NAT'L INST OF STAND & TECH
A11106 979273

NBS
PUBLICATIONS

NBSIR 83-2660

A Discussion of Gridnet Algorithms and Simulation Results

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Center for Computer Systems Engineering
Washington, DC 20234

November 1982

Issued February 1983

Task Code RF
Work Unit 00065

Sponsored by
Defense Nuclear Agency
Washington, DC 20305

QC
100
.U56
83-2660
1983
C.2

NATIONAL BUREAU
OF STANDARDS
LIBRARY

MAR 21 1983

not acc-circ
QC 100
.U56
83-2660
1983
c.2

NBSIR 83-2660

A DISCUSSION OF GRIDNET ALGORITHMS AND SIMULATION RESULTS

J. A. Epstein

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Center for Computer Systems Engineering
Washington, DC 20234

November 1982

Issued February 1983

Task Code RF
Work Unit 00065

Sponsored by
Defense Nuclear Agency
Washington, DC 20305

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

TABLE OF CONTENTS

LIST OF FIGURES

A Discussion of Gridnet Algorithms and Simulation Results

J. A. Epstein

This report is an evaluation of the results of computer simulation of GRIDNET conducted during the period from 17 May 1982 to 12 November 1982.

This report describes the testing and modification of algorithms which permit messages in a GRIDNET to be routed from any source to any destination, in a network having thousands of nodes, and to accomplish this routing in an efficient manner using only limited local knowledge of network operability status. Estimates were developed for both algorithm performance and runtime efficiency. Additional studies were made concerning network connectivity, reducing packet overhead, network topology, and resolving packet overflow.

Key words: alternate routing; communications networks; distributed control; network connectivity; packet overhead; packet switching; survivability.

INTRODUCTION

The studies described in this report followed the completion of U.S. Department of Commerce contract NB80SBCA0477 by the General Electric Company, Space Systems Division, Huntsville Alabama. (5,6)

Under the above contract, General Electric (henceforth, G. E.) developed and tested algorithms for routing packets in a GRIDNET. GRIDNET is a highly survivable network containing thousands of nodes. G. E. also performed studies on network traffic loading, a subject which is beyond the scope of this paper.

The target objectives of NB80SBCA0477 included that the routing algorithms developed deliver messages in all routable networks and that the path length taken was not to exceed the minimum path length by more than 30%. G. E. delivered a heuristic algorithm in September 1981 which met these objectives for the 72 arbitrarily selected networks which were the basis for evaluation, delivering all routable messages with an average path length which was 25% greater than the minimum path length. G. E. believed that the algorithm would properly route messages in any routable network, although this later turned out not to be the case. Following completion of the contract, N. Geer of G. E., on his own initiative, delivered a non-heuristic algorithm to NBS. Further investigation showed this algorithm to be equivalent to Lee's algorithm, a classic routing algorithm used primarily for wiring design (7).

All routing simulations discussed in this report are for a 150 loop GRIDNET network, although only a portion of this network is shown in some of the graphics. This was an expansion of a 60 loop GRIDNET used for the 72 test networks.

The first part of this paper describes the studies of the routing algorithms performed from 17 May 1982 to 30 September 1982. A later section entitled "Subjects for Further Study" introduces new subjects which were developed and studied from 30 June 1982 to 12 November 1982. These topics include methods of maintaining information about inoperative stations, reducing packet overhead, resolving packet overflow, and a proposed format for the information field for an interloop packet.

A review of GRIDNET for the uninitiated precedes all other discussions. The following section, "Review of GRIDNET," is an excerpt from pp. 3-7 from the section of the same name in (3), taken with permission.

REVIEW OF GRIDNET

In 1979, a novel data communication concept called CROSSFIRE was proposed (1) for an application requiring a high degree of security and integrity. In this system, a primary station controls communications with a number of associated secondary stations using a bit-oriented, link level, protocol. Communication takes place over two loops, each carrying the same data. One loop transfers the data in a clockwise direction and the other carries the same data in a counterclockwise direction. Communication is between any of the several secondary stations and the single primary station that supervises and controls the flow of traffic on the loop. The secondary stations receive identical data that arrives at slightly different times on each of the two loops, and they regenerate the binary signals before forwarding each bit stream to the next adjacent station in the proper direction. This function is performed on all data received by a secondary station, even data addressed to it, without regard to its source. The primary station accepts, but does not regenerate and retransmit, data received on each loop from the secondary stations. Each time a primary station receives a correct copy of a message that it has previously transmitted on one of the loops it serves to confirm that the loops and all of the enroute repeaters are functioning correctly. As a result of the redundant transmission paths, the delivery of a packet to its destination on the loops cannot be prevented by cutting both loops or disabling a node at any single geographic location. The occurrence of such a cut or break can be immediately detected by the primary station since it will cease to receive one or both of the delayed copies of its outgoing transmission. Then, by polling each secondary station on the loop, and determining whether their response is on the clockwise or the counterclockwise loop, the location of the break can be localized to a region between two secondary stations. This region may consist of a single link, or it may include a node and the links that are immediately adjacent to that node.

