UCCS EAS Tech-Report 92-18. Results published in the Proceedings of Second IEEE Network Management and Control Workshop, Westchester, New York, September 21-23, 1993.

RREACT: A Distributed Protocol for Rapid Restoration of Active Communication Trunks

C. Edward Chow, 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 Corporation
4678 Alpine Meadows Lane
Colorado Springs, CO 80919
TEL: (719) 592-1461
FAX: (719) 592-1851
Email: 0002912712@mcimail.com

Abstract

Commercial telecommunications networks have tight real-time requirements for restoration after a failure. The problem of finding the available restoration paths and reassigning the interrupted traffic within such tight real-time requirements places difficult demands on the restoration protocol employed. This paper reviews several distributed network restoration protocols and presents a new distributed protocol called RREACT, for performing this function with a distributed algorithm which uses no prior network status or topography knowledge and which supports multiple simultaneous link restorations. Simulation results show that this protocol significantly outperforms other existing algorithms based on the Sender-Chooser approach, usually completing the total restoration in well under one second. In addition, enhancements to the basic algorithm are described which help to ensure near-optimal use of network spare channel resources and address such restoration situations where a complete restoration is possible but not achieved due to the poor choice of initial restoration path(s).

1. Introduction

To meet the increasing demands of telecommunications customers, both commercial and governmental, improvements must be made in the restoration capabilities of existing digital telecommunication networks. With the widespread deployment of fiber optic transmission systems and the alarming rate of outages due to fiber cuts [WROB90], there is great interest in strategies for improving the process of restoring disrupted traffic from minutes to sub-seconds following a fiber cut [WU92]. 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 [DHR90] [PH90] [DP92] [COAN91][BAKE91][GROV89][YH88] [KCOMS90]. The centralized DCS-based network restoration approach [DHR90] [PH90][DP92] 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 [COAN91]. In this paper, we will focus on distributed network restoration algorithms for 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 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 [DHR90][DP92] 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 [DP84] classified the shortest path algorithms into four classes:

1) one (node) to all (nodes) [DIJK59],

2) one to one [NICH66],

3) all to all [CARR71], and

4) k-shortest path algorithms [SUUR74][ST84][TOPK88].

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 [DIJK59], 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 [MP88] indicated that for finding the one-to-one shortest path, Nicholson's shortest path algorithm [NICH66] is about 60% faster than Dijkstra's and Bovet's algorithm [BOVE86] 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.

Routing algorithms proposed for computer networks [MFR78][MRR80][SS80] [BGGJP85][ABG91] find the shortest paths from each node to all other nodes in a network. The approximate distributed Bellman-Ford Algorithms in [ABG91] 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.

The network restoration problem can be formulated as a maximum flow problem [CLR90] [BAKE91] where the goal is to find all the disjoint paths with the maximum flow between the two disrupted nodes. The efficient distributed maximum flow algorithm proposed by Goldberg and Tarjan [GT88] requires $O(n^2)$ waves of messages and $O(n^3)$ messages, where n is the number of nodes in the network. This algorithm, while finding the potential paths for restoration, can only be applied in small networks and is used in conjunction with a path selection algorithm to generate results as a benchmark to compare the efficiency of other network restoration algorithms.

In this paper we presents a distributed protocol called RREACT which can perform a complete restoration of fiber optic cable failures in under one second using conventional inter-nodal messages between the switching nodes such as the Digital Cross-connect Systems (DCSs). RREACT also supports restorations with multiple simultaneous link failures and provides efficient utilization of network spare channel resources. The protocol performs this function using a distributed algorithm which uses no prior network status or topography knowledge other than what is present locally at the switching node. In addition, enhancements to the basic algorithm are described which help to ensure near-optimal use of network spare channel resources and address such restoration difficulties as "Whalen's Dilemma" in [WK90], while still achieving the high levels of restoration performance.

RREACT outperforms the FITNESS network restoration algorithm [YH88] both in terms of restoration time and spare channel usage. It performs better than the Two-Prong [CHOW93] in some cases on a small test network and proves to be very reliable. When the network size increases, its performance degrades due to the large increases in message volume. Several heuristics are proposed and implemented to cut down the message volume. Note that the above distributed network restoration algorithms can also be applied to ATM-based networks for restoring virtual paths after a network failure.

The rest of this paper is organized as follows: Section 2 defines the basic terms of the network model. Section 3 reviews three existing distributed network restoration algorithms [GROV89][YH88][CHOW93]. Section 4 briefly describes a network restoration simulation system called NETRESTORE which facilitates the design and comparison of network restoration algorithms. Section 5 presents the RREACT network restoration algorithm. Section 6 compares RREACT's performance with those of other three algorithms. Section 7 discusses the enhancements of RREACT. Section 8 is the summary.

2. Definitions and Network Models

To facilitate the presentation and the analysis of network restoration algorithms, let us define some of the basic terms used in this paper.

A **network** is an augmented undirected graph G(N, L) where N is the set of nodes and L is the set of links in the network. A **link** is associated with a certain bandwidth and this bandwidth is divided into **channels**. Each channel in the links of a network contains the same size bandwidth unit. A channel can only be in one of two states, working or spare. A channel which is in the working state is called a **working channel**. A channel which is in the spare state is called a **spare channel**. A link is labelled with a triple where the first label represent the number of working channels in the link, the second label represents the number of spare channels in the link, and the third label represents the

transmission delay on the link. A link can be uniquely identified by the two end nodes of the link. The channels in a link are given a unique ID and can be uniquely identified in a network by concatenating the link ID with the channel ID.

A **route** is specified by the 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. We only consider paths which do not contain cycles. The **hop-count of a path** is the number of channels in the path. A **working path** is a path where all its concatenated channels are all working channels. A **spare path** is a path where all its concatenated channels are spare channels. A **restoration path** is a spare path being designated for restoring a disrupted working channel due to a network failure.

3. Reviews of existing distributed network restoration algorithms

Grover's Self-Healing Network Algorithm

The first distributed network restoration algorithm for a DCS-based fiber network was proposed by Grover in [GROV87] and detailed in his Ph.D. dissertation [GROV89]. 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 the computation resource in each DCS node for processing.

Figure 3.1 shows the execution steps of Grover's algorithm 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 restoration path directly to the Chooser without the need for message processing in the

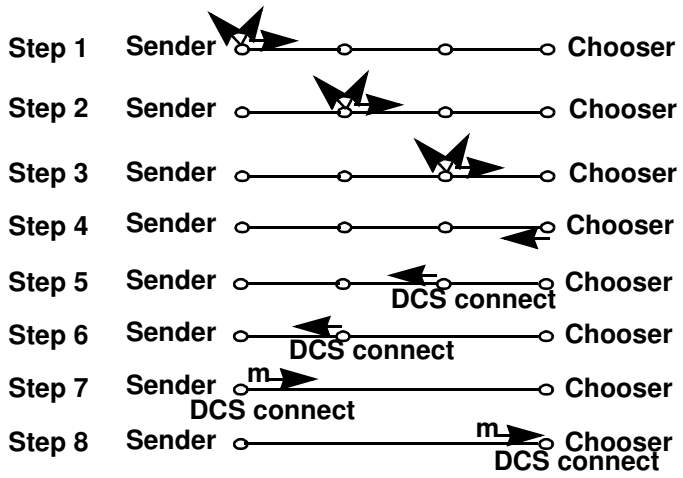

Figure 3.1. The execution steps of Grover's SHN algorithm.

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.

Bellcore's FITNESS Algorithm

Following Grover's publication, another distributed network restoration algorithm for DCS-based fiber networks was proposed by Yang and Hasegawa in [YH88]. This method is called FITNESS and also uses a Sender—Chooser relationship for the nodes adjacent to the fiber link cut. FITNESS reduces the potential 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 with 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.

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-connection 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 3.2 shows the execution steps of the FITNESS algorithm on a restoration route. Note that

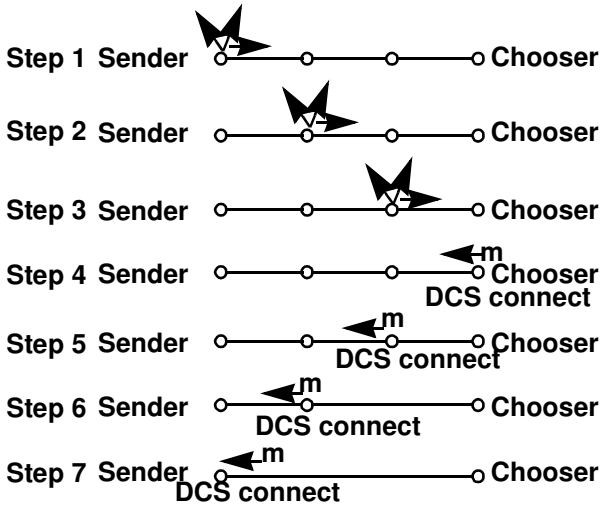

Figure 3.2. The execution steps of FITNESS algorithm.

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 that 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.

