Performance Analysis of Fast Distributed Link Restoration Algorithms[1]

C. Edward Chow, John D. Bicknell[2], Steve McCaughey
Department of Computer Science
University of Colorado at Colorado Springs
Colorado Springs, CO 80933-7150
TEL: (719) 593-3110
FAX: (719) 593-3542
Email: chow@quandary.uccs.edu

and

Sami Syed
MCI Telecommunications
4678 Alpine Meadows Lane
Colorado Springs, CO 80919
Phone: (719) 592-1461
Email: 2912712@mcimail.com

Abstract

Four distributed link restoration algorithms are analyzed in detail using a set of important performance metrics and functional characteristics. The functional characteristics are used to explain how these algorithms function and provide insight into their performance. The analysis and simulation results indicate that the Two Prong link restoration algorithm, which is based on issuing aggregate restoration requests from both ends of the disruption and on an intelligent backtracking mechanism, outperforms the other three algorithms in terms of restoration time. The RREACT link restoration algorithm consistently found paths that use fewer spares.

I. Introduction

With the widespread deployment of fiber optic transmission systems and the alarming rate of outages due to fiber cuts [1], there is great interest in strategies for improving the process of restoring disrupted traffic from minutes to sub-seconds following a fiber cut [2]. Automatic protection switching probably is the fastest technique and can switch the disrupted traffic to dedicated spare links in under 50 milliseconds. However, it requires high dedicated spare capacity. With recent advances in digital cross-connect systems, DCS, there is increasing interest in using DCS in network restoration [3,4,5,6,7,8,9,10]. The centralized DCS-based network restoration approach [3,4,5] requires reliable telemetric links between the DCS nodes and the network operation center. It is slower than distributed DCS-based network restoration, where the affected DCS nodes exchange messages directly to restore the disrupted traffic. The hybrid preplanned approach proposed in Bellcore's NETSPAR uses a distributed topology update protocol to identify the fault and then downloads a precomputed routing table according to the fault. The problem with this approach is that the memory required for storing the routing tables is too great [6]. In this paper, we will focus on distributed network restoration algorithms for DCS-based fiber networks.

There are two basic approaches to reroute the disrupted traffic due to a fiber span cut. The *link restoration* approach replaces the affected link segment of a disrupted channel by a spare path between the two disrupted ends. The *path restoration* approach releases each disrupted channel and

1. This research was supported by MCI with grant #01-80046.

2. John Bicknell was a graduate student at UCCS when this work was done and is now with MCI.

lets the source and destination end of the channel re-establish the connection. With the additional release phase the path restoration will take more time than the link restoration. However, the path restoration can find more efficient spare paths with fewer link segments and can handle the node failure situation with the same logic. The network restoration techniques described in [3,5] fall in the path restoration category. In order to achieve fast network restoration, we focus on the link restoration approach in this paper.

The existing shortest path algorithms provide the basic building blocks or inspire the basic approaches adopted by the existing network restoration algorithms. The survey article of Deo and Pang [11] classified the shortest path algorithms into four classes: 1) one (node) to all (nodes) [12], 2) one to one [13], 3) all to all [14], and 4) k-shortest path algorithms [15,16,17]. The k-shortest path algorithms try to find k paths between the source node and the sink node. Some of them find k shortest paths sequentially. Others find k paths where the total length of k paths are minimum. They can further be divided by the nature of the path found. Some find node disjoint paths. Others find edge disjoint paths. Dijkstra's shortest path algorithm [12], which is a labelling process initiated at the source node, has had great influence on the design of these k-shortest path algorithms and hence the design of existing network restoration algorithms. Mohr and Pasche [18] indicated that for finding the one-to-one shortest path, Nicholson's shortest path algorithm [13] is about 60% faster than Dijkstra's and Bovet's algorithm [19] performs far fewer iterations than Dijkstra's. The basic idea behind Nicholson's algorithm is a two-prong labelling process that is initiated simultaneously from both the source and the sink. This two-prong labelling idea inspired us to look at the network restoration problem from a different angle than the existing distributed network restoration algorithms which initiate the network restoration process from only one of the two disrupted nodes.

Routing algorithms proposed for computer networks [20,21,22,23,24] find the shortest paths from each node to all other nodes in a network. The approximate distributed Bellman-Ford Algorithms in [24] has polynomial message complexity and very fast response time. However, the goal of routing algorithms is quite different from that of the distributed network restoration, which finds the shortest paths between two disrupted nodes. The response time requirement of the network restoration algorithm is almost two orders of magnitude of that imposed on routing algorithms.

Since the publication of Grover's Self-Healing Network [25], a number of distributed network restoration algorithms have been proposed for DCS and ATM based networks. These algorithms include the Self-Healing Network (SHN) [8,25], FITNESS [9], RREACT [26], Komine [10] and Two Prong [27]. A good tutorial on the basic characteristics of these network restoration algorithms is provided in [2]. Most of these algorithms only handle link failures. Komine [10] also deals with node failures but it is not very efficient in handling the link failure situations due to its complicated, variable length, message structure. Since the algorithms for handling node failures are different from the link restoration algorithm in terms of the restoration strategy and the network status information to be maintained, in this paper, we will concentrate on the analysis of the link restoration algorithms. We have identified five important performance metrics to be used to evaluate these distributed link restoration algorithms and to compare their relative efficiency. In addition to these metrics, we have introduced a set of functional characteristics to be used for explaining how distributed link restoration algorithms function, their differences and why these algorithms perform as they do. Throughout the paper, comparisons and observations are made relative to the functional characteristics and performances of the four distributed link restoration algorithms.

In [27] a distributed link restoration algorithm called Two Prong was proposed and preliminary simulation results were presented. The Two Prong algorithm is based on issuing aggregated bandwidth requests from both ends of the disrupted connections and on an intelligent backtracing mechanism which resolves the conflicts in the aggressive bandwidth reservation process deployed by the algorithm. It was indicated that although the Two-Prong outperforms the other algorithms in most cases, there are cases in which the Two Prong did not achieve 100% restoration when it could have.

To understand and to improve the performance of the Two Prong algorithm, the Two Prong and other distributed network restoration algorithms were systematically examined in detail. The functional characteristics and the performance metrics were applied to explain how the Two Prong algorithm functions and performs. It was found that the analysis based on these functional characteristics gave us insight on the behavior of distributed network restoration algorithms and helpful to improve the Two Prong algorithm to achieve a higher restoration level.

In this paper, three network topological problems are identified to be among the causes of low network restoration level. It is shown how the improved Two Prong algorithm solves them and how these solutions are related to the algorithm’s functional characteristics.

The rest of this paper is organized as follows: Section II gives definitions of the terms related to the link restoration. Section III provides a brief overview of the link restoration algorithms. Section IV defines the performance metrics used to evaluate how well a link restoration algorithm performs. Section V defines and describes the functional characteristics used to analyze these algorithms. Section VI contains a brief analysis of the link restoration algorithms using the functional characteristics. Examples of how the Two Prong algorithm solves specific, commonly occurring problems are also included. Section VII contains an evaluation of the Two Prong algorithm with tables comparing its performance with the other algorithms. It also discusses the impact of parallel DCS op0ating mode on restoration time. Section VIII contains a brief summary.

II. Network Restoration Definitions

A *network* is defined as a system of switching nodes connected by communication lines, and can be represented as an augmented undirected graph with a set of nodes and a set of links. See Figure 1. A link connects two nodes in the network and has an associated bandwidth. The bandwidth of a

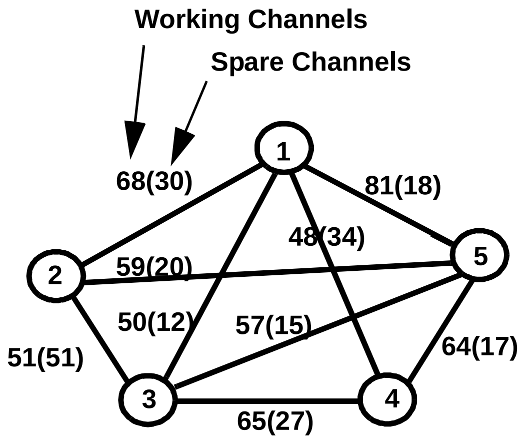

Figure 1: An Example Network.

link is divided into basic units called *channels*. Each channel is in one of two states: *working* or *spare*. Each link in the network is labeled with two numbers, representing the number of working and spare channels in the link. A *route* is specified as an ordered set of concatenated link IDs. The *hop count of a route* is the number of links in the route. A *path* is specified by an ordered set of concatenated channel IDs. The *hop count of a path* is the number of channels in the path. A *working path* is a path where all channels are working channels while a *spare path* is a path where all the channels are spare channels. A *restoration path* is a spare path that is designated for restoring a disrupted working channel due to a network failure.

The *distributed link restoration algorithms* are distributed algorithms that on detecting a link failure, initiate restoration requests, find as many restoration paths to replace the disrupted channels, and issue the connection re-establishment commands to the involved switching nodes. The *restoration time* of a link restoration algorithm for a given link failure, is the period between the detection of the link failure to the time the last restoration path is connected to the disrupted working path. Given a link failure, the *restoration level* achieved by a link restoration algorithm is the percentage between the number of the restoration paths found by an algorithm and the number of the disrupted working channels on the failed link. The *spare usage* of a link restoration is the total number of spare channels in the restoration paths found by a link restoration algorithm.

III. Overview of the Distributed Link Restoration Algorithms

A. Grover's Self-Healing Network Approach

The first distributed link restoration approach for a DCS-based fiber network was proposed by Grover in [25] and detailed in his Ph.D. dissertation [8]. The protocol associated with the algorithm is called the Self-Healing Network (SHN) protocol. In the SHN protocol, one of the two DCS nodes on detecting the fiber cut becomes the Sender based on some arbitration rule, such as larger DCS network ID, and the other becomes the Chooser. Then, the request messages, called *signatures*, will be sent out along all the spare channels on all outgoing fibers. Each of these signatures will bear different indices. These signatures will be broadcasted to the intermediate nodes between the Sender and the Chooser. The Chooser on receiving a signature, will check the index. If it is the first time the Chooser has received a signature with this index number, then a reply signature will be sent back through the same request path. Each of the intermediate nodes on receiving a reply signature will generate a switch command to the DCS to connect the ports of the two spare channels. This is called reverse linking. When the Sender receives the reply signature, it will reconnect one of the disrupted channels to the new spare path and send the information of the restored channel ID through the spare path back to the Chooser. The Chooser, on receiving the information of the restored channel ID from the spare path, will reconnect the corresponding ports to restore the disrupted channel. This protocol basically requires three message transmissions between the Sender and the Chooser for each disrupted channel.

Note that one signature is sent out for each spare channel connected to Sender and is indexed for distinction. On the same route, these signatures compete for computation resources in each DCS node for processing.

Figure 2 shows the execution steps of Grover's approach for a single channel restoration route. At Step 1, the Sender broadcasts the signature towards the Chooser. At Steps 2 and 3, the signature will be propagated along the available spare channels on other outgoing links. At Step 4, after receiving the signature, the Chooser will send a reply signature back towards the Sender. At Steps 5 and 6, the intermediate node on receiving the reply signature will make the DCS connection and forward the reply signature along the restoration path. At Step 7, the Sender receives the reply signature and will select one disrupted channel to connect the restoration path. A mapping information message including the ID of this disrupted channel will then be sent through the connected restora-

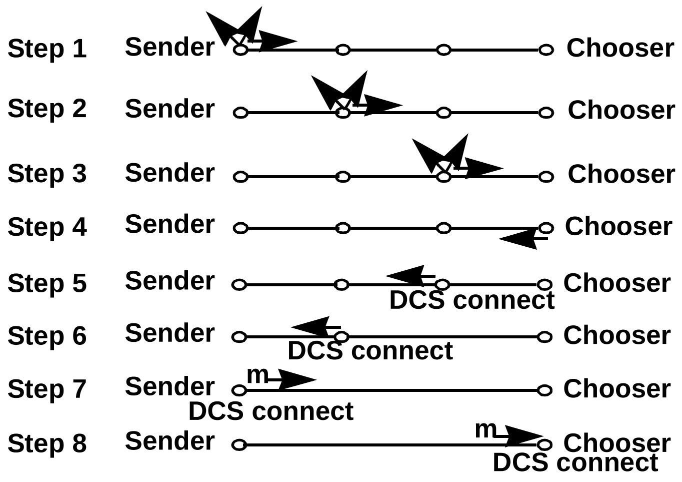

Figure 2. The execution steps of Grover's SHN algorithm.

tion path directly to the Chooser without the need for message processing in the intermediate nodes. At Step 8, the Chooser on receiving the mapping information message will connect the restoration path to the corresponding disrupted channel and the restoration will be complete.