In addition, by conducting continuous polling during the intervals between the transmission of normal traffic, the loops are never idle for a significant period of time. This aids in the almost immediate detection of any attempt to cut or jam a loop. Further protection against the injection of spurious or unauthorized data is provided by comparing the contents of the data streams that are received over the clockwise and counterclockwise loops on a bit by bit basis. This is in addition to the use of the normal cyclical redundancy frame check sequence. Collectively, these procedures make it virtually certain that any fault, outage, or adversary's action will be quickly detected.

These CROSSFIRE advantages can be extended to large networks by interconnecting a number of CROSSFIRE type systems together using gateway stations at their intersections. Multiple interconnections of the loops and adaptive routing using distributed processing provide the potential for establishing alternate routes between distant pairs of stations despite simultaneous interruptions to the continuity of multiple loops. This conceptual approach has been termed GRIDNET (2).

Network Configuration

In GRIDNET, loops are interconnected in the regular hexagonal array pattern shown in figure 1, where stations have not been shown and where loops are shown as smooth, uniform shapes. Gateway stations are located at each point where two loops touch.

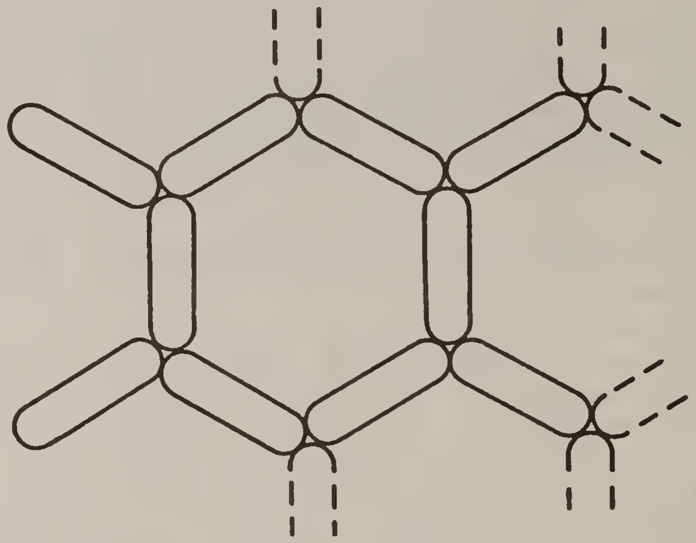


Figure 1. Loops Connected to Form GRIDNET

If this structure is flattened slightly, and the loops are collapsed and depicted as straight lines, the configuration of figure 2 results and this graphic representation provides the basis for an orderly addressing scheme.

Each GRIDNET address has three parts designated S, L, and N. The S address component represents the "station number" on a loop. The gateway station in the first quadrant on a loop is assigned the value of one for its S address component. The other stations are assigned increasingly higher S component addresses in accordance with their location on the loop as measured in a clockwise direction looking at their graphical representation on the array. The four lowest station numbers are reserved for assignment to the (up to four) gateway stations. The secondary stations are then numbered starting with the value 5. For a 12 station loop, with two secondaries between each pair of gateways, the station number sequence in the clockwise direction is 1, 5, 6, 2, 7, 8, 3, 9, 10, 4, 11, 12. The S component of the address will range from one to some maximum value as defined for the system. Typically, this maximum value will be less than 30. The L component of the address represents a "level" counting from the bottom upward on a topological graph of the system. In figure 2, the L values for the loops appear along the Y axis of the array. The final address component, N, represents a "loop number". These are shown along the X axis of the figure.

The L and N address components are not assigned values beginning with the origin of the coordinate system, and this provides room for future expansion of the network in any direction. In a similar fashion, all L,N values need not be occupied provided that lack of occupancy does not fragment the network. The origin of both L and N are such that the loops that appear as vertical interconnectors in figure 2 have even values for both L and N, while the horizontal interconnector loops have odd values for these address components. All of the address components must have integer values greater than zero. These address components are used in developing routing information for messages that flow through the network. The L and N portions of the address of a message are used to determine to which enroute loop it should be forwarded in order to reduce the distance to its destination L,N. When a message reaches its destination L,N, then the S address component designates the station on that loop to which delivery is made.

