DRS: A Fault Tolerant Network Routing System For Mission Critical Distributed Applications

Abdur Chowdhury
The Telephone Connection
1375 Piccard Drive, Ste. 150
Rockville, Maryland
achowdhu@telecnnct.com
301-417-0700 (phone)
301-417-0707 (fax)

David Grossman
Office of Information Technology
Washington, DC

Eric Burger
The Telephone Connection
Rockville, Maryland

Ophir Frieder[1]
Florida Institute of Technology
Melbourne, Florida

1. Abstract

We present a novel proactive routing algorithm that consistently searches for failures via frequent ICMP echo requests. Our algorithm differs from its predecessors in that it is proactive instead of reactive by looking for failures before they affect message transmissions. When a failure is detected, an alternative route is identified and used. Our algorithm is currently deployed by a leading telecommunications company. In production use, the net result has improved availability by a 13% increase.

Keywords: routing, networking, fault tolerance, ICMP, IP

1 This work is supported in part by matching funds from the National Science Foundation under the National Young Investigator Program under contract number IRI-9357785 and by TTC. Ophir Frieder is currently on leave from the Department of Computer Science at George Mason University.

2. Introduction

With the current trend in distributed computing, large supercomputers are becoming scarce. These large centralized solutions of yesterday are being replaced by smaller computers coupled together by networks to achieve the same objective at substantially lower cost. The Berkley NOW (Network Of Workstations) project [29], PVM (Parallel Virtual Machine) [32, 33] and MPI (Message Passing Interface) [30] approaches use the network to provide a communication interface to link computers to act as a single large supercomputer. New operating systems, like Spring from Sun [31], focus on distributed computing via the network, both these approaches have one thing in common - the network.

We developed a network routing algorithm to provide fault-tolerance to relatively small networks by proactively monitoring network communication links between servers rather than reactive routing techniques. A reactive technique waits for a failure to occur and then *reacts* by finding an alternative route. Our proactive algorithm constantly looks for errors via continuous ICMP echo requests – when a failure is identified, a new route is selected around the failed portion of the network.

We have deployed this system in a production environment in over 25 separate installations running distributed server applications. The need to ensure that these servers are able to communicate with one another is of mission-critical concern. The DRS (Dynamic Routing System) is built on-top of existing hardware and a variety of operating systems (Solaris, SunOS, LynxOS) making its use and deployment economical. The DRS has been deployed in a distributed commercial application of a major telecommunications vendor for 24 months. During that time, outages due to networks hardware failures were averted. We achieved a 13% increase in availability of the network with the use of the.

Our algorithm further improves reliability via two network interface cards per server to provide an alternate method of physical communications in case of hardware failure. The DRS works by frequent link checks between all pairs of nodes to determine if the link between pairs of computers is valid. This algorithm is redundant because multiple links between two nodes are defined. When one link fails the second direct link is used. However, if no link exists, a broadcast is done to identify whether or not some other node is able to act as a router to create a path between the sender and the proposed recipient. Our algorithm differs from others in that links are constantly being checked. Other approaches [1, 2, 3, 4, 5, 6] take a more optimistic approach and simply re-route when "discovery" messages are not received for a specified time. Our algorithm discovers the failure before application performance is affected. The essential goal of our algorithm is to hide network failures from distributed applications.

Based on our actual implementation, we developed an analytical model of the DRS to evaluate its potential use for large networks. Using this model, we computed for various network sizes the fault identification times given a percentage of network usage. For a typical 10Mbs Ethernet, we compute that a sixteen node network has sub-second fault identification when using 10 percent of the total theoretical packet throughput of a ethernet network. Sixteen nodes seems small given that large corporations often have tens of thousands of workstations all on various LANs, where each LAN has 50-100 workstations. However, the application domain of this solution is distributed server applications running on a separate network. A characteristic of DSA (Distributed Server Applications) is a tightly coupled server node array, where clients exist apart from the server network and the server array appears as a single serving entity. Examples of this are large scale DNS server handling distributed data and requests. A word processor running off a file server is not a DSA.

In Section 3, we overview previous related efforts. In section 4, we describe the DRS algorithm details. In Section 5 we highlight our results obtained using a simulation of the DRS. Finally, in Section 6, we conclude and outline directions for future work.

3. Prior Work

There is an abundance of literature on routing algorithms and protocols. Research dealing with hardware fault tolerance has been studied in particular by the telecommunications industry. Prior literature can be partitioned into routing algorithms, routing demons, and hardware solutions. We first discuss routing algorithms developed for network communications, then the routing demons that implement them, and finish off with hardware solutions provided and developed by the telecommunications industry.

3.1 Routing Algorithms

The most common routing solution today is the Routing Information Protocol (RIP) [1, 2]. RIP automatically creates and maintains network routes. RIP is a dynamic routing protocol that is commonly employed in the Internet since it is included in most versions of UNIX. RIP, although popular, has many shortcomings. Several key problems with RIP is that failed network links will take minutes to fix, RIP does not work with subnets, and RIP is a reactive routing protocol to network failures. [7, 8]

Open Shortest Path First (OSPF) is a routing protocol for IP networks based on the DARPA Internet Protocol (IP) network layer. The basic routing algorithm is called the Shortest Path First Algorithm [3]. OSPF is an Interior Gateway Protocol and is intended to be used within an IP network under common administration, such as a campus, corporate, or regional network. The OSPF approach is a passive approach. Therefore, an OSPF routing demon does not know that a problem has occurred until a time-out value

has been reached, meaning a node has not sent a discovery message in the specified time period, before a new route is sought out. [9, 10, 11, 12, 13, 14]

The External Gateway Protocol (EGP), sometimes referred to as BGP (Border Gateway Protocol), enables networks to exchange information on how to reach other networks. This is useful in constructing Wide Area Networks, however, it does not provide fault tolerance to a small server network. [15, 16, 17, 18, 19, 20, 21, 22, 23]

3.2 Routing Demons

Routed, gated, in.rdisc, [1, 2, 3, 4, 5, 6] are routing demons that implement RIP, OSPF, EGP, and BGP. Given the algorithms they are based on, however, they do not provide a proactive fault detection schema that protects distributed applications from network failures.

3.3 Hardware Routing

The telecommunications industry has been interested in fault tolerance for their networks of systems for many years. Telecommunications protocols are inherently different from routing protocols in that they focus predominantly on hardware solutions. The most widely used solutions are SONET and DXC [26, 25]. Others include double-loop, forward hop, and fiber. Survivable network architectures for traffic restoration are generally divided into two categories: ring-based dedicated restoration and mesh restoration. Rings with redundant capacity and automatic protection switching are capable of healing by themselves and hence, are called self-healing rings (SHR). Mesh restoration uses digital cross-connect systems to reroute traffic around a failure point.