B. Bellcore's FITNESS Approach

Following Grover's publication, another distributed network restoration process for DCS-based fiber networks was proposed by Yang and Hasegawa in [9]. This method is called FITNESS and also uses a Sender—Chooser relationship for the nodes adjacent to the fiber link cut. FITNESS reduces the potentially large number of signatures that may be generated in SHN by requesting the aggregated maximum bandwidth that is allowed on a restoration route. The restoration process is initiated by the Sender which will broadcast restoration request messages, called help messages, on all links which contain spare channels. Each help message will contain the Sender address, Sender-Chooser pair ID, source of the message, destination of the message, requested bandwidth and hop count. Requested bandwidth will be the minimum of the working channels lost due to the fiber link cut and the spare capacity of the particular link over which the specific help message is being broadcasted.

Help messages will be selectively broadcasted by intermediate nodes. Each intermediate node will maintain a table of the help messages it has received. This table contains the source of the help message, requested bandwidth and hop-count of the path from the Sender. The first help message received will always be broadcasted. Successive help messages are broadcasted only if the requested bandwidth is greater than all previously received messages. When help messages with a requested bandwidth equal to earlier messages, but with a lower hop count are received, such messages will not be broadcasted. Instead, table hop count and source entries will be modified to reflect the discovery of the shorter path. Help messages with lower bandwidth than any table entries or equal bandwidth and higher hop count will be ignored. The requested bandwidth in help messages broadcasted by intermediate nodes will be the minimum of the arriving message's requested bandwidth and the spare capacity of the link over which the help message is being sent.

On detection of fiber link failure, the node which becomes the Chooser will set a fixed time-out. The length of time for the time-out must be determined empirically, but appears optimal in the range of 250 to 350 msec. During the time-out, the Chooser will maintain a table of all help messages received. On the termination of the time-out, the Chooser will select the table entry corresponding to the largest requested bandwidth and send an acknowledgment message to the source of the selected help message.

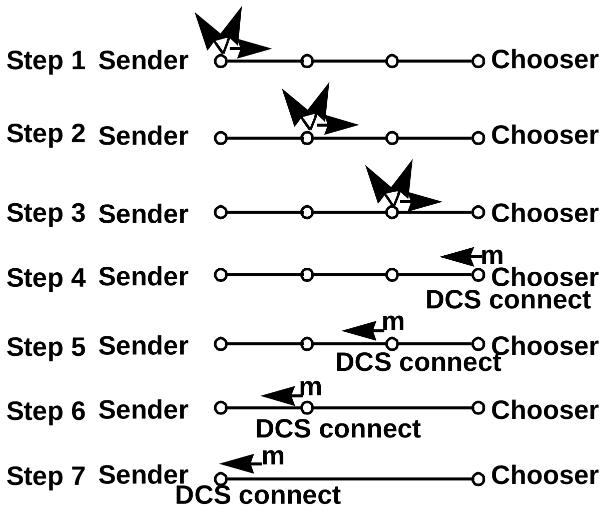

Figure 3. The execution steps of FITNESS approach.

On receipt of an acknowledgment message, each intermediate node will reply with a confirmation message and then will send the acknowledgment message to the next node along the path to the Sender. As this process continues, each node on receipt of a confirmation message will make cross-connections to restore lost working channels. If a single restoration path provides insufficient bandwidth to affect full restoration of all lost working channels, the Sender initiates a new wave of help messages and the process is repeated until all channels are restored, or no new paths can be found.

Figure 3 shows the execution steps of the FITNESS algorithm on a restoration route. Note that these steps are similar to those of SHN except that at Step 4 the Chooser makes the DCS connection according to the available bandwidth in this restoration route and sends the mapping information message with the IDs of the restored channels to the Sender. This allows FITNESS to have one less step than SHN. The messages in FITNESS are longer than those in SHN. The request message contains the maximum capacity and hop count of the restoration route being explored. The acknowledgment message in FITNESS contains the mapping information of a set of restored channels instead of one in SHN. The message transmission time in FITNESS therefore is longer than that of SHN but the message processing frequency in each node is greatly reduced.

C. RREACT Approach

RREACT is another distributed approach to network restoration and is described in detail in [26]. This method also uses a Sender—Chooser and flooding approach as in the FITNESS and Self-Healing approaches. What distinguishes this approach is that the restoration request messages, called seek messages, include information about the path which each seek message has traversed from the Sender node to the Chooser node. Seek messages are propagated at each node as they arrive. As they are transmitted from node to node, the node ID's of the nodes they have visited and the number of spare channels available over each link they have traversed is added to a variable length field in the seek message format. This path information is inspected at each node as a seek message is received. If the message has visited the node before, it is discarded.

The effect of this flooding approach is that all possible paths between the Sender and Chooser nodes are discovered and information is received at the Chooser node which enables it to create a matrix describing the current topology of the network. As each seek message arrives at the Chooser with a new path description, the matrix is updated and the path is selected, if possible, for use as restoration path and the matrix again updated to account for the spare channels taken to create the new restoration path.

The actual path building process is very similar to that of the FITNESS approach as shown, previously, in Figure 3.

D. Two Prong Network Restoration Approach

For comparison purposes, Figure 4 is provided to show the execution steps of the Two-Prong approach on a restoration route. Note that by letting both ends broadcast the request message the inter-

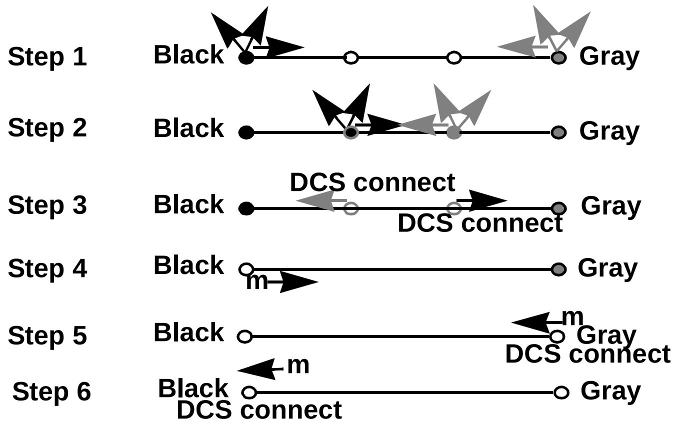

Figure 4. The execution steps of Two-Prong approach.

mediate nodes start the DCS connection as early as at Step 3, compared with FITNESS that performs DCS connection after Step 4.

The Two Prong algorithm’s name is derived from its fundamental approach to finding network restoration paths. Unlike other distributed approaches, the Two Prong algorithm does not use a *Sender-Chooser* relationship for the nodes adjacent to the fiber link cut to select one node to be the Sender and only to allow the Sender to initiate the restoration process. Instead, the two nodes perform nearly symmetrical roles throughout the execution of the Two Prong algorithm. On detecting a fiber link cut, one adjacent node will be arbitrarily designated the *Black-Origin* node. The other node will be designated the *Gray-Origin* node. Restoration will be initiated from both nodes with the Black-Origin node broadcasting *Black* restoration request messages and the Gray-Origin node broadcasting *Gray* restoration request messages. These messages are sent out on all links which contain spare channels. Each restoration request message (Black or Gray) will contain the pair ID of the Black-Origin and Gray-Origin nodes, the source of the message, the destination of the message and the requested bandwidth. Requested bandwidth will be the minimum of the working channels lost due to the fiber link cut and the spare capacity of the particular link over which the specific Black or Gray message is being broadcasted.

On receipt of a Black or Gray message, an intermediate node becomes designated as a Black or Gray node, respectively. Black intermediate nodes will selectively broadcast Black messages and Gray intermediate nodes will selectively broadcast Gray messages. A table is maintained of all messages which have been received, whether they are broadcasted or not. Selection for broadcasting is determined by the available spare channel capacity on links to neighboring nodes. If there are any available spare channels over a given link, the Black or Gray message will be broadcasted. The requested bandwidth will be the minimum of the arriving message's requested bandwidth and the remaining unrequested spare capacity of the link over which the request message is being sent. A 'scratch-pad' record of unrequested spare capacity is maintained so that successive Black or Gray messages can also be broadcasted. This scratch pad record ensures that the total amount of restoration bandwidth requested over any given link does not exceed the total number of available spare channels in that single link. We call this mechanism *floodgating*.

As Black and Gray messages propagate across the network, Black messages will begin arriving at Gray nodes and Gray messages will begin arriving at Black nodes. A node, upon receiving a different 'colored' request message, will make appropriate cross connections between the links over which the two different (Black and Gray) requests were received. Once the cross connection has been made, the request message will be forwarded over the newly connected link to the next node in the restoration path. A single message could trigger cross-connections for multiple paths branching from a single node. If this occurs, each node on the other end of these newly connected links would have to be informed of the number of spare channels which had been connected from that link. Any requested bandwidth which cannot be accommodated is reported back to the requesting node in a Backtrack message of the appropriate color. An example based on Figure 5 will make these operations clear.

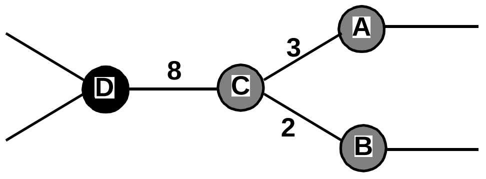

Figure 5: Two Prong Message Processing

Assume node C first receives a Gray message requesting 3 channels from node A. C will then record this request in its message table and become a *Gray* node. This request will also be propagated to nodes B and D. Later, node C receives another Gray message requesting 2 channels from node B. Again C will record this request and propagate the message to nodes D and A. After this, assume node C receives a Black message requesting 8 channels from node D. Node C will then cross connect three spare channels to node D with the three spare channels to node A. Node C also will forward a Black message to node A requesting 3 channels. Node C then will cross connect two spare channels to node D with the two spare channels to node B. Node C also will forward a Black message to node B requesting 2 channels. Since node C could support only 5 of the 8 channels contained in the Black request, node C will send a Backtrack Black message to node D reporting that only 5 channels were connected.

Backtracking will occur only when a request is forwarded and connections are not available to accommodate all or a portion of the requested bandwidth. The short fall will be reported in a Backtrack message as described in the example, above. Upon receipt of a Backtrack message, the node will disconnect any channels to the false path and redirect the restoration path in another direction by generating a new request message of the same color as the Backtrack message. If no alternative path exists, the Backtrack message will be sent back along the path over which the original request message came. When a Backtrack message is received by a Black-Origin or Gray-Origin, disconnections will be made, as required.

As restoration paths become established between the Black-Origin and the Gray-Origin, Black messages will be converging on the Gray-Origin and Gray messages converging on the Black-Origin. On receiving a Gray message, the Black-Origin will insert an Ack message into the channels of the new restoration path. On receiving a Black message, the Gray-Origin will begin *listening* to the spare channels in that link for an Ack message from the Black-Origin. Once this message is received, the Gray-Origin will forward a *Confirm* message containing mapping information informing the Black Origin of which disrupted channels are to be connected to which restoration paths. Then the Gray-Origin will make final connections to restore its disrupted channels. The Black-Origin, upon receipt of the Confirm message from the Gray-Origin will make its final connections to restore its disrupted channels. Note that these Ack and Confirm messages are sent in-band and do not require message processing by intermediate nodes.

Once the Black-Origin has restored all lost channels, it will begin flooding *Cancel* messages to all neighbors. Cancel messages are used to terminate the algorithm and return the network to normal operation. Cancel messages also ensure that any cross-connections which are not used by paths in the final restoration solution are disconnected. Cancel messages contain the source and destination of the message and a bandwidth value. The value of this bandwidth is set to the number of working channels in the link over which the Cancel message is sent from the Black-Origin. After broadcasting these messages, the Black-Origin returns to normal operation or *White* state.