Each loop that is "interior" to the network and fully connected has four gateway stations. Peripheral loops that are only partially connected have fewer gateway stations, normally two. A gateway is logically two stations, one on each loop. Each of the two logical stations has a different address but they can communicate directly with each other by

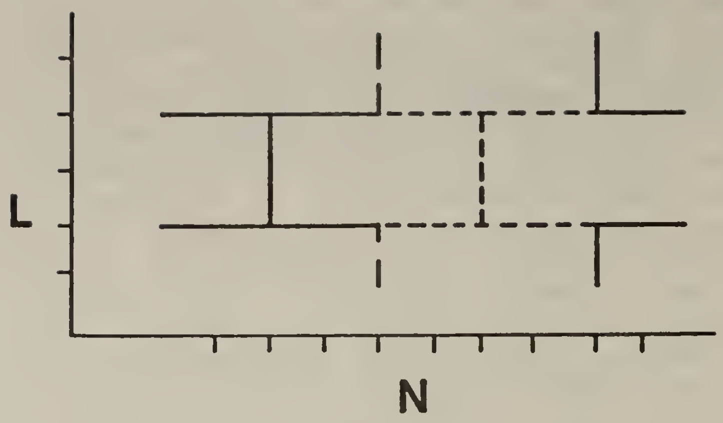


Figure 2. Graphic Representation of GRIDNET

exchanging ownership of buffers containing information that is to be transferred between the two loops. All link control, flow control and routing functions are performed by the gateways. They also perform certain of the network management functions. Each gateway keeps track of the operational status of all of the stations on its home loop and adjacent loops and of the ability of each of the second adjacent loops to communicate with their foreign loops in order to accomplish traffic routing.

The graphic representation of GRIDNET shown in figure 2 provides a basis for identifying three types of loops based on the orientation of their connections to adjacent loops. The Type I, II, and III loops are shown in figure 3 with their respective gate numbers. The connecting adjacent loops are given as relative L and N values. The connecting loop gateway station number is also shown. These gateway station numbers are used in the routing procedure.

DISCUSSION OF ROUTING ALGORITHMS

After both a routing heuristic (from DOC contract #NB80SBCA0477) and a routing algorithm, based on Lee's

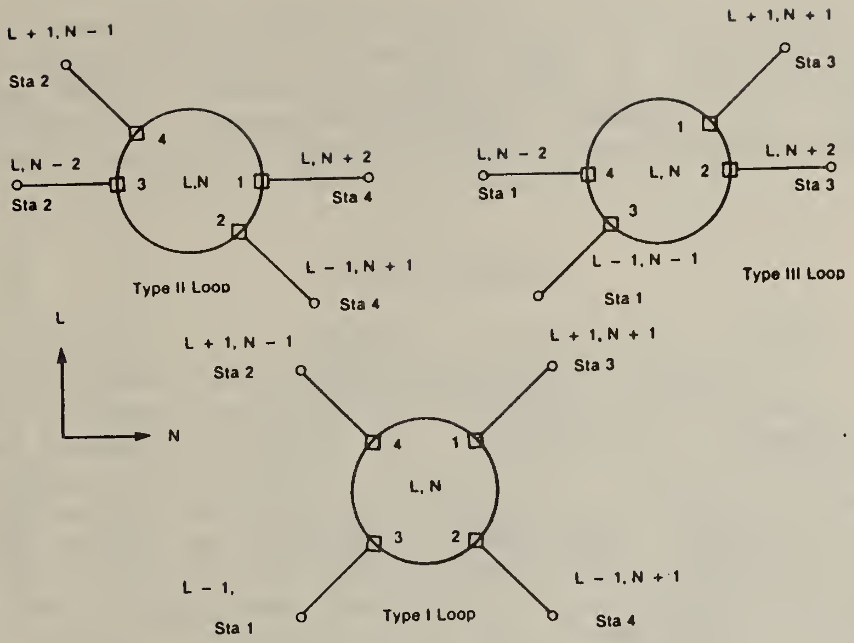


Figure 3. Loop Types and Gateway Station Numbers

Algorithm (7) (submitted later by N. Geer of G. E. after completion of G. E.'s contract) were available, there was some question as to which was superior. For the networks tested by G. E., both delivered all routable messages, which was one of the goals of the GRIDNET specifications. On those networks, the heuristic delivered the messages more efficiently by traversing fewer loops, but its runtime was considerably longer.

It was decided to perform a statistical analysis on the routers, to determine both relative and absolute performance. In addition, a third router was proposed combining the other two by making an initial heuristic computation to guess which of the original routers should be used. A software package was developed which tested the three routers on the same random networks. After running this software package on a few hundred randomly created networks, a sophisticated network was created which caused the original heuristic to fail. Further investigation showed that no fine tuning of this heuristic could be used to guarantee delivery [Appendices A,B]. So Lee's algorithm was, for the present, the only available choice. Examples of Lee's algorithm appear later in the text.

