The Journal of Supercomputing, 21, 239–256, 2002

On the Design, Development, Deployment, and Network Survivability Analysis of the Dynamic Routing System Protocol

ABDUR CHOWDHURY, OPHIR FRIEDER, AND PENG-JUN WAN {abdur, ophir, wan@cs.iit.edu}

Department of Computer Science, Illinois Institute of Technology

Abstract. With the ever-increasing demands on server applications, reliability is of paramount importance. Often these services are implemented using a distributed server cluster architecture where many servers act together providing end user services. We evaluated one hundred deployed systems and found that over a one-year period, thirteen percent of the hardware failures were network related. To reliably provide end-user services, the server clusters must guarantee server-to-server communication in the presence of these network failures. We describe a protocol designed to provide proactive dynamic routing for server clusters architectures called the Dynamic Routing System (DRS) protocol and present analysis to its survivability in the presence of network failure. Our experiments show that, for an eight-node server cluster with three concurrent network failures, the DRS provides a 267% improvement in the probability of server to server communication over a traditional network topology. Additionally, the proactive routing approach of the DRS performs better than traditional routing systems by fixing network problems before they affect application communication.

Keywords: network survivability, dynamic routing, fault tolerance, distributed server clusters

1. Introduction

With the ever-growing compute needs, traditional supercomputers are becoming scarce and distributed server clusters are becoming the solution of choice. These smaller computers are coupled by networks to achieve the same objective at a substantially lower cost. The Berkley NOW (Network Of Workstations) project was one of the first projects pushing this solution [1]. PVM (Parallel Virtual Machine) [4] and MPI (Message Passing Interface) [10] libraries provide messaging and synchronization constructs that are needed for distributed parallel computing with NOW solutions. Projects like Beowulf [22] for Linux are continuing the distributed/parallel approach. All of these approaches have one thing in common, the use of a network as a communication media. Much of the prior work, however, focuses on efficient communication methods [19] rather than providing fault tolerance and redundancy for the network of workstations.

We developed a network routing algorithm to provide fault-tolerance to networks by proactively monitoring network communication links between servers. This is different from reactive routing techniques [2, 3, 9, 17] that wait for a failure to occur and then react by finding an alternative route and different from wireless approaches that minimize power consumption [5]. Our proactive algorithm constantly looks for errors via continuous ICMP echo requests. When a failure is

identified, a new route bypassing the failed portion of the network is selected. This new route is often found in the time of a TCP retransmit, so server applications are unaware that a network failure has occurred.

Our algorithm improves reliability via two network interface cards per server to provide an alternate method of physical communications in the 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 uses redundant network links between two nodes to provide multiple communication channels. When one link fails, the second direct link is checked and used if possible. 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 new path between the sender and the proposed recipient. Our algorithm discovers the failure before application performance is affected. The essential goal of our algorithm is to hide network failures from distributed applications.

After describing the DRS approach, we provide a Network Survivability Analysis (NSA) of the DRS algorithm. Our evaluation of is based on a hub topology and thus is fundamentally different from other circuit analysis evaluations [8, 25]. We show that the DRS algorithm provides a more resilient solution to network failures than a single network, and a simple dual network solution [14]. While these topologies may seam simple, server clusters are generally tightly coupled, and this topology represents the norm. We present our results as a probability model showing the probability of success of the system as a whole and in terms of the number of network failures.

The DRS is currently deployed in 27 local voice mail-server clusters by MCI WorldCom; each cluster contains between 8 and 12 servers. Thus, understanding the reliability supported is not only of theoretical interest but of practical interest as well. In prior work [7], we showed that, over a one-year period, 13% of hardware failures for 100 compute servers were network related, i.e., network interface cards, hubs, etc. This likelihood of failure provides motivation to improve the resilience of server clusters where services need to be guaranteed. We show that for an eight-node cluster with three network failures, the DRS provides a 267% increase in the probability of server to server communication over traditional server cluster topologies. The proactive routing policy of the DRS performs better than traditional routing systems by fixing network problems before they affect application communication.

In Section 2, we overview prior work and describe the DRS protocol in Section 3. In Section 4, we present an example of how the DRS handles network failure scenarios, and in Section 5, we present two DRS probability models. In Section 6, we provide an analysis of our results and a comparison to other approaches, and finally, we conclude with our observations.

2. Prior work

One of the most common routing solutions today is the Routing Information Protocol (RIP) [13]. Its popularity stems from the fact that RIP is included in most

versions of UNIX. RIP is a dynamic routing protocol that automatically creates and maintains network routes. Although popular, RIP has many shortcomings. One of the major problems of RIP is its reactive nature. When a link is not heard from for a predetermine amount of time, it is considered down and an alternative route is sought.

Open Shortest Path First (OSPF) [20] 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. OSPF is an Interior Gateway Protocol, and its intended use is 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 daemon does not know that a problem has occurred until a time-out value is reached before a new route is sought out.

