FINAL REPORT

Colorado Advanced Software Institute

NETSIM: A network simulation system for the design and analysis of network survival techniques

Principal Investigator: **C. Edward Chow**
Associate Professor
Department of Computer Science
University of Colorado at Colorado Springs

Graduate Student: **Dianne Ouderkirk**
Department of Computer Science
University of Colorado at Colorado Springs

Collaborating Company: **US West Advanced Technologies**
Dr. George I. Bell
Member of Technical Staff
Representative

Project Title:	NETSIM: A network simulation system for the design and analysis of network survival techniques
Principal Investigator:	**C. Edward Chow, Ph.D.**
University:	**University of Colorado at Colorado Springs**
Collaborating Company:	**US West Advanced Technologies**
Representative of Collaborating Company:	**George I. Bell, Ph.D.**

As authorized representative of the collaborating company, I have reviewed this report and approve it for release to the Colorado Advanced Software Institute.

Signature: _____

Date:_____

Table of Contents

Abstract

The proposed research deals with the development of algorithms and simulation tools for the design and analysis of network survival approaches. A software simulation system called NETSIM was built to facilitate the construction of network models and the fast distributed network restoration algorithms. With NETSIM, we have designed and implemented three distributed network restoration algorithms: Two Prong path-based restoration algorithm (Two Prong), One Prong path-based restoration algorithm (One Prong), and a modified version of Grover's Self-Healing Network (SHN) algorithm (modified SHN). To evaluate the efficiency of these algorithms in terms of spare capacity usage, a front end interface to Bertseka and Tseng's Relax-III code was implemented to calculate the optimal spare usage for a given link break. A graphical user interface program, called nstool, was built to edit the network topology and status. It also provides an easy to use panel to allow user to specify the restoration algorithms and network parameters for a simulation run. It helps launch the simulation and collects simulation results. Nstool is integrated with gnuplot to plot the simulation results.

With the NETSIM and the simulators that simulate link-based and path-based restoration algorithms, the user will be able to compare these different approaches under the same network assumptions. By integrating with the algorithm that generate optimal spare usage for link break cases, the user can evaluate the efficiency of these heuristic distributed algorithms in terms of spare usage.

The NETSIM system can assist the network administrators in their network planning and management tasks. Its simulation results can facilitate the network designers to improve the network reliability and efficiency.

1. Objective

The general objective of the proposed research was to develop a set of algorithms and a network software library for the design and analysis of reliable, efficient, survivable network systems that provide fast restoration of disrupted traffic, and efficient utilization of network resources. The network failures to be examined include single link failures, multiple link failures, single node failures and area failures. The objective described here was slightly different from the original proposal due to the discussion in the kick-off meeting of the project. Based on the discussion, we have decided to focus on network restoration and especially the path-based network restoration approaches.

First, we proposed to perform a literature survey for the related works on network restoration. Objective measures was to be defined and used in the comparison of the network restoration methods. The measures include at least: 1) Time required to restore disrupted traffic to a defined level, 2) Efficiency of restoration in terms of resources required for a given level of restoration, 3) Efficiency of restoration in terms of being able to accommodate different priorities of channels and different levels of restoration goals for each priority class, and 4) Stability of the restoration techniques in terms of mishandled circuits and abilities to converge to solution without overload or overshoot.

Second, we proposed to build a network model that can simulate different survivable network topology and status. Efficient network restoration algorithms were to be designed for the network model. This research was based upon the work we have been doing on path finding algorithms and distributed network restoration algorithms. The theoretical limits of these survivable network algorithms were to be investigated, and the performance of existing survivable network algorithms was to be push closer to these limits or new and better approaches to be invented. An extensible network simulation system with an enhanced graphic user interface to demonstrate and to compare different algorithms for survivable networks was to be built. The performance of the new network restoration algorithms would be compared with that of the existing network restoration algorithms. A network software library for constructing the network simulation prototype would be improved during the life of this project, which integrates the basic procedures for path finding, message sending, multicasting, broadcasting, data collection, and debugging.

The algorithms and the network simulation prototype created in the proposed research would form a technology and knowledge base to enable and facilitate network designers to design reliable, efficient, survivable networks. They would provide network administrators with efficient tools to plan or utilize more efficiently the available bandwidth in the networks and to provide reliable network services to the users.

2. Approach

The proposed project proceeded with the following three phases:

2.1 Definition Phase

Based on the survey of the general characteristics of survivable networks and our research on restoration techniques, we defined a common terminology and a set of quantitative measures of restoration performance. [Wrob90][Wu92] provides a good introduction to the problems and the solutions.

2.2 Modelling Phase

We defined a network model with the following parameterized variables:

1) Channel type and priority.
   Channels can be of different types, such as DS-3 or STS-N.
   The network model should be able to handle various priority classes.
2) working and spare channels on a span.
3) Network routes.
4) Time/methods required to detect fault.
5) Time/methods required to propagate detection to the end node of a disrupted path and other points of intelligence.
6) Times/methods required for various computational steps in finding reroute channels.
7) Upper limits on number of computational steps which can be achieved per second in reroute calculations.
8) Speeds of transmission, switching, and processing equipments.
9) Time/methods required to send the reroute channel information to network elements that are responsible for establishing the reroute channels.
10) Times/methods required to set up reroute channels.
11) Times/methods required to verify that reroutes are established.
12) Types of signalling channels among network nodes.

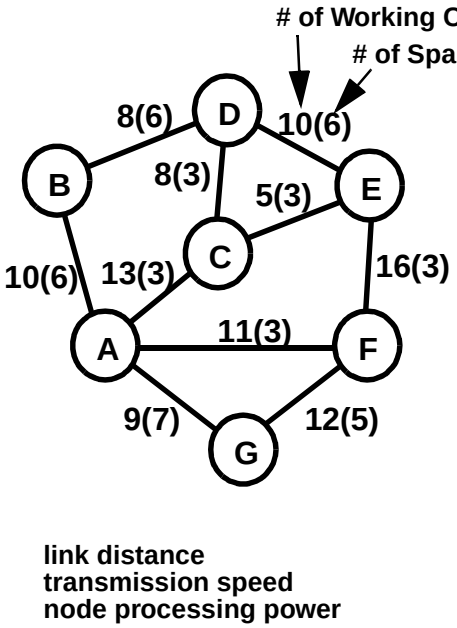
The model was constructed to study the performance of the approaches for restoring single link failures, multiple link failures, and node/area failures. A generic network file format was designed to capture the network topology and status.