SIMULATION RESULTS

Since the results of the simulations done by G. E. relied only on 72 test networks, it was decided to do a more in-depth study of algorithm performance, based upon a large number of randomly created networks. In addition, it was decided to perform a limited study of network survivability.

Connectivity

A major objective of GRIDNET is to provide the capability to transfer a message between two stations in the network if any path between those stations exists. Thus, it is of interest to develop estimates of network connectivity characteristics as increasing numbers of loops are destroyed. That is, it was decided to determine how many disjoint connected components a GRIDNET is broken into when a certain number of loops within the network are destroyed, and how many loops are fractured away from the main body (largest component) of the network. Figure 4 shows the graphs produced by running this experiment on the 150 loop network with 200 random networks for each number of missing loops (1-150) for a total of 30,000 networks. Each random network was created by individually removing a fixed number of randomly selected loops from an originally intact network.

The graph shows that between 25 and 30 missing loops the number of loops being fractured away from the main network accelerates rapidly. That is, when 25 or 30 loops (about 20%) are missing, the network begins to fragment.

Random Networks

In GRIDNET, each gateway has knowledge of its "double adjacency neighborhood." This means that connecting CROSSFIRE loops share information about the status of the loops to which they are connected, out to a level of two loops. Each packet passing through a gateway acquires that local knowledge. This allows a gateway to make a routing decision based upon the information contained in a packet. This mechanism has been implemented in various ways for the computer simulations. Each routing decision is based only upon the information contained in the current packet, and the knowledge of the gateway's local neighbors.

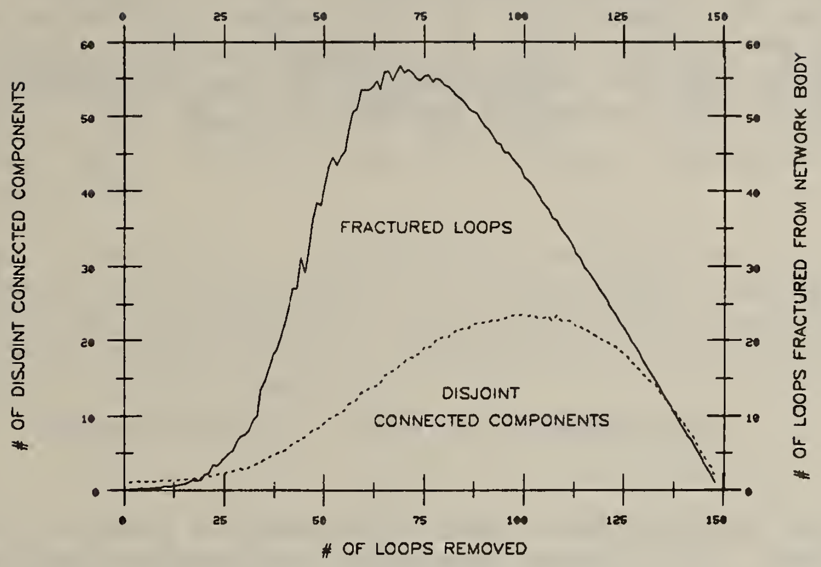


Figure 4. Connectivity of fractured GRIDNETs

After it was determined that the heuristic delivered by G. E. could not guarantee delivery for all networks, a study was made to compare the performance of Lee's algorithm with the minimum path length. For each random network created, one source and ten destinations were tried. This is because in order to compute minimum paths for a network with n nodes, it is necessary to perform $O(n^2)$ calculations, if the destination is selected at random. By performing just twice the expected number of calculations for one destination (still $O(n^2)$) the minimum paths from the source to all nodes in the network can be determined.

One thousand different random networks were created as above, except that the number of loops removed was also selected randomly. For each of these a single random source and ten random destinations were randomly selected to provide a total of ten thousand network configurations. Twenty-six percent were unroutable, of which 75% were unroutable at least partially due to the fact that the destination loop was missing, i.e., network fragmentation occured. In fact, the random networks in this simulation averaged 27 outages in the 150 loop network. Recall from the previous section that at around this number of outages the network begins to fragment, which makes it an interesting region to study. This is because the networks examined are unfragmented or are just beginning to fragment.

Of the routable networks, the routing algorithm delivered its message in all cases, taking an average of only 17% more loops than the minimum path. This is less than both the 30% target in the G. E. contract and the 25% observed for the networks tested by G. E. (4).

IMPROVEMENTS ON THE SIMULATION MODEL

This section describes improvements made on the simulation model to better simulate the routing operations in a GRIDNET. These improvements focused primarily upon the representation of information concerning inoperative stations and communication links, and sharing this information in a more realistic fashion.