The External Gateway Protocol (EGP) suite [24], sometimes referred to as Border Gateway Protocol (BGP), are network to network routing protocols as opposed to host to host routing protocols. These protocols are used in constructing Wide Area Networks, however, they do not provide fault tolerance to small server network clusters [18].

While RIP, OSPF, EGP and BGP are routing solutions to many different routing problems, they do not address the needs of a high availability, server cluster environment. Their primary goal is to provide routing updates to other routers on the network to find alternative routes to the same network. The general design goal is based on reactively rerouting when a specified timeout period is reached. So, if a destination network does not respond to a route query, after some time quantum, it is considered down and a new route is sought after. The DRS algorithm is proactive and node oriented where each node of a server cluster is constantly monitored to maintain a communications link. If that link does not work, a redundant route is sought after in a distributed manner [16]. The DRS system is similar in approach to some telecommunication approaches without using specialized hardware [6, 11, 12, 15].

Network Survivability Analysis (NSA) [23] was developed to quantitatively evaluate different network topologies. NSA numbers increase with redundancy and decrease with series components. More redundancy with less hardware becomes the design objective instead of the use of redundancy to correct a reliability or survivability deficiency. The DRS system uses redundant hardware to provide alternative routes to network nodes. We provide a NSA type evaluation of the DRS providing several probability evaluation techniques for the DRS. We can then quantify the DRS improvements to a server cluster system with different network topologies.

3. DRS algorithm

The Dynamic Routing System (DRS) improves fault tolerance via proactive failure recognition and the use of a redundant network. Thus, each computer has two network interface cards connected to two separate networks. It is the task of the routing daemons to monitor the connections between two servers. If a failure occurs,

the daemons set up new point-to-point routes around the problem before network applications are aware that a problem occurred.

The DRS runs on every node in the server array. Each DRS daemon is configured to monitor hosts on the networks and executes a two stage run process. In the first phase, the communication links between the local host and all other hosts that is it is configured to monitor are checked. These checks are accomplished using the ICMP (Internet Control Message Protocol) [21] echo request. 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 and host kernel, are operational. The DRS continues to test all known hosts on all known networks in the same manner.

Each daemon keeps track of which hosts to monitor and the state that they are in (i.e., "up," "down"). If a failure occurs, the DRS daemon 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. The execution flow of the DRS algorithm in illustrated Figure 1.

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

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

Step 2. Sleep: Each DRS is dormant at startup. The dormant initial condition prevents false negative results of a ping at startup. Periodically a system might 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 daemon sleep for a predetermined amount of time at startup, false failures are avoided.

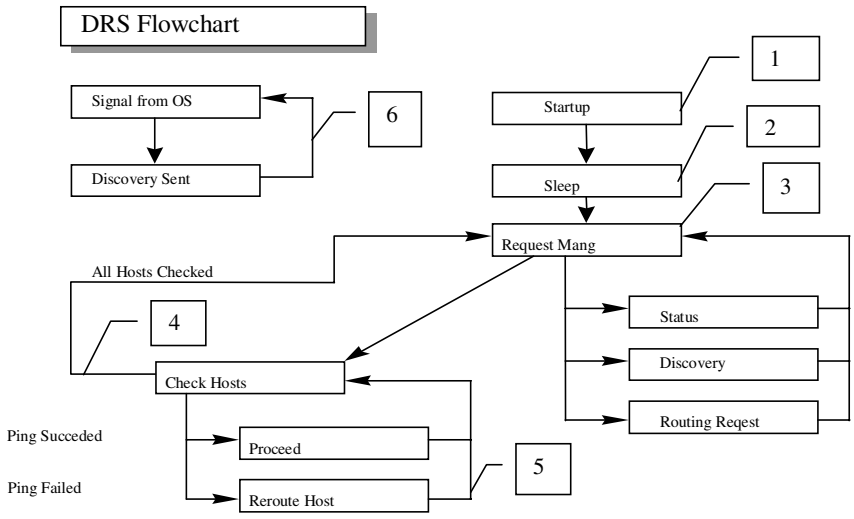


Figure 1. DRS Algorithm.

Step 3. Monitor Messages: The DRS is a routing daemon. 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. A "request for information" can be an administrator contacting the daemon and requesting a view of its routing tables, or a remote server that is 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 greater detail during Step 6.

Step 4. Link Status: 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 milliseconds where X is a configurable setting. Note that X affects the speed an error is detected and affects the amount of network overhead incurred.

Step 5. Fix Communication Errors: In this step, the DRS 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 daemons to see if they can communicate with the host or network in question. The first DRS daemon to respond is used as a router to the lost host, and a new route is added to the kernel's routing table. If no one responds, the host in question has suffered at least two hardware failures and has become disjoint from the rest of the network. If this has happened, the only remaining option is to notify the system administrator of a catastrophic failure.