UCCS' Two-Prong: A fast distributed algorithm based on a Two-Prong approach

In [CHOW93] we propose a new fast distributed algorithm based on a Two-Prong approach. Unlike the SHN and FITNESS algorithms, the Two-Prong algorithm does not use a Sender—Chooser relationship for the nodes adjacent to the fiber link cut 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 labelled restoration request messages and the Gray-Origin node broadcasting Gray labelled 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 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 will be 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.

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 differently labelled request message, will make appropriate cross connections between the links over which the two different 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. It is possible that cross connections could involve a multiple number of links. 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. The selection of specific channels to connect can be based upon an implementation specific convention. An example based on Figure 3.3 will make these operations clear.

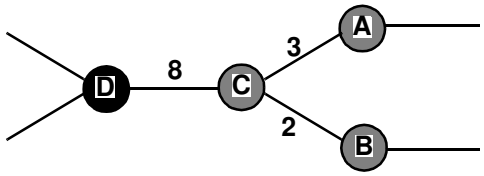

Figure 3.3. Backtracking in the Two-Prong network restoration algorithm.

Assume node C first receives a Gray message requesting 3 bandwidth from node A. C then will record this request in its Gray message table and become a Gray labelled node. This request will also be propagated to node B and D. Later, node C receives another Gray message requesting 2 bandwidth from node B. Again C will record this request and propagate the message to node D. After this, assume node C receives a Black message requesting 8 bandwidth 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 bandwidth. 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 bandwidth. Since node C only could support 5 of the original Black request for 8 channels, node C will send a Backtrack Black message to node D reporting that 3 bandwidth could not be connected.

Backtracking only will occur 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. The specific identification of which channels are to be disconnected can be based upon an implementation specific convention.

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 connect disrupted channels to the restoration channels identified by the Gray message. Then the Black-Origin will insert a mapping information signal into the channels of the new restoration path. This mapping information will inform the Gray-Origin of which disrupted channels have been connected to which restoration channels.

On receiving a Black message, the Gray-Origin will begin 'listening' to the spare channels in that link for the mapping information signal from the Black-Origin. Once this information is received, the Gray-Origin will make final connections to restore its disrupted channels.

Once the Black-Origin has restored all lost channels, it will begin flooding Cancel messages to all neighbors. Cancel messages contain the source and destination of the message and a bandwidth value. The value of this bandwidth is initially set to the number of remaining spare 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 cancellation, they will compare the value contained in the bandwidth field to the number of spare channels on the link over which the message was sent. If there are fewer spare channels than the number specified in the Cancel message, disconnections will be made to raise the number of spare 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 cancellation, it will make any necessary disconnections. A White node will only selectively flood a Cancel message to a neighboring node with which it made a disconnection. The bandwidth specified in this message will be equal to the number of spare channels existing on the link connecting the White node to its neighbor after the disconnection has been made.

Figure 3.4 shows the execution steps of the Two-Prong algorithm on a restoration route. Note that

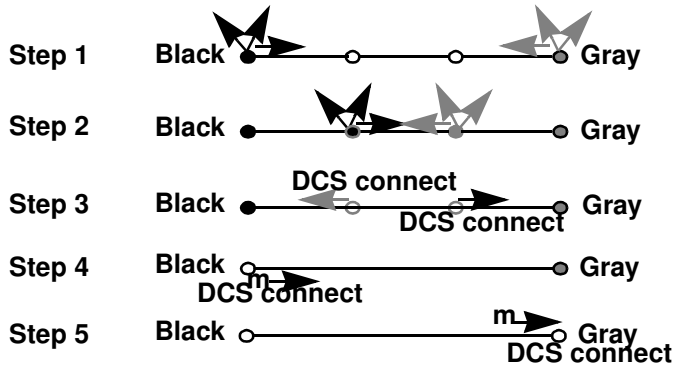

Figure 3.4. The execution steps of Two-Prong algorithm.

by letting both ends broadcast the request message the intermediate nodes start the DCS connection as early as at Step 3, compared with FITNESS that performs DCS connection after Step 4. Also notice that the connections are made concurrently on both ends.

The analysis and simulation results have revealed many interesting properties of the Two-Prong algorithm. The most significant property is the speed with which the algorithm discovers restoration paths and connects them. This speed is brought about by three factors. The first is the aggressive nature of the algorithm as it explores restoration paths from two directions. The second is the true concurrency which occurs as a result of the distributed, parallel nature of the algorithm. The last factor affecting the speed is the early initiation of parallel channel connection procedures once a path has been established. Simulation results also have revealed that Two-Prong is very resource efficient in that the number of available spare channels in a network which are used to achieve restoration is relatively low. This is based upon comparisons with SHN and FITNESS in which Two-Prong always used an equal or smaller number of channels to achieve restoration. It should also be noted that Two-Prong uses no time-outs and requires no special hardware additions or adaptations to existing DCS equipment.

The simulation results have also revealed that implementation of the Two-Prong algorithm is nontrivial. In some simulation results Two-Prong failed to find all possible restoration paths due to the timing of the request message flows in the network studied. We are currently studying ways to improve the algorithm's backtracking logic for better performance. Two-Prong also requires a cancellation phase to 'undo' connections which are made in the course of the restoration process, but aren't part of the paths used in the solution. However, this cancellation phase does not affect the speed of restoration since it is initiated after restoration is completed.

4. NETRESTORE Network Restoration Simulation System

An ongoing research effort at UCCS has developed a network restoration simulation tool to examine existing and proposed telecommunications restoration protocols. This tool, NETRESTORE, is based on a conventional discrete event simulation approach, runs on both PC and Unix workstation. The version on the UNIX workstation features an object-oriented X-windows user interface for specifying the network topology, control the simulation, and displaying the simulation results. NETRESTORE allows any network to be easily defined and then evaluated using user-definable restoration protocols.

Features of the prototype NETRESTORE simulation program include:

Discrete Event Simulation

Prototype Models Network Processes To Node Level Entities

Passive Links Modeled As Aggregate Spare And Working Channels

Protocol Implementation Changes Isolated In Defined Set Of Procedures And Globals

Networks Are Defined By Simple Text File

Tool Developed To Construct Network Definition Files

Operating features include:

Break Can Be Specified In Any Link

Command Line Interface Implemented To Specify Parameters For Simulation Run

Pre- & Post- Run Network States Included In NETRESTORE Simulation Log

NETRESTORE has proved invaluable for examining the strengths and weaknesses of various restoration protocols, developing and debugging new protocols, and playing “what if” scenarios to determine protocol robustness, efficiency, and performance.

After analyzing various proposed protocols which address network restoration, we determined that significant improvements could be made in many areas. One new protocol we developed uses an unconventional Two-Prong restoration approach [CHOW93], while the other, RREACT, follows a more conventional methodology but offers substantial performance and network resource utilization improvements over similar protocols.

5. RREACT distributed network restoration protocol

The RREACT protocol uses a distributed approach to network restoration. Upon detecting link failure, one node enters the SENDER state while the other node enters the CHOOSER state. RREACT utilizes the conventional inter-nodal message passing capabilities to identify all paths between the DCS nodes involved in the restoration effort with a single wave of SEEK messages. As paths are identified by the SEEK messages arriving at the CHOOSER node, a CONNECTION

message retraces the original messages' path to set up the alternative connectivity to effect the restoration using whatever spare channel capacity is presently available in the network.

Main RREACT features:

Distributed Algorithm

Sender-Chooser Methodology

Bandwidth Requested In Aggregate

Two Phase Approach

Variable Message Size - Path And BW Data

Chooser Node Builds "Smart" Picture Of Current Local Topology Via Table

Advantages:

Conventional Message Passing Approach

Fast Restoration

No Local Or Global Databases Required

Finds Shortest Paths - Good Use Of Links

Establishes All Paths In One Wave

Explicit Messages Establish Restoration

Fully Supports Multiple Link Failures

Dynamically Provides For New Link Failures

Chooser node builds current picture of local spare channel availability for use in optimization options or enhancements.

Disadvantages:

Message Volume Sensitive To Number Of Spans Per Node In Highly Connected Meshes

Key Results:

Very Reliable And Fast

Heuristics To Control Message Volume During Flooding Implemented

5.1 The Basic RREACT Algorithm

RREACT operates as follows:

Upon detecting a link failure with another switching node, the processor of a digital cross-connect switching node will determine whether it should change its state to either the SENDER or

CHOOSER mode. This determination may be based on any convenient arbitration convention (e.g., node ID, link ID, etc.) such that each node directly involved with the restoration has a unique state.

Both nodes will determine the number of channels needing restoration and save this information for future reference.

The SENDER node will flood out SEEK messages over all its links which are capable of supporting restoration channels. Each SEEK message contains fields of source, destination, message type, restoration ID, restoration start time, size, and two variable length vectors representing each node (ID) the message has passed through and the corresponding spare channel capacity for each node pair along the path it has followed.

The CHOOSER node will create a N by N network capacity table (where N is the number of nodes in the network) and initialize all entries to the value of the MAXIMUM spare channel capacity possible (e.g., MAXINT).