Partial outages

Until June 1982, all routing simulation considered entire GRIDNET loops as operative or inoperative. This was acceptable as an initial simplifying assumption even though it is not consistent with the basic notion of GRIDNET as a highly survivable network, in which portions of loops can continue to function. So, it was decided to update the simulation package and routing algorithms to account for partial outages, i.e., loops which have some or all of their gateways and links between gateways inoperative.

A method has been developed to internally represent partial outages, where each loop in the network has an outage byte associated with it. The high order four bits correspond to the four gateways. The low order four bits correspond to the link between, say, gateway n and gateway n+1 (modulo 4). Where a bit is set, the corresponding gateway or link is inoperative. For example, if gateway 4 is known to be inoperative and so are both links adjacent to gateway 4, the outage byte would be 0001 1001. As another example, if all gates are operative but the link between gates 3 and 4 is broken, the outage byte would be 0000 0010. It is important to note that the outage byte for a loop which is (at least thought to be) fully intact is 0000 0000.

In the graphic representation used in the simulation model, a gateway outage is represented by a single hash mark in the appropriate position on the loop. A link outage is represented by a double hash mark in the appropriate position on the loop.

In this simulation, eliminating one half of a gateway is considered to be equivalent to eliminating the connection between the two halves of the gateway, and the CROSSFIRE links between the eliminated gateway half and the two adjacent gateways on the same loop.

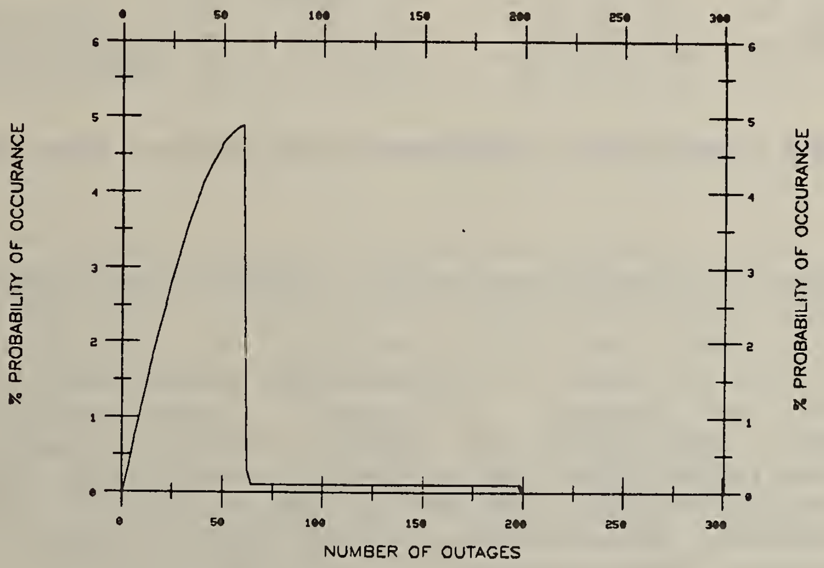
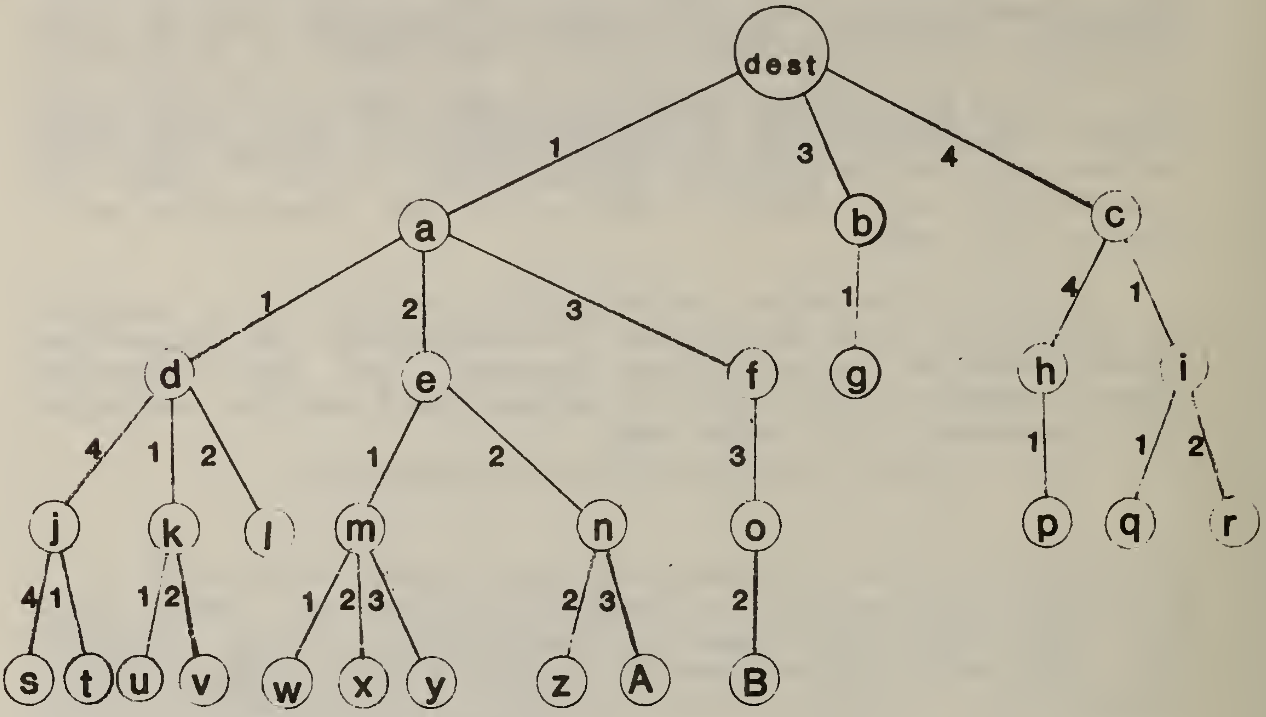