When Black and Gray nodes receive a Cancel message, they will compare the value contained in the bandwidth field to the number of working channels on the link over which the message was received. If there are more working channels than the number specified in the Cancel message, disconnections will be made to reduce the number of working channels so that they match the number contained in the Cancel message. The node will then flood Cancel messages in the same manner as the Black-Origin node and then will go to a White state. On receiving a Cancel message, the Gray-Origin will make any necessary disconnections and go the White state.

When a White node receives a Cancel message, it will make any necessary disconnections. A White node will selectively flood a Cancel message only to a neighboring node with which it made a disconnection. The bandwidth specified in this message will be equal to the number of working channels existing on the link connecting the White node to its neighbor after the disconnection has been made.

Table 4 shows the state transitions of the Two Prong algorithm.

IV. Performance Metrics for Link Restoration Algorithms

The five performance metrics we have identified for evaluating Link Restoration algorithms are:

1) Time to restoration.

2) Restoration level.

3) Spare usage.

4) Range of application.

5) Message volume.

Performance metric one, time to restoration, refers to the time required by the algorithm to complete execution to whatever level of restoration it can achieve. Since it is desirable to accomplish restoration as quickly as possible to avoid call dropping, this is an extremely important metric. Ideally, an algorithm must achieve full possible restoration in less than two seconds, including completion of any required cross-connections.

Performance metric two, restoration level, refers to how many of the lost working channels are restored. The ideal is that all lost channels be restored. This may not always be possible. Three situations can occur which limit restoration. The first situation is that there is not sufficient spare capacity in the network to support restoration of the lost working channels, even with an optimal algorithm. Another situation can occur that while there is considerable spare capacity within the network, overall, it is distributed in such a way that restoration cannot be achieved in a specific failure scenario. Typically, this occurs when a node adjacent to the link failure has too few spare channels to its other neighbors to restore the lost working channels. Allocation strategies for spare channels is a complete area in itself for research (see [28,29]). Certain network topologies can create situations which are pitfalls for distributed algorithms using a heuristic approach to path finding or restoration path selection (see [30]). Such situations can result in the algorithm achieving a less than optimal level of restoration.

Performance metrics 1 and 2, when combined, are the most critical performance criteria for any network restoration algorithm. The ideal is a 100% restoration within two seconds. In situations in which an algorithm cannot achieve full restoration within two seconds, the rate at which the algorithm restores lost channels can be of important. Figure 6 illustrates this point. The vertical axis rep-

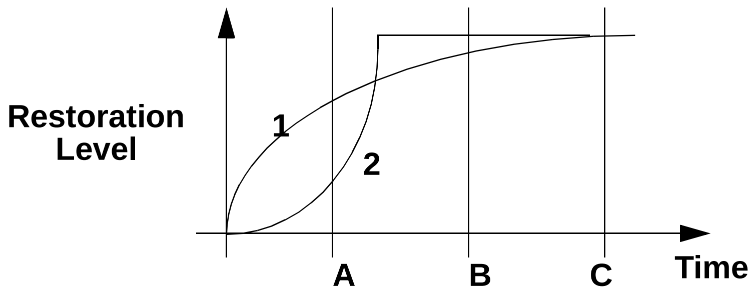

Figure 6: Time to Restoration Level

resents level of restoration and the horizontal axis the time required to achieve that level of restoration. The two curves represent the rate at which two algorithms, 1 and 2, achieve increasing levels of restoration.Three time marks are shown, A, B, and C. If two seconds of elapsed time occurs at time mark C, then both algorithms have restored 100%. If, however, two seconds of elapsed time occurs at time mark A, then algorithm 1 is clearly superior to algorithm 2 as it achieves a higher level of restoration. On the contrary, if two seconds of elapsed time occurs at time mark B, then the restoration level achieved by algorithm 2 is higher than algorithm 1.

Performance metric 3, spare usage, refers to how many spare channels are switched to working channels to replace lost working channels. In the link restoration approach, it requires at least twice as many spare channels to replace the working channels lost. Since bandwidth is a limited (and expensive) resource within the network, it is desirable that as few spare channels as possible be employed in the restoration solution.

Performance metric 4, range of application, refers to what different kinds of failure scenarios the algorithm can be applied to affect restoration. A number of the proposed distributed algorithms can only address single link failures. A limited number of algorithms can be used to restore lost working channels in multiple link failure and node failure scenarios.

Performance metric 5, message volume, refers to how many network restoration messages are generated by a restoration algorithm. It is desirable that the number of messages an algorithm generates be as few as possible. Not only does message volume affect performance metric 1 (time to restore), it also limits other network restoration message traffic flow during the restoration process which may be of high or critical priority.

It should be noted that we have not included among the performance metrics the number of distinct paths an algorithm uses in its restoration solution. Although we have found a high correlation between the number of paths used in a restoration solution and the time to restoration metric of a specific algorithm, when comparing across algorithms, this correlation does not exist. This particular observation is further discussed in Section V. As regards the merits, in and of itself, for having fewer or greater numbers of paths in a restoration solution, we have found no particular benefit to either one. In general, while we have found some differences among the several algorithms in the number of distinct paths which are used in final restoration solutions, they often are the same and reflect more the topology of the network and the location of the link failure, rather than the heuristic methods of the algorithms.

V. Functional Characteristics of Link Restoration Algorithms

We have identified seven functional characteristics which can be attributed to distributed link restoration algorithms. These are related to certain fundamental tasks which an algorithm must perform in the restoration process when using a distributed approach. These functional characteristics can be used to systematically analyze distributed link restoration algorithms to understand how they function and perform. These functional characteristics are:

1) Find paths.

2) Resolve spare channel contention.

3) Select restoration paths.

4) Control message volume.

5) Control congestion.

6) Counter race conditions.

7) Connect restored paths.

Functional characteristic 1, find paths, relates to how an algorithm identifies possible restoration paths. Most distributed algorithms use some form of flooding to do this. Ideally, all paths which can be used in restoration are identified. There are also two approaches in finding paths. One is to limit, through some heuristic mechanism, path finding to a subset of all paths in a network, the other approach is to perform an exhaustive tracing of all paths in the network. This functional characteristic is fundamental to the performance of the algorithm, affecting all other functional characteristics and the final performance metrics.

Functional characteristic 2, resolve spare channel contention, relates to how an algorithm resolves multiple requests for the same spare channel. Since all the candidate restoration paths in a restoration solution usually are not disjoint paths, there is some competition among these paths for available spare channels. At some point in the execution of these distributed algorithms there exists the potential that the same spare channel may have more than one reservation or attempted reservation made to use it in the restoration solution. Such contention must either be avoided or resolved by the algorithm. The method used to resolve spare channel contention can have critical impact upon the level of restoration achieved and message volume.

Functional characteristic 3, select restoration paths, refers to the process the algorithm uses to select from among the identified candidate paths those which are to be used in the restoration process. Most of the distributed algorithms use a ‘first come, first served’ approach which is effectively a shortest path heuristic. Some, however, do use other heuristics based on other factors such as the bandwidth of the path. Since the selection process involves contention for spare channels, there is often a strong relationship between selection of restoration paths and resolution of spare channel contention. In fact, in some algorithms, both are simultaneously executed in some centralized body of code in a node which is given specific responsibility to select restoration paths from those available. Path selection can have critical impact upon the restoration level which is achieved and the utilization of spare channels.

Functional characteristic 4, control message volume, refers to what mechanisms the algorithm uses to reduce the number of messages generated during execution of the algorithm. It is desirable that algorithms generate as few messages as possible. A large number of messages has an adverse impact upon the time to restore metric and affects the network’s capacity to process other time critical network restoration messages not involved in the restoration algorithm. Some algorithms inherently generate fewer messages than others, but most employ some heuristic mechanism to further reduce message volume. The most commonly used heuristic is hop count. That is, a message used to reserve spare bandwidth contains a field which counts the number of nodes through which it has been transmitted. When some predefined limit is reached, the message is ignored and not propagated further. A similar heuristic is a time stamp which lets a message die after a certain time period such as 1 second (half the 2 second time constraint). Some algorithms use more sophisticated methods.

Functional characteristic 5, control congestion, refers to what techniques an algorithm uses to control message congestion at critical nodes. This characteristic is related to functional characteristic 4, control message volume and is strongly affected by functional characteristic 1, find paths. Critical nodes are nodes which, because of their location within the network, will process a larger number of messages than other nodes in the network. Typically these are nodes in a ‘hub’ type position, i.e. a node with a relatively high connectivity, or nodes which are adjacent to the link or node which failed. Nodes adjacent to a failure site are usually more critical than ‘hub’ nodes in that they normally assume special roles and have additional responsibilities in the restoration process. In some distributed algorithms, congestion is the key component affecting time to restoration. An algorithm which produces high congestion at a critical node loses many of the benefits of parallelism which are sought by using a distributed approach. All the other nodes are able to quickly process their messages and completion of the algorithm’s execution becomes dependent upon a single node’s ability to process a much greater number of messages.

Functional characteristic 6, counter race conditions, refers to an algorithm’s method to control or respond to race conditions during execution of the restoration algorithm. A network topology affects the speed with which messages propagate through a network. The timely arrival of a message, or the relative arrival times of messages to a certain node may affect performance of an algorithm. Particularly affected may be the time to restore and level of restoration performance of the algorithm. Some algorithms are not adversely affected by race conditions while others may be severely affected. Those which can be affected by race conditions require certain mechanisms to counter race conditions. These typically are tables which record information from messages received earlier to compare with messages arriving later or some time out mechanism to allow the arrival of a larger number of messages before taking some specified action.

Functional characteristic 7, connect restored paths, refers to the method used by the algorithm to make final restoration path connections. Most algorithms use a three phased approach: identify a path between the two end nodes, an acknowledgment is sent from one node to the other, the second node then sends a confirmation that the path has been connected. This process is usually handled by network restoration messages which are relayed from node to node over the restored path.

VI. Analysis of the Distributed Link Restoration Algorithms Using Functional Characteristics

In this Section, we analyze the distributed link restoration algorithms with special emphasis on the performance of the Two Prong algorithm using the functional characteristics discussed in Section V. The Two Prong algorithm differs from other algorithms in several respects. These differences are best identified by analyzing its functional characteristics, that is, how it finds paths, resolves spare channel contention, selects restoration paths, controls message volume, controls congestion, counters race conditions and connects restored paths.

A. Path Finding

The Two Prong algorithm finds paths by flooding Black and Gray messages across the network. The path traveled by a Gray message, from the Gray-Origin to the Black-Origin, defines a path. Initially, this flooding is very similar, in most respects, to the flooding mechanisms used by other distributed network restoration algorithms. However, once a Gray message encounters a Black node, it is no longer flooded to all the Black node’s neighbors, instead it is forwarded to the first neighbor to have sent a Black message to the Black node. This results in the Gray message *homing in* towards the Black-Origin, directly following a path that has been traversed earlier by a Black message. This also helps to reduce message volume. The same process is also happening to the Black messages which are travelling a path from the Black-Origin to the Gray-Origin. The resulting effect is a shortest-path heuristic as the messages which arrive the earliest at an origin node have traversed a shorter route than later arriving messages.

It should be noted that many of the paths which are used in the final restoration solution are what we describe as *natural paths*. That is, paths which are easy to find using a shortest path methodology. In our test results, we have found that 75% to 100% of the paths which are used in a restoration solution are easily found by simple flooding. Some restoration scenarios, however, have more complicated solutions. To find these other paths, Two Prong uses flooding and sometimes backtracking logic to attempt connections among alternate paths. In comparison, the Self-Healing Network and RREACT algorithms use an exhaustive search of all possible paths to find all the restoration paths. The FITNESS algorithm uses multiple waves to find all restoration paths.

Figure 7 is given as an example of how complicated a restoration solution can be in even a simple network. It demonstrates what we call the *Three Finger Problem* and explains how the Two Prong algorithm finds the restoration paths in the network.