2.2.1 The network model

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 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 first node in a path is called the *Start* node. The last node in the path is called the *Finish* node.

In order to accurately calculate the performance of a network restoration algorithm, we need to include link distances, transmission speeds, and node processing powers in the model. These parameters will allow us to compute the propagation delay, transmission delay, and queueing delay for the message exchanges and hence the restoration time.

The *distributed network restoration algorithms* are distributed algorithms that on detecting a network failure, initiate restoration requests, find as many restoration paths to



Traffic pattern

Path	BW	Path	BW	Path	BW	Path	BW
A-B	5	A-G	4	B-A-G	3	D-E	3
A-C	6	B-D-C	2	C-D	3	D-E-F	2
A-C-D	3	B-D	4	C-E	2	D-E-F-G	3
A-C-E	2	B-D-E	2	C-E-F	1	E-F	4
A-F	9	B-A-F	2	C-A-G	2	E-F-G	6
						F-G	3

Figure 1: An Example Network.

replace the disrupted channels, and issue the connection re-establishment commands to the involved switching nodes. The *restoration time* of a network restoration algorithm for a given network failure, is the period between the detection of the network failure to the time the last restoration path is connected to the disrupted working path. Given a network failure, the *restoration level* achieved by a network restoration algorithm is the percentage between the number of the restoration paths found by an algorithm and the number of the disrupted working channels affected by the failure. The *spare usage* of a network restoration is the total number of spare channels in the restoration paths found by a network restoration algorithm.

2.2.2 Performance metrics

The five major performance metrics we have identified for evaluating Network 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. Certain network topologies can create situations which are pitfalls for distributed algorithms using a heuristic approach to path finding or restoration path selection (see [WK90]). 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 2 illustrates this point. The vertical axis represents level of restoration and the horizontal

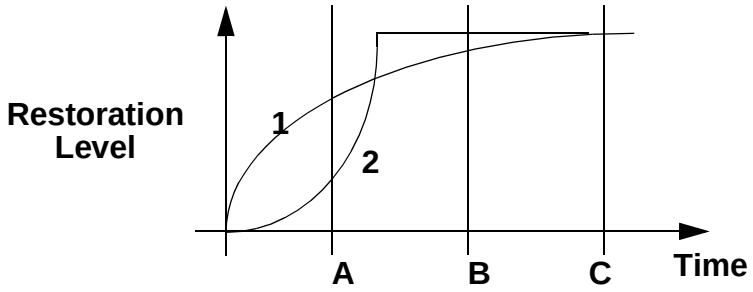


Figure 2: Selection of network restoration approaches

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. If there are priority traffic to be restored earlier, algorithm 1 will be a better choice. The selection of network restoration approaches depends on the real-time requirement, the restoration curve, and whether there is a need to support priority traffic.

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. 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.

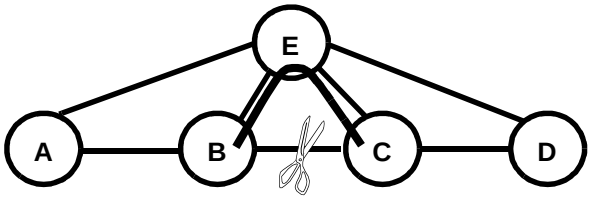
2.2.3 Link-based restoration vs. path-based restoration

There are two basic restoration approaches to restore the disrupted traffic. Figure 3 shows a simple comparison that highlights the difference between these two approaches. The link based restoration approach tries to find paths around the disrupted areas and keeps the working channels of the disrupted paths intact. The restoration is initiated by the nodes that are adjacent to the disrupted area. For a single link cut, the replacement path for the disrupted link segment will be at least two hops.

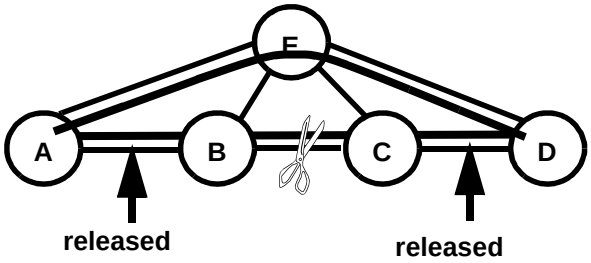
The path-based restoration approach lets the end nodes of the paths initiate the restoration process. Since it takes some time for the signals to propagate back to the end nodes of the path, therefore the initiation will be started much later, compared with the link-based approach. However, it is possible for the end nodes to find a new path that is much shorter than the original path. Hence it is more efficient in terms of spare usage. Actually, we did observe the negative spare usage in the simulation of New Jersey test network. This is due the working channels on the disrupted paths are released and can be used for restoration.

In order to have a fair comparison, the spare usage of the path-based approach is defined to be the number of spares used in the restoration paths minus the number of the released working channels. It is possible to have negative spare usage for certain network failure scenarios.

Note that the path-based restoration approach requires detailed traffic information including the individual paths and their bandwidth. The link-based restoration approach does not need those information. The path-based restoration approach can handle single



The link based restoration approach tries to find paths around the disrupted areas and keeps other working channels intact.



The path based restoration approach releases the working channels of the disrupted paths and has original sources re-establish connections.

Figure 3. link-based restoration vs. path-based restoration

link, multiple links, node and area failures, while the link-based approach cannot handle the node or area failures.

2.3 Design and Analysis Phase

2.3.1 Related work

In [Chow93a] we propose a new fast distributed algorithm based on a Two-Pronged approach. Unlike the Self-Healing Network [Grover 89] and Bellcore's FITNESS [Yang88] 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. Although the Two-Prong approach is the fastest in terms of time to restore, it does require sophisticated backtracking to achieve the higher restoration level. Further development of the Two-Prong approach should succeed in not only reducing susceptibility to racing conditions, but also improving spare channel resource utilization in the final restoration solution. One avenue to be explored is an optimization based upon exchanging sub-path connections following a backtracking episode. These exchanges would preserve the shortest possible paths and reduce connections to longer sub-paths. Such optimization procedures may, however, increase message volume during the restoration process and possibly degrade the time performance of the Two-Prong approach. Further research is needed to develop efficient optimizations and explore these related trade-offs.

In a comparison study of our current implementations of the Two-Prong and the RREACT approaches [Chow93b], we found that on the large US network test model Two-Prong generates fewer messages while on a smaller network such as the New Jersey LATA network the RREACT generates half as many messages. The messages generated by the

RREACT approach are variable length since they contain path information. The messages generated by Two-Prong are fixed length and shorter than those of RREACT. The RREACT implementation was able to achieve full restoration on all single-link failures of the New Jersey LATA network while Two-Prong fails to achieve full restoration on 6 out of 23 link failure cases. The average level of restoration in the 6 partial restoration cases is around 86%. The current implementation of Two-Prong tries to minimize the message length and the restoration table maintained in each node. One possible improvement on the Two-Prong is to include additional information in the messages to allow the intelligent backtracking mechanism to make better decisions. Determining the minimum information exchange that would allow the faster Two-Prong approach to achieve full network restoration is an interesting and challenging research issue.

The RREACT algorithm has also proven to be a very stable, robust algorithm, achieving full restoration in all the single-link failure scenarios on the New Jersey LATA network. Time performance was also excellent, outperforming the FITNESS algorithm in all but one scenario and beating the Two-Prong algorithm in six instances. Spare channel utilization is also excellent, the best of the three algorithms. Unfortunately, our research has shown that RREACT's performance begins to degrade as the network grows in size. This is due to longer message lengths and dramatically increased message volume as solution paths become longer and more numerous in the larger network. We will examine techniques to improve RREACT's performance in these larger network scenarios, such as setting time-of-life for the restoration request messages and restricting the hop counts which the messages can traverse.

Two-Prong has proven to be the fastest of all the algorithms we have implemented and tested. In fact, Two-Prong achieves restoration faster than FITNESS in all instances and is faster than RREACT in all but six instances. Good time performance remains consistent, even as network size increases. Spare channel utilization is better than FITNESS and nearly as good as RREACT. Message volume appears high in the New Jersey LATA network, but is very stable and does not grow as quickly as RREACT when network size is increased. Unfortunately, the Two-Prong algorithm has proven to be very susceptible to message racing conditions which affect its performance occasionally resulting in less than full restoration. Current implementations have restricted message content to preserve fixed length messages and to limit storage and processing requirements in the individual nodes. Such restrictions inherently require sophisticated connection and back-tracking logic for the algorithm to perform correctly in all topologies. The perfecting of this logic with these constraints has proven to be a challenging and non-trivial research effort.

In [Chow93c] we provide a detailed analysis of the basic factors that impact the performance of network restoration approaches. These factors include 1) path finding, 2) spare channel contention resolution, 3) restoration paths selection, 4) message volume control, 5) congestion control, 6) race condition control, and 7) paths re-establishment approaches. This gives us a systematic way to analyze a network survivable approach in the proposed project.

As it is indicated in [Chow93d], the existing distributed network restoration approach perform well in link restoration cases but there is a lot of room for improvement in the node and area failure cases. The major cause of poor performance in node/area restoration cases is the uncoordinated contention among the requests for different

restoration paths. Strict priority-based contention solution can guarantee the restoration of paths in sequential but it will have poor restoration time performance. The major thrust of research effort will be to explore new distributed contention resolutions approaches in the network restoration setting. Examining existing distributed contention resolutions or deadlock avoidance algorithms in the literature will be a starting point.

2.3.2 The design and analysis of modified SHN, One Prong, and Two Prong

Modified SHN was implemented by following the PASCAL pseudo code in Grover's dissertation and filled in out interpretation of the logic that is not clear specified. We then went through the debugging process to verify the simulation results and improve its performance.

Two Prong extends the idea of link based Two Prong. But with the additional traffic pattern and the need to handle the resource contention among the competing paths, the software for the protocol simulation is completely designed from scratch. We did reuse the discrete event simulation kernel from previous project.

One Prong was designed with the goal to allow multiple path restoration to coexists. We reuse most of the Two Prong code include the table management routines, but the logic is much simplified.

2.4 Detailed plan for the NETSIM system

NETSIM was built using the algorithms, software libraries, and tools provided by the Computer-Aided Network Design & Analysis Research Environment, CANDARE, which was developed at UCCS and was used successfully to develop generalized resource allocation systems. CANDARE facilitated us to construct the network models and to design the network restoration algorithms. We designed the proposed algorithms with the needed controllable parameters for modelling the message processing delays, message transmission delays, and restoration time, and for collecting the statistics of the objective measures mentioned above.

3. Results

We have designed and implemented two versions of path-based distributed network restoration algorithms. One is based on the Two Prong link-based distributed network restoration algorithm. The other is based on a one prong idea with simplified logic. We also implemented a version of Grover's Self-Healing Network and included enhancement to improve the clean up phase. A graphical user interface based on XVIEW was built to facilitate the editing of network topology and the control of the simulation runs.

3.1 Path-based Two Prong distributed network restoration algorithm

In this section the path-based Two Prong algorithm is briefly introduced.

3.1.1 Two Prong network restoration protocol

The protocol has five phases: Notification Phase, Tentative Connection Set Up Phase, Path Trace-out Phase, Path Confirmation Phase, and Clean Up Phase.

Notification Phase

On detecting the network failure, a node broadcasts the network with PACK messages that identify the paths that are affected and the relative position of the node on the path, i.e., upstream or downstream relative to the disrupted area. It also forwards FAIL

messages along the path to the end node still connected. The FAIL messages trigger the release of working channels of the disrupted paths. To reduce the number of messages and the protocol processing time, one PACK message carries up information up to 10 maximum disrupted paths. 10 was chosen based on trials on New Jersey network. Network with large number of paths and faster transmission speed may increase the number for better performance.

Based on the information in the PACK messages, each node maintains a table that keeps track of the upsteam or downstream nodes to a disrupted path.

Tentative Connection Setup Phase

When a tandem node receives PACK messages from both the upstream and downstream nodes related to a particular path, it knows there is a potential restoration path, sets up tentative connection, and sends CREQ messages to its neighbors along the potential restoration. The CREQ message will be replied with a CRACK message indicating the actual number of reserved spares. The exchange of CREQ/CRACK messages will be propagated to the end node of the path.

Path Trace-out Phase

On receiving a CREQ message, the Finish node of the path sends an ACK message upstream and reserves the bandwidth along the way.

On receiving an ACK message, the tandem node updates the table to indicate the path bandwidth is now reserved and forwards the ACK message upstream. Note that an ACK message may be split into several ACK messages with smaller bandwidth request.

In case of resource contention among paths, an ACK message may arrive and find the previous reserved spare are taken over by the higher priority path. A CONFC message will be returned to the sender of the ACK message.

Path Confirmation Phase

On receiving an ACK message, the Start node sends a CONF message along the restoration path and confirms the establishment of the restoration path. If all the bandwidth of the path is found, a “Final” flag will be set in the CONF message. The CONF message with the Final flag is called CONFF message. The reception of any further ACK message will be replied with an NAK message to tear down the tentative connections.

On receiving a CONFF message, the tandem node will release bandwidth that is not on the confirmed path and indicate in its table that the path has restored all its bandwidth. The reception of any further request for the path will be rejected. The CONFF messages are used by the end node to inform the tandem nodes to stop the bandwidth reservation process.

When the Finish node receives CONF message, part or all the bandwidth of a path is restored.

Clean up Phase

When all the bandwidth of a path is restored, a PDONE message is flooded out to speed up the release of unneeded bandwidth.

3.1.2 Lessons learned and further improvement directions

The arrival of a CONF message may trigger the release of bandwidth on some outgoing links and hence the initiation of the pending restoration requests on other paths. Multiple messages were observed to be sent over an outgoing link. One possible improvement of Two Prong is to consolidate all the operations in those messages into a single message. When the protocol processing time is the dominant factor in the total restoration time, this message consolidation effort can have a big pay-off.

One of the important protocol parameters is the retrial number of the ACK requests. Allowing the retrial along a rejected path has shown to increase the restoration level in some cases. But in other cases, it not only decreased the restoration level but also slowed down the restoration process due to the additional message volume. Further research is needed to decide how many retrials are needed and when to retry.

The hop count limit on the PACK message also impact the restoration level and restoration time. The original hypothesis, that the larger hop count should yield the higher restoration level, was proven to be false. In some of the simulation runs, we have observed that after a certain threshold on the hop count, both the restoration level and the restoration time went down. This is due to the increase of message volume and racing situations. Also the optimal hop counts are not the same for all failure cases.

3.2 Path-based One Prong distributed network restoration algorithm

One Prong was built based on the reuse of Two Prong software and implemented a much simplified logic. The protocol has only four phases: Notification Phases, Path Trace-out Phase, Path Confirmation Phase, and Clean Up Phase.

Notification Phase

On detecting the network failure, a node broadcasts the network with PACK messages. The PACK messages identify the paths that are affected and the relative position of the node on the path, i.e., upstream or downstream relative to the disrupted area. It also forwards FAIL messages along the path to the end node still connected. The FAIL messages trigger the release of working channels of the disrupted paths. To reduce the number of messages and the protocol processing time, one PACK message carries up information up to 10 maximum disrupted paths.

Based on the information in the PACK messages, each node maintains a table that keeps track of the upstream or downstream nodes to a disrupted path.

Path Trace-out Phase

On receiving a PACK message, the Finish node of the path sends an ACK message upstream and reserved the bandwidth along the way. The ACK message also carries an ACKHOP field that indicated the hop count of its traversed path. When the sum of ACKHOP count and the HOP count field on the table, which was created by the PACK message, exceeds the hop count limit, the ACK was rejected with a CONF with zero bandwidth. This cuts down quite a lot of message volume.

To avoid forwarding an ACK message to a node which is its traversed path, a TRAIL field which contains the traversed node list is included in the ACK message.

On receiving an ACK message, the tandem node updates the table indicating the path bandwidth is now reserved and forwards the ACK message upstream. Note that an ACK message may be split into several ACK messages with smaller bandwidth request.

In case of resource contention among paths, an ACK message may arrive and find that there are not enough spares to satisfy its bandwidth request. If there is no spare left, a CONF with 0 bandwidth is returned. If there are some spares left, a CONFX with the unsatisfied bandwidth of the ACK request is returned.

Path Confirmation Phase

On receiving an ACK message, the Start node of the path sends a CONF message along the restoration path and confirms the establishment of the restoration path. If all the bandwidth of the path is restored, a “Final” flag will be set in the CONF message. The CONF message with the Final flag is called CONFF message. The reception of any further ACK message will be replied with an NAK message to tear down the tentative connections.

On receiving a CONFF message, the tandem node will release bandwidth that is not on the confirmed path and indicate in its table that the path has restored all its bandwidth. The reception of any further request for the path will be rejected. The exchange of CONFF messages is a mechanism to allow the end node to inform the tandem nodes to stop the bandwidth reservation process.

When the Finish node receives CONF message, part or all the bandwidth of a path is restored.

Clean up Phase

When all the bandwidth of a path restored, a PDONE message is flooded out to speed up the release of unneeded bandwidth.

3.2.1 Lessons learned and further improvement directions

In the original Two Prong logic, a node throws away the PACK message that it has seen before. The early implementation of One Prong follows the same logic. Through the debugging of a network failure scenario which is supposed to have full restoration, we discovered that the second PACK message needs to be forwarded (not broadcast) to the sender of the first PACK message to enable the finding of additional restoration paths.

This subtle modification is an evidence of the difficulty of distributed programming. Many subtle modifications on One Prong were based on tedious and long debugging. The debugging facility that prints out the simulation events of a selected path or node proved to be instrumental to those subtle but important improvements in One Prong.

One Prong uses a very conservative approach in finding paths between the Finish node and the Start node. At any instance of time, the aggregated bandwidth in the ACK messages does not exceed that of the disrupted path. Only when the ACK message got rejected, the retrial on other outgoing links will be initiated. The reason here is to allow more paths to be concurrently restored, since in path-based approach there may be multiple paths exploring the restoration paths. A more aggressive approach is to use the flooding approach used by most the link-based approach to explore the restoration path. We have done some preliminary study of this aggressive approach and the early simulation

results seems to indicate late restoration time and lower restoration level. But without additional study, this may be premature conclusion.

The current One Prong tries to strike a balance between restoration time and restoration level. The spare usage is not a major concern. If the goal of the restoration is to have spare usage as a top priority, restoration level second, and restoration time last, then a new path-based algorithm needs to be designed. One possible design is to wait for the PACK message to arrive from all the possible paths to the Finish node before sending the first ACK message. To find out shorter path first, an ACK request follows the bread-first search pattern should be sent. This however does not guarantee the optimal spare usage as it was indicated by [WK90].

3.3 An enhanced implementation of Grover's SHN.

One of the early efforts of the project was to develop a simulator for Grover's SHN before we shifted our focus on the path-based approach. Based on the description in Grover's Ph.D. dissertation, we implemented a version of SHN and through debugging we enhanced its clean up phase operation. We observed that the old signature may chase the bandwidth just released by ACK of the same index. The virtual looping will not terminate until the old signature exceeds the hop count limit. We modified the logic to have a new sig-cancel event and allow the tandem node to keep a list of signatures that got canceled. We did find the restoration level improvement and the fast convergence in the clean up process with this new enhancement. However the problem of when to release this cancelation list remains to be an open issue.

With this implementation, it allows us to compare the major link-based restoration approaches with path-based approaches under the same network assumptions.

3.4 NETSIM simulation system

We have built a network restoration simulation system called NETSIM where the major link restoration approaches, such as the FITNESS, SHN, RREACT, and link-based Two Prong were implemented as simulators. We also implemented versions of path-based Two Prong and One Prong. These simulators were built with unified command interface where the simulation parameters can be consistent specified. A generic set of simulation parameters include transmission speed, refractive index (related to the propagation delay), message/protocol processing time, hop count, DCS cross connect time, and DCS operating mode (parallel or sequential).

A set of network file conversion utilities was built to convert other network file formats to that of NETSIM. These utilities also generate the script files with commands that simulate each of the link breaks and node failures with default simulation parameters. Simulation with different simulation parameters can be carried out by modifying these script files.

A graphical user interface called nstool was built that can edit the network topology, launch multiple simulation, collect simulation results, and display them using gnuplot.

3.5 Simulation Results

Table I shows the samples of the simulation results on several link breaks of the New Jersey Test Network. We compared the performance of a centralized link-based restoration algorithm, Path-based Two Prong and One Prong, and four link-based restoration algorithms, i.e., the link-based Two Prong, modified SHN, FITNESS, and RREACT. The

simulation parameters are 8 kbps transmission speed, 1.4 refraction index, 10 msec protocol processing time, 10 msec DCS cross connect time with the sequential operating mode.

Table 1: Comparison of Network Restoration Algorithms
Link Failure Cases

Scenario	Perf. Metric	Centralized Link-based	Two Prong Link/Path based	One Prong Path Based	modified Grover's SHN	FITNESS	RREACT
New Jersey Single Link Failure N01 - N02	Time msec Level Spares Used # of Msgs	257 100% 312 12	482/2959 100% 318/160 126/674	793 90.5% 226 313	3126 100% 312 4269	1646 100% 339 185	592 100% 312 78
New Jersey Single Link Failure N04- N05	Time msec Level Spares Used # of Msgs	127 100% 237 12	107/874 100% 204/196 89/222	940 100% 209 106	3173 100% 212 2753	1151 100% 269 30	275 100% 204 107
New Jersey Single Link Failure N08 - N11	Time msec Level Spares Used # of Msgs	205 100% 301 13	475/4836 100%/98% 215/175 133/914	786 100% 221 266	2387 93.75% 192 3177	1827 100% 239 197	739 100% 233 114

Restoration Time: centralized methods faster, link-based Two prong close.

Number of Messages: centralized methods have 8-10 times fewer messages.

Among the 23 link breaks of New Jersey test network, path based One Prong fully restored all 22 link breaks except N01-N02 break. This indicates there is still room for it to be improved. Compared with fast link-based Two Prong, in worst case, path-based Two Prong and One Prong are about 8~9 times slower. The big message volume on the modified SHN is due to the large number of spares on the New Jersey test network. The queueing delay slows down the time performance of the modified SHN.

Table 2 shows the simulation results on node restoration cases of New Jersey test networks. Here path-based Two Prong suffers due to large number of message exchanges and resource contention. The path-based One Prong with simplified logic outperforms the path-based Two Prong.

Table 3 shows the One Prong simulation results on link breaks of Bellcore test network referenced in Grover's dissertation. Here it achieves full restoration on all cases. In terms of the path length efficiency which compare spare usage with optimal solution, One Prong and Grover's SHN are about even. In four cases One Prong performs better. In the other four cases, Grover's SHN performs better. In terms of time to restoration, One prong takes about 5~7 times more time than Grover's SHN.

Table 4 shows the impact of parallel DCS operating mode on the restoration time. Table 4a shows the restoration results of One Prong on N04-N03 cut with sequential DCS operating mode and 0.01msec per DCS connection. Table 4b shows the restoration time results of One Prong on the same cut but each node is assumed to have 4 parallel DCS connection servers, each performs one DCS connection (DS3) in 0.01 msec. In some of

the link breaks, the last restoration path only have one bandwidth and the parallel DCS does not help the shorten the time to restoration number. It does shorten the restoration time those restoration paths with multiple bandwidth.

Table 2: Comparison of Network Restoration Algorithms
Node Failure Cases

Scenario	Perf. Metric	Centralized Path based	Centralized Link based	Centralized Combined	Komine*	Two Prong Path based	One Prong Path Based
New Jersey Single Node Failure N04	Time msec Level # of Msgs Spare Usage	191 100% 33 -	153 100% 28 -	167 100% 11 -	2025 100% 1647 -	3175 100% 596 61(r61)	466 100% 209 63(r61)
New Jersey Single Node Failure N05	Time msec Level # of Msgs Spare Usage	416 100% 18 -	370 90. 5% 18 -	426 100% 18 -	2337 91% 1633 -	18954 69% 5246 72(r102)	1077 78.38% 478 61(r102)

*** These are the simulation results of our implementation of Fujitsu's KOMINE network restoration algorithm.**

Table 3. One Prong restoration results on bellcore.net [Grover'89]

with 8kbps, dxct=0.01ms, ppt=0.01ms

==========

N01-N00 p=18/5/0/13 bw=5/18= 27.78% @0.730550 m=1115 s=0/10/10, N=5/5, L=10/10
N02-N01 p=13/5/0/8 bw=5/13= 38.46% @0.796250 m=847 s=0/10/10, N=5/5, L=10/10
N02-N00 p=18/7/0/11 bw=7/18= 38.89% @2.586373 m=2337 s=0/14/14, N=7/7, L=14/14
N03-N00 p=18/11/0/7 bw=11/18= 61.11% @2.203324 m=949 s=0/35/35, N=11/11, L=31/35*
N04-N00 p=13/13/0/0 bw=13/13=100.00% @2.387124 m=674 s=0/29/29, N=13/13, L=29/29
N04-N02 p=4/4/0/0 bw=4/4=100.00% @0.820675 m=224 s=0/8/8, N=4/4, L=8/8
N04-N03 p=17/17/0/0 bw=17/17=100.00% @2.013275 m=785 s=0/37/37, N=17/17, L=36/37+
N05-N00 p=14/11/0/3 bw=11/14= 78.57% @1.738950 m=1017 s=0/29/29, N=11/11, L=28/29*
N05-N04 p=12/8/0/4 bw=8/12= 66.67% @2.480748 m=702 s=0/16/16, N=8/8, L=16/16
N06-N04 p=4/3/0/1 bw=3/4= 75.00% @0.374275 m=402 s=0/6/6, N=3/3, L=6/6
N07-N00 p=13/13/0/0 bw=13/13=100.00% @2.440523 m=663 s=0/26/26, N=13/13, L=26/26+
N07-N05 p=12/11/0/1 bw=11/12= 91.67% @2.332974 m=736 s=0/31/31, N=11/11, L=27/31
N07-N04 p=20/20/0/0 bw=20/20=100.00% @2.138399 m=892 s=0/53/53, N=20/20, L=40/53*
N07-N06 p=10/2/0/8 bw=2/10= 20.00% @1.250725 m=602 s=0/4/4, N=2/2, L=4/4
N07-N03 p=15/15/0/0 bw=15/15=100.00% @2.692998 m=837 s=0/36/36, N=15/15, L=34/36+
N08-N04 p=12/12/0/0 bw=12/12=100.00% @1.300680 m=528 s=0/24/24, N=12/12, L=24/24
N08-N03 p=13/13/0/0 bw=13/13=100.00% @2.223129 m=757 s=0/34/34, N=13/13, L=34/34
N08-N07 p=14/14/0/0 bw=14/14=100.00% @1.780604 m=702 s=0/29/29, N=14/14, L=28/29*
N09-N08 p=20/2/0/18 bw=2/20= 10.00% @0.814250 m=3503 s=0/4/4, N=2/2, L=4/4
N10-N04 p=12/12/0/0 bw=12/12=100.00% @1.180575 m=540 s=0/24/24, N=12/12, L=24/24
N10-N07 p=16/16/0/0 bw=16/16=100.00% @1.929555 m=822 s=0/39/39, N=16/16, L=37/39+
N10-N08 p=16/16/0/0 bw=16/16=100.00% @2.127549 m=891 s=0/33/33, N=16/16, L=33/33
N10-N09 p=9/7/0/2 bw=7/9= 77.78% @0.986239 m=830 s=0/14/14, N=7/7, L=14/14

Here the path network efficiency and path length efficiency are represented by the numbers after "N=" and "L="

The optimal spare usages are generated by a front end interface to Bertsekas and Tseng's relax-III code.

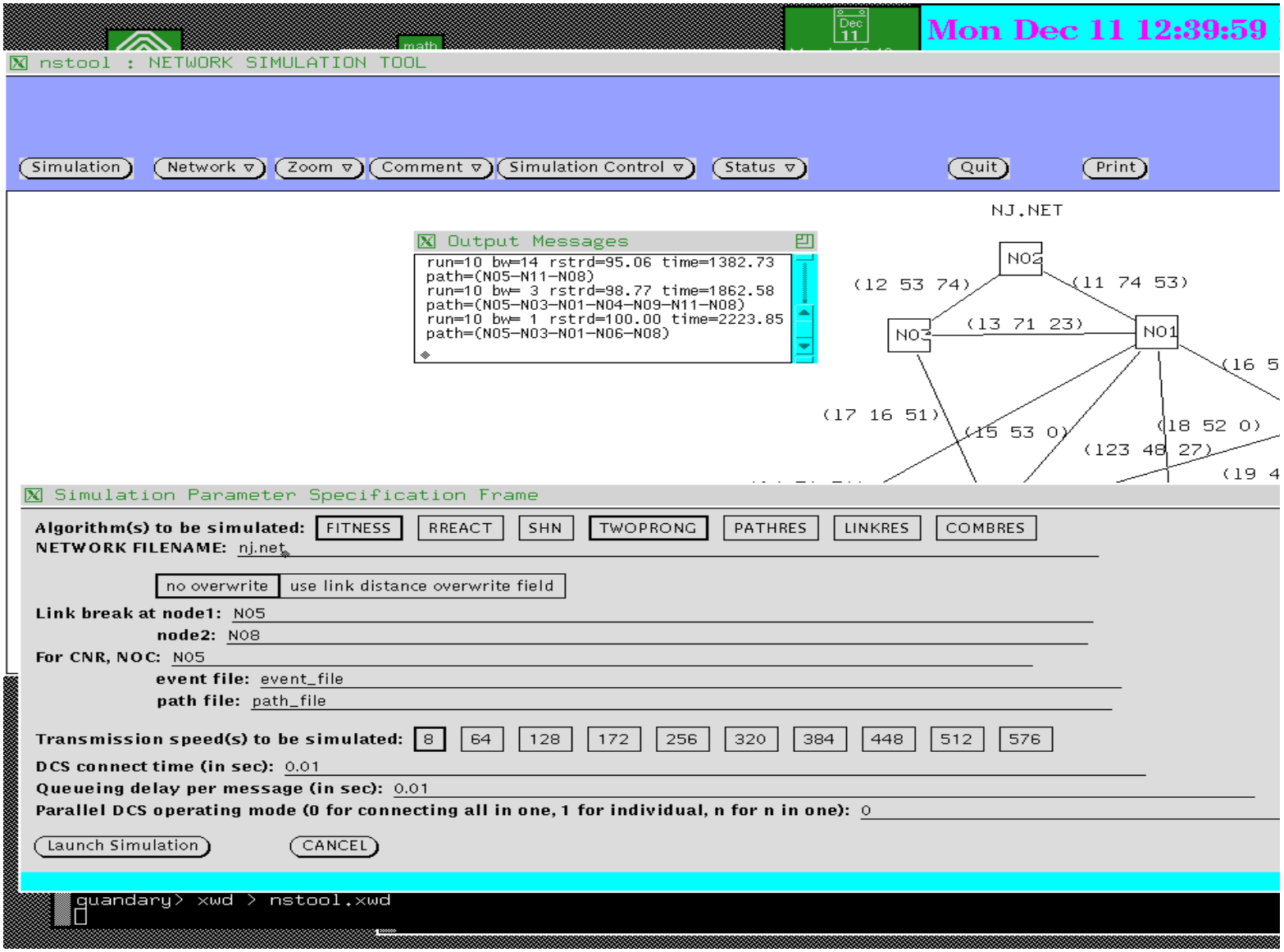
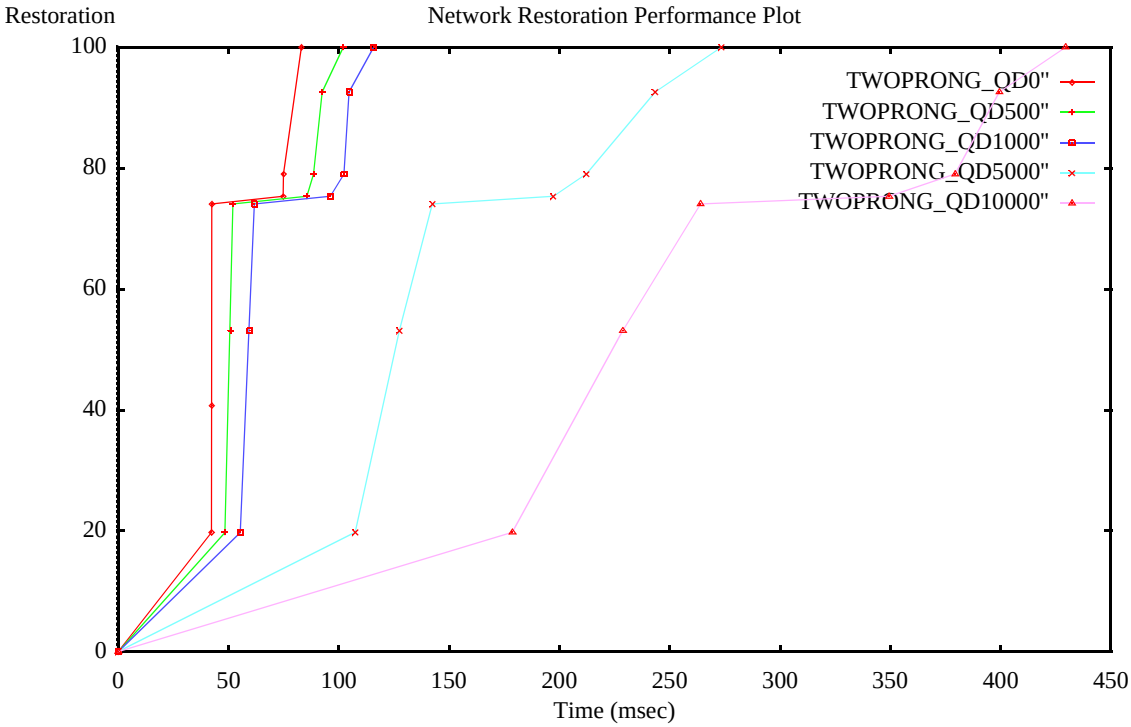


Figure 4. nstool



Generated by GNUPLOT

Figure 5. study the performance limits of network restoration approaches.

Table 4. Impact of parallel DCS operating mode

N04-N03cut on New Jersey Network, 10 msec DCS connection time.

a) restoration time with sequential operating mode

Restoration Paths

Path #	BW Restored	% Restored	Time (msec)	Restoration Path
28	4	100.00	305	N01-N00-N03
8	7	10.94	369	N03-N07-N10-N04
8	27	42.19	639	N03-N08-N10-N04
8	13	20.31	769	N03-N07-N06-N04
8	14	21.88	909	N03-N00-N02-N04
8	3	4.69	939	N03-N08-N09-N10-N07-N06-N04

b) restoration time with 4 parallel DCS connection servers

Restoration Paths

Path #	BW Restored	% Restored	Time (msec)	Restoration Path
28	4	100.00	245	N01-N00-N03
8	7	10.94	298	N03-N07-N10-N04
8	27	42.19	398	N03-N08-N10-N04
8	13	20.31	448	N03-N07-N06-N04
8	14	21.88	498	N03-N00-N02-N04
8	3	4.69	508	N03-N08-N09-N10-N07-N06-N04

Figure 4 shows the screendump of nstool with its canvas window display New Jersey test network and an output message window showing part of the simulation results. The simulation parameter specification frame allows a user to select one or several network restoration algorithms to be launched in one simulation run. It also allows the user to specify the network parameters to be specified. For the link distances between nodes, it provides option to use the link distance field in the network file or calculate the distance based on the coordinates of the two end nodes. NETSIM allows uses to specify the location of nodes based on the V&H coordinates or the longitude/latitude system.

Figure 5 shows the plotting feature of nstool. After nstool launched the simulation, it collects the simulation results which are returned via a Berkeley socket, generates the gnuplot commands and data files, and execute the gnuplot to display the simulation results. This facilitates the user to analyze the effect of changing certain network parameters. Figure 5 shows the impact of changing the message processing time from 10 msec to 0 msec on the performance of Two Prong over a specific link cut. It shows that changing the message processing time from 10 msec to 1 msec, the restoration time dramatically reduces from 430 msec to 100 msec. After that, other network parameter such as the transmission speed becomes the dominating factor.

4. Evaluation

The success of the proposed project can be assessed by the performance of the network survival algorithms generated by this project, and by the usefulness of NETSIM and the analysis report on the comparison of various network survival approaches. The performance of the proposed network survival algorithms and their comparison with the existing algorithms can be demonstrated by using NETSIM. Based on the results we discussed in Section 3, the project is quite successful since we now have a tool to compare the link-based network restoration with the path-based network restoration under the same network parameter assumptions. The NETSIM will allow network designers/planners to check the reliability of a network topology/status in terms of the restoration level and restoration time. The front end interface to the relax-III code can serve as a tool for planning the spare capacity in the network.

US West will provide feedbacks on the NETSIM usage in their research and network management organizations, and on the usefulness of the analysis report. Those feedbacks will further indicate the degree of success of the project.

5. Intellectual property developed under sponsorship of this grant.

We have designed path-based Two Prong and One Prong network restoration approach. The NETSIM software can be licensed to companies in telecommunication industry that operate or plan networks. To obtain the source code or the object code of NETSIM system, send email to chow@quandary.uccs.edu. A copy of the final report will be sent to the Larry Anderson, who is our campus Technology Transfer officer.

6. Technology Transfer section that describes the technology exchange going in both direction between UCCS and the Collaborating Company.

We have given presentations of our research results to researchers at US West Advanced Technologies twice and got valuable feedbacks from them. From them, we know how the V&H Coordinate system works. Their feedbacks encouraged us to focus on the One Prong development

in the second half of the project, since it can be used both in the network restoration and in the network provisioning. Based on the feedback, we have focused on improving the restoration level. We have achieved significant improvement on the restoration level, while maintaining, or in some cases improving the restoration time. We have delivered the NETSIM manual and software to US West Advanced Technologies.

Acknowledgment

We would like to thank Dr. Dimitri Bertsekas for showing us how to use their efficient relax-III code. We would like to thank Dr. George I. Bell and Dr. Steve Chiu for their feedback and suggestions on the project. This work can not complete with the contributions from the following graduate students: Ron Gray's work on the Gnuplot interface, Mark Bracco and Al Backmann's work on the front end interface to relax-III, John Bicknell's work on link-based Two Prong and early version of path-based Two Prong.

References

[Bert88] D. P. Bertsekas and P. Tseng, "Relaxation Methods for Minimum Cost Ordinary and Centralized Network Flow Problems," *Operations Research Journal*, Vol. 36, 1988, pp. 93-114.

[Chow93a] C.-H. E. Chow, J. Bicknell, S. McCaughey, and S. Syed, "A Fast Distributed Network Restoration Algorithm," *Proceedings of 12th International Phoenix Conference on Computers and Communications*, March 24-26, 1993, Scottsdale, Arizona.

[Chow93b] C.-H. E. Chow, S. McCaughey, and S. Syed, "RREACT: A Distributed Protocol for Rapid Restoration of Active Communication Trunks," *Proceedings of 2nd IEEE Network Management and Control Workshop*, Sept. 21-23, 1993.

[Chow93c] C.-H. E. Chow, J. Bicknell, and S. Syed, "Performance Analysis of Fast Link Restoration Algorithms," accepted to be published in 1994 on *Journal of Digital and Analog Communication Systems.* Part of the research results published in *Proceedings of Globecom 93*.

[Chow93d] C.-H. E. Chow, V. Narasimhan, and S. Syed, "Analysis of Centralized Network Restoration," *Proceedings of 2nd International Conference on Computer Communications and Networks*, June 28-30, 1993, San Diego.

[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.

[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] Wrobel, L. A., "Disaster Recovery Planning for Telecommunication," Artech House, Inc., Norwood, MA, 1990.

[Wu92] Wu, T. H., "Fiber Network Service Survivability," Artech House, Inc., Norwood, MA, 1992.

[Yang88] 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.