Figure 5. Distribution for outage selection for new simulation

Gate outages are selected for the random networks with the probabilities shown in figure 5. Link outages which are not part of gate outages are introduced independantly according to the same distribution. The average (expected value) occurs at about 42 outage items.



Edge markings indicate gate taken from parent node

Figure 6. Tree illustrating one-way Lee's algorithm

The tree in figure 6 illustrates the routing procedure used for the network in figure 7. Each node in the tree represents a gateway in the network which has been included as a possible link in a path. The algorithm makes a breadth-first-search[1] from the destination in all directions until it reaches the current loop or until all possibilities have been exhausted. If the latter is true, no

[1] Starting at vertex v and marking it as visited, in a breadth-first-search all unvisited vertices adjacent to v are visited next. Then unvisited vertices adjacent to these vertices are visited and so on. This is analogous to an ever widening circle, similar to a wave caused by throwing a stone into water.

path exists from the current loop to the destination, so an attempt is made to find a path back to the source by swapping the source with the destination and running the algorithm again. If neither path exists, the message is killed. This condition is not possible in the simulation, since outages do not occur dynamically in the model.

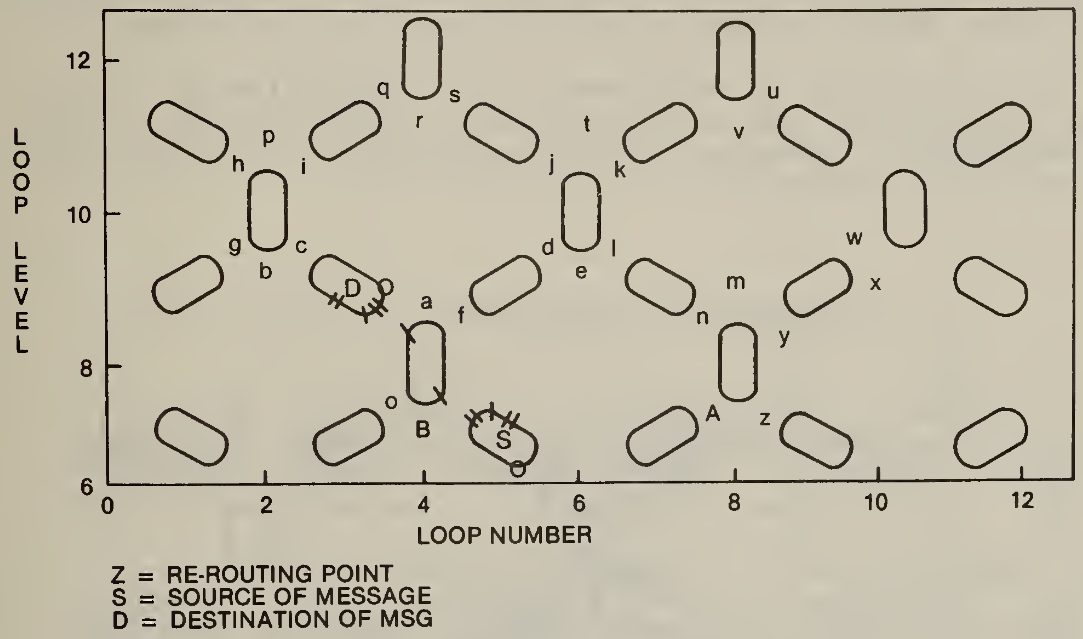


Figure 7. Network used in Fig. 6 illustration of one-way Lee's algorithm

If a path is selected, it is the shortest path available based upon current knowledge. If two or more paths are of equal length, priority is assigned to paths leading directly into the destination gate, and clockwise from that gate and other potential gates in the path.

For the example in figures 6 and 7, the path from the destination passed through the tree nodes a,f, and o, to node B, which is on the source loop. The gate routing list from the destination to the source is 1,3,3,2. The gateway numbering scheme of GRIDNET implies that the routing list from the source to the destination is 3,1,1,4.

While the simulation model based on complete loop outages sequentially searched a list to determine whether or not an entry was already in the routing table, the partial loop outage model uses a direct lookup table instead, in the interests of runtime efficiency.

Improved Integrity of Simulation

Earlier simulations assumed that each loop has knowledge of all loops within a double adjacency neighborhood of itself, regardless of their operability status. It seems unreasonable to assume this, since if a station is inoperative it cannot relay any information to its neighbors. Therefore, to improve the integrity of the new simulation it is assumed that a station does not obtain outage information whose transmission is blocked by an outage.

This condition is enforced in the simulation by internally traversing the double adjacency neighborhood to collect information, passing through only those links and gateways which are operative.

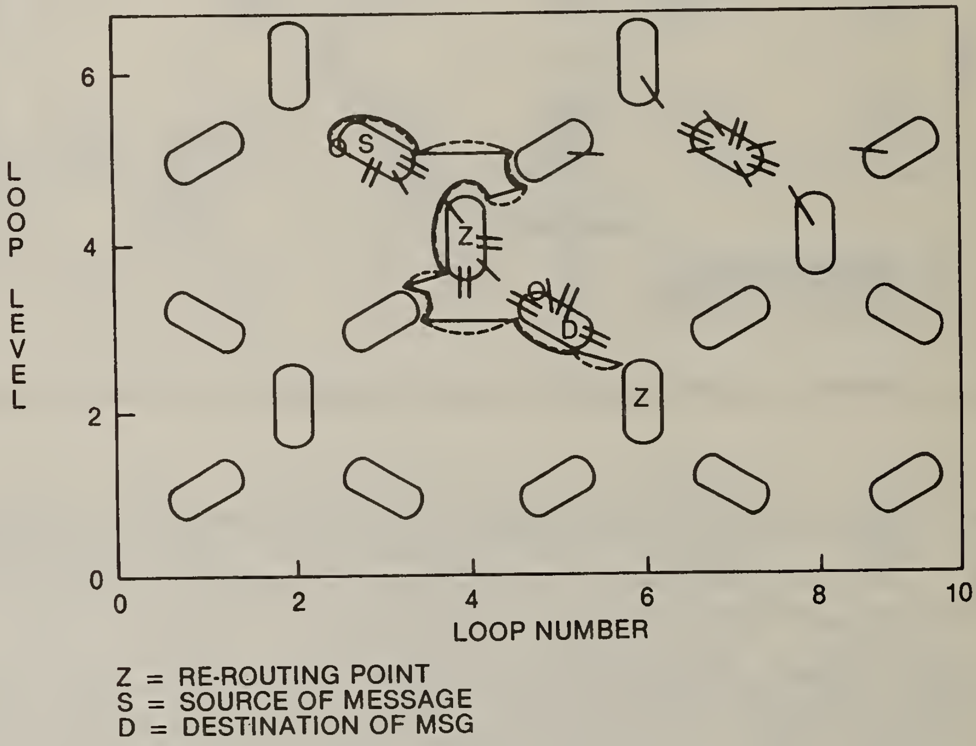


Figure 8. Illustration of integrity of new simulation

Figure 8 illustrates the improved integrity. The circles at 3,5,3 and 4,3,5 represent the source and destination gates. Although the outages on the destination loop are within a double adjacency neighborhood of the source loop, the packet is unaware of their presence until it reaches loop 4,4, since the outage transmission is blocked due to the inoperative gateway 2,4,4. From there, it attempts to route through gate 1 on the destination loop, until it reaches loop 2,6, where it learns that the link between gates 1 and 4 is broken, and returns the message back to the sender since there is no possible direction of approach remaining.

PROCESSOR REQUIREMENTS

After the simulation results showed that the routing algorithms performed well for badly fractured networks, emphasis was shifted to CPU runtime, especially since using partial outages adds to algorithm complexity.

An initial attempt was made to estimate CPU time by counting CPU instructions. This was done by looking thorough the assembly language cross-listing from an efficient version of the simulation, written in C (9). C was used largely because Fortran lacks the efficient "bit-fiddling" capability to manipulate the data representations used for partial outages. Assembly language instructions were counted, and an attempt was made to compensate for the fact that the assembler (VAX 11/780 Macro)[2] was somewhat more sophisticated than the assemblers to be used in actually implementing GRIDNET on microprocessors. Statements incrementing instruction counters were inserted at the appropriate places in the earlier simulation, which was written in Fortran.

[2] Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the either the National Bureau of Standards or the Defense Nuclear Agency, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

The findings were that about 260,000 microprocessor instructions are required to route a message from source to destination for the random networks with partial outages described in the previous section. Assuming about 3 microseconds per microprocessor instruction indicates that it requires 0.78 seconds of microprocessor CPU time to route a message from source to destination.

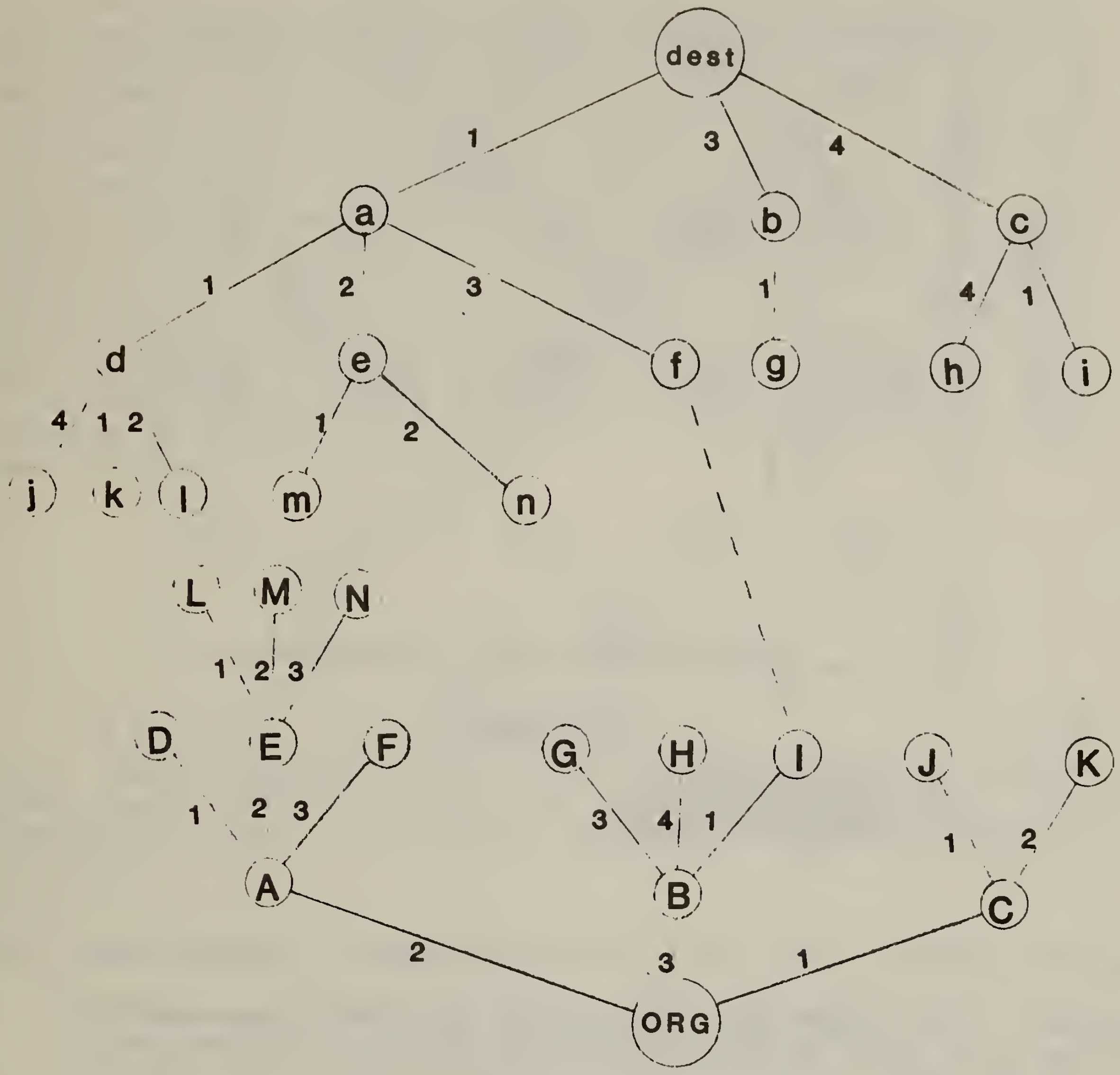
These findings did not correlate with performance as observed by waiting for large batches of random networks to be computed, and based upon knowledge of the system loading factor. The resulting time (0.78 seconds of CPU time) was much longer than expected. So, it was determined to do a more accurate study, based upon the CPU time of the VAