switch, the RFTS can be used to test the health of the routers as well as that of the switches.

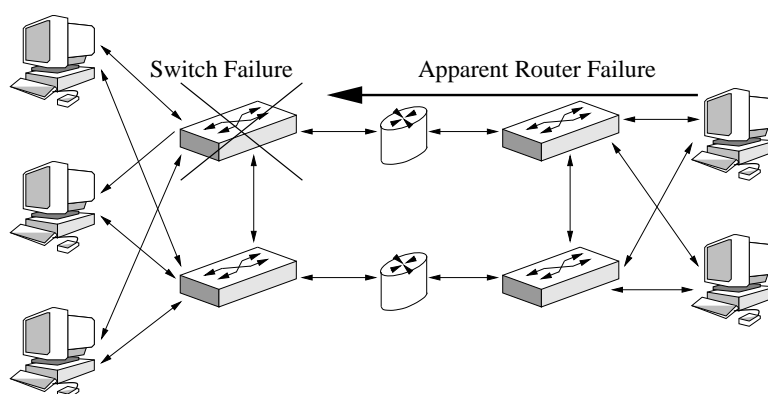


Figure 9.1: The failure of a switch in the Switch Fault Tolerance System also causes the apparent failure of a router.

9.2 Integrated Network Fault Tolerance System

Figure 9.2 shows one way in which the Router and Switch Fault Tolerance Systems can be integrated so as to provide fault tolerance for the routers, switches, NICs, and cables in a network while minimizing the network load imposed by the fault tolerance system. In this implementation, each router interface is given two IP addresses, similar to those of each NIC in every NFTS-enabled host: one address is for the main subnet, while the other is for a switch, NIC, and cable-testing subnet. However, since the switches in this network are not connected, the main subnet IP addresses for each router interface are identical.

The NIC Tester of the SFTS can now be replaced by the RFTS itself: if a router, switch,

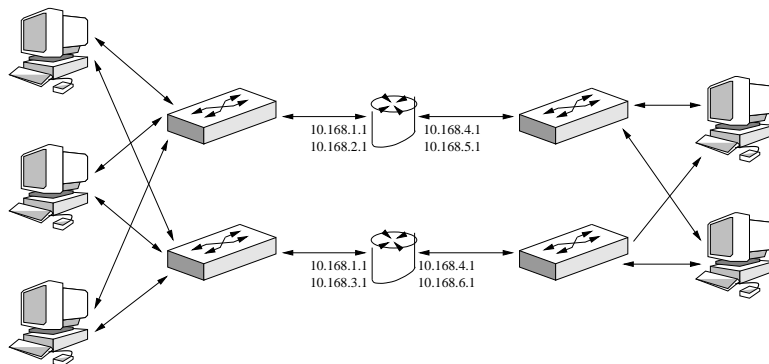


Figure 9.2: The Network Fault Tolerance System, combining the RFTS and the SFTS to provide fault tolerance at the router, switch, NIC, and network cable levels.

NIC, or cable fails, the RFTS will detect a router failure, and the SFTS will switch NICs to correct for that failure. When the route table of a host is updated to route packets out of a functioning NIC, all communications are repaired: no static routes on the host have to be updated to reflect new gateways to a remote subnet since all routers have the same IP address, and no SNMP calls have to be made to each router when a NIC change occurs since any necessary static routes to alternate NICs in the hosts can be determined upon installation of the system and saved in the routers' configuration files. This integrated fault tolerance system scales identically to the RFTS, and unlike the SFTS, does not require more than one host to be active in order for the system to function properly.

The concepts behind the Router and Switch Fault Tolerance Systems are not limited to Windows NT 4.0. Since any platform that supports IP networking also must have route tables, the RFTS and SFTS software can be ported or rewritten for other operating systems, such as UNIX. Also, the systems do not require special hardware, and although they were designed and tested on a 100Mbps Ethernet network, they can be used on other types of computer networks with few if any modifications.

With an integrated fault tolerance system, a network of computers will be a much more reliable tool for local and remote users. This system can find applications in military communications, video conferencing, and World Wide Web based businesses that require constant connectivity in order to function.

Bibliography

- [1] Alexander S. and Droms R. *DHCP Options and BOOTP Vendor Extensions*. Request for Comment 2132, <http://rfc.roxen.com/rfc/rfc2132.html>.
- [2] Ballew, Scott M. *Managing IP Networks with Cisco Routers*. Sebastopol, California, 1997, O'Reilly.
- [3] Beveridge, Jim, and Wiener, Robert. *Multithreading Applications in Win32*. Reading, Massachusetts, 1997, Addison-Wesley.
- [4] Bonner, Pat. *Network Programming with Windows Sockets*. Upper Saddle River, New Jersey, 1996, Prentice Hall.
- [5] Braden, R., ed. *Requirements for Internet Hosts – Communication Layers*. Request for Comment 1122, <http://rfc.roxen.com/rfc/rfc1122.html>.
- [6] Braden, R., ed. *Requirements for Internet Hosts – Application and Support*. Request for Comment 1123, <http://rfc.roxen.com/rfc/rfc1123.html>.
- [7] Brain, Marshall. *Win32 System Services: The Heart of Windows 95 & Windows NT*. Upper Saddle River, New Jersey, 1996, Prentice Hall.
- [8] Case, J., et. al. *A Simple Network Management Protocol*. Request for Comment 1067, <http://rfc.roxen.com/rfc/rfc1067.html>.
- [9] Case, J., et. al. *A Simple Network Management Protocol (SNMP)*. Request for Comment 1098, <http://rfc.roxen.com/rfc/rfc1098.html>.
- [10] Case, J., et. al. *A Simple Network Management Protocol (SNMP)*. Request for Comment 1157, <http://rfc.roxen.com/rfc/rfc1157.html>.

- [11] Cerf, V. *IAB Recommendations for the Development of Internet Network Management Standards*. Request for Comment 1052, <http://rfc.roxen.com/rfc/rfc1052.html>.
- [12] *Cisco IOS Software Command Summary*.
<http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/sbook/sip.pdf>.
- [13] Comer, Douglas E. *Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architecture*. Englewood Cliffs, New Jersey, 1995, Prentice Hall.
- [14] Conta, A. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. Request for Comment 1885, <http://rfc.roxen.com/rfc/rfc1885.html>.
- [15] Droms, R. *Dynamic Host Configuration Protocol*. Request for Comment 1531, <http://rfc.roxen.com/rfc/rfc1531.html>.
- [16] Droms, R. *Dynamic Host Configuration Protocol* Request for Comment 1541, <http://rfc.roxen.com/rfc/rfc1541.html>.
- [17] Droms, R. *Dynamic Host Configuration Protocol* Request for Comment 2131, <http://rfc.roxen.com/rfc/rfc2131.html>.
- [18] Hart, Johnson M. *Win32 System Programming*. Reading, Massachusetts, 1997, Addison-Wesley.
- [19] Higginson, Peter L., and Shand, Michael C. *Development of Router Clusters to Provide Fast Failover in IP Networks*. <http://www.digital.com/DTJR02/DTJR02HM.HTM>.
- [20] Knight, S., et. al. *Virtual Router Redundancy Protocol*. Request for Comment 2338, <http://www.cis.ohio-state.edu/htbin/rfc/rfc2338.html>.
- [21] Lewis, Chris. *Cisco TCP/IP Routing Professional Reference*. New York, New York, 1998, McGraw-Hill.
- [22] Maufer, Thomas A. *Deploying IP Multicast in the Enterprise*. Upper Saddle River, New Jersey, 1998, Prentice Hall.
- [23] McCloghrie, K. *Management Information Base for Network Management of TCP/IP-Based Internets*. <http://rfc.roxen.com/rfc/rfc1156.html>

- [24] Microsoft Product Support Services. *NetBIOS Name Conflicts When NetBEUI Used on Multiple NICs*. <http://support.microsoft.com/support/kb/articles/Q103/4/70.asp>.
- [25] Miller, Mark A. *Managing Internetworks with SNMP*. New York, New York, 1993, M&T Books.
- [26] Minasi, Mark, et. al. *Master TCP/IP for NT Server*. San Francisco, California, 1997, Sybex Network Press.
- [27] Murray, James D. *Windows NT SNMP*.
- [28] *NetWare 3.2 Product Details*. <http://www.novell.com/products/netware3/details.html>.
- [29] *NetWare 4.2 Product Details*. <http://www.novell.com/products/netware4/details.html>.
- [30] *NetWare 5.1 Product Details*. <http://www.novell.com/netware5/details.html>.
- [31] *NetWare Small Business Suite 5 Product Details*.
<http://www.novell.com/products/smallbiz/details.html>.
- [32] Perkins, David, and McGinnis, Evan. *Understanding SNMP MIBs*. Upper Saddle River, New Jersey, 1997, Prentice Hall.
- [33] Postel, J. *User Datagram Protocol*. Request for Comment 768,
<http://rfc.roxen.com/rfc/rfc768.html>.
- [34] Postel, J. *Internet Control Message Protocol*. Request for Comment 777,
<http://rfc.roxen.com/rfc/rfc777.html>.
- [35] Postel, Jon, ed. *Internet Protocol*. Request for Comment 791,
<http://rfc.roxen.com/rfc/rfc760.html>.
- [36] Postel, J. *Internet Control Message Protocol*. Request for Comment 792,
<http://rfc.roxen.com/rfc/rfc792.html>.
- [37] Quinn, Bob, and Shute, Dave. *Windows Sockets Network Programming*. Reading, Massachusetts, 1996, Addison-Wesley.
- [38] Roberts, Dave. *Developing for the Internet with Winsock*. Scottsdale, Arizona, 1995, Coriolis Group Books.

- [39] Simoneau, Paul. *SNMP Network Management*. New York, New York, 1999, McGraw-Hill.
- [40] Stallings, William. *Network and Internetwork Security: Principles and Practice*. Englewood Cliffs, New Jersey, 1995, Prentice Hall.
- [41] Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts, 1994, Addison-Wesley.
- [42] Tanenbaum, Andrew S. *Computer Networks: Third Edition*. Upper Saddle River, New Jersey, 1996, Prentice Hall.
- [43] *Using HSRP for Fault-Tolerant IP Routing*.
<http://www.cisco.com/cpress/cc/td/cpress/ccie/ndcs978/nd2022.htm>.