There are two types of self-healing rings, unidirectional self-healing rings (USHR) and bi-directional self-healing rings (BSHR). They both restore 100% of the traffic under a single network failure condition. In USHR's, working traffic is carried around the ring in one direction while a second communications ring is for protection. This second ring transmits in the opposite direction of the working ring and is only used in times of failure of the first ring. In BSHR's, working traffic travels in both directions around the ring [24]. Two rings exist in BSHR as well; however, either ring may be used at any time; no ring is set aside for fault tolerance.

The self-healing mesh network architecture using Digital Cross-Connect systems (DXC's) [25] is a crucial part of some integrated network restoration systems. The mesh is constructed such that each node has two connections to each neighboring node. Message passing is fundamentally different from routing protocols because switches are used at each node such that a direct connection exists between the sender and the recipient. The message travels over the direct connection; it is not subdivided into

individual packets. The primary connection is used until a failure occurs, at which time, the backup connection is used.

A conventional DXC self-healing network using logical channel protection requires substantial network hardware because for *n* nodes, there are two physical connections among each pair of nodes, or *(n*(n-1))* total connections. This is a large amount of spare capacity for network components. Originally, self-healing meshes used a centralized database to track failures and reconfigure in the event of a failure for the entire mesh. This centralization was a bottleneck and was itself prone to failure. Hence, a distributed approach is now used. With a distributed approach, each node determines rerouting patterns and fault detection [25].

In the class of double-loop networks, referred to as forward-loop backward-hop (FLBH) [26], each node is connected via unidirectional links to the node one hop in front of it and to the node *X* hops in back of it. This type of system is similar in topology design to FDDI and other fiber type hardware solutions. These hardware solutions run on the order of tens of thousands of dollars to implement.

A variety of hardware solutions have been used by the telecommunications industry to route phone calls. These include SONET, DXC's, FDDI, and SHR. Each of these is highly fault tolerant due to the numerous paths that exist between every source and destination node. Fault tolerant routing is provided via hardware mirroring, rendering this approach as very expensive. Such solutions have not been implemented for use with computer networks using IP protocols.

4. DRS Algorithm

DRS improves fault tolerance via proactive failure recognition and the use of a completely redundant network.

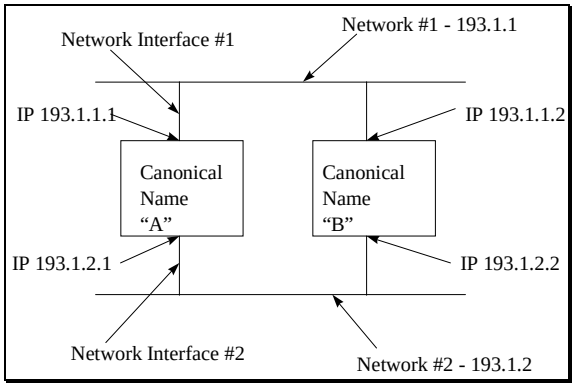

Figure 1: Dual Network Setup Details

The diagram above illustrates a dual network setup for two computers. Each computer has two network interface cards connected to two separate networks. It is the task of the

routing demons to monitor the connections between “A” and “B”. If a failure occurs, the demons set up routes to route around the problem before network applications are aware that a problem occurred.

The DRS runs on every node in the server array. Each DRS demon is configured to monitor hosts on the networks and executes a two stage run process. In the first phase, the communications links between the local host and all other hosts that is it has been configured to monitor are checked. These checks are accomplished using the ICMP (Internet Control Message Protocol) echo request [7, 8]. Host “A” sends an ICMP echo request to host “B” via the first network. If the echo is returned, the DRS can assume that the hub, wiring, network interface card, device driver, network protocol stack, host kernel, and the DRS Demon are all operational. The DRS then tests all known hosts and all known networks in the above example. The next check is for host “B” network number two.

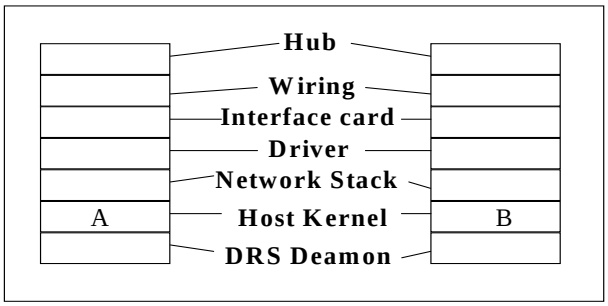

Figure 2: DRS Network Stack

Each demon keeps track of which hosts to monitor and the state that they are in, (i.e., “up”, “down”). If a failure occurs, the DRS demon must determine a new route of communication between host “A” and “B”. The next section describes different failure scenarios and how the new route is calculated and resolved. Figure 3, gives a high level overview of the execution flow of the DRS algorithm.

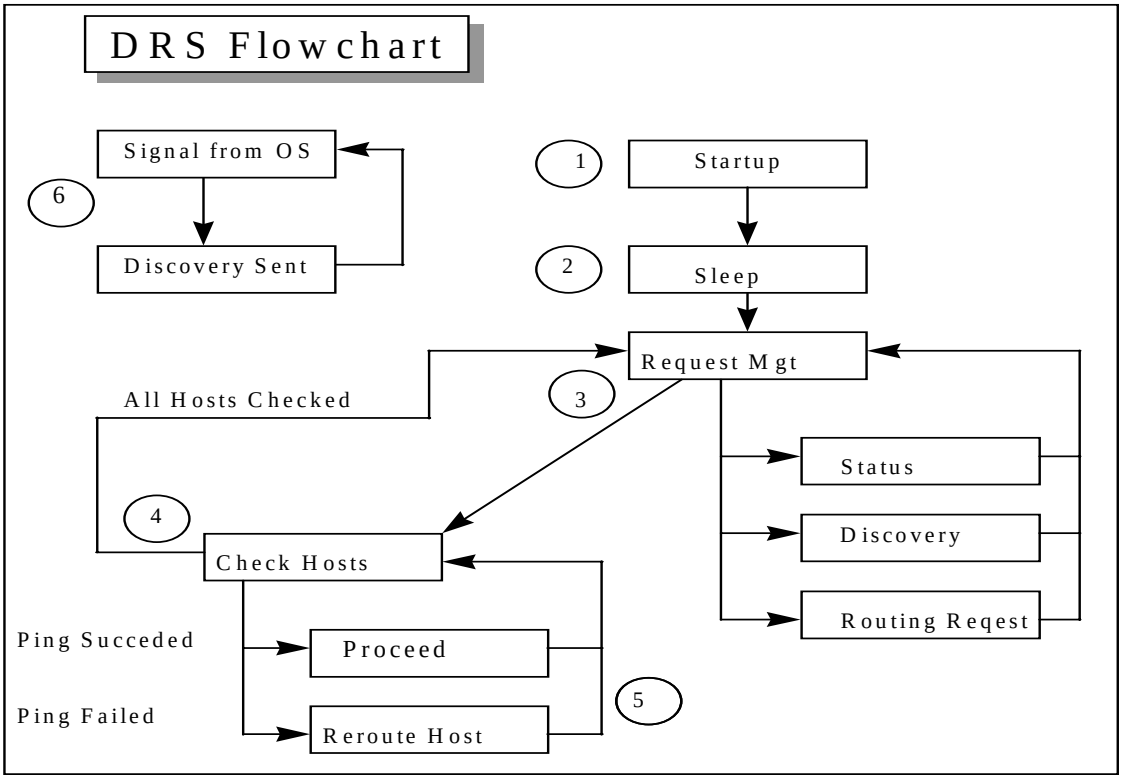

Figure 3: DRS Architecture

The following is a detailed description of each step in the DRS execution.

Step 1. System Initialization. All DRS system variables are initialized, i.e., read in configuration variables, setup network communication modules, etc.

Step 2. Initial Sleep period. Each DRS is dormant at startup. The dormant initial condition prevents false negative results of a ping at startup. Periodically a system could be powered down during routine maintenance. When the servers are restarted, not all may start at the same time or boot at the same speed. By having the DRS demon sleep for a predetermined amount of time at start-up, false failures are not reported.

Step 3. Listen for incoming "Request" or "Discovery" messages. The DRS is a routing demon. Thus, part of its job is to handle requests for information or to add new information to its internal database of network hosts and configurations. "Request for information" may be an administrator contacting the demon and requesting a view of its routing tables, or a remote server may be unable to contact another server and is asking all other servers if they are able to communicate to the server in question. "Discovery" messages are used for detection of fixed routes and new servers on the network. This is covered in more detail during step 6.

Step 4. Check all communication links. Link status verification, the key to the DRS, is the proactive monitoring of communication links between each server. This verification enables the DRS to quickly find and fix network failures. The DRS starts with a list of hosts to monitor. This list is known at start time, but may be added to in the future by a "Discovery" message. The DRS sends an ICMP echo request to the host in question. If the echo is successful, the route is marked as "up"; if it is unsuccessful, several more attempts are made. If none are successful, the route is marked as "down". All links are continuously monitored. Checks occur every *X* seconds where *X* is a configurable setting. Note that *X* affects the speed an error is detected and also effects the amount of network utilization that will occur.

Step 5. Fix identified communication errors. In this step, the DRS's attempts to fix known "down" routes. Each host has two network interfaces. Once one interface is not responding, the second interface is sent an ICMP echo request to verify that it is working. If that is successful, the DRS modifies its internal routing tables to move all communication to say "B" network 1 to "B" network 2. Note that in step 4, the new route was checked. The second check guarantees that a failure did not occur in-between steps. If the second interface did not respond to the ICMP request, a broadcast is sent on all connected networks. This broadcast asks all other DRS demons to see if they can communicate with the host or network in question. The first DRS demon to respond is used as a router to the lost host. If no one responds, the host in question has suffered at least two hardware failures and has become completely separated from the network. If this has happened, the only thing left to do is to send an alarm message to the system administrator notifying him/her of the catastrophic failure.

Step 6. Send "Discovery messages". This stage runs as a separate thread of execution in the demon. The DRS sends out a broadcast message on all of its network interfaces stating the server, and the server's network interfaces. This message has several functions for the DRS. The most important is that if a network failure did occur and was fixed, the DRS would never know, because "down" routes are not checked. By sending this message, the other DRS demons become aware that a "down" interface is now working again, and the demon corrects all rerouted communications of that host to the original routes.

The DRS loops through this six step cycle monitoring communication links, answering requests, and fixing problems as they occur.

4.1 DRS in the presence a single network failures

We now describe the action of the DRS in the presence of a single failure. Upon startup (before the network error occurs), the DRS establishes communication links to each host.

Consider a failure to node *B's* interface #1. Since every node on the network is implementing the same algorithm, we only discuss the events as they happen for node A.

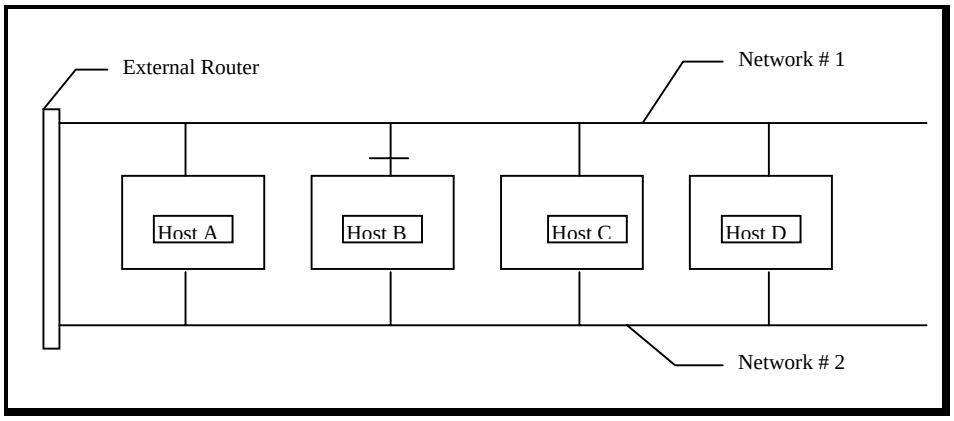

Figure 4: Single Network Failure on Node B

Node *A* sends an ICMP echo request for each node and link in its routing table (part of step 4). The ICMP echo request is not responded to by node *B* for network #1 because of the failure. Node A then looks for another route. Note that one does exist because node B's second interface is operational. Node A now identifies that this is a potential route because it is listed as being in the UP status. However, this status is not guaranteed to be current so an additional check of the proposed alternative route is made in the later phase of step number four. In this case, an ICMP echo request (or ping) is sent to host B along the alternative route. If this succeeds, A's routing table is updated to reflect the newly identified static route that circumvents the failure.

At this point, network communication is not impeded by the failure. However, it is important to identify where the failure exists so it may be repaired. The fault is isolated by examining the routing tables of each node. Looking at node A, C, and D's routing table, it becomes apparent that each node of these two nodes is able to communicate with all other nodes directly on network #1. However, A, C, and D's routing tables now contains an alternative route to node B. For diagnostic purposes, it is reasonable to assume that the routing tables are current enough to isolate the problem.

For nodes A, C, and D, the only failure is the route to B for interface #1, while host B has failure for every interface on the first network. This tells us that the error has occurred with host B's network interface, ethernet wire, or network hub port. At this point, node B is examined and the exact nature of the problem (i.e., interface card, network cable, hub port, etc.) is determined and repaired.

Alarm Management for the DRS has been implemented on a proprietary system that has the ability to page system administrators and send e-mail messages in the case of problems. The DRS also logs errors and messages to a log file. The DRS is easily

modified to use any other alarm management system that uses standard inter-process communication facilities provided by most UNIX systems.

4.2 Multiple Failures

A single failure is the most common. However, multiple failures do occur, and the DRS is resilient to them. There are two kinds of multiple failures. Those that are equivalent in algorithm perception and handled by the DRS as any single network failure (i.e., do not require a router) and those where each failure is routed via a routing element.

4.3 Non-Router Failures

A hub or a situation where all interfaces on a single network fail at once is treated by the DRS algorithm as a single failure. All these failures are handled in the same fashion and appear like the single failure example above. All network communication for the first network is rerouted to the second network via each host's second interface card.

4.4 Multiple Network Failures

Multiple failures, although unlikely, are not always as well behaved. The likelihood of an error occurring on the primary or secondary network only is smaller than the possibility of it happening randomly to both networks. Thus, the DRS must be able to handle staggered network failures. Here is an example of a staggered multiple network failure.

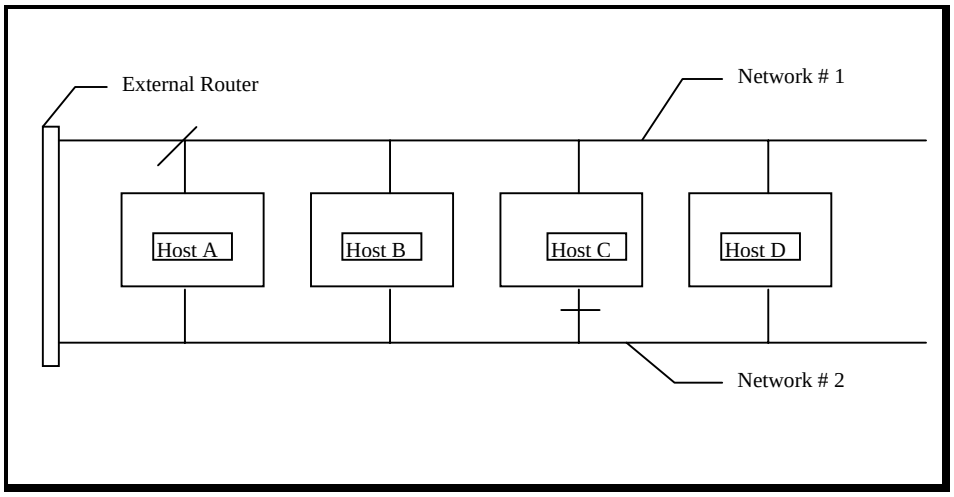

Figure 5: Dual Network Failure on two different networks

In this example, a dual network failure occurs. A port on the network hub one to host A interface 193.1.1.1 has failed, and, at the same time, the network interface card for host C interface 193.1.2.3 has failed.

The problem is that host A and host C cannot directly communicate to each other. Host A cannot communicate on network #1 and host C cannot communicate on network #2.

We only discuss the details of host A and its procedure in correcting the network communication from failures since the same algorithm is applied to each host.

In step 4, each host's communication link that is in an "UP" state is checked. Every host that is on network #1 fails because the problem is with the hub. Host A is then placed in the "DOWN" state. Host C's interface on network #2 also failed and is placed in the "DOWN" state. This reflects the new information that shows many communication paths have failed. At this point the "Fix Identified Communication Errors" step is executed. Host B and host D have direct routes (using interface 2) that appear to be usable. This corrects the communication link failure on network #1 from A to B and from A to D

Notice that we still do not have a means of communicating from A to C. The reason for this is that the interface card on network #2 for host C is identified as failed for both interface one and interface two (this is our second failure).

Host A now attempts to find some means of communicating with C on interface #1. It broadcasts a routing request along both network #1 and network #2. The first "CANYOUROUTE" broadcast is blocked by the failure on node A interface one. The second "CANYOUROUTE" broadcast is sent out as a routing request for node C on the second network. The first node to respond to the plea for help is used as a router for communication to host C. Assume B is the first node to respond for communications routing for host C interface one. A static route using node B is added to node A's routing table. See Figure 6.

Since two routes must be know for each node, A must find a route for host C interface number two. The DRS does not distinguish that the different interfaces are connected to the same host in this instance. Again a broadcast is issued on both networks. The first broadcast goes unacknowledged. Assume node D answers the second request for help in routing to node C. See Figure 7. Now all communication routes to host C were restored using a remote host as routers between the nodes.

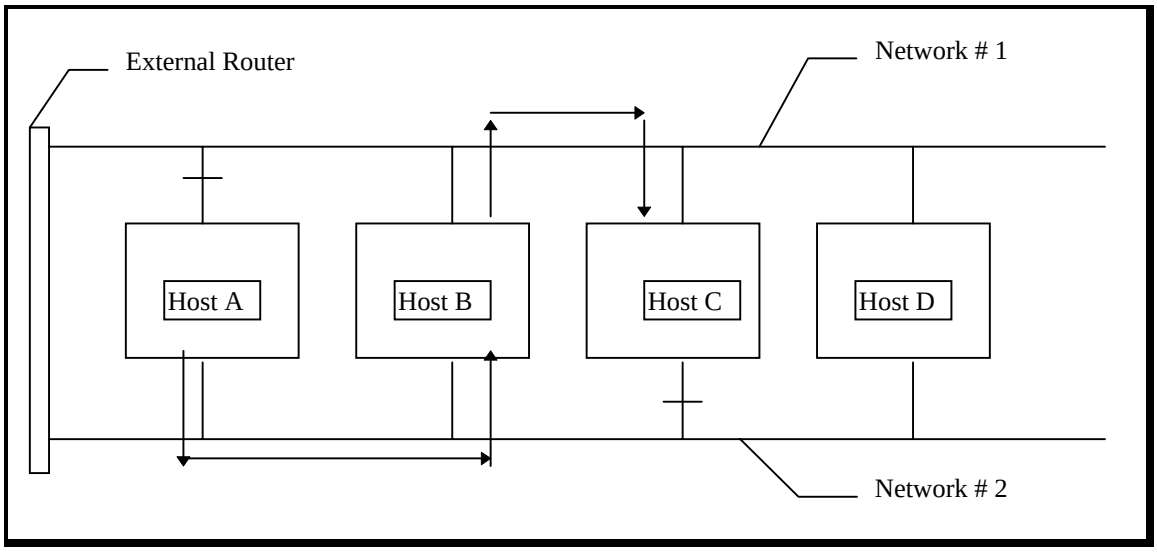

Figure 6: Communication route diagram for route from A to C via B

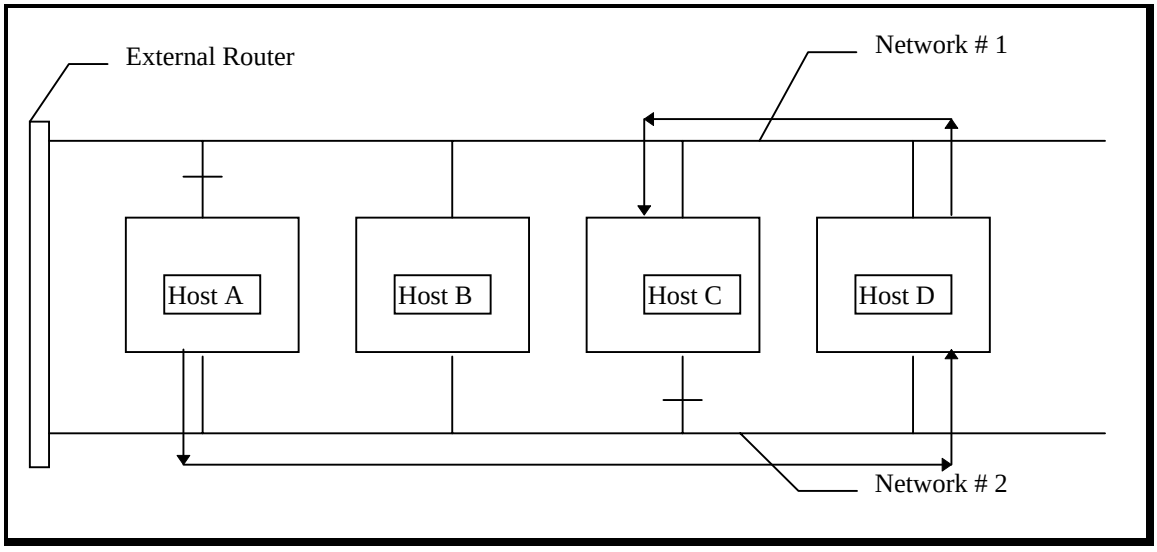

Figure 7: Communication route diagram for route from A to C via D

4.5 Detection of Network Repairs

The DRS algorithm has the ability to detect the reconnection or repair of a failed network route. In the previous example, assume the cause of node A's problem is that the port on

hub one was accidentally turned off. Further, assume the problem was detected, circumvented and an alarm was sent by the DRS demon.

The system administrators resolve the problem by turning the port back on. Once this is done, they do not need to examine the routing tables of the hosts because the DRS is self-correcting. That is, the DRS uses discovery messages to identify the correction and return the routing tables to their original state (i.e., direct network routes instead of static alternative routes).

The self-correcting recovery occurs because each host periodically broadcasts its own discovery message on all of its network interfaces. When node A receives discovery messages on interface 1 from B, C, and D, it examines the routing table to determine if any corrections or updates are needed.

The DRS identifies that static routes exist for nodes B, C, and D. The DRS now removes them from the local kernels routing table. This removal is to restore the original routing tables state after a failure has been fixed. The DRS's routing table is then updated to reflect the newly repaired communication path.

Notice that host C interface #2 is still down. Host A has not received a status discovery message from that link because the network link is still not functioning correctly. With this algorithm when network failures are corrected, the system is able to return to its original state without manual intervention.

4.6 Network Failures not addressed by the DRS

There are some sequences of failures that even the use of an external router does not address. The DRS cannot resolve network failures that entail the complete separation of a network node. A total network failure means that there are no physical connections to a particular host available. Below is an example of a complete network failure for host A. In this case, A cannot communicate on any interface and has suffered a non-repairable network failure. Note that B, C, and D will alarm the system administrator that this error has occurred.

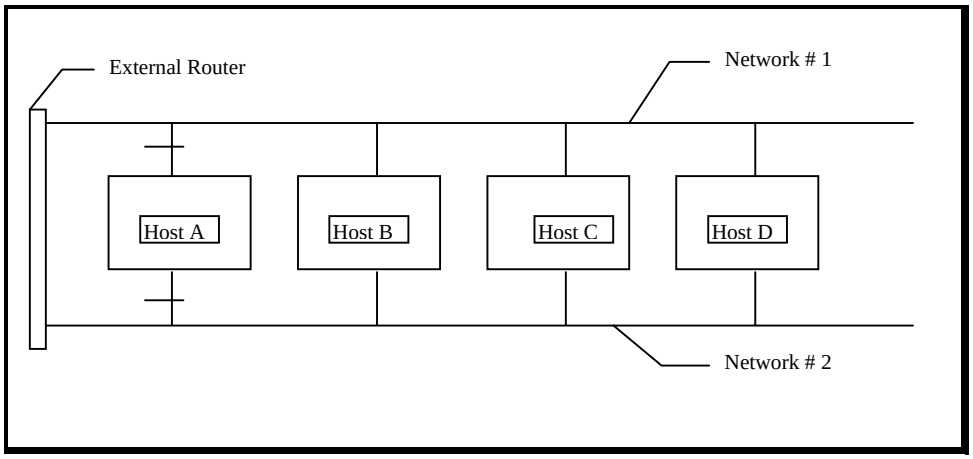

Figure 8: Total Network Isolation of node A

5. DRS Performance Results

5.1 Scaling Results

The DRS checks each "up" link it is attached to (Step 4). Therefore, on average, the fastest a failure in a communication link can be determined is the amount of time it takes to check all links divided by two. When scaling the DRS, it is important to examine the time needed to determine that a failure has occurred and the amount of bandwidth the DRS will use to achieve that goal. A trade-off exists between how quickly a failure is discovered and how much network overhead (bandwidth) is required.

5.2 DRS Model System

We developed an analytical model to estimate the network usage of the DRS given the number of nodes on the network and delay time between node checks. The DRS model achieves many levels of performance, as shown below, by modifying the frequency (time) at which hosts are checked. Details of this model can be found in "A Fault Tolerant Network Routing System For Mission Critical Distributed Applications", masters thesis [33].

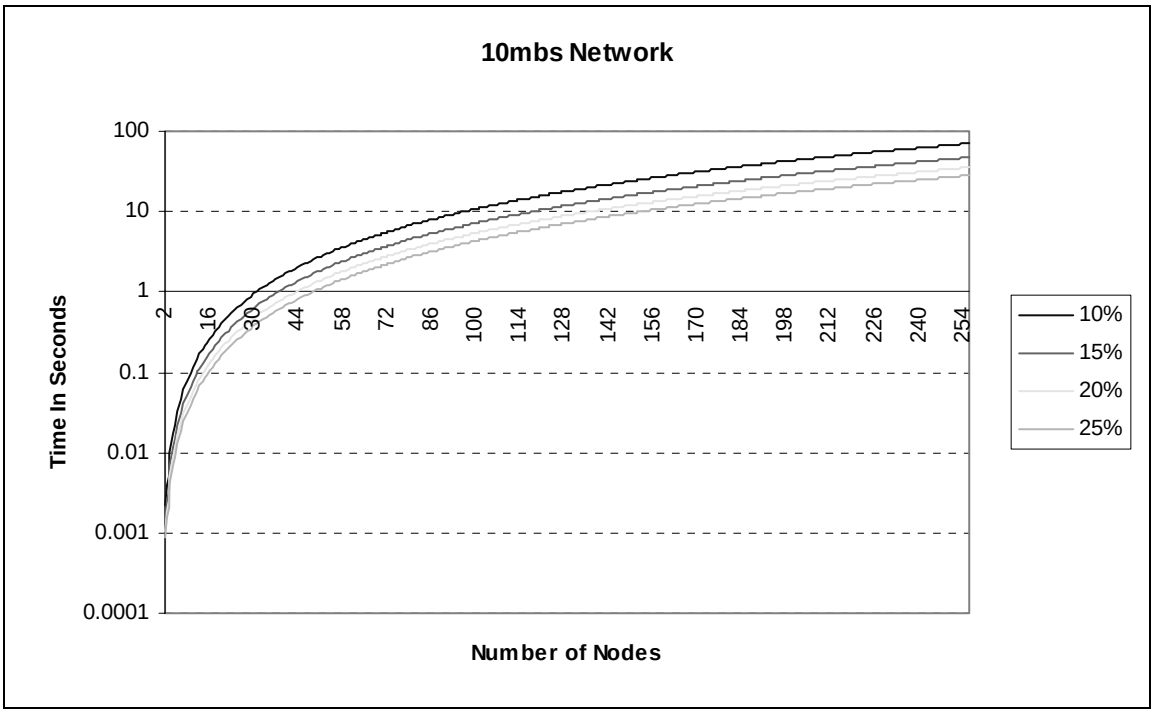

Figure 9: 10Mb Network Performance

The graph above shows that as the number of nodes increases the rate of network monitoring must decrease to maintain a constant network usage. The results presented in, Figure 9, and Figure 10 were obtained using the DRS model. A comparison of actual performance versus the performance predicted by our simulation is presented in Figure 11.

Figure 9 shows that on a 10Mb network, 32 servers can coexist while still being able to detect network failures in around one second. If there is a need for more servers, either a higher network utilization factor is used or a higher bandwidth network is required. In Figure 10, the DRS performance using a 100Mbs ethernet network, is shown.

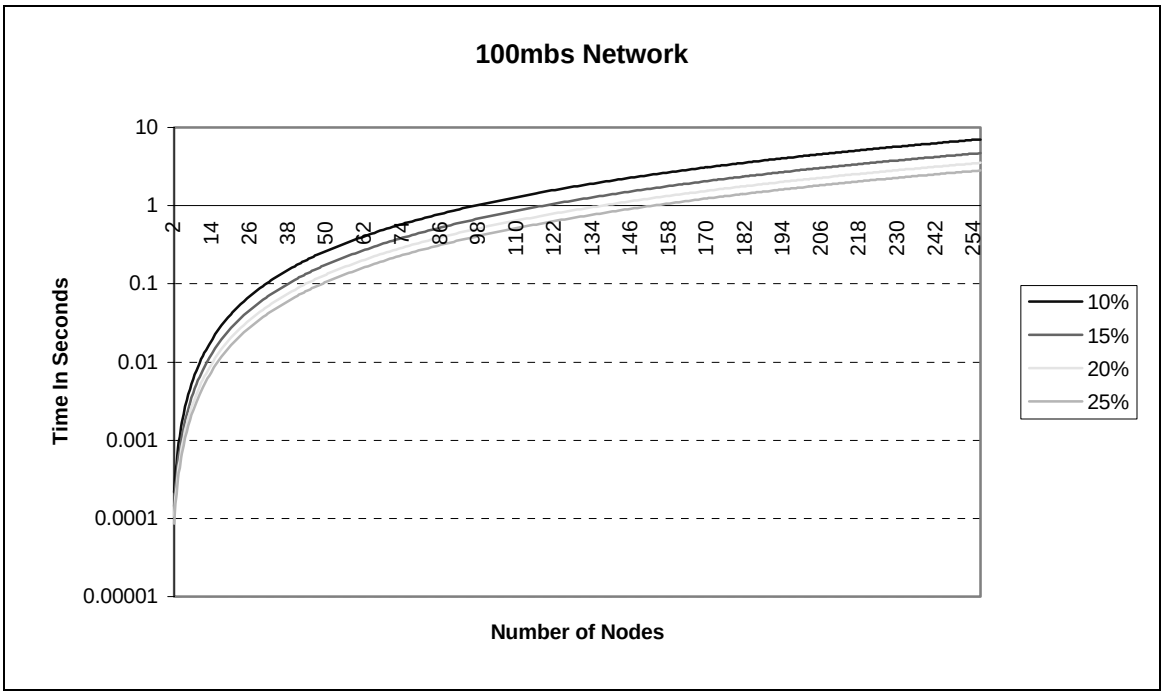

Figure 10: 100Mb Network Performance

By using a 100Mbs network the performance of the DRS greatly improves. As seen from Figure 10, 254 nodes can be monitored by the DRS with less than 10% network usage, and failures are detected in less than 10 seconds. Ninety nodes are able to detect an error in less than 1 second. The number 254 was chosen because it is the size of a class "C" network.

5.3 DRS Network Measurements

We evaluated the performance of the DRS system and compared our expected network to actual network usage. We did this to (a) find an acceptable level of performance for our system that allowed for sub-second error detection and (b) find a level of network usage that did not adversely affect the other applications performance.

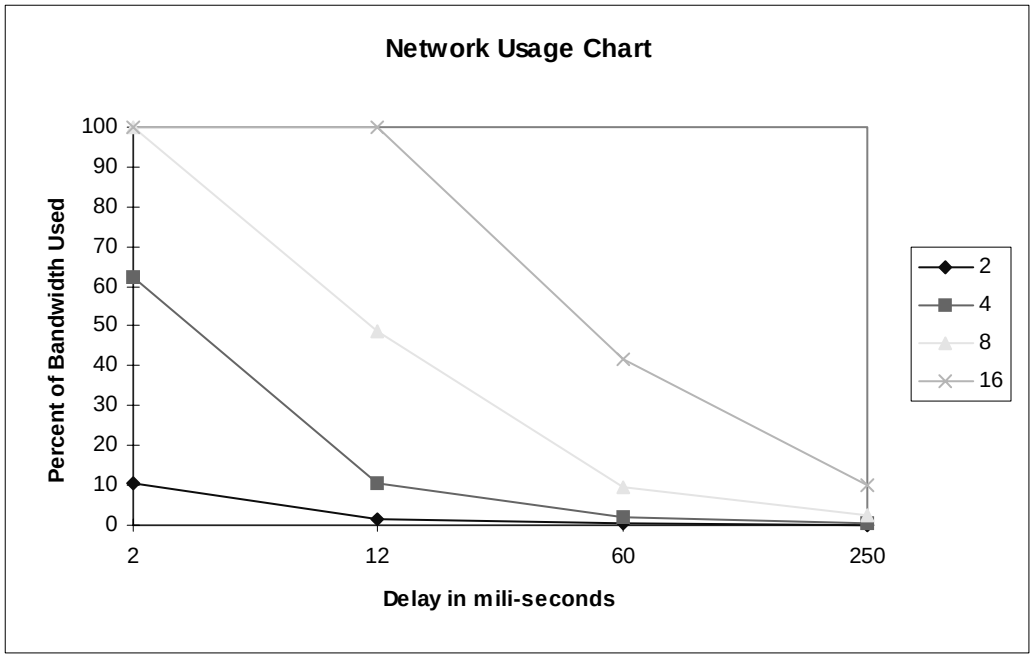

Figure 11: Bandwidth Usage VS Delay

Figure 9, and Figure 10 show that as the number of nodes increases the time between checks decreases to maintain a constant network usage. We tested that hypotheses by counting bytes sent on the network and averaging over time. Figure 11, shows the tested results. Note that 10% usage is usable by a network of less than 16 nodes and provides a less than one second error detection rate.

5.4 Hardware Failure Rates:

The DRS has been implemented in a commercial system. The primary goal of the DRS was to provide a fault tolerance for network hardware failures. The commercial system does not have any system administrators on premises. In fact the closest repair engineer is at least one day away. Because of the mission critical nature of the system, downtime of one day is very expensive.

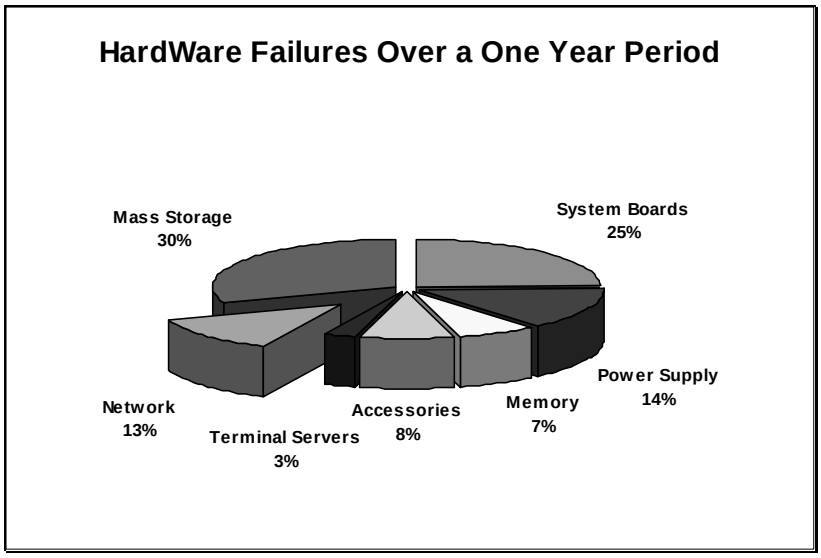

Figure 12: Percentage of failures by type

Over a 12 month period, hardware failures were tracked to determine the usefulness of the DRS. Hardware failures were categorized into seven classes.

- System Boards (mother boards, CPU chips, etc.)
- Mass Storage (SCSI controllers, hard drives, tape drives)
- Network hardware (hubs, ethernet cards, ethernet wires, etc.)
- Terminal Servers
- Accessories (specialized hardware)
- Memory (RAM)
- Power Supplies

Examining 50 systems, we experienced 70 hardware failures over a 12 month period. These failures showed us that 13% of our hardware failures were network related.

By using the DRS the network failures did not affect system performance, effectively eliminating this class of failures. A reduction of system down time of 13 percent was considered useful and worth the added cost of hardware and network bandwidth. It is reasonable to expect the same amount of hardware failures in the future.

6. Conclusion and Future Work

The DRS is a routing protocol that uses existing hardware and networking protocols to provide a fault tolerant network system for distributed applications and operating systems.

The DRS is unlike routed and gated approaches which passively monitor network links, because it proactively monitors each host and its communication links. The DRS also checks alternate routes before using them to achieve an additional level of fault tolerance without the use of special hardware. This fault tolerance comes at the price of some network bandwidth usage. We found this to be a reasonable trade off given that tightly coupled server arrays tend to be smaller than client server networks. Our production implementation ran on a four node system and created no network problems. We have calculated that the same system could be run with 32 server nodes and still achieve sub-second response time to network failures. The DRS is designed for the current trend of distributed computing systems. By using the DRS in a tightly clustered server system remote clients are unaffected during a network failure. The future of this research will focus on the need for a more efficient means of checking a large amounts of servers, i.e., lower than (n* (n-1)) messages. Also we will explore the use of RIP II in conjunction with DRS to add support for nodes outside of the network.

7. References

1 Request For Comment 1246, ‘‘Experience with the OSPF Protocol’’, 08/08/1991, J.Moy.

2 Request For Comment 1253, ‘‘OSPF Version 2 Management Information Base’’, 08/30/1991, F. Baker, R.Coltun.

3 Request For Comment 1370, ‘‘Applicability Statement for OSPF’’, 10/23/1992, Internet Architecture Board.

4 Request For Comment 1583, ‘‘OSPF Version 2’’, 03/23/1994, J.Moy.

5 Request For Comment 1403, ‘‘BGP OSPF Interaction’’, 01/14/1993, K.Varadhan.

6 Request For Comment 1397, ‘‘Default Route Advertisement In BGP2 And BGP3 Versions Of The Border Gateway Protocol’’, 01/13/1993, D. Haskin.

7 Request For Comment 1058, ‘‘Routing Information Protocol’’, 06/01/1988, C.Hedrick.

8 Request For Comment 1388, ‘‘RIP Version 2 Carrying Additional Information’’, 01/06/1993, G.Malkin.

9 Request For Comment 1246, ‘‘Experience with the OSPF Protocol’’, 08/08/1991, J.Moy.

10 Request For Comment 1253, ‘‘OSPF Version 2 Management Information Base’’, 08/30/1991, F. Baker, R.Coltun.

11 Request For Comment 1370, ‘‘Applicability Statement for OSPF’’, 10/23/1992, Internet Architecture Board.

12 Request For Comment 1583, ‘‘OSPF Version 2’’, 03/23/1994, J.Moy.

13 Request For Comment 1403, ‘‘BGP OSPF Interaction’’, 01/14/1993, K.Varadhan.

14 Request For Comment 1397, ‘‘Default Route Advertisement In BGP2 And BGP3 Versions Of The Border Gateway Protocol’’, 01/13/1993, D. Haskin.

15 Request For Comment 904, ‘‘Exterior Gateway Protocol formal specification’’, 04/01/1984, International Telegraph and Telephone Co, D. Mills.

16 Request For Comment 1388, ‘‘Exterior Gateway Protocol EGP’’, 10/01/1982, E.Rosen.

17 Request For Comment 1105, ‘‘Border Gateway Protocol BGP’’, 06/01/1989, K.Lougheed, Y. Rekhter.

18 Request For Comment 1163, ‘‘A Border Gateway Protocol (BGP)’’, 06/20/1990, K.Lougheed, Y. Rekhter.

19 Request For Comment 1164, ‘‘Application of the Border Gateway Protocol in the Internet’’, 06/20/1990, J.Honig, D. Katz, M. Mathis, Y. Rekhter, J. Yu.

20 Request For Comment 1265, ‘‘BGP Protocol Analysis’’, 10/28/1991, Y.Rekhter.

21 Request For Comment 1267, ‘‘A Border Gateway Protocol 3 (BGP-3)’’, 10/25/1991, K.Lougheed, Y. Rekhter.

22 Request For Comment 1268, ‘‘Application of the Border Gateway Protocol in the Internet’’, 10/25/1991, P. Gross, Y.Rekhter.

23 Request For Comment 1269, ‘‘Definitions of Managed Objects for the Border Gateway Protocol (Version 3)’’, 10/26/1991, J. Burruss, S. Willis.

24 Jane M. Simmons, Member, IEEE, and Robert G. Gallager, Fellow, IEEE, “Design Error Detection Scheme for Class C Service in ATM”, IEEE/ACM Transactions on Networking, Vol 2, No. 1, February 1994.

25 Tsong-Ho Wu, Senior Member, IEEE, “A Passsive Protected Self-Healing Mesh Network Architecture and Applications”, IEEE/ACM Transactions on Networking, Vol. 2, No. 1, February 1994.

26 Jon M. Peha, Member, IEEE, and Fouad A. Tobagi, Fellow, IEEE, “Analyzing the Fault Tolerance of Double-Loop Networks”, IEEE/ACM Transactions on Networking, Vol. 2, No. 4, August 1994.

22 R. Metcalfe and D. Boggs, Ethernet: Distributed Packet Switching for Local Computer Networks, Communications of the ACM, Vol. 19, No. 7, Jul. 1979, pp.395

23 A. Tanenbaum, Computer Networks

24 J. Shoch, Y. Dalal, and D. Redell, Evolution of the Ethernet Local Computer Network, Computer, Aug. 1982, pp.10

25 T. Gonsalves, and F. Tobagi, On the Performance Effects of Station Locations and Access Protocol Parameters in Ethernet Networks , IEEE Transactions on Communications, Vol. 36, No. 4, Apr. 1988, pp. 441

26 K. Chua, and K. Lye, Backoff Considerations in CSMA/CD LAN With Single Time-Varying Channel, Electronic Letters 25th Apr. 1991 Vol. 27 No. 9

27 D. Chorafas, Local Area Network Reference

28 W. Stallings, Local And Metropolitan Area Networks

29 T. Anderson, D. Culler, D. Patterson, ..., “A Case for NOW (Networks of Workstations)”, December 9, 1984.

30 J. Dongarra, S. Otto, M. Snir, “An Introduction to the MPI Standard”, January 16, 1995.

31 J. Mitchell, J. Gibbons, G. Hamilton, P. Kessler, ..., “An Overview of the Spring System” 1995.

32 Jeremy Casas, Ravi Konuru, Steve Otto, Robert Prouty, and Jonathon Walpole. Adaptive Load Migration Systems for PVM. In *Proceedings of Supercomputing* 94, Pages 390-399, Washington D.C., November 1994.

33 Abdur Chowdhury, Lisa Nicklas, Sanjeev Setia, Elizabeth White, “Supporting Dynamic Space-sharing on Clusters of Non-dedicated Workstations”.

34 Abdur Chowdhury, “DRS A Fault Tolerant Network Routing System For Mission Critical Distributed Applications”, Masters Thesis George Mason University.