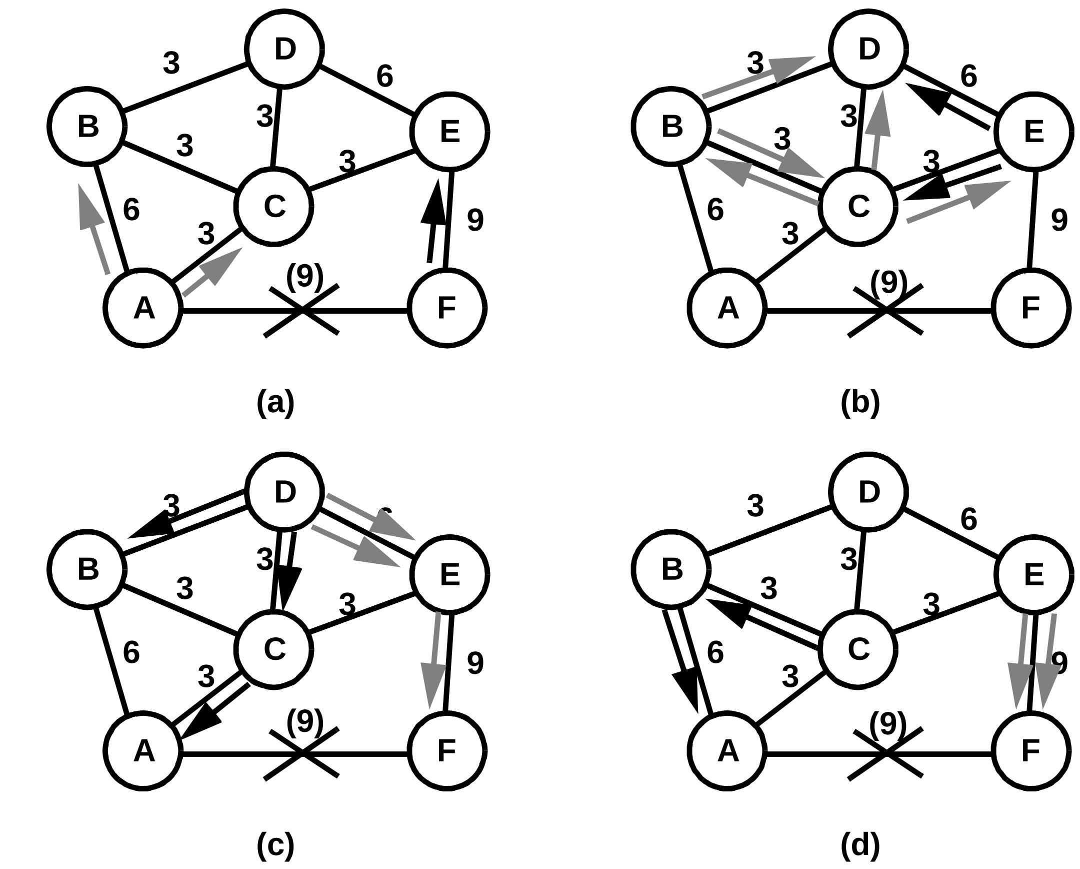

Figure 7: Three Finger Problem.

Refer to Figure 7a and assume that a single link failure has occurred between Nodes A and F with the loss of 9 working channels. Further assume that Node A becomes the 'Gray-Origin' and Node F becomes the 'Black-Origin'. Both nodes immediately, upon detection of the fault, begin flooding messages to their neighbors. Figure 7a shows the initial propagation of messages: Gray message for 3 channels from Node A to C, Gray message for 6 channels from Node A to B, and Black message for 9 channels from Node F to E.

In Figure 7b, the second stage of message propagation is shown: Gray message for 3 channels from Node B to D, Gray message for 3 channels from Node B to C, Gray message for 3 channels from Node C to B, Gray message for 3 channels from Node C to D, Gray message for 3 channels from Node C to E, Black message for 3 channels from Node E to C, Black message for 6 channels from Node E to D.

Node C has now received Gray messages from Nodes A and B. Node C has also received a Black message from Node E. Since the Gray message from Node A arrived earlier than the Gray message from Node B, Node C makes cross connections for 3 channels on the links connecting Node C with Nodes A and E. Node C will also forward the Black message from Node E to Node A. Node C will not broadcast the Black message.

Concurrent Path Setup

Note that Node E is able to, concurrently, make cross connections between Node C and Node F for the same path Node C is building. Similarly, Node D makes cross connections for the Gray and

Black messages it has received from Node B and Node E, respectively and also for the Gray and Black messages it has received from Node C and Node E, respectively.

Figure 7c shows the resulting message flows: Gray message for 3 channels from Node E to F, Two Gray messages for 3 channels each from Node D to E, Black message for 3 channels from Node D to B, Black message for 3 channels from Node D to C, and Black message for 3 channels from Node C to A. Node C will now be able to make cross connections for the Gray message it received earlier from Node B and the Black message now arriving from Node D.

Figure 7d shows the Black and Gray messages now converging on the Origin nodes. At this point the final solution is clear. There will be three paths used in the final restoration solution: Path #1 is A-C-E-F for 3 channels, Path #2 is A-B-D-E-F for 3 channels, and Path #3 is A-B-C-D-E-F for 3 channels. A total of 36 spare channels are used in this restoration solution.

B. Spare Channel Contention Resolution

The Two Prong algorithm uses several mechanisms to resolve spare channel contention. The first is the use of ‘floodgates’ to reduce spare channel contention. Unlike most other distributed algorithms which will flood multiple messages over a link without regard to the total bandwidth which has been requested by these messages, Two Prong keeps track of the total requested bandwidth over a single link and will not forward requests for more bandwidth than the link can support. This mechanism in itself will not stop spare channel contention, but substantially reduces the problem. Spare channel contention occurs in the Two Prong algorithm when multiple paths have a common upstream node which flooded the Gray or Black message to its neighbors. Then, because of this propagation, the node has generated more bandwidth requests than it had received from the node before it and has ‘over committed’ the link from which it received the message. We have identified and named this the ‘Funnel Problem’. To overcome this situation, the Two Prong algorithm uses a backtracking logic to reroute messages to other available paths. Such backtracking usually happens at the node at which the propagation occurred and caused the over capacitation of the link. A simple example of the Funnel Problem is illustrated in Figure 8 and explained in the following text.

Note that although the network topologies of the *Three Finger Problem* and the *Funnel Problem* are very similar, the actions taken by Node C (a ‘hub’ node) are quite different. Assume again a single link failure between Nodes A and F. Again these two nodes flood messages to their neighbors. This initial flooding is shown in Figure 8a. Note the creation of the *Funnel Problem* which occurs because Node C received a Gray message for 3 channels from Node A and, because it hasn’t yet received any Black messages, floods this message to Nodes D and E. Each flooded Gray message from Node C is for 3 channels. This has actually multiplied the original request for 3 channels and results in a situation in which both Nodes D and E think they can each make a connection for 3 channels with Node C.

At this point, Node C has already made cross connections between Nodes A and E. Node D has received Gray messages from Nodes C and B and has received a Black message from Node E. Node D will try to first make connections between Nodes C and E (we’re assuming here, that the Gray message from Node C arrived before the Gray message from Node B). So Node D will try to connect 3 channels from Node C with 3 channels from Node E and will connect 3 channels from Node B with 3 channels from Node E.

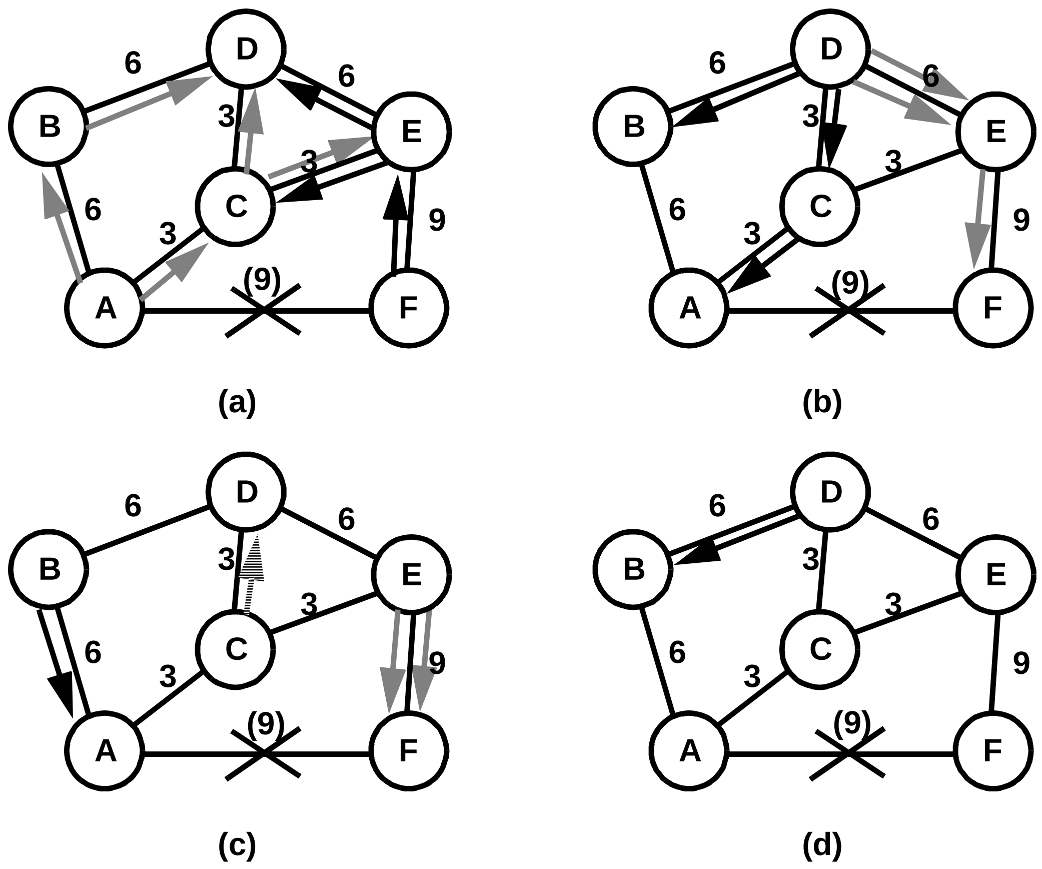

Figure 8: Funnel Problem.

Figure 8b shows the resulting messages: Gray message for 3 channels from Node E to F, Two Gray messages for 3 channels from Node D to E, Black message for 3 channels from Node C to A, Black message for 3 channels from Node D to C, and Black message for 3 channels from Node D to B.

At this point, Node C receives the Black message which it cannot connect since it has already allocated all available spare channels with Node A. Therefore, Node C sends a Backtrack Black message to Node D. All the other Gray and Black messages are processed in ways previously described.

Figure 8c shows the resulting messages: Black message for 3 channels from Node B to A, two Gray messages for 3 channels each from Node E to F, and a Backtrack Black message from Node C to D.

Node D, upon receipt of the Backtrack Black message from Node C, will try to redirect the original Black message request for (now) 3 channels in another direction. The node has kept a record of the Gray messages it has seen and will redirect the request to Node B by forwarding a Black message for 3 channels. Figure 8d shows this taking place.

The Final restoration solution for the Funnel Problem are three paths: Path #1: A-C-E-F for 3 channels, Path #2: A-B-D-E-F for 3 channels and Path #3: A-B-D-E-F for 3 channels. Note that Paths 2 and 3 can be thought of as a single path with a bandwidth of 6 channels, but since they are separately created, we count such paths as distinct paths in our performance analysis.

The Self Healing Network algorithm resolves spare channel contention by reserving specific spares to restore specific disrupted channels. Self Healing is able to do this since it does not use aggregate bandwidth request messages, but instead uses single signatures for each individual channel lost. RREACT resolves spare channel contention by building a complete data base of the current state of the network and updating this data base as it selects restoration paths. FITNESS resolves spare channel contention by heuristic methods which limit request message flooding and by selecting a single restoration path during each wave of request messages.

C. Restoration Path Selection

The Two Prong Algorithm selects restoration paths on a ‘first come, first served’ basis. As each individual Gray message arrives at the Black-Origin node, it sends an Ack message which is inserted into the established channel on a path which is usually connected all the way to the Gray-Origin. Since cross-connections are made very early in the restoration process and many restoration paths are quickly identified, Two Prong is able to begin restoration of the lost channels very quickly. This is in effect a shortest path heuristic and also favorably affects the utilization of spare channel resources, in most instances.

The Self-Healing Network similarly uses a ‘first come, first served’ restoration path selection mechanism. This results in some of the same features as the Two Prong algorithm, that is, early restoration of lost channels and good utilization of spare channel resources. The RREACT algorithm, in its basic form, also uses a first come, first served process. There are improvements to the algorithm under study which use a more advanced restoration path selection process. It should be noted, also, that the RREACT algorithm generally has the best spare channel utilization of all the algorithms tested. This is due to its path finding, spare channel contention resolution, and restoration path selection processes. The FITNESS algorithm uses a time-out to collect a number of request messages from the Sender node at the Chooser node. The designers of the FITNESS algorithm consider that it is beneficial that the FITNESS algorithm select from among the available paths the one with the greatest bandwidth. This is, in part, necessary to the performance of the algorithm which restores only a single path in each wave of messages. The greater bandwidth heuristic helps to reduce the number of paths and waves the algorithm uses to restore the lost channels. Unfortunately, this also adversely impacts upon the utilization of spare channels, since the path with the largest bandwidth is often not the shortest path, and contains links that are part of shorter paths with less bandwidth (over their entire lengths). These shorter paths often don’t get chosen until later waves, by which time some of their bandwidth has already been lost to earlier selections which results in a less resource efficient solution. The multi-wave, time-out path selection process, which the FITNESS algorithm uses, also has an adverse effect upon the time to restoration metric. FITNESS generally performs well behind the other algorithms in this regard.

D. Message Volume Control

The Two Prong algorithm controls message volume through the heuristics of message forwarding and floodgating. As already described, once a Gray message encounters a Black node, it is no longer flooded, it is forwarded and vice versa for a Black message. This in itself considerably reduces message volume since a message is flooded across only about half the network. Floodgating also serves to reduce message volume, since many late arriving messages aren’t flooded once all available spare channel capacity on all links from a node has already been reserved.

The Self-Healing Algorithm does not have a mechanism to control message volume. The algorithm is dependent upon low-level hardware adaptations to process signature messages as they propagate across the network. It should be noted that the Self-Healing algorithm's signature messages only request a single bandwidth. All the other algorithms make aggregate bandwidth requests. The RREACT algorithm generates a great number of messages since it performs an exhaustive trace of all possible paths from the Sender node to the Chooser node. However, it does prevent messages from tracing duplicate paths and a time value heuristic is applied which kills 'Seek' messages which are more than one second old. The FITNESS algorithm requires each node to keep a table of the messages it has seen. Messages which arrive and request a smaller bandwidth than earlier messages are thrown away. FITNESS also employs a hopcount mechanism and can be implemented with a limit that throws away request messages which have a hopcount greater than the limit.

E. Congestion Control

The Two Prong algorithm controls congestion, again through the message forwarding and floodgating heuristics. The relatively even distribution of messages in the algorithm helps to reduce congestion at 'hub' nodes. At the critical Gray-Origin and Black-Origin nodes, message volume is very low, and is related to the number of paths used in the final restoration solution. This is because the only Gray messages which travel all the way to the Black-Origin have probably (though not always) traveled over a path which has been connected all the way to the Gray-Origin and can be used as a final solution path. This is a significant factor in the time to restore performance of the algorithm, particularly as network size increases.

In the Self Healing Network algorithm, congestion is not controlled. Signatures are flooded throughout the network and can result in as many signatures arriving at a critical 'Chooser' node as there are spare channels connected to the node. As stated, this algorithm is dependent upon special hardware adaptations to the DCS equipment to process these signatures in a timely manner. Even though the RREACT algorithm uses a heuristic to kill off duplicate and messages older than one second, there is still a high volume of messages, particularly at the 'Chooser' node, which must select from all identified paths, those which are to be used in the restoration solution. This creates considerable congestion which adversely affects the algorithm's time to restoration performance, particularly in large networks. Because of their message flooding heuristics and multiple waves approaches, the FITNESS algorithm have very low congestion, particularly at critical nodes.

F. Race Condition Control

Because intermediate nodes must make path connection decisions, without global knowledge of the network's topology, the Two Prong algorithm is the most sensitive to race conditions of the distributed network restoration algorithms discussed. To counter race conditions, each node keeps a table of all the messages it has seen. A sophisticated logic is required to correctly act upon each message as it arrives. The logic depends upon the message type received, its source, the requested bandwidth, the spare channel capacities of all links connected to the node, and the state of the entries in the message table in the node. An example of a simple race condition problem is provided in Figure 9.

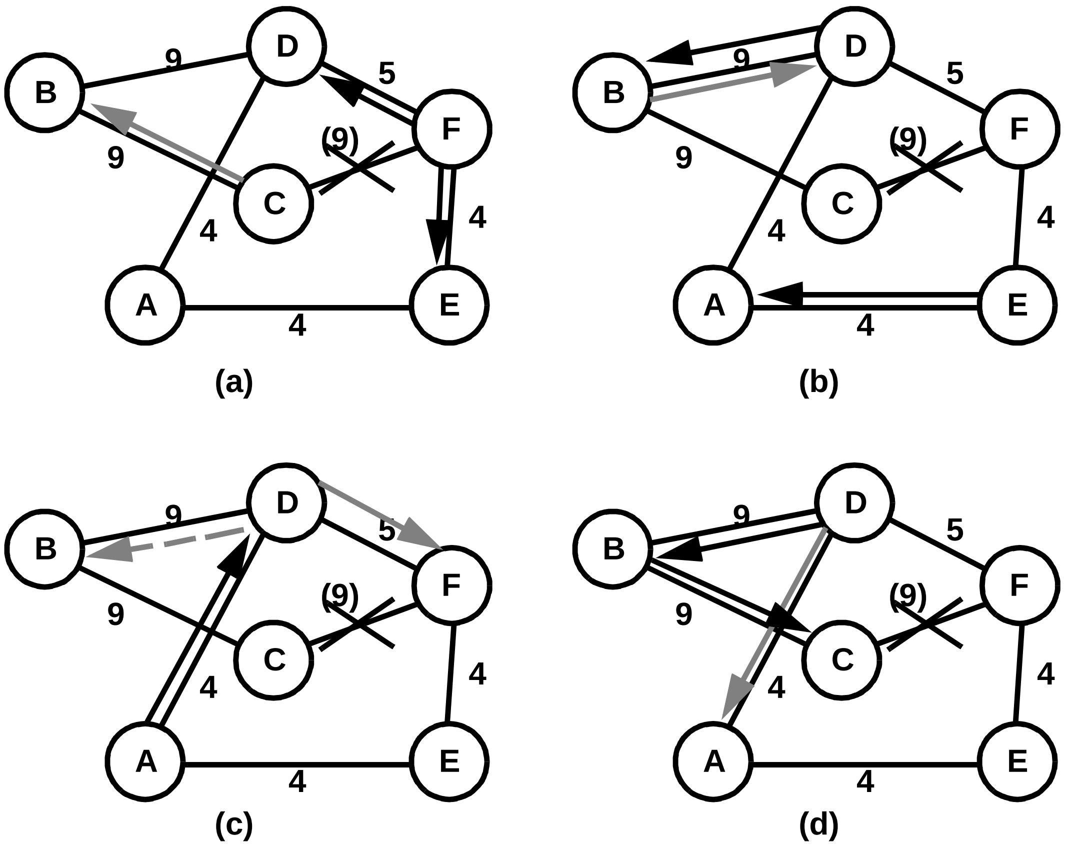

Figure 9: Race Condition Problem.

Assume for this problem that a single link failure has occurred between Nodes C and F with the loss of 9 channels. Assume Node C becomes the Gray-Origin and Node F the Black-Origin and they begin normal flooding to their neighbors. This flooding is shown in Figures 9a and 9b.

At this point, Node D has received a Gray message for 9 channels from Node B and a Black message for 5 channels from Node F. Assume that the Black message arrived first and that Node D is therefore a ‘Black’ node. Node D can now make cross-connections between Nodes B and F for 5 channels. Since Node D is a Black node, it cannot flood (broadcast) the Gray message, it can forward it only in response to a Black message. In the earliest versions of the Two Prong algorithm, Node D would simply Backtrack the Gray message and not record having seen it. This would result in less than full restoration. The current version, however, would send a Backtrack Gray message to Node B, but would also keep a record of the Gray message it received from Node B.

Figure 9c shows the resulting messages: Gray message for 5 channels from Node D to F, Backtrack Gray message that only 5 channels were connected from Node D to B, and Black message for 4 channels from Node A to D.

Now Node D can connect the rest of the channels requested from Node B. The resulting messages are shown in Figure 9d: Black message for 5 channels from Node B to C, Black message for 4 channels from Node D to B and Gray message for 4 channels from Node D to A.

The Self Healing Network and RREACT algorithms are not adversely affected by race conditions and have no special mechanisms to counter them. It should be noted that in the case of the FITNESS algorithm, its multiple waves are controlled by a time-out mechanism which is sensitive to race conditions. The performance of this algorithm can vary considerably due to network dimension and topology, failure location within the network, and the value selected for the time-out period.

G. Restored Path Connection

The Two Prong algorithm is significantly different from the other algorithms in the way in which it connects restored paths. Path connection begins in the Two Prong algorithm from the moment a node has received both a Gray and a Black message. This means that path connection begins earlier than in other network restoration algorithms and a high degree of parallelism is generated as multiple paths are connected in parallel and connections along the same path are done concurrently. Final connections at the disrupted ends are made as each restoration path is identified by both the Gray-Origin and Black-Origin nodes. Since connections through the intermediate nodes are made even before the path is recognized by the origin nodes, Ack and Confirm messages are transmitted ‘in-band’, that is directly from one origin node to the other without processing by intermediate nodes. This significantly improves the Two Prong algorithm’s time to restoration performance over other algorithms which initiate path connection only after the path is identified and make connections in a sequential manner.

In contrast, the Self Healing Network initiates DCS cross connections once a path has been traced by a signature message from the Sender node to the Chooser node. The connections along this path are then sequentially initiated as an acknowledgment message traverses the path from the Chooser node to the Sender. FITNESS is similar, except that each path is of some aggregate bandwidth, whereas the Self Healing Network connects individual channels. RREACT is again similar, initiating cross-connections as each path trace arrives at the Chooser node. It should be noted that in the case of the Self Healing Network and RREACT, some parallelism is preserved since multiple paths are likely to be in the process of making cross connections. In the case of FITNESS, however, only one single path is created at a time, with those cross-connections being implemented in a sequential manner.

VII. Evaluation of the Two Prong Algorithm using Performance Metrics

To compare the performance of the Two Prong, FITNESS, RREACT, and Self Healing Network (SHN) algorithms, we have implemented four simulators with the corresponding protocols using the NETRESTORE simulation system developed at the University of Colorado at Colorado Springs. All algorithm simulators were tested on three networks: the ‘New Jersey Network’ LATA test network which was defined in the FITNESS [9] paper and is shown in Figure 10, the LATA ‘X’

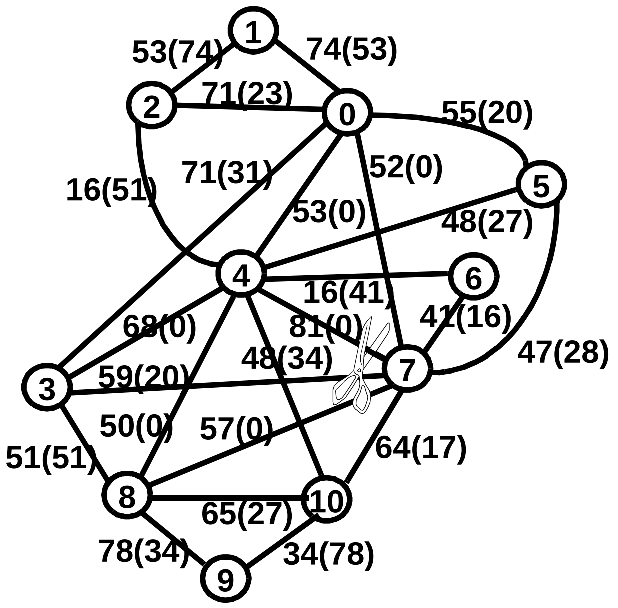

Figure 10: The New Jersey LATA Network.

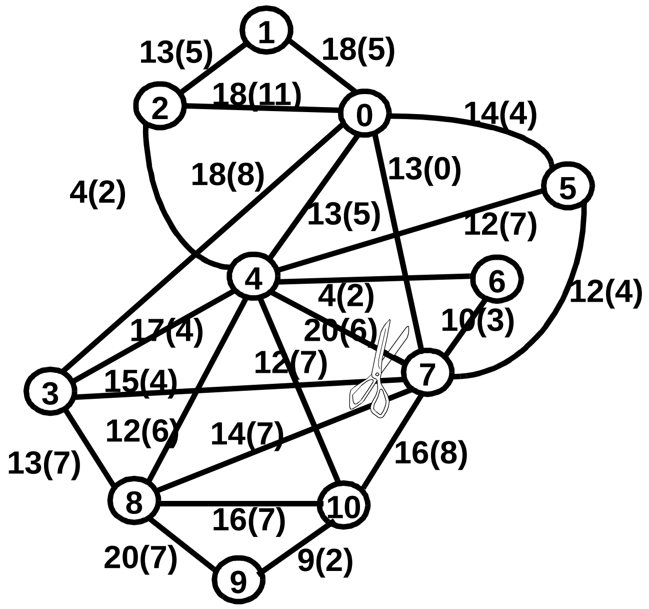

Figure 11: LATA ‘X’ Network.

test network which is defined in [8] and is shown in Figure 11, and the ‘US Net’ test network which is also defined in [8] and is shown in Figure 12.

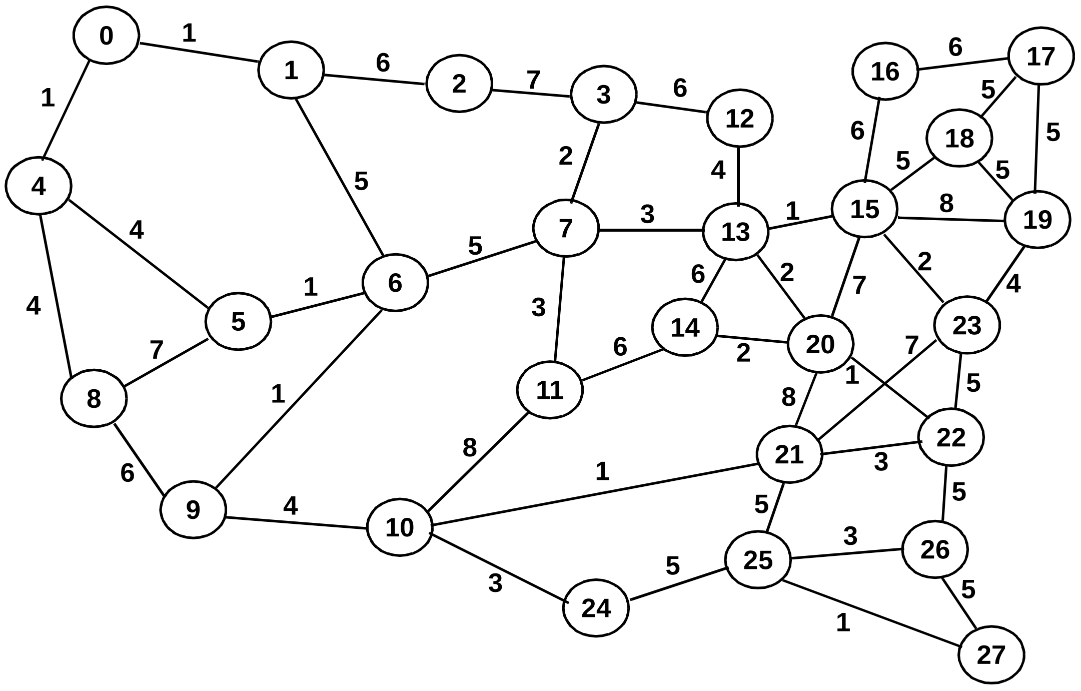

Figure 12: US Net.

For these four simulators, the simulation control parameters, that can be changed via the same command line interface, include message processing time, message length, transmission speed, DCS cross connect time, refractive index of fiber medium, DCS operating mode (parallel or sequential), and message repeat limit (hop count).

All algorithms were identically tested under the following assumptions:

a. All messages have equal priority.

b. All messages are serviced by a node in the order they are received.

c. It requires 10 msec to process any incoming messages.

d. Propagation speed of messages is 200,000 km/sec.

e. Transmission delays are computed for variable length messages, using a 8Kbps.

f. It takes 10 msec for DCS to make a connection in sequential operating mode.

g. These results do not include the time required for fault detection. Assume that the sending of confirmation messages and the processing of the cross connection commands can be done concurrently.

In order to evaluate the spare usage for the link restoration, we used the optimal spare usage results generated by the RELAXT-III program, which was implemented by Bertsekas and Tseng [31,32] using the relaxation method for the minimum cost network flow problem. The front end interface that maps the link restoration problem to the minimum cost network flow problem was done by a group of graduate students at UCCS.

In general, the Two Prong algorithm performed better than the other algorithms with respect to the time to restoration performance metric. This is primarily due to the aggressive nature of the algorithm in identifying, selecting and connecting restoration paths. The algorithm's time to restoration performance is also enhanced in that the hand shaking required to make final connection of the disrupted ends of the lost channels is done over connected paths. In smaller networks, the RREACT algorithm is able to compete relatively closely to the Two Prong algorithm in terms of time to restoration, but as network size increases (as shown by the US Net results), the Two Prong algorithm is able to clearly outperform RREACT in this regard. This is due to a high degree of congestion at the Chooser node in the RREACT algorithm, while the Origin nodes in the Two Prong algorithm have a much lower level of congestion.

In analyzing the performance of the Two Prong algorithm, we have found that its performance is related to the number of paths used in the final restoration solution. Further analysis has indicated that the best possible time to completion of any given $path_i$ is:

$T_i = \max (h_i(2t) + 4t + c + 3p_i, T_{i-1} + t)$

Where:

T_i is the Time required to complete $path_i$

h is the hopcount of $path_i$

t is the time required to process a single message

c is the time required to complete a path's cross connection at a single node

p_i is the propagation time required over $path_i$

Since the best possible time for completion of full restoration is dependent the number of paths in the final restoration solution, the best time performance of the Two Prong algorithm is that of the last (nth) path to be connected plus the time required for fault diagnosis (f). This then becomes:

$T_n = f + \max (h_n(2t) + 4t + c + 3p_n, T_{n-1} + t)$

The formulas above hold for all networks where $c <= 2t$. In networks where $c > 2t$, the following best possible time to restoration formula holds:

$T_n = f + \max (h_n(2t) + 2t + 2c + 3p_n, T_{n-1} + t)$

Worst case performance of the Two Prong algorithm is hard to determine and is quite dependent upon the topology of the network. We are still working towards determining a hard upper bound for the time to restore performance of the algorithm. Generally, we have found in the networks we have studied so far, the Two Prong algorithm performs within a factor of 2 of its best case time performance.

In the tested networks, the Two Prong algorithm has achieved a better level of restoration compared with earlier versions of the Two Prong algorithm which had difficulty with race conditions and resulted in instable performance and occasionally low restoration levels.

While the Two Prong algorithm has comparably economical utilization of spare channel resources, it is sometimes beaten by RREACT. This is due to the aggressive commitment of links to possible restoration paths which aren't included in the final restoration solution. Because these links are temporarily 'committed' to a restoration path, another candidate path, which could have included them, makes another choice which becomes part of the final restoration solution. Ultimately, the original links aren't used, get released, but it is too late to include them in the final solution since they have already been 'bypassed'. We are still improving the Two Prong algorithm's performance, by recognizing more efficient paths and by making necessary reconnections for better spare utilization.

At present, the Two Prong algorithm is designed only for single link failure recovery. Of the other algorithms, only the Self-Healing Network can automatically recover multiple link failures. The FITNESS and RREACT algorithms would require some modification to handle multiple link failures. Only the Komine algorithm can currently handle the node failure scenario. It is also capable of recovery from multiple link failures. None of these algorithms address area failures.

We are currently in the early development stage of a modified version of the Two Prong algorithm which will address multiple link and node failure situations. Preliminary analysis indicates that the Two Prong approach offers implicit benefits over the Komine approach in that it is able to automatically distinguish between a single link failure and the node failure scenario. This is quite beneficial in that it reduces the message volume involved in recovery from these failure scenarios and wasted effort as two restoration activities compete for message processing time and spare channel capacity.

The Two Prong algorithm does have a fairly high message volume. However, the increase in message volume seems linearly related to the number of links in a network and does not demonstrate exponential growth. Further, the message processing is fairly evenly distributed across the network with low congestion at critical nodes so that message volume seems to have minimal impact upon the time to restoration performance of the algorithm. The FITNESS algorithm has the lowest message volume of all the algorithms studied. RREACT has low message volume in small networks, but appears to have an exponential growth as the network size increases. This is due to the exhaustive trace of all paths in the network. Due to the restoration path selection process RREACT is based on, congestion is also high at the critical Sender node in these large networks. Published results of the Self-Healing Network do not report message volume. From analyzing its functional characteristics, however, it can be estimated to be quite high and probably demonstrates exponential growth relative to the number of links in the network.

Tables 1 through 3 show the performance of the four algorithms on the three test networks. The DCS' are assumed to operate at sequential mode, connecting one channel at a time. The tables compare the performance in time to restoration (in msec), restoration level, spare usage and number of messages. These are performance metrics 1, 2, 3 and 5.

In the column with SHN simulation results, we list the simulation results of our implementation of SHN and those reported in Grover's dissertation. If they are the same, only one number is presented. If they are different, two numbers are presented. The number on the left side of '/' is that of UCCS' implementation and the number on the right side of '/' is from Grover's dissertation. There is no message number reporting on Grover's dissertation. Our implementation includes improvements such as mechanism for faster signature cancellation, avoidance mechanism to prevent old signatures from chasing the bandwidth just being released, and faster tandem logic. We assume a fix message processing time which can be changed from command line interface. Grover's SHN simulation measured the execution time of its logic. Also Grover's simulation results did not include the DCS connect time. These differences are reflected in the results of restoration times. The results on the restoration level and spare usage were very close.

The last columns of Tables 1 and 3 show the optimal spare usages for link failure scenarios generated by the RELAXT-III program. The data indicate that the performance of the four distributed link restoration algorithms in terms of spare usage is very close to, or the same as the optimal spare usage.

Impact of parallel DCS operating modes on restoration time

In [33], it was shown that with DCS operating at sequential mode (one channel at a time) and with long DCS connect time such as 100 msec, the DCS cross connect time becomes the dominating time factor. There are several ways to improve this situation. One is to use algorithms that allow more concurrent DCS executions in the network. Another is to design new DCS that can accept multiple connection requests and perform DCS connections in parallel. A parallel DCS that has k number of DCS servers can make k connections simultaneously. One of the basic approaches is to improve the DCS cross connect time. As it demonstrated in Tables 1-3, with DCS connect time at 10 msec, the four algorithms can meet the two second real time constraints in almost all cases. SHN has problems meeting the real time constraints with the New Jersey test network due to large spare capacity and its channel by channel restoration strategy. RREACT has very few cases in US net that miss the deadline, because it explored all possible paths and resulted in network congestion.

Figure 13 shows the impact of parallel DCS operating modes on the restoration times of the four algorithm simulation on the N04-N07 span cut of New Jersey test network with message processing time set at 0.01 seconds and DCS cross connect time set at 0.01 seconds. Due to the time-out and multiple wave mechanism used by FITNESS, here the last wave happened to find one path and therefore the parallel DCS can not improve the restoration time. This is an extreme case but it can happen. SHN finds path one at a time and in this case, many path pairs happened to be found with

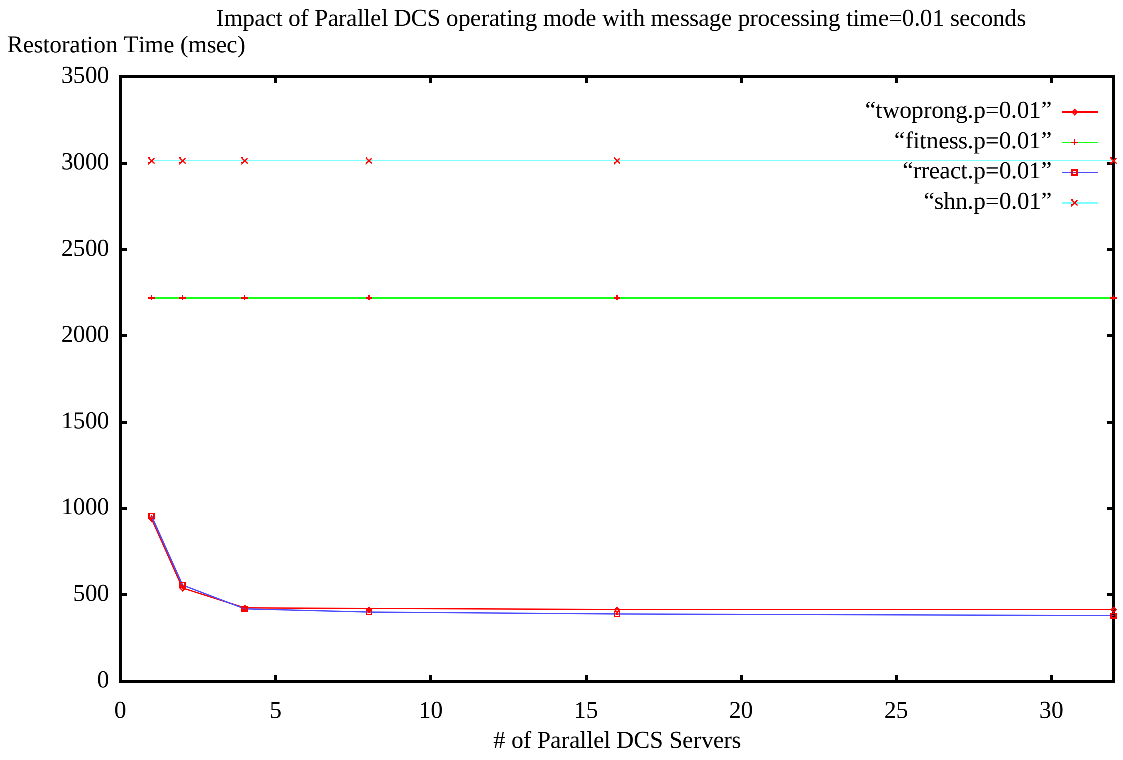

Figure 13. Impact of parallel DCS on restoration time with message processing time = 0.01 seconds

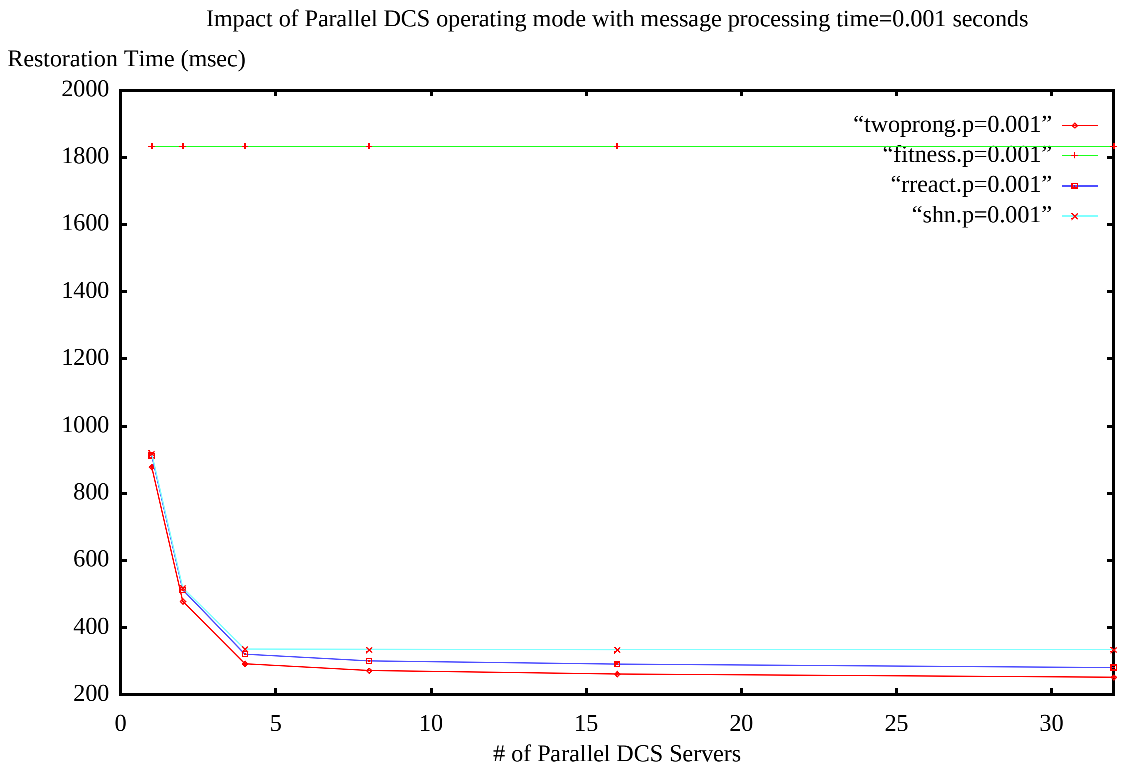

Figure 14. Impact of parallel DCS on restoration time with message processing time = 0.001 seconds.

more than 0.01 seconds apart and the parallel DCS can not help either. Two Prong and RREACT did show the benefit of using the parallel DCS. The restoration times improve dramatically from the one DCS server to two Parallel DCS servers. The improvements level off after the number of parallel DCS servers increases to 4.

Figure 14 shows the impact of parallel DCS operating modes on the restoration times of the four algorithm simulation on the N04-N07 span cut of New Jersey test network with message processing time set at 0.001 seconds. Here with shorter message processing delay, more paths were found in SHN to be within 0.01 seconds DCS cross connect time and therefore the parallel DCS can improve its performance.

VIII. Summary

In this paper we have provided a brief overview to three distributed link restoration algorithms and presented the Two Prong link restoration algorithm in detail. We have identified five important metrics which can be used to evaluate the performance of distributed network restoration algorithms. We have further identified seven functional characteristics and discussed them in the context of the four link restoration algorithms and have shown the effect they have on the performance of these algorithms. These metrics and characteristics, together, provide the developers of such algorithms, or anyone studying distributed link restoration algorithms, a systematic method to analyze an algorithm and to relate its design to its performance.

Table 1: Comparison of Two Prong, FITNESS, RREACT, and SHN algorithms on the 'New Jersey' Network

Scenario	Perf. Metric	Two Prong	FITNESS	RREACT	SHN	RELAXT-III
New Jersey N00 - N01 Failure 74 ch. lost	Time msec Level Spare Usage # of Msgs	652 74.32% 198 154/165	1676 100% 339 185/186	1155 100% 312 203/203	3126 100% 312 4269/4281	- 100% 312 -
New Jersey N00 - N02 Failure 71 ch. lost	Time msec Level Spare Usage # of Msgs	822 100% 160 87/140	1046 100% 160 62/63	836 100% 160 39/142	836 100% 177 2328/2374	- 100% 160 -
New Jersey N00 - N04 Failure 53 ch. lost	Time msec Level Spare Usage # of Msgs	662 100% 116 70/126	904 100% 157 56/57	667 100% 116 37/63	2375 100% 124 1994/1994	- 100% 116 -

Table 1: Comparison of Two Prong, FITNESS, RREACT, and SHN algorithms on the 'New Jersey' Network

Scenario	Perf. Metric	Two Prong	FITNESS	RREACT	SHN	RELAXT-III
New Jersey	Time msec	544	1030	851	2481	-
N01 - N02	Level	81.13%	94.34%	100%	100%	100%
Failure	Spare Usage	126	235	189	186	186
53 ch. lost	# of Msgs	90/139	108/270	149/149	3864/4141	-
New Jersey	Time msec	377	535	376	1043	-
N02 - N04	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	48	48	48	48	48
16 ch. lost	# of Msgs	84/137	26/27	29/107	1030/1643	-
New Jersey	Time msec	658	1398	804	2384	-
N03 - N07	Level	81.36%	100%	100%	100%	100%
Failure	Spare Usage	166	278	222	224	222
59 ch. lost	# of Msgs	131/138	111/112	96/96	3158/3174	-
New Jersey	Time msec	941	2221	957	3014	-
N04 - N07	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	210	216	204	204	204
81 ch. lost	# of Msgs	142/183	199/200	71/73	3051/3051	-
New Jersey	Time msec	691	1183	710	1744	-
N05 - N07	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	141	141	141	141	141
47 ch. lost	# of Msgs	126/181	95/96	64/93	2799/3159	-
New Jersey	Time msec	575	869	621	2164	-
N06 - N07	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	123	123	123	127	123
41 ch. lost	# of Msgs	120/170	83/84	35/68	3333/3372	-
New Jersey	Time msec	745	1305	868	2553	-
N07 - N10	Level	84.38%	84.38%	100%	100%	100%
Failure	Spare Usage	162	210	215	235	215
64 ch. lost	# of Msgs	112/135	130/158	102/103	3264/3482	-
New Jersey	Time msec	762	1125	786	1920	-
N08 - N10	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	189	254	189	204	189
65 ch. lost	# of Msgs	122/175	87/88	90/90	2425/2425	-

Table 2: Comparison of Two Prong, FITNESS, RREACT and Self-Healing Network (SHN) algorithms on the LATA ‘X’ Network

Scenario	Perf. Metric	Two Prong	FITNESS	RREACT	SHN	RELAX-III
LATA ‘X’	Time msec	182	386	177	234/167	-
N00 - N01	Level	27.78%	27.78%	27.78%	27.78%	27.78%
Failure	Spare Usage	10	10	10	10	10
18 ch. lost	# of Msgs	57/106	69/135	7/196	188/554	-
LATA ‘X’	Time msec	324	682	341	333/144	-
N00 - N02	Level	38.89%	38.89%	38.89%	38.89%	38.89%
Failure	Spare Usage	14	14	14	14	14
18 ch. lost	# of Msgs	97/146	63/64	92/624	319/363	-
LATA ‘X’	Time msec	369	1064	358	864/254	-
N00 - N04	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	29	40	29	29	29
13 ch. lost	# of Msgs	79/120	107/108	70/103	675/750	-
LATA ‘X’	Time msec	204	386	177	233/218	-
N01 - N02	Level	38.46%	38.46%	38.46%	38.46%	38.46%
Failure	Spare Usage	10	10	10	10	10
13 ch. lost	# of Msgs	73/247	85/4029	10/659	268/602	-
LATA ‘X’	Time msec	405	1391	481	863/294	-
N03 - N04	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	36	42	36	36/43	36
17 ch. lost	# of Msgs	119/170	117/118	129/214	580/580	-
LATA ‘X’	Time msec	459	1046	544	675/352	-
N03 - N07	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	34	40	34	34/38	34
15 ch. lost	# of Msgs	148/250	142/143	139/184	769/908	-
LATA ‘X’	Time msec	385	1345	387	663/259	-
N04 - N07	Level	100%	100%	100%	100%	100%
Failure	Spare Usage	40	40	40	40	40
20 ch. lost	# of Msgs	102/174	148/149	92/126	513/513	-
LATA ‘X’	Time msec	483	769	452	654/365	-
N05 - N07	Level	91.67%	91.67%	91.67%	91.67%	91.67%
Failure	Spare Usage	30	33	27	27/37	27
12 ch. lost	# of Msgs	135/195	97/125	133/253	856/961	-
LATA ‘X’	Time msec	265	404	187	175/189	-
N06 - N07	Level	20%	20%	20%	20%	20%
Failure	Spare Usage	4	6	4	4/6	4
10 ch. lost	# of Msgs	98/170	67/123	30/127	127/267	-

Table 2: Comparison of Two Prong, FITNESS, RREACT and Self-Healing Network (SHN) algorithms on the LATA 'X' Network

Scenario	Perf. Metric	Two Prong	FITNESS	RREACT	SHN	RELAX-III
LATA 'X' N07 - N10 Failure 16 ch. lost	Time msec Level Spare Usage # of Msgs	510 100% 37 154/255	1419 100% 37 165/166	943 100% 37 279/367	703/465 100% 37/41 677/677	- 100% 37 -
LATA 'X' N08 - N10 Failure 16 ch. lost	Time msec Level Spare Usage # of Msgs	389 100% 33 110/171	1363 100% 33 146/147	585 100% 33 200/466	593/288 100% 33 791/881	- 100% 33 -

Table 3: Comparison of Two Prong, FITNESS, RREACT and Self-Healing Network (SHN) algorithms on the 'US' Network

Scenario	Perf. Metric	Two Prong	FITNESS	RREACT	SHN	RELAXT-III
US Net N04 - N05 Failure 100 ch. lost	Time msec Level Spare Usage # of Msgs	292 5% 12 46/50	768 5% 12 64/74	328 5% 12 25/25	245/207 5% 12 629/1112	- 5% 12 -
US Net N07 - N11 Failure 100 ch. lost	Time msec Level Spare Usage # of Msgs	637 8% 38 514/688	1206 6% 23 363/467	1352 9% 45 1129/2346	416/878 6/10% 23/69 897/922	- 10% 55 -
US Net N08 - N09 Failure 100 ch. lost	Time msec Level Spare Usage # of Msgs	197 1% 3 166/292	528 1% 6 135/263	792 2% 11 379/399	449 2% 11 593/593	- 2% 11 -
US Net N09 - N10 Failure 100 ch. lost	Time msec Level Spare Usage # of Msgs	585 2% 10 303/311	442 1% 4 125/246	863 3% 18 532/613	699/421 3% 18 739/739	- 3% 18 -
US Net N10 - N21 Failure 100 ch. lost	Time msec Level Spare Usage # of Msgs	498 8% 33 209/215	1244 7% 27 267/334	1590 8% 33 828/854	765/363 8% 33 1002/1002	- 8% 33 -

Table 3: Comparison of Two Prong, FITNESS, RREACT and Self-Healing Network (SHN) algorithms on the 'US' Network

Scenario	Perf. Metric	Two Prong	FITNESS	RREACT	SHN	RELAXT-III
US Net	Time msec	695	988	2833	1306/192	-
N13-N14	Level	8%	5%	8%	8%/5%	8%
Failure	Spare Usage	42	23	38	42/13	37
100 ch. lost	# of Msgs	503/583	228/294	2902/3736	1843/1843	-
US Net	Time msec	289	765	317	512/209	-
N15-N18	Level	10%	10%	10%	10%/6%	10%
Failure	Spare Usage	25	25	25	25/13	25
100 ch. lost	# of Msgs	72/75	26/27	114/1982	606/606	-
US Net	Time msec	372	1152	670	493/216	-
N15 - N19	Level	14%	14%	14%	14%	14%
Failure	Spare Usage	37	41	37	37/45	37
100 ch. lost	# of Msgs	125/131	97/98	564/1596	881/882	-
US Net	Time msec	285	750	319	492/246	-
N19 - N17	Level	11%	11%	11%	11%/7%	11%
Failure	Spare Usage	28	28	28	28/20	28
100 ch. lost	# of Msgs	79/82	117/164	83/119	813/814	-
US Net	Time msec	284	702	307	437/160	-
N19 - N18	Level	10%	8%	10%	10%/6%	10%
Failure	Spare Usage	20	26	20	20/12	20
100 ch. lost	# of Msgs	86/90	115/158	80/102	762/768	-
US Net	Time msec	505	1598	1444	538/409	-
N20 - N21	Level	10%	8%	11%	11%	11%
Failure	Spare Usage	42	30	48	48	48
100 ch. lost	# of Msgs	299/312	434/532	1399/1871	1672/1849	-
US Net	Time msec	537	1226	2230	529/614	-
N22 - N20	Level	11%	10%	12%	10%/11%	12%
Failure	Spare Usage	42	33	48	29/42	40
100 ch. lost	# of Msgs	343/372	320/374	2326/4675	1558/1710	-
US Net	Time msec	416	533	402	362/153	-
N27 - N25	Level	5%	5%	5%	5%/3%	5%
Failure	Spare Usage	14	25	14	14/6	14
100 ch. lost	# of Msgs	169/173	92/93	116/136	971/1056	-

Acknowledgment

The authors would like to thank Al Beckman and Mark Bracco for teaching us how to use their front end interface to the RELAXT-III program. They also would like to thank Professors Bertsekas

Table 4. Two-Prong Algorithm State Chart

MsgType / State	**Link Break**	**Black**	**Gray**	**Backtrack Black**	**Backtrack Gray**	**Cancel**
White	Go to Gray-Origin State and Flood Gray Messages -or- Go to Black-Origin State and Flood Black Messages	Go to Black State Make table record and propagate the Black Message	Go to Gray State Make table record and propagate the Gray Message	Ignore	Ignore	Disconnect redundant connections and selectively propagate the Cancel Message
Black	N/A	Update table record and propagate the Black Message	Connect and forward according to the Black message table record—send Backtrack Gray Message for unconnected requested bandwidth	Disconnect and redirect, if redirect is not possible, send Backtrack Black Message to the next Black node	Disconnect and redirect as if receiving a Gray message, if redirect is not possible send Backtrack Gray Message	Disconnect redundant connections, propagate the Cancel Message and go to White State
Gray	N/A	Connect and forward according to the Gray message table record—send Backtrack Black Message for unconnected requested bandwidth	Update table record and propagate the Gray Message	Disconnect and redirect as if receiving a Black message, if redirect is not possible send Backtrack Black Message	Disconnect and redirect, if redirect is not possible, send Backtrack Gray Message to the next Gray node	Disconnect redundant connections, propagate the Cancel Message and go to White State
Black-Origin	N/A	Ignore	Connect and initiate mapping information signal —if all lost channels have been restored, flood Cancel Messages and go to White State	Disconnect redundant connections	Ignore	Ignore
Gray-Origin	N/A	Initiate 'listening' for mapping information signal (in-band)—Make final connection on receipt	Ignore	Ignore	Ignore	Disconnect redundant connections and go to White State

and Tseng for sending us the RELAXT-III user's guide and the well designed RELAXT-III program.

References

1. L. A. Wrobel, *Disaster Recovery Planning for Telecommunications*, Artech House, 1990.

2. T.-H. Wu, *Fiber Network Service Survivability*, Artech House, 1992.

3. D. K. Doherty, W. D. Hutcheson, and K. K. Raychaudhuri, 'High capacity digital network management and control', *Proceedings of GlobalCom '90*, (San Diego), 301.3.1–301.3.5 (1990).

4. C. Palmer and F. Hummel, 'Restoration in a partitioned multi-bandwidth cross-connect network', *Proceedings of GlobalCom '90*, (San Diego), 301.7.1–301.7.5 (1990).

5. R. D. Doverspike and C. D. Pack, 'Using SONET for Dynamic Bandwidth Management in Local Exchange Network', *Proceedings of 5th International Network Planning Symposium*, Kobe, Japan, (1992).

6. B. Coan, W. E. Leland, M. P. Vecchi, A. Weinrib, and L. T. Wu, 'Using Distributed Topology Update and Preplanned Configurations to Achieve Trunk Network Survivability', *IEEE Trans. on Reliability*, **40**, (4), (October 1991).
7. J. E. Baker, 'Distributed Link Restoration With Robust Preplanning', *Proceedings of GlobalCom '91*, 306-311 (1991).
8. W.D. Grover, 'SELFHEALING NETWORKS: A Distributed Algorithm for k-shortest link-disjoint paths in a multi-graph with applications in real time network restoration', in *Doctoral Dissertation for the Department of Electrical Engineering, University of Alberta*, Fall 1989.
9. Yang, C. H. and S. Hasegawa, 'FITNESS: Failure Immunization Technology for Network Service Survivability', *Proceedings of GlobalCom '88*, 47.3.1-47.3.6 (1988).
10. H. Komine, T. Chujo, T. Ogura, K. Miyazaki, and T. Soejima, 'A distributed restoration algorithm for multiple-link and node failures of transport networks', *Proceedings of GlobalCom '90*, (San Diego), 403.4.1–403.4.5 (1990).
11. N. Deo and C.-Y. Pang, 'Shortest-path algorithms: Taxonomy and annotation', *Networks*, **14**, 275–323 (1984).
12. E. W. Dijkstra, 'A note on two problems in connexion with graphs', *Numer. Math.*, **1**, 269-271 (1959).
13. T. A. Nicholson, 'Finding the shortest route between two points in a network', *Computer J.*, **9**, 276-280 (1966).
14. B. A. Carré, 'An Algebra for network routing problems', *Journal Inst. Math. Appl.* **7**, 273-294 (1971).
15. J. W. Suurballe, 'Disjoint paths in a network', *Networks*, **4**, 125-145 (1974).
16. J. W. Suurballe and R. E. Tarjan, 'A quick method for finding shortest pairs of disjoint paths', *Networks*, **14**, 325-336 (1984).
17. D. M. Topkis, 'A k shortest path algorithm for adaptive routing in communications networks', *IEEE Trans. on Communications*, **36**, 855–859 (July 1988).
18. T. Mohr and C. Pasche, 'A parallel shortest path algorithm', *Computing*, **40**, 281-292 (1988).
19. J. Bovet, 'Une amélioration de la methode de Dijkstra pour la recherche d'un plus court dans un réseau', *Discrete Applied Math.*, **13**, 93-96 (1986).
20. J.M. McQuillan, G. Falk and I. Richer, 'A review of the development and performance of the ARPANET routing algorithm', *IEEE Trans. on Communications*, **COM-26**, (12), 1802-1811 (December 1978).
21. J.M. McQuillan, I. Richer and E.C. Rosen, 'The new routing algorithm for the ARPANET', *IEEE Trans. on Communications*, **COM-28**, (5), 711-719 (May 1980).
22. M. Schwartz and T. E. Stern, 'Routing techniques used in computer communications networks', *IEEE Trans. on Communications*, **COM-28**, (4), 539-552 (April 1980).
23. A. E. Baratz, J. P. Gray, P. E. Green, Jr., J. M. Jaffe, and D. P. Pozefsky, 'SNA networks of small systems', *IEEE Journal on Selected Areas in Communications*, **SAC-3**, (3), 416-426 (May 1985).

24. B. Awerbuch, A. Bar-Noy, and M. Gopal, ‘Approximate distributed bellman-ford algorithms’, *Proceedings of IEEE INFOCOM*, 1206–1213 (1991).

25. W. D. Grover, ‘The self-healing network: A fast distributed restoration technique for networks using digital cross-connect machines’, *Proceedings of GLOBECOM ‘87*, 28.2.1–28.2.6 (1987).

26. C.-H. E. Chow, S. McCaughey, and S. Syed, ‘RREACT: A Distributed Protocol for Rapid Restoration of Active Communication Trunks’, *Proceedings of 2nd Network Management and Control Workshop*, 391-406 (1993).

27. C.-H. E. Chow, J. Bicknell, S. McCaughey, and S. Syed, ‘A Fast Distributed Network Restoration Algorithm’, *Proceeding of 12th International Phoenix Conference on Computers and Communications*, 261-267 (March 1993).

28. H. Sakauchi, Y. Nishimura, and S. Hasegawa, ‘A self-healing network with an economical spare-channel assignment’, *Proceedings of GLOBECOM ‘90*, (San Diego), 403.1.1–403.1.6 (1990).

29. W.D. Grover, ‘Near Optimal Spare Capacity Planning In A Mesh Restorable Network’, *Proceedings of GLOBECOM ‘91*, 57.1.1-57.1.6 (1991).

30. J. S. Whalen and J. Kenney, ‘Finding maximal link disjoint paths in a multigraph’, *Proceedings of GlobalCom ‘90*, (San Diego), 403.6.1–403.6.5 (1990).

31. Bertsekas, D. P., and P. Tseng, ‘Relaxation Methods for Minimum Cost Ordinary and Centralized Network Flow Problems’, *Operations Research Journal*, **36**, 93-114 (1988).

32. Bertsekas, D. P., and P. Tseng, ‘RELAXT-III: A New and Improved Version of the RELAX Code’, Massachusetts Institute of Technology, 1992.

33. H. Kobrinski and M. Azuma, ‘Distributed Control Algorithms for Dynamic Restoration in DCS Mesh Networks: Performance Evaluation’, *Proceedings of GlobalCom ‘93*, 1584-1588 (1993).