All intermediate (TANDEM) nodes between the SENDER and the CHOOSER nodes which receive the SEEK messages will flood out a modified copy of the original message over all their links (except the incoming link) which are capable of supporting restoration channels AND are not in identified in the incoming message's path description vector (i.e., no cycles in path). Message modifications will encompass increasing the message size field by one, adding the node's ID to the path description vector, and adding the spare channel capacity of the link over which it is to be sent to the spare channel capacity vector. As a simple heuristic to limit total message volume, if more than half the allowable time for the network restoration to occur has elapsed, the incoming message is killed and not flooded out.

* * *

When the CHOOSER node receives any SEEK message, the following actions are performed:

Appropriate entries in the network capacity table will be updated with the spare channel capacity defined by the path vector and spare channel capacity vectors if each "new" value is less than the present value in the table. Each node pair along the path will define the row/column entry to be updated. The lessor of the two node IDs is always used to index the row.

The updated network capacity table will then be used to update all fields in the spare channel capacity vector of the message currently being processed. The spare channel capacity of a link in the message can not be larger than that of the same link in the network capacity table, since the information in the network capacity table represents the most recent spare channel allocation status.

The maximum channel capacity of the path described in the message is then determined by finding the minimum value among the elements in the updated spare channel capacity vector.

At this point, only non-zero capacity paths are processed further.

If the number of channels needing restoration equals zero, the new path and capacity vectors are saved in a list for possible future use. The list will be reexamined when a REJECT message is received.

The actual bandwidth of the new restoration path is determined by taking the minimum of the outgoing link's spare channel value, the number of channels remaining to be restored, and the

calculated maximum capacity of the new path. If this restoration bandwidth value is greater than zero, then a CONNECT message is created. The CONNECT message contains fields of source, destination, message type, the calculated maximum path capacity value, and a fixed length vector representing each node (ID) which the original message has passed through. Spare channels on the outgoing link of the CHOOSER for this restoration path are allocated based on the path capacity value and the corresponding DCS switch connect command is issued.

The total number of channels to be restored is decreased by the path capacity value of the new CONNECT message. The network capacity table is then updated to reflect the use of these spares.

The CONNECT message is sent to the neighboring node of the CHOOSER in the path description vector.

* * *

When a TANDEM node receives a CONNECT message, the following actions are performed:

Spare channels on the incoming and outgoing links are allocated based on the path capacity value and the corresponding DCS switch connect command is issued.

If for any reason the channel capacity on the designated outgoing link to the previous node is less than the path capacity value included in the CONNECT message, special exception processing occurs. If only part of the path capacity value can be supported, the CONNECT message is changed to reflect this new limit and the CONNECT message is forwarded as originally intended. The number of channels which could not be supported by the changed link status are entered into a REJECT message which is sent back to the CHOOSER node along the original path using the CONNECT message's path vector. Examples of events which could cause changes in the channel capacity of a link could include additional link/channel failures or simultaneous link/channel restoration efforts originated elsewhere in the network. This feature allows for handling multiple link failures.

This CONNECT message is sent to the neighboring node (toward the SENDER side) of the current TANDEM node in the path description vector.

* * *

When a TANDEM node receives a REJECT message, the following actions are performed:

The REJECT message is sent to the neighboring node (toward the CHOOSER side) of the current TANDEM node in the original path description vector. Spare channels on the incoming and outgoing links are deallocated based on the rejected path capacity value and the corresponding DCS switch disconnect command is issued.

* * *

When the CHOOSER node receives a REJECT message, the following actions are performed:

Appropriate entries in the network capacity table will be updated with the rejected channel capacity. Each node pair along the path after the REJECT message originator will determine the row/column entry to be updated. The lessor of the two node IDs is always used to index the row.

The total number of channels to be restored is increased by the rejected path capacity value of the REJECT message. If available, path and capacity vectors are retrieved from the saved list for originating new CONNECT message(s). As many of these saved paths are used to replace the rejected

bandwidth value as needed; the original methodology used by the CHOOSER is employed to accomplish this.

* * *

When any SENDER node receives a CONNECT message, the following actions are performed:

Spare channels on the incoming link are allocated based on the path capacity value and the corresponding DCS switch connect command is issued. The total number of channels to be restored is updated.

* * *

Network restoration is complete when the number of channels to be restored is reduced to zero and all associated CONNECT messages have successfully reached the SENDER node.

5.2 Comparison of SHN, FITNESS, Two-Prong, and RREACT

The following simple analysis gives us insight on the comparative performance of the above four distributed network restoration algorithms. Let the hop count of the restoration route be *n* and each link in the route be the same length, the message processing time in each node be *p*, and the message transportation time on each link be *t*, which includes the transmission and propagation time. The time for restoring traffic using Grover's SHN algorithm is $2(n+1)kp + 3nt$ where *k* is the additional queueing delay overhead caused by the competition for the DCS operation system with other signatures at each node. Let *s* be the number of spare channels on the link connected to the Sender. The Sender will send out *s* signatures in sequence and these *s* restoration requests will be processed concurrently along this route. For high speed fiber optical transmission systems, *k*, which is related to the number of spare channels, *s* can be pretty big. There will be $2(n+1)$ instances of message processing for each spare path along this route. The *3nt* term is the message transportation time and reflects the three phase nature of the SHN protocol. The time for restoring traffic on this route using FITNESS is the time-out period set by the algorithm, which is not less than $(2n+1)p + 2nt$. The time for restoring traffic on this route using Two-Prong algorithm is $(n+1)p + 2nt$. The time for restoring traffic on this route using RREACT algorithm is $(2n+1)p + 2nt + 2t'$, where t' is the message transportation time due to the overhead of transmitting path information in the request message. Assume two bytes are used to encode a node in the path and *T* is the transmission speed of the network management channel, t' is equal to $\frac{2 \times (1 + 2 + 3 + \ldots + (n-1))}{T} = \frac{n \times (n-1)}{T}$. Note that we only have to specify the immediate nodes in the path information. The first intermediate node will send out a request message with its node ID in the path information field. The second intermediate node in the path will send out a request message with two node IDs. If we use 3 bytes of the DC fields in the SONET frame, *T* is 24Kbytes per seconds. For n=6, t' is only 1.25 mseconds. If the time-out of FITNESS is set as $(2n+1)p + 2nt$, then FITNESS outperforms SHN. For *p*=10 mseconds, the rank of the restoration performance for the four algorithms from the fastest to the slowest is Two-Prong, FITNESS, RREACT, and SHN. The Two-Prong has the lowest restoration time among the four algorithms.

Note that after the first time-out, if the selected restoration route has less bandwidth than that of the disrupted working channels, then FITNESS will issue another wave of restoration request messages. Let w be the number of waves of restore request messages. The overall time for restoring traffic using FITNESS is $w \cdot time\text{-}out$. For RREACT, Two-Prong and SHN, the overall time for restoring traffic is the restoration time along the longest restoration path. In this situation the SHN can outperform FITNESS.

6. Simulation Results

In the following sections we present the simulation results of FITNESS, Two-Prong, and RREACT on two test networks: one is the New Jersey LATA test network defined in the FITNESS [YH88] paper and shown in Figure 6.1. The New Jersey LATA test network was designed to be fully restorable for any single link span failure. The other is the US Continental test network defined in the GROVER [GROV89] paper and shown in Figure 6.2. The US Continental test network was designed to approximate typical national network connectivity with fewer span degree in each node.

6.1 Simulation results on the New Jersey LATA test network

6.1.1 RREACT Simulation Results

For the following simulation runs of the RREACT protocol, the following assumptions were used:

a. All messages have equal priority.

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

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

d. It requires an additional 10 msec to generate each outgoing message.

e. Propagation speed of messages is 200000 km/sec.

f. Transmission delays are computed for variable length messages, using a 10000 baud transfer rate (i.e., 1 msec per byte).

The following table provides some examples of the restoration result output of the RREACT protocol.

TABLE 1 - Sample of NJ Network RREACT Restoration Solutions

```
Path# Break = N02 - N03 (53 working channels lost)
1   (23 bw) 44% restored at   120 msec -> N03 N01 N02
2   (20 bw) 82% restored at   267 msec -> N03 N05 N06 N01 N02
3   (7 bw)  95% restored at   545 msec -> N03 N05 N06 N08 N04 N01 N02
4   (3 bw) 100% restored at   741 msec -> N03 N05 N11 N09 N04 N01 N02
```

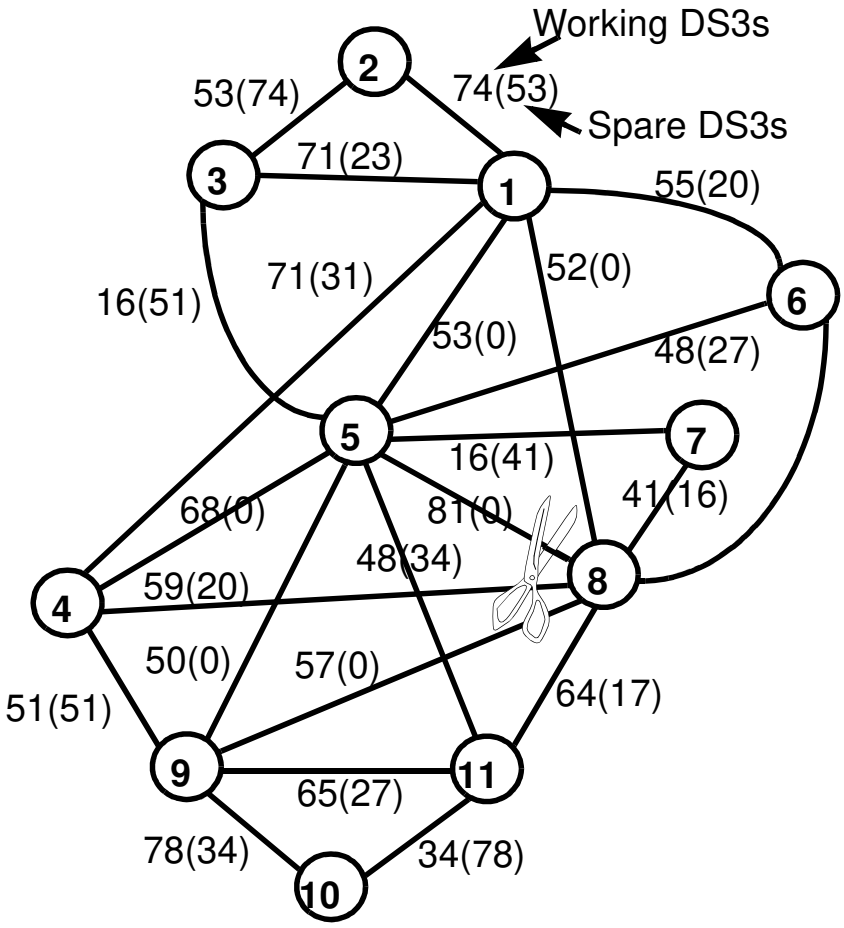

Figure 6.1. The New Jersey LATA Network.

```
Path# Break = N03 - N01 (71 working channels lost)
1   (53 bw) 75% restored at   119 msec -> N01 N02 N03
2   (18 bw)100% restored at   195 msec -> N03 N06 N05 N03


Path# Break = N05 - N08 (81 working channels lost)
1   (16 bw) 20% restored at   140 msec -> N08 N07 N05
2   (27 bw) 54% restored at   161 msec -> N08 N06 N05
3   (17 bw) 75% restored at   182 msec -> N08 N11 N05
4   (1 bw)  76% restored at   318 msec -> N08 N06 N01 N03 N05
5   (17 bw) 97% restored at   396 msec -> N08 N04 N09 N11 N05
6   (3 bw) 100% restored at   402 msec -> N08 N04 N01 N03 N05


Path# Break = N01 - N04 (71 working channels lost)
1   (20 bw) 29% restored at   256 msec -> N04 N08 N06 N01
2   (23 bw) 61% restored at   599 msec -> N04 N09 N11 N05 N03 N01
```

```
3   (4 bw)  67% restored at   802 msec -> N04 N09 N11 N05 N03 N02 N01
4   (7 bw)  77% restored at   997 msec -> N04 N09 N10 N11 N05 N03 N02 N01
5   ( 8 bw) 88% restored at  1123 msec -> N04 N09 N10 N11 N08 N06 N05 N03 N02 N01
6   ( 9 bw)100% restored at  1144 msec -> N04 N09 N10 N11 N08 N07 N05 N03 N02 N01

Path# Break = N08 - N04 (59 working channels lost)
1   (20 bw) 34% restored at   210 msec -> N04 N01 N06 N08
2   (17 bw) 63% restored at   252 msec -> N04 N09 N11 N08
3   (8 bw)  77% restored at   419 msec -> N04 N09 N11 N05 N06 N08
4   (2 bw)  80% restored at   446 msec -> N04 N09 N11 N05 N07 N08
5   (11 bw) 99% restored at   512 msec -> N04 N01 N03 N05 N07 N08
6   (1 bw) 100% restored at   594 msec -> N04 N09 N10 N05 N05 N07 N08

Path# Break = N07 - N08 (41 working channels lost)
1   (17 bw) 42% restored at   207 msec -> N08 N11 N05 N07
2   (24 bw) 100% restored at  228 msec -> N08 N06 N05 N07
```

The following table summarizes the restoration performance of the RREACT protocol on NJ net for all link breaks.

Table 2: NJ Network RREACT Restoration Performance

Break	Work	Restored	#Paths	Spares	Msgs	Time(msec)
N01-N02	74	100%	4	252	78	582
N01-N03	71	100%	2	160	73	195
N01-N04	71	100%	6	401	152	1144
N01-N05	53	100%	3	116	63	214
N01-N06	55	100%	4	185	110	559
N01-N08	52	100%	4	128	77	344
N02-N03	53	100%	4	189	149	741
N03-N05	16	100%	1	48	107	207
N04-N05	68	100%	4	204	81	318
N04-N08	59	100%	6	222	108	594
N04-N09	51	100%	5	245	149	684
N05-N06	48	100%	3	144	75	253

Table 2: NJ Network RREACT Restoration Performance

Break	Work	Restored	#Paths	Spares	Msgs	Time(msec)
N05-N07	16	100%	1	48	64	195
N05-N08	81	100%	6	204	79	402
N05-N09	50	100%	3	139	113	442
N05-N11	48	100%	5	210	105	559
N06-N08	47	100%	3	141	94	301
N07-N08	41	100%	2	123	68	228
N08-N09	57	100%	3	154	119	327
N08-N11	64	100%	6	233	114	739
N09-N10	78	100%	5	326	237	984
N09-N11	65	100%	4	189	90	592
N10-N11	34	100%	2	82	200	361

6.1.2 FITNESS Simulation Results

The UCCS implementation of the FITNESS protocol was tested using NETRESTORE against the same New Jersey network.

For the following simulation runs, the same assumptions as per the RREACT testing were used:

a. All messages have equal priority.

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

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

d. It requires an additional 10 msec to generate each outgoing message.

e. Propagation speed of messages is 200000 km/sec.

f. Transmission delays are computed for variable length messages, using a 10000 baud transfer rate (i.e., 1 msec per byte).

The following table provides some examples of the restoration solution output of the FITNESS protocol. These examples correspond to the RREACT examples given in Section 6.1 above.

TABLE 3 - Sample NJ Network FITNESS Restoration Solutions

```
Path# Break = N02 - N03 (53 working channels lost)
1  (31 bw) 59% restored at   468 msec -> N02 N01 N04 N09 N10 N11 N05 N03
```

```
2 (22 bw) 100% restored at  724 msec -> N02 N01 N03

Path# Break = N03 - N01 (71 working channels lost)
1  (53 bw) 75% restored at  312 msec -> N01 N02 N03
2 (18 bw) 100% restored at  654 msec -> N03 N06 N05 N03

Path# Break = N05 - N08 (81 working channels lost)
1 (27 bw) 34% restored at   312 msec -> N05 N06 N08
2 (20 bw) 59% restored at   685 msec -> N05 N11 N09 N04 N08
3 (16 bw) 78% restored at   972 msec -> N05 N07 N08
4 (17 bw) 99% restored at  1439 msec -> N05 N03 N01 N04 N09 N10 N11 N08
5 (1 bw) 100% restored at  1756 msec -> N05 N03 N01 N06 N08

Path# Break = N01 - N04 (71 working channels lost)
1 (34 bw) 48% restored at   468 msec -> N01 N02 N03 N05 N11 N10 N09 N04
2 (20 bw) 77% restored at   754 msec -> N01 N06 N08 N04
3 (16 bw) 99% restored at  1208 msec -> N01 N03 N05 N07 N08 N11 N09 N04
4 (1 bw) 100% restored at  1618 msec -> N01 N03 N05 N06 N08 N11 N09 N04

Path# Break = N08 - N04 (59 working channels lost)
1 (27 bw) 46% restored at   406 msec -> N04 N09 N11 N05 N06 N08
2 (17 bw) 75% restored at   746 msec -> N04 N09 N10 N11 N08
3 (15 bw) 100% restored at 1127 msec -> N04 N01 N03 N05 N07 N08

Path# Break = N07 - N08 (41 working channels lost)
1 (24 bw) 66% restored at   342 msec -> N07 N05 N06 N08
2 (17 bw) 100% restored at  672 msec -> N07 N05 N11 N08
```

The following table summarizes the restoration performance of the FITNESS protocol on the NJ net for all link breaks.

Table 4: NJ Network FITNESS Restoration Performance

Break	Work	Restored	#Paths	Spares	Msgs	Time(msec)
N01-N02	74	100%	3	343	136	1096
N01-N03	71	100%	2	160	64	654
N01-N04	71	100%	4	417	149	1618
N01-N05	53	100%	2	157	61	645

Table 4: NJ Network FITNESS Restoration Performance

Break	Work	Restored	#Paths	Spares	Msgs	Time(msec)
N01-N06	55	100%	3	208	104	1086
N01-N08	52	100%	3	195	119	1055
N02-N03	53	100%	2	261	101	724
N03-N05	16	100%	1	48	30	343
N04-N05	68	100%	3	269	89	1044
N04-N08	59	100%	3	278	112	1127
N04-N09	51	100%	4	272	164	1568
N05-N06	48	100%	3	184	121	1045
N05-N07	16	100%	1	48	29	344
N05-N08	81	100%	5	289	200	1756
N05-N09	50	100%	2	166	62	705
N05-N11	48	100%	2	268	85	754
N06-N08	47	100%	3	141	101	1004
N07-N08	41	100%	2	123	86	672
N08-N09	57	100%	3	178	105	1005
N08-N11	64	100%	5	239	197	1827
N09-N10	78	100%	4	357	154	1509
N09-N11	65	100%	2	254	88	748
N10-N11	34	100%	2	82	72	683

6.1.3 Two-Prong Simulation Results

The Two-Prong protocol was tested using NETRESTORE against the same New Jersey network. For the following simulation runs, the same assumptions as per the RREACT testing were used:

a. All messages have equal priority.

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

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

d. It requires an additional 10 msec to generate each outgoing message.

e. Propagation speed of messages is 200000 km/sec.

f. Transmission delays are computed for variable length messages, using a 10000 baud transfer rate (i.e., 1 msec per byte).

The following table provides some examples of the restoration solution output of the Two-Prong protocol. These examples correspond to the RREACT examples given in Section 6.1 above.

TABLE 5 - Sample NJ Network Two-Prong Restoration Solutions

```
Path# Break = N02 - N03 (53 working channels lost)
1 (23 bw) 43% restored at   147 msec -> N03 N01 N02
2 (20 bw) 81% restored at   304 msec -> N03 N05 N06 N01 N02
3 (10 bw)100% restored at   447 msec -> N03 N05 N11 N09 N04 N01 N02

Path# Break = N03 - N01 (71 working channels lost)
1 (53 bw) 75% restored at   107 msec -> N03 N02 N01
2 (18 bw) 100% restored at  200 msec -> N03 N05 N06 N01

Path# Break = N05 - N08 (81 working channels lost)
1 (16 bw) 20% restored at   115 msec -> N08 N07 N05
2 (27 bw) 53% restored at   158 msec -> N08 N06 N05
3 (17 bw) 74% restored at   231 msec -> N08 N11 N05
4 ( 1 bw) 75% restored at   304 msec -> N08 N06 N01 N03 N05
5 ( 3 bw) 79% restored at   325 msec -> N08 N04 N01 N03 N05
6 (11 bw) 93% restored at   345 msec -> N08 N04 N01 N03 N05
7 ( 6 bw)100% restored at   386 msec -> N08 N04 N01 N02 N03 N05

Path# Break = N01 - N04 (71 working channels lost)
1 (20 bw) 28% restored at   197 msec -> N04 N08 N06 N01
2 (20 bw) 56% restored at   426 msec -> N04 N09 N11 N05 N03 N01
3 ( 3 bw) 61% restored at   446 msec -> N04 N09 N11 N08 N07 N05 N03 N01
4 ( 4 bw) 66% restored at   477 msec -> N04 N09 N11 N08 N07 N05 N03 N02 N01
5 (14 bw) 86% restored at   540 msec -> N04 N09 N10 N11 N05 N03 N02 N01
6 ( 9 bw) 99% restored at   560 msec -> N04 N09 N10 N11 N08 N07 N05 N03 N02 N01

Path# Break = N04 - N08 (59 working channels lost)
1 (20 bw) 34% restored at   172 msec -> N08 N06 N01 N04
2 (17 bw) 63% restored at   239 msec -> N08 N11 N09 N04
3 (11 bw) 81% restored at   465 msec -> N08 N07 N05 N03 N01 N04

Path# Break = N07 - N08 (41 working channels lost)
```

```
1 (27 bw) 66% restored at   167 msec -> N08 N06 N05 N07
2 (14 bw) 100% restored at  274 msec -> N08 N11 N05 N07
```

The following table summarizes the restoration performance of the Two-Prong protocol on the NJ net for all link breaks.

Table 6: NJ Network Two-Prong Restoration Performance

Break	Work	Restored	#Paths	Spares	Msgs	Time(msec)
N01-N02	74	74%	4	198	181	445
N01-N03	71	100%	2	160	147	200
N01-N04	71	99%	6	392	214	560
N01-N05	53	100%	3	116	126	178
N01-N06	55	85%	3	145	206	329
N01-N08	52	100%	3	128	167	312
N02-N03	53	100%	3	186	117	447
N03-N05	16	100%	1	48	127	211
N04-N05	68	100%	5	204	132	272
N04-N08	59	81%	4	166	169	465
N04-N09	51	100%	5	245	184	493
N05-N06	48	100%	3	144	170	276
N05-N07	16	100%	1	48	114	179
N05-N08	81	100%	7	210	192	386
N05-N09	50	100%	4	139	163	271
N05-N11	48	100%	5	218	178	388
N06-N08	47	100%	3	141	185	360
N07-N08	41	100%	2	123	158	274
N08-N09	57	100%	6	163	154	362
N08-N11	64	84%	3	162	155	252
N09-N10	78	96%	4	308	192	622
N09-N11	65	100%	3	189	173	331

Table 6: NJ Network Two-Prong Restoration Performance

Break	Work	Restored	#Paths	Spares	Msgs	Time(msec)
N10-N11	34	100%	2	82	156	278

6.2 Simulation Results on the US network

6.2.1 RREACT Simulation Results

The RREACT protocol was also tested against the US network (Figure 6.2) defined in the

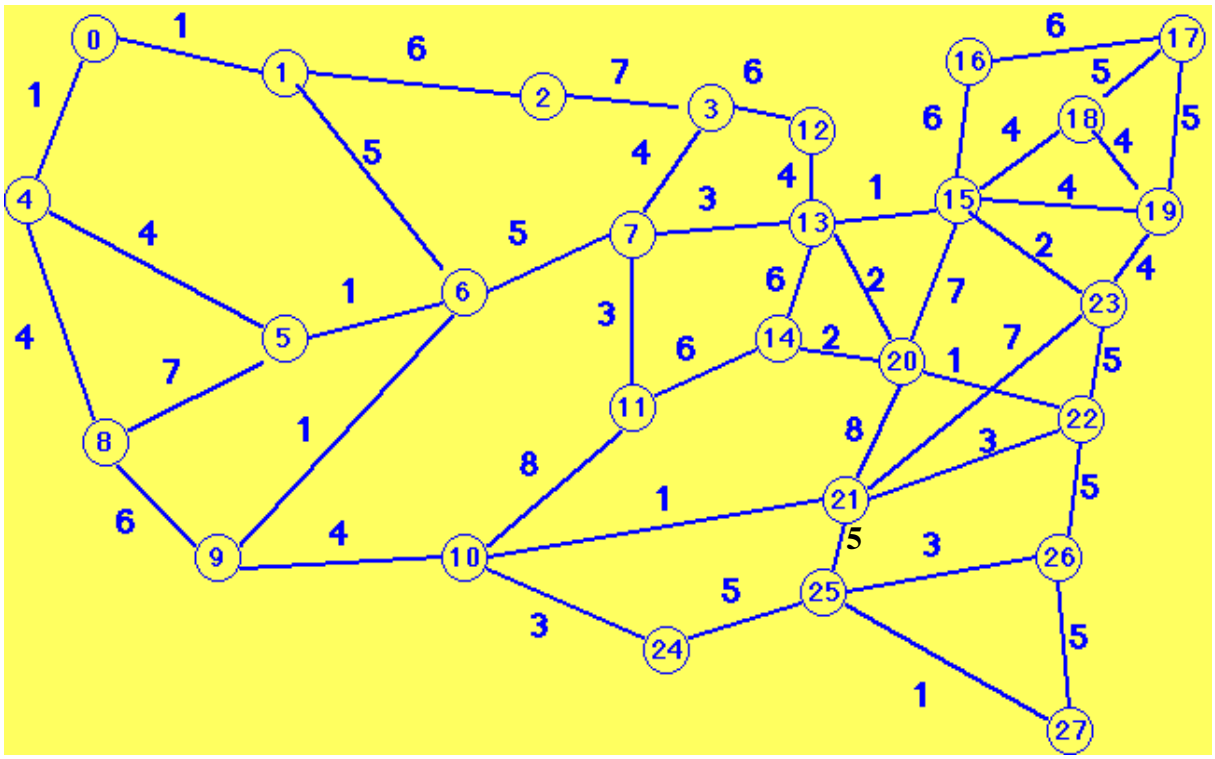

Figure 6.2. The Continental US Study Network Model (US net).

GROVER [GROV89] paper. This network was designed to approximate typical national network connectivity (with fewer span degree in each node). This evaluation was run using slightly different criteria, where the number of working channels between any two nodes is assumed to be 100 and the objective of the test is to find ALL possible alternative paths between the two nodes involved with the break.

The following table shows examples of the restoration result output of the RREACT protocol. for the US network.

TABLE 7 - Sample of US Network RREACT Restoration Solutions

```
Path# Break = N11 - N07 (100 working channels lost)
1   (3 bw)   3% restored at   239 msec -> N07 N13 N14 N11
2   (1 bw)   4% restored at   361 msec -> N07 N06 N09 N10 N11
3   (3 bw)   7% restored at   417 msec -> N07 N03 N12 N13 N14 N11
4   (1 bw)   8% restored at  1039 msec -> N07 N06 N05 N08 N09 N10 N11
5   (1 bw)   9% restored at  1231 msec -> N07 N03 N12 N13 N20 N21 N10 N11
6   (1 bw)  10% restored at  1254 msec -> N07 N06 N01 N00 N04 N08 N09 N10 N11
Message Traffic = 1070 Spares Used = 49

Path# Break = N21 - N10 (100 working channels lost)
1   (3 bw)   3% restored at   265 msec -> N10 N24 N25 N21
2   (2 bw)   5% restored at   423 msec -> N10 N11 N14 N20 N21
3   (2 bw)   7% restored at  1047 msec -> N10 N11 N14 N13 N20 N21
4   (1 bw)   8% restored at  1341 msec -> N10 N11 N14 N13 N15 N20 N21
Message Traffic = 1079 Spares Used = 33

Path# Break = N09 - N10 (100 working channels lost)
1   (1 bw)   1% restored at   373 msec -> N10 N11 N07 N06 N09
2   (1 bw)   2% restored at   674 msec -> N10 N11 N07 N06 N05 N08 N09
3   (1 bw)   3% restored at   928 msec -> N10 N11 N07 N06 N01 N00 N04 N08 N09
Message Traffic = 897 Spares Used = 18

Path# Break = N09 - N08 (100 working channels lost)
1   (1 bw)   1% restored at   249 msec -> N08 N05 N06 N09
2   (1 bw)   2% restored at   999 msec -> N08 N04 N00 N01 N06 N07 N11 N10 N09
Message Traffic = 1069 Spares Used = 11

Path# Break = N05 - N06 (100 working channels lost)
1   (1 bw)   1% restored at   231 msec -> N06 N09 N08 N05
2   (1 bw)   2% restored at   300 msec -> N06 N01 N00 N04 N05
3   (3 bw)   5% restored at   621 msec -> N06 N07 N11 N10 N09 N08 N05
4   (1 bw)   6% restored at  1375 msec -> N06 N07 N13 N14 N11 N10 N09 N08 N05
Message Traffic = 838 Spares Used = 33

Path# Break = N15 - N19 (100 working channels lost)
1   (4 bw)   4% restored at   157 msec -> N19 N18 N15
2   (2 bw)   6% restored at   172 msec -> N19 N23 N15
3   (5 bw)  11% restored at   231 msec -> N19 N17 N16 N15
4   (2 bw)  13% restored at   592 msec -> N19 N23 N21 N20 N15
Message Traffic = 1052 Spares Used = 35
```

Path# Break = N21 - N20 (100 working channels lost)

```
1   (1 bw)   1% restored at   152 msec -> N20 N22 N21
2   (2 bw)   3% restored at   335 msec -> N20 N15 N23 N21
3   (1 bw)   4% restored at   510 msec -> N20 N14 N11 N10 N21
4   (1 bw)   5% restored at   669 msec -> N20 N14 N11 N10 N24 N25 N21
5   (4 bw)   9% restored at   797 msec -> N20 N15 N19 N23 N21
6   (2 bw)  11% restored at  1305 msec -> N20 N13 N14 N11 N10 N24 N25 N21
Message Traffic = 1055 Spares Used = 48
```

6.2.2 FITNESS **Simulation Results**

The FITNESS protocol was also tested against the US Continental test network This evaluation criteria was the same as used in the RREACT simulation runs on this network.

The following table shows examples of the solution output of the FITNESS protocol for the US network.

TABLE 8 - Sample US Network FITNESS Restoration Solutions

```
Path# Break = N11 - N07 (100 working channels lost)
1  (4 bw)   4% restored at   418 msec -> N07 N03 N12 N13 N14 N11
2  (2 bw)   6% restored at   723 msec -> N07 N13 N14 N11
2  (1 bw)   7% restored at  1094 msec -> N07 N06 N09 N10 N11
4  (1 bw)   8% restored at  1478 msec -> N07 N13 N20 N21 N10 N11
5  (1 bw)   9% restored at  1884 msec -> N07 N06 N05 N08 N09 N10 N11
Message Traffic = 511 Spares Used = 41
```

One solution path not found. Time constraint encountered.

```
Path# Break = N21 - N10 (100 working channels lost)
1  (3 bw)   3% restored at   352 msec -> N10 N24 N25 N21
2  (2 bw)   5% restored at   717 msec -> N10 N11 N14 N20 N21
3  (2 bw)   7% restored at  1110 msec -> N10 N11 N14 N13 N20 N21
4  (1 bw)   8% restored at  1515 msec -> N10 N11 N14 N13 N15 N20 N21
Message Traffic = 452 Spares Used = 33

Path# Break = N09 - N10 (100 working channels lost)
1  (1 bw)   1% restored at   390 msec -> N10 N11 N07 N06 N09
2  (1 bw)   2% restored at   805 msec -> N10 N11 N07 N06 N05 N08 N09
3  (1 bw)   3% restored at  1528 msec -> N10 N11 N07 N06 N01 N00 N04 N08 N09
Message Traffic = 478 Spares Used = 18

Path# Break = N09 - N08 (100 working channels lost)
```

```
 1 (1 bw)   1% restored at   363 msec -> N08 N05 N06 N09
 2 (1 bw)   2% restored at  1121 msec -> N08 N04 N00 N01 N06 N07 N11 N10 N09
 Message Traffic = 373 Spares Used = 11

 Path# Break = N05 - N06 (100 working channels lost)
 1 (3 bw)   3% restored at   457 msec -> N06 N07 N11 N10 N09 N08 N05
 2 (1 bw)   4% restored at   746 msec -> N06 N09 N08 N05
 3 (1 bw)   5% restored at  1113 msec -> N06 N01 N00 N04 N05
 4 (1 bw)   6% restored at  1857 msec -> N06 N07 N13 N14 N11 N10 N09 N08 N05
 Message Traffic = 357 Spares Used = 33

Path# Break = N15 - N19 (100 working channels lost)
1  (5 bw)   5% restored at   355 msec -> N19 N17 N16 N15
2  (4 bw)   9% restored at   648 msec -> N19 N18 N15
3  (4 bw)  13% restored at  1030 msec -> N19 N23 N21 N20 N15
Message Traffic = 316 Spares Used = 39

Path# Break = N21 - N20 (100 working channels lost)
 1 (4 bw)   4% restored at   385 msec -> N20 N15 N19 N23 N21
 2 (2 bw)   6% restored at   703 msec -> N20 N15 N23 N21
 3 (2 bw)   8% restored at  1133 msec -> N20 N14 N11 N10 N24 N25 N21
 4 (1 bw)   9% restored at  1395 msec -> N20 N22 N21
 5 (1 bw)  10% restored at  1809 msec -> N20 N13 N14 N11 N10 N21
Message Traffic = 481 Spares Used = 41
```

One solution path not found. Time constraint encountered.

6.2.3 Two-Prong Simulation Results

The Two-Prong protocol was also tested against the US Continental test network. This evaluation criteria was the same as used in the RREACT simulation runs on this network.

The following table shows examples of the solution output of the Two-Prong protocol for the US network.

TABLE 7 - Sample of US Network Two-Prong Restoration Solutions

```
Path# Break = N11 - N07 (100 working channels lost)
1  (3 bw)   3% restored at  236 msec -> N07 N13 N14 N11
2  (1 bw)   4% restored at  275 msec -> N07 N06 N09 N10 N11
3  (1 bw)   5% restored at  333 msec -> N07 N03 N12 N13 N14 N11
4  (1 bw)   6% restored at  454 msec -> N07 N06 N05 N08 N09 N10 N11
5  (1 bw)   7% restored at  471 msec -> N07 N03 N12 N13 N20 N14 N11
6  (1 bw)   8% restored at  545 msec -> N07 N06 N01 N00 N04 N08 N09 N10 N11
```

```
Message Traffic = 738 Spares Used = 38

Path# Break = N21 – N10 (100 working channels lost)
1   (3 bw)   3% restored at   210 msec -> N10 N24 N25 N21
2   (2 bw)   5% restored at   286 msec -> N10 N11 N14 N20 N21
3   (2 bw)   7% restored at   363 msec -> N10 N11 N14 N13 N20 N21
4   (1 bw)   8% restored at   465 msec -> N10 N11 N14 N13 N15 N20 N21
Message Traffic = 321 Spares Used = 33

Path# Break = N09 – N10 (100 working channels lost)
1   (1 bw)   1% restored at   345 msec -> N10 N11 N07 N06 N09
2   (1 bw)   2% restored at   546 msec -> N10 N11 N07 N06 N05 N08 N09
3   (1 bw)   3% restored at   660 msec -> N10 N11 N07 N03 N02 N01 N00 N04 N08 N09
Message Traffic =423 Spares Used = 19

Path# Break = N09 – N08 (100 working channels lost)
1   (1 bw)   1% restored at   183 msec -> N08 N05 N06 N09
Message Traffic = 403 Spares Used = 3

Path# Break = N05 – N06 (100 working channels lost)
1   (1 bw)   1% restored at   212 msec -> N06 N09 N08 N05
2   (1 bw)   2% restored at   236 msec -> N06 N01 N00 N04 N05
3   (3 bw)   5% restored at   432 msec -> N06 N07 N11 N10 N09 N08 N05
4   (1 bw)   6% restored at   500 msec -> N06 N07 N13 N14 N11 N10 N09 N08 N05
Message Traffic = 384 Spares Used = 33

Path# Break = N15 – N19 (100 working channels lost)
1   (4 bw)   4% restored at   129 msec -> N19 N18 N15
2   (2 bw)   6% restored at   157 msec -> N19 N23 N15
3   (5 bw)  11% restored at   169 msec -> N19 N17 N16 N15
4   (1 bw)  12% restored at   387 msec -> N19 N23 N22 N20 N15
5   (1 bw)  13% restored at   542 msec -> N19 N23 N21 N20 N15
Message Traffic = 591 Spares Used = 35

 Path# Break = N21 – N20 (100 working channels lost)
1   (1 bw)   1% restored at   155 msec -> N20 N22 N21
2   (2 bw)   3% restored at   266 msec -> N20 N15 N23 N21
3   (4 bw)   7% restored at   328 msec -> N20 N15 N19 N23 N21
4   (1 bw)   8% restored at   352 msec -> N20 N14 N11 N10 N21
5   (1 bw)   9% restored at   424 msec -> N20 N14 N11 N10 N24 N25 N21
6   (1 bw)  10% restored at   531 msec -> N20 N13 N14 N11 N10 N24 N25 N21
```

```
7   (1 bw)  11% restored at   615 msec -> N20 N13 N07 N06 N09 N10 N24 N25 N21
Message Traffic = 520 Spares Used = 49
```

6.3 Discussion of Simulation Results

The RREACT results show promise for improvement in overall network restoration performance in a robust manner. For the typical two second restoration requirement, RREACT's performance exceeded this requirement and our initial expectations by a wide margin. The RREACT solution versus those obtained from the FITNESS protocol shows its strength in a number of different areas:

Time to full (and intermediate stage) restoration.

The number of messages used to accomplish restoration were comparable.

The number of spare channels used to affect each restoration was considerably better.

The largest cause of these differences centers on FITNESS's use of multiple message waves and the penalty it exacts. The FITNESS algorithm chooses the largest identified restoration path during each wave of messages so that the number of discrete message waves required is minimized, and therefore the total time needed to complete the restoration. Unfortunately, this will (usually) result in inefficient use of the network's spare channel links for the restoration. Its performance directly scales with the number of discrete paths needed. Results for the simulation runs against the US network show slightly more favorable performance under most conditions, but when the number of needed restoration paths are above a certain threshold (approximately 5), FITNESS does not have enough time to complete the full restoration.

The simulation results on US Continental test network show that the number of messages generated by Two-Prong is about one half of that generated by RREACT and that the restoration time of Two-Prong is about half of that spent by RREACT. RREACT in nature will find all the paths between the disrupted nodes therefore as the network size increases the message volume increases dramatically. The Two-Prong on the other hand stops the flooding when the message reaches a node with a different color. However, in two cases, RREACT is able to find one additional path. The simulation results on New Jersey LATA test network show that the RREACT is quite robust and in some cases the restoration time and spare capacity usage are both better than the Two-Prong.

The overall excellent results of the basic RREACT protocol in initial simulation and evaluation prompted further investigation into use of available time to provide a more optimal solution to network restoration problem. The next section discusses enhancements to the basic protocol which greatly improve the optimality of the final restoration solution with minimal performance penalties.

7. RREACT Enhancements

Versions of RREACT are being developed at UCCS to address the problem of optimal use of network resources (i.e., spare channels) during fast restorations.

The first version, called RREACT+, is based on a simple heuristic which addresses problems such as Whalen's Dilemma where selection of one restoration path early in the process has a negative

impact on the protocol's ability to fully restore all working channels even when simple inspection reveals that the capacity exists for a full restoration and should be possible.

The second version uses the local topology information inherent in the incoming SEEK messages to derive an optimal allocation of spares (i.e., minimize the total number utilized for a given restoration) based on a dynamic programming solution using a variation of the 0/1 Knapsack algorithm [HS78]. Instead of instantly making connections as paths are identified with incoming SEEK messages at the CHOOSER node, the CHOOSER node continually computes the optimal restoration solution (i.e., minimal use of network spare channels) by a variation of the 0/1 Knapsack Problem dynamic programming algorithm. Any trigger event causes the CHOOSER node to select the optimal solution set (or the best possible at the present time) and begin sending CONNECT messages per the basic RREACT protocol corresponding to the selected solution. After this version of RREACT was implemented, we tried in on several networks and soon found that it quickly ran out of memory on NJ net and can only work on very small networks such as that in Figure 7.1.

RREACT+ Distributed Network Restoration Algorithm

RREACT+ operates as follows:

Instead of instantly making connections as paths are identified with incoming SEEK messages at the CHOOSER node, the CHOOSER node only updates the link capacity table, determines a path weight W_p for the new path, and saves this potential restoration path to some memory store which is ordered by the W_p values.

On receiving a trigger event:

The path with the best W_p is selected.

A CONNECT message sent over the restoration path.

The link capacity table is updated.

The path memory store is purged of any paths which are now non-viable.

The W_p values of all paths remaining in the store are updated.

The path memory store is sorted by W_p.

A trigger event may be defined as one or more of the following:

Hardware TIMEOUT.

Some maximum number of stored paths.

A constant parameter based on the local network topology.

A parameter based on SEEK message arrivals over incoming links.

Some other dynamically or statically determined event...

The W_p value for a path is heuristically calculated based on the following factors:

Total number of restoration channels a path can support.

Path length (i.e., hops).

Utilization of all links along the path.

Impact of path's selection on other potential restoration paths.

There Wp is calculated as follows:

W_{bw} = Path channel restoration capacity / Number of channels to restore

W_{pl} = 2 / Path length, #hops

W_{eff} = $\frac{\text{SUM (Path channel restoration capacity / Link spare channel count)}}{\text{Path length, \#hops}}$

W_w = $W_{bw} * W_{pl} * W_{eff}$

W_{imp} = SUM (All other known paths W_w's with common links)

W_p = $W_w - W_{imp}$

This improved algorithm's methodology and usefulness is best illustrated by a simple example on Figure 7.1:

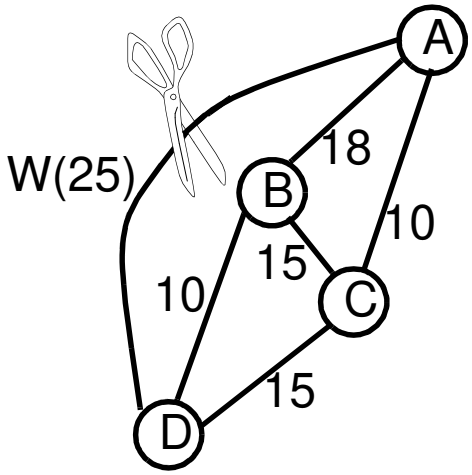

Figure 7.1. A simple example illustrates the RREACT+ algorithm.

Initially, 25 channels need to be restored between Nodes A and D, and the following table shows

Path	BW	W_{bw}	W_{pl}	W_{eff}	W_w	W_{imp}	Wp
ABD	10	$\frac{10}{25}$	1	$\frac{\frac{10}{18}+\frac{10}{10}}{2}$	0.311	0.614	-0.303
ABCD	15	$\frac{15}{25}$	$\frac{2}{3}$	$\frac{\frac{10}{18}+\frac{15}{15}+\frac{15}{15}}{3}$	0.377	0.881	-0.504
ACBD	10	$\frac{10}{25}$	$\frac{2}{3}$	$\frac{\frac{10}{18}+\frac{15}{15}+\frac{15}{15}}{3}$	0.237	1.021	-0.784
ĐACD	10	$\frac{10}{25}$	1	$\frac{\frac{10}{10}+\frac{10}{15}}{2}$	0.333	0.614	-0.281

the calculation of weights.

After ACD is used, ACBD is deleted; 15 channels need to be restored and the following table shows the calculation of weights.

Path	BW	W_{bw}	W_{pl}	W_{eff}	W_w	W_{imp}	Wp
ĐABD	10	$\frac{10}{15}$	1	$\frac{\frac{10}{18}+\frac{10}{10}}{2}$	0.518	0.119	0.398
ABCD	5	$\frac{5}{15}$	$\frac{2}{3}$	$\frac{\frac{5}{18}+\frac{5}{15}+\frac{5}{5}}{3}$	0.119	0.518	-0.398

Next, after ABD is used, only ABCD left.

In this example, if path ACBD was identified first by the basic RREACT protocol, using this restoration path would have made complete restoration impossible due to the impact of this path on other restoration paths. In the improved protocol, using ACBD would be deferred and ultimately not selected due to its poor W_p value.

A preliminary version of the RREACT+ algorithm has been implemented for the NETRESTORE system. For this version, a simulated hardware timer was used to determine when the CHOOSER should start making connections; if no new SEEK messages have arrived in a window of 100 msec since the last one, all paths are assumed to be identified and all CONNECT messages are promulgated.

This version of RREACT+ was validated against the simple network given in Figure 7.1 above and identical results obtained. RREACT+ also was evaluated against the NJ network defined in

Figure 6.1. As expected, this modification to the basic RREACT algorithm imposed a considerable penalty in restoration times. All restoration times for the NJ net increased into the 800 to 1300 msec range. The spare utilization efficiency of RREACT+ proved to be somewhat disappointing as almost half the link breaks resulted in identical spare usage to the basic RREACT algorithm and the other half exhibited net increases in spare use (between 8 and 60 more). Three cases showed minor improvements in spare usage. The overall message count for each case remained virtually identical.

RREACT+ was designed to provide complete restoration solutions where other protocols would fail or otherwise perform very poorly. Further study and simulation is being performed to determine RREACT+'s performance, robustness and other characteristics under difficult network restoration topologies (e.g., Whalen's Dilemma) and various message processing delay scenarios,

8. Summary

We have presented a distributed network restoration protocol called RREACT based on the Sender-Chooser paradigm. It tries to find all the paths between the two disrupted nodes by attaching path information in the broadcast request messages. The simulation results indicate that RREACT due to its simplicity is very robust and with single wave of request message is able to perform single link restoration under two seconds. Mutiple simultaneous link failures are supported as well as other types of realtime changes in network configuration during restoration. A comparison with other important distributed restoration algorithms on a number of network configurations was also given.

References

[ABG91] B. Awerbuch, A. Bar-Noy, and M. Gopal, "Approximate distributed bellman-ford algorithms," in *Proceedings of IEEE INFOCOM*, pp. 1206–1213, April 1991.

[BAKE91] J. E. Baker, "Distributed Link Restoration With Robust Preplanning," in *Proceedings of GlobalCom '91*, pp. 306-311, Dec. 1991.

[BGGJP85] 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*, vol. SAC-3, No. 3, pp. 416-426, May 1985.

[BOVE86] J. Bovet, "Une amélioration de la methode de Dijkstra pour la recherche d'un plus court dans un réseau," *Discrete Applied Math.*, vol. 13, pp. 93-96, 1986.

[CARR71] B. A. Carré, "An Algebra for network routing problems," *Journal Inst. Math. Appl.* vol. 7, pp. 273-294, 1971.

[CLR90] T. H. Cormen, C. E. Leiserson, R. L. Riovest, *Introduction to Algorithms*, The MIT Press, 1990.

[COAN 91] Coan, B, et al, "Using Distributed Topology Update and Preplanned Configurations to Achieve Trunk Network Survivability," *IEEE Trans. on Reliability*, vol. 40, No. 4, October 1991.

[CHOW93] C.-H. E. Chow, J. Bicknell, S. McCaughey, and S. Syed,
"A Fast Distributed Network Restoration Algorithm," to be published in *Proceeding of*

12th International Phoenix Conference on Computers and Communications, March 24-26, 1993, Scottsdale, Arizona.

[DHR90] D. K. Doherty, W. D. Hutcheson, and K. K. Raychaudhuri, "High capacity digital network management and control," in *Proceedings of GlobalCom '90*, (San Diego), pp. 301.3.1–301.3.5, Dec. 1990.

[DIJK59] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math., vol. 1, pp. 269-271, 1959.

[DP84] N. Deo and C.-Y. Pang, "Shortest-path algorithms: Taxonomy and annotation," *Networks*, vol. 14, pp. 275–323, 1984.

[DP92] R. D. Doverspike and C. D. Pack, "Using SONET for Dynamic Bandwidth Management in Local Exchange Network," in *Proceedings of 5th International Network Planning Symposium*, Kobe, Japan, June 1992.

[GROV87] W. D. Grover, "The self-healing network: A fast distributed restoration technique for networks using digital cross-connect machines," in *Proceedings of GlobalCom '87*, pp. 28.2.1–28.2.6, 1987.

[GROV89] 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.

[GT88] A. V. Goldberg and R. E. Tarjan, "A New Approach to the Maximum-Flow problem," *Journal of the Association for Computing Machinery*, vol. 35, No. 4, pp. 921-940, Oct. 1988.

[HS78] E. Horowitz and S. Sahni, "Fundamentals of Computer Algorithms", pp. 219-228, *Computer Science Press*, 1978.

[KCOMS90] H. Komine, T. Chujo, T. Ogura, K. Miyazaki, and T. Soejima, "A distributed restoration algorithm for multiple-link and node failures of transport networks," in *Proceedings of GlobalCom '90*, (San Diego), pp. 403.4.1–403.4.5, Dec. 1990.

[MFR78] J.M. McQuillan, G. Falk and I. Richer, "A review of the development and performance of the ARPANET routing algorithm, *IEEE Trans. on Communications*, vol. COM-26, No. 12, pp. 1802-1811, December 1978.

[MP88] T. Mohr and C. Pasche, "A parallel shortest path algorithm," *Computing*, vol. 40, pp. 281-292, 1988.

[MRR80] J.M. McQuillan, I. Richer and E.C. Rosen, "The new routing algorithm for the ARPANET, *IEEE Trans. on Communications*, vol. COM-28, No. 5, pp. 711-719, May 1980.

[NICH66] T. A. Nicholson, "Finding the shortest route between two points in a network," Computer J., vol. 9, pp. 276-280, 1966.

[PH90] C. Palmer and F. Hummel, "Restoration in a partitioned multi-bandwidth cross-connect network," in *Proceedings of GlobalCom '90*, (San Diego), pp. 301.7.1–301.7.5, Dec. 1990.

[SNH90] H. Sakauchi, Y. Nishimura, and S. Hasegawa, "A self-healing network with an economical spare-channel assignment," in *Proceedings of GlobalCom '90*, (San Diego), pp. 403.1.1–403.1.6, Dec. 1990.

[SS80] M. Schwartz and T. E. Stern, "Routing techniques used in computer communications networks," *IEEE Trans. on Communications*, vol. COM-28, No. 4, pp. 539-552, April 1980.

[ST84] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325-336 1984.

[SUUR74] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, pp. 125-145, 1974

[TOPK88] D. M. Topkis, "A k shortest path algorithm for adaptive routing in communications networks," *IEEE Trans. on Communications*, vol. 36, pp. 855–859, July 1988.

[WK90] J. S. Whalen and J. Kenney, "Finding maximal link disjoint paths in a multigraph," in *Proceedings of GlobalCom '90*, (San Diego), pp. 403.6.1–403.6.5, Dec. 1990.

[WROB90] L. A. Wrobel, *Disaster Recovery Planning for Telecommunications*, Artech House, 1992.

[WU92] T.-H. Wu, *Fiber Network Service Survivability*, Artech House, 1992.

[YH88] Yang, C. H. and S. Hasegawa, "FITNESS: Failure Immunization Technology for Network Service Survivability," *Proc. of GlobalCom '88*, pp. 47.3.1-47.3.6, November 1988.

BIBLOGRAPHIC DATA SHEET	1. Report No. **EAS-CS-92-18**	2.	3. Recipient's Accession Number

4. Title and Subtitle

RREACT: A Distributed Protocol for Rapid Restoration of Active Communication Trunks

5. Report Date
November 1992

6.

7. Author(s)

C. Edward Chow, Steve McCaughey, and Sami Syed

8. Performing Organization Rept. No.
EAS-CS-92-18

9. Performing Organization Name and Address

Department of Computer Science
University of Colorado at Colorado Springs
Colorado Springs, CO 80933-7150

10. Project/Task/Work Unit No.
MCI IXC Network Resoration

11. Contract/Grant No.
MCI Grant # 01-80046

12. Sponsoring Organization Name and Address

MCI Telecommuncations Corporation
4678 Alpine Meadows Lane
Colorado Springs, CO 80919

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts

Commercial telecommunications networks have tight real-time requirements for restoration after a failure. The problem of finding the available restoration paths and reassigning the interrupted traffic within such tight real-time requirements places difficult demands on the restoration protocol employed. This paper reviews several distributed network restoration protocols and presents a new distributed protocol called RREACT, for performing this function with a distributed algorithm which uses no prior network status or topography knowledge and which supports multiple simultaneous link restorations. Simulation results show that this protocol significantly outperforms other existing algorithms based on the Sender-Chooser approach, usually completing the total restoration in well under one second. In addition, enhancements to the basic algorithm are described which help to ensure near-optimal use of network spare channel resources and address such restoration situations where a complete restoration is possible but not achieved due to the poor choice of initial restoration path(s).

17. Key Words and Document Analysis. 17a. Descriptors

Distributed Algorithms
Network Restoration
DCS based Networks
Network Simulation
Protocol for network management
Fault Management
Real-time performance

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
		33
	20. Security Class (This Page)	22. Price