Step 6. Send "Discovery messages": This stage runs as a separate thread of execution in the daemon. The DRS sends a broadcast message on all available network interfaces stating its own server identification and its server's network interfaces addresses. This message is crucial for several reasons, the most significant being that if a network failure did occur and was fixed, the DRS would otherwise not be updated of the fixed status because "down" routes are not checked. By sending this message, the other DRS daemons become aware that a "down" interface is now working again, and the daemon corrects all rerouted communications of that host to the original routes.

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

continuous, its execution interval is very low thus introducing very little overhead on the network.

4. DRS failure analysis

The DRS handles many different failure situations. We now briefly describe several common failure situations and the solutions the DRS algorithm will compute. Network failures can be categorized into three scenarios:

- Single network failure
- Multiple network failures
- Complete network separation failures

We describe the action of the DRS in each of these scenarios. We use a four-node server cluster as our cluster to demonstrate failure scenarios, where the servers' conical names are "A," "B," "C," and "D," and each server node is connected to network 1 and network 2. The single network failure is the most common and simplest failure (see Figure 2). 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.

Server A sends an ICMP echo request for each node and link in its routing table (part of Step 4). The ICMP echo request does not responded for server 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 there 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 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.

Single network failures are the most common. However, multiple network failures occur, and the DRS reduces the likelihood of a system failure in most multiple failure situations. There are two kinds of multiple failures: those failures that are equivalent in algorithm perception and handled by the DRS as any single network failure and those where each failure is routed via a routing element.

Hub or switch failure situations where all interfaces on a single network fail at once are treated by the DRS algorithm as a single failure. Multiple failures, although

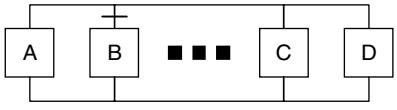


Figure 2. Single network failure.

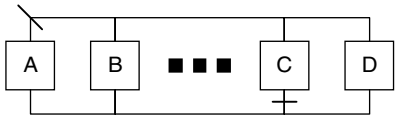


Figure 3. Multiple network failure.

unlikely, are not always as well behaved. The likelihood of an error occurring on the primary or secondary network is smaller than the possibility of it happening randomly to both networks. Thus, the DRS must be able to handle staggered network failures. An example of a staggered multiple network failure would be server A's network port failing and server C's network card failing. Note in the example, a simple two-network solution would fail because no direct network communication route is available (see Figure 3).

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 and is placed in the "DOWN" state. Host C's interface on network #2 also fails and is placed in the "DOWN" state. The DRS now attempts to find alternative routes. 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. 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's interface #1. A static route-using node B is added to node A's routing table.

Since two routes must be known for each node, A must find a route for host C interface #2. 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. Now all communication routes to host C are restored using a remote host as routers between the nodes.

The final failure scenario is a complete network separation or node failure (Figure 4). A complete node failure cannot be solved by any routing solution, and we base our survivability analysis and routing on the assumption that the sender and receiver are working. A complete network separation is a failure scenario where both network interfaces for a single node fail, thus isolating the node from the system. We address the probability of these cases in our probability model. In

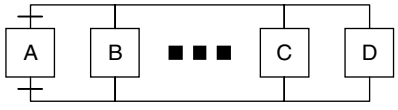


Figure 4. Complete network separation.

the next section, we provide a network survivability analysis of the DRS. We represent the network communication link as a single line connecting all nodes. This is a single network element and not a connected link topology, and we have chosen this representation because of its frequent use in IP network diagrams.

5. DRS proactive cost

The DRS's proactive monitoring of network links comes at a cost of network bandwidth. To find errors before they effect network communication, the links must be checked frequently. If the links are not checked frequently, the DRS is equivalent to a reactive routing protocol. As the number of nodes increases, the bandwidth required to support the frequent checks likewise increases. In Figure 5, we present the maximum number of servers in the cluster that the DRS supports given a requirement for error resolution in X time units and the percentage of network bandwidth useable by the DRS. As shown in Figure 5, ninety hosts are supported in less than 1 second with only 10% of the bandwidth usage.

Each ICMP echo request is 64 bytes in length. As the number of nodes increases the number of checks required to maintain link connectivity status increases. Thus, for each node there are $2(n-1)$ messages and a total of $2n(n-1)$ messages for the system. For a given number of nodes and a frequency rate of checks the amount of bandwidth used can be calculated. In Figure 5, we show that relationship. Our production machines did not see any degradation in performance from the added network usage.

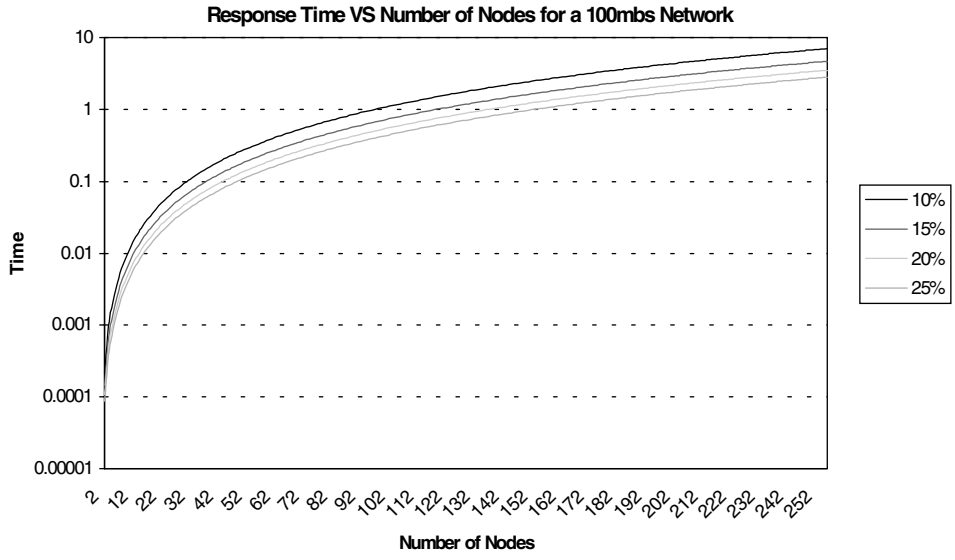


Figure 5. Bandwidth usage.

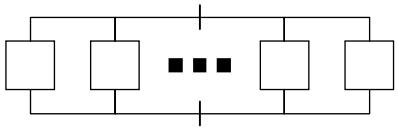
6. Network survivability analysis

In the prior sections, we described the DRS algorithm, and examined its solution to network failure scenarios. We now present two probability models to quantitatively compare the DRS algorithm to single network approaches, and dual-network approaches. The first model gives us a success probability based upon the unconditional failure probability for the system as a whole. The second model gives us the probability of a successful connection between any two nodes given a system with N nodes and f network failures.

6.1. Unconditional failure probability model

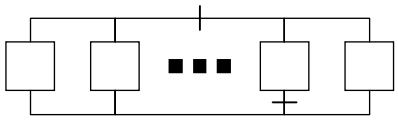
The following analysis describes the probability of system failure based upon the success and failure probabilities of each individual component. We assign p as the probability that a component will function properly, and q as the probability that a component will fail, with $p + q = 1$. The formula can be extended trivially if the components have distinct failure probabilities.

Case 1. Both backplanes fail (q^2). In this case, the system fails and the probability is q^2.



Case 1—Both backplanes.

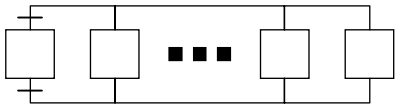
Case 2. Exactly one backplane fails ($2pq$). In this case, the system fails if and only if at least one of the two interfaces of the pairs connecting to the working backplane fails. So the probability is $2^* p^* q^* (1 - p^* p) = 2pq(1 - p^2)$.



Case 2—Hub and NIC failure.

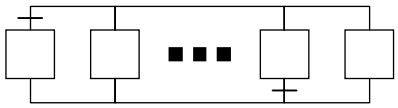
Case 3. Neither of the backplanes fails (p^2).

Case 3.1. Both interfaces of the source node fail (q^2). So the failure probability is $p^{2*} q^2 = p^2 q^2$.



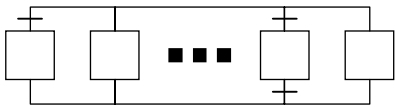
Cases 3.1, 3.3—Both NIC.

Case 3.2. Exactly one interface of the source node fails ($2pq$). In this case, the interface of the destination node at the same side of the working interface of the source node must be down (q).



Case 3.2.

Case 3.2.1. The other interface of the destination node fails (q). So the failure probability is $p^{2*}2pq^*q^*q = 2p^3q^3$.



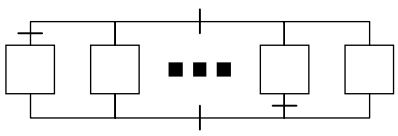
Case 3.2.1.

Case 3.2.2. The other interface of the destination node works (p). In this case, all other $N-2$ bridges must be down $(1-p^2)$. Thus the failure probability is

$$p^{2*}2pq^*q^*p^*(1-p^2)^{N-2} = 2p^4q^2(1-p^2)^{N-2}.$$

So the total failure probability in Case 3.2 is

$$2p^3q^3 + 2p^4q^2(1-p^2)^{N-2}.$$



Case 3.2.2.

Case 3.3. Neither of the interfaces of the source node fails (p^2) (see Case 3.1). In this case, both interfaces of the destination node must fail (q^2). So the failure probability is

$$p^{2*}p^{2*}q^2 = p^4q^2.$$

Therefore, the total failure probability in Case 3 is

$$p^2q^2 + 2p^3q^3 + 2p^4q^2(1-p^2)^{N-2} + p^4q^2.$$

So the total failure probability is

$$q^2 + 2pq(1-p^2) + p^2q^2 + 2p^3q^3 + p^4q^2 + 2p^4q^2(1-p^2)^{N-2}.$$

Therefore, we can write the probability of success as

$$P[\text{Success}] = 1 - [q^2 + 2pq(1-p^2) + p^2q^2 + 2p^3q^3 + p^4q^2 + 2p^4q^2(1-p^2)^{N-2}]. \quad (1)$$

DRS Unconditional Failure Probability

To compare results, we also provide equations for a dual network system and a single network system. Using the same methods, the probability of success of a dual network can be written:

$$P[\text{Success}] = 1 - [q^2 + 2pq(1-p^2) + p^2q^2 + 2p^3q^2 + p^4q^2]. \quad (2)$$

Dual Network Unconditional Failure Probability

Likewise, the probability of success for a single network system can be written:

$$P[\text{Success}] = 1 - [q + pq + p^2q]. \quad (3)$$

Single Network Unconditional Failure Probability

The dual and single networks are independent of N because they do not have the re-routing algorithm, while the DRS equation will approach a specific probability as $N \to \infty$.

We examined twenty-seven field deployed server clusters containing over one hundred servers for hardware failures. In a one-year period, 13% of the hardware failures were network related, i.e., network hubs, network switches, network interface cards, etc. Given this actual usage data, we define $q = 0.13$ and $p = 0.87$. Using Eq. (1), we can calculate the unconditional failure probability of a given system. Using a cluster size of 20 servers and a probability of failure of 13%, we find that the DRS system yields an unconditional failure probability 37% greater than a single network topology for 20 clustered server nodes.

If no prior data are available, we can assign an equal probability of success and failure to each node, i.e., $p = q = 0.5$ and evaluate the probability of unconditional failure at any moment for our system. When evaluating our systems probability of success with Eq. (1), the DRS for any given moment is 112% better than a single network topology and 13% better than a dual-network topology. In Figure 6, we illustrate a comparison of the DRS versus dual and single network topologies presented in Eqs. (1), (2) and (3) for $p = q = 0.5$ and X nodes, respectively.

6.2. *Conditional failure probability model*

The unconditional failure probability gives us a total probability of success model for the entire system. We now present a quantitative probability model to evaluate systems in terms of a given number of network failures occurring at a given instance. In this model, we determine the probability of success, independent of time, of a system with N nodes and f failures. We assume that, in a system with N nodes, there are exactly $2N$ interface connections and 2 non-meshed back planes, each with equal probability of failure, say q, for $0 \leq q \leq 1$. Therefore, the probability of 2 failures in any system will be q^2, the probability of 3 failures will be q^3, and the probability of f failures will be q^f. It follows that

$$\lim_{f \to \infty} q^f = 0.$$

Consequently, the probability of multiple failures decreases exponentially.

As the number of nodes in a system increases, the probability of that system maintaining a successful connection between any two nodes at any given time will approach 1 for a fixed number of failures, using the DRS. Since there are $2N + 2$

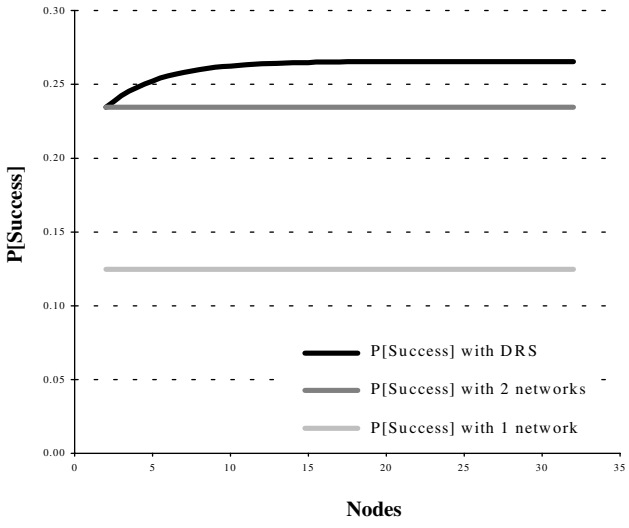


Figure 6. Unconditional failure probability.

total connections that the f failures can be distributed among, the total number of combinations of f failures in the system is $\binom{2N+2}{f}$. We now count the number of failure combinations that result in the failure of the communication between a specific pair of nodes.

Case 1. Both backplanes fail. In this case, the remaining $f-2$ failures appear in the $2N$ components. The total number of such combinations is $\binom{2N}{f-2}$.

Case 2. Exactly one backplane fails. The total number of combinations of f-failure is $\binom{2N}{f-1}$. To make the specific pair unable to communicate, at least one of the two interfaces of the pairs connecting to the working backplane fails. Therefore, the number of failures that will not result in the failure of communication between the given specific pair of nodes is $\binom{2N-2}{f-1}$. The total number of such combinations is $\binom{2N}{f-1} - \binom{2N-2}{f-1}$.

There are two cases in which exactly one backplane fails, so the total number of combinations is $2 \cdot \left[\binom{2N}{f-1} - \binom{2N-2}{f-1}\right]$.

Case 3. Neither of the backplanes fail.

Case 3.1. Both interfaces of the source node fail. The total number of combinations is $\binom{2N-2}{f-2}$.

Case 3.2. Exactly one interface of the source node fails. In this case, the interface of the destination node at the same side of the working interface of the source node must be down.

Case 3.2.1. The other interface of the destination node fails. So the remaining $f-3$ failures appear in $2N-4$ components. Thus, the total number of combinations is $\binom{2N-4}{f-3}$.

Case 3.2.2. The other interface of the destination node works. In this case, all other $N-2$ bridges must be down. Therefore, the remaining $f-2$ failures appear in the $N-2$ bridges, and each bridge must contain at least one failure. Among them $(f-2) \bmod (N-2) = f-N$ bridges contain two failures, and $(N-2)-(f-N) = 2N-f-2$ bridges contain exactly one failure. There are $\binom{N-2}{f-N}$ choices of the bridges with both failed links. For each such choice, there are 2^{2N-f-2} configurations of the remaining single-failure bridges. So, the total number of combinations is $\binom{N-2}{f-N} \cdot 2^{2N-f-2}$.

There are two cases in which exactly one interface of the source node fails so the total of combinations in Case 3.2 is $2 \cdot \left[\binom{2N-4}{f-3} + \binom{N-2}{f-N} \cdot 2^{2N-f-2}\right]$.

Case 3.3. Neither of the interfaces of the source node fails. In this case, both interfaces of the destination node must fail. So the total number of failures is $\binom{2N-4}{f-2}$.

Therefore, the total number of failures in Case 3 is

$$\binom{2N-2}{f-2} + 2 \cdot \left[\binom{2N-4}{f-3} + \binom{N-2}{f-N} \cdot 2^{2N-f-2}\right] + \binom{2N-4}{f-2}.$$

Therefore, the total number of failure combinations is

$$F(N,f)=\binom{2N}{f-2}+2\cdot\left[\binom{2N}{f-1}-\binom{2N-2}{f-1}\right]+\binom{2N-2}{f-2}$$
$$+2\cdot\left[\binom{2N-4}{f-3}+\binom{N-2}{f-N}\cdot 2^{2N-f-2}\right]+\binom{2N-4}{f-2}$$

and the probability of success for N nodes and f failures can be written as:

$$P[\text{Success}]=\frac{\binom{2N+2}{f}-F(N,f)}{\binom{2N+2}{f}}. \tag{4}$$

Probability of Success

Using Eq. (4), it is readily apparent that, as shown in Figure 7, the probability of success converges to 1 as N gets large for fixed values of f. More specifically, for $f=2$ the P[S] surpasses 0.99 at 18 nodes. For $f=3$ the P[S] surpasses 0.99 at 32 nodes, and for $f=4$ the P[S] surpasses 0.99 at 45 nodes. Given that $\lim_{f\to\infty} q^f=0$ and that $\lim_{N\to\infty} P[\text{S}]=1$, a system implementing the DRS has a high probability of resilience to network failures.

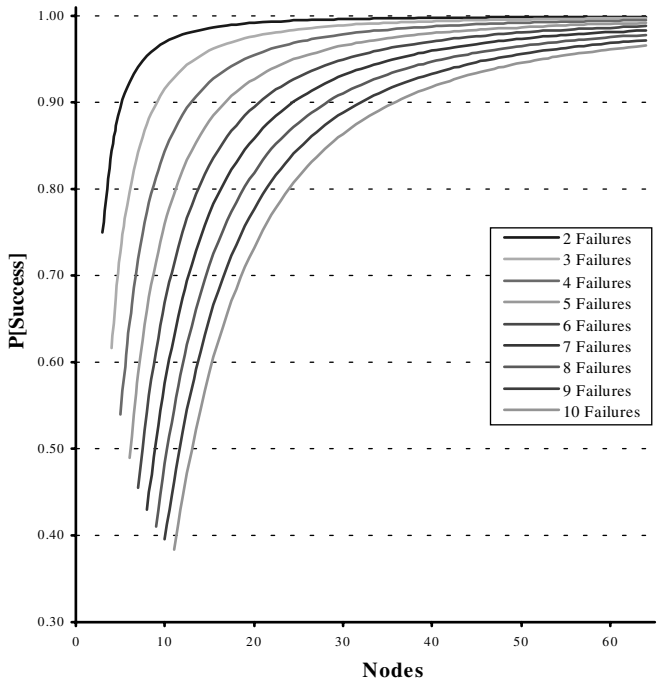


Figure 7. Convergence of P[Success] to 1.

7. DRS comparisons

Previously, we provided two probability models of the DRS algorithm. The first was an unconditional failure probability of the entire system. We compared the DRS system to a dual network topology and a single network topology. We showed that for the DRS system the probability of success for any given moment is 112% better than a single network topology and 13% better than a dual-network topology that do not have proactive or reactive routing. We then provided a conditional failure probability model to evaluate the system in terms of number of network failures at a given instance.

We now provide a quantitative comparison of the DRS to other solutions. The simplest solution is a single network topology where all nodes are connected via a shared hub. Although the cheapest in terms of hardware costs, as shown by our probability model evaluation, this solution is the least reliable of the viable solutions. A solution is to add a second network to the system to provide a redundant communication path. This approach requires changes to most operating systems since they are traditionally not designed to detect a network failure. Hence, the ability to select the appropriate network card once a failure occurs requires modification to the kernel and also introduces overhead to detect the failure. Although possible, MCI WorldCom was not interested in modifying the operating system. Our solution assumes that a second network is available. This allows the DRS to *reactively* re-route a connection if a network failure occurs.

With the probability model provided above, we now compare the DRS to the previously discussed solutions in terms of probability of success, number of nodes, and number of failures, f.

In Figures 8 and 9, comparisons of three network topologies are shown for two and three network failures. The first is a single network topology. The DRS for

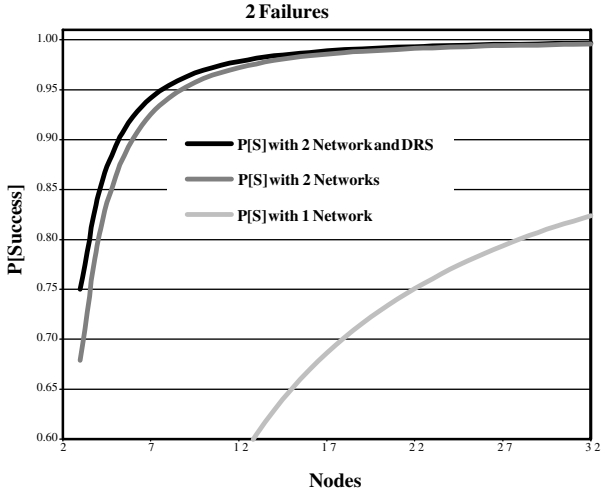


Figure 8. Probability of success with 2 failures.

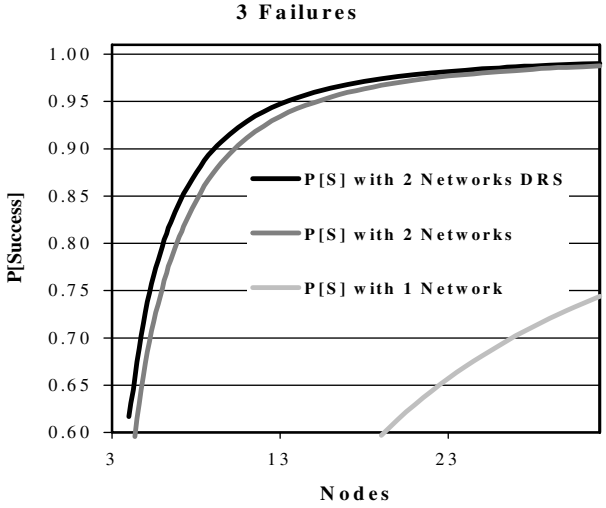


Figure 9. Probability of success with 3 failures.

an eight-node cluster with three simultaneous failures is 267% more reliable that a single network topology 129% more reliable with two network failures.

While re-routing without routing software does not work with most operating systems, the DRS only nominally improves on simple dual network topologies when comparing with a fixed number of failures. The reason is that DRS handling of opposite network failure situations is only a small percentage of the total number of network failure scenarios when the communication backplane is shared. Most switches use a fully connected mesh to create more communication channels to avoid channel contention. Therefore, the assumption that the communication link is one link is true for non-switched hub topologies where the hub uses a shared media for all port-to-port communications.

Our existing models are a lower bound result for shared backplane topologies; in reality, the DRS probability of success is better because of the ability of DRS to route around single channel failures in a backplane mesh.

8. Conclusions

We briefly described the Dynamic Routing System designed as developed for MCI WorldCom. Further details can be found in [7]. As of date, the DRS approach is deployed in at least twenty-seven installations nationwide for enhanced voice services. Each deployed instillation consists of eight to twelve servers.

We provided two probability models to quantitatively compare different network topologies. The first model gives an unconditional failure probability of the entire system. We compared the DRS system to a dual network topology and a single network topology. We showed that the probability of success of the DRS for any given moment is 112% better than a single network topology and 13% better than

a dual-network topology. We also presented a second probability model to evaluate the system in terms of the number of failures. Using Eq. (4), we showed that the probability of success converges to 1 as N gets large for fixed values of f. More specifically, for $f = 2$ the $P[S]$ surpasses 0.99 at 18 nodes. For $f = 3$ the $P[S]$ surpasses 0.99 at 32 nodes, and for $f = 4$ the $P[S]$ surpasses 0.99 at 45 nodes. Given that $\lim_{f\to\infty} q^f = 0$ and that $\lim_{N\to\infty} P[S] = 1$, a system implementing the DRS has a high probability of resilience to network failure.

We compared the DRS to single network topologies and showed that for an eight-node cluster with three simultaneous failures, the DRS is 267% more reliable than a single network topology and 129% more reliable with only two network failures. We also demonstrated that the DRS results are a lower bound result when compared to dual network topologies for a fixed number of failures and would provide greater resilience to network failures than other topologies.

The DRS proactive approach comes at the price of network bandwidth. We show in Figure 5, that ninety hosts using the DRS can detect and fix most network failures in less than 1 second with only 10% of the bandwidth of a 100Mbs network.

In conclusion, based on field data, while the DRS does use network bandwidth to pro-actively monitor network communication links, this added overhead has not effected the overall performance of the deployed systems, and the network failures that did occur in the field did not affect performance. This improved reliability is because only one network is relied upon, thus failed links used the redundant network and provided the same bandwidth for the applications. As shown by our quantitative analysis, the DRS performance is better in terms of fault tolerance added to distributed-server clusters than other topologies or solutions. The timeliness of the DRS reroute solution prevented the performance degradation present in solutions where timeout approaches are used.

References

1. T. Anderson, D. Culler, and D. Patterson. A case for NOW (Networks of Workstations). *IEEE Micro* 15, 1995.
2. G. R. Ash. *Dynamic Routing in Telecommunications Networks*, McGraw Hill, 1998.
3. S. Bahk and M. E. Zarki. Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *Proceedings of ACM SIGCOMM*, August 1992.
4. J. Casas, R. Konuru, S. Otto, R. Prouty, and J. Walpole. Adaptive load migration systems for PVM. In *Proceedings of Supercomputing 94*, pp. 390–399, Washington D.C., November 1994.
5. J. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks, *INFOCOM*, 22–31, 2000.
6. I. Chlamtac, A. Farag'o, and T. Zhang. Optimizing the system of virtual paths. *IEEE/ACM Transactions on Networking*, 2(6):581–587, 1994.
7. A. Chowdhury, O. Frieder, E. Burger, D. Grossman, and K. Makki. *Dynamic Routing System (DRS): Fault Tolerance in Network Routing*, Computer Networks and ISDN, Elsevier Science Publishing Co., North-Holland, 31(1–2), January 1999.
8. I. Cidon, R. Rom, and Y. Shavitt. Analysis of multi-path routing. *IEEE/ACM Transactions on Networking*, 7(6):885–896, 1999.
9. W. Dees and R. Smith. Performance of Interconnection Rip-Up and Reroute Strategies. In *Proceedings 18th Design Automation Conference*, June 1981, pp. 382–390.

10. J. Dongarra, S. Otto, and M. Snir. *MPI: The Complete Reference*, The MIT Press, 1996.
11. D. Z. Du, D. F. Hsu, and F. K. Hwang. Doubly link ring networks. *IEEE Transactions on Computers*, C-34(9):853–855, 1985.
12. R. J. Gibbens and F. P. Kelly. Dynamic routing in fully connected networks. *IMA Journal of Math. Control & Information*, 7:77–111, 1990.
13. C. Hedrick. Request For Comment 1058, "Routing Information Protocol," 06/01/1988, http://ds.internic.net/ds/dspg2intdoc.html
14. B. R. Hurley, C. J. R. Seidl, and W. F. Sewell. A survey of dynamic routing methods for circuit-switched traffic. *IEEE Communications Magazine*, 25(9), 1991.
15. F. P. Kelly. Bounds on the performance of dynamic routing schemes for highly connected networks. *Mathematics of Operations Research*, 19:1–20, 1994.
16. P. B. Key and G. A. Cope. Distributed dynamic routing schemes. *IEEE Communications Magazine*, 28:54–64, 1990.
17. K. R. Krishnan, R. Doverspike, and C. Pack. Improved survivability with multilayer dynamic routing. *IEEE Communications Magazine*, July 1995.
18. S. Low and P. Varaiya. Stability of a class of dynamic routing protocols (IGRP). In *IEEE Proceedings of the INFOCOM*, vol. 2, pp. 610–616, March 1993.
19. K. Makki, N. Pissinou, and O. Frieder. Efficient solutions to multicast routing in communication networks. *ACM Journal on Mobile Networks and Nomadic Applications*, 1(4):1996.
20. J. Moy. Request For Comment 1583, "OSPF Version 2," 03/23/1994, http://ds.internic.net/ds/dspg2intdoc.html
21. J. Postel. *Internet Control Message Protocol (ICMP)*, RFC 792, 1981.
22. C. Reschke, T. Sterling, and D. Ridge. A design study of alternative network topologies for the beowulf parallel workstation. In *Proceedings of the Fifth IEEE Symposium on High Performance Distributed Computing*, 1996.
23. R. Talbott. Network survivability analysis. *Fiber and Integrated Optics*, 8:13–43, 1988.
24. K. Varadhan. Request For comment 1503, "BGP OSPF Interaction," 01/14/1993, http://ds.internic.net/ds/ dspg2intdoc.html
25. E. Wong, A. Chan, and T. Yum. Analysis of rerouting in circuit-switched networks. *IEEE/ACM Transactions on Networking*, 8(6):419–427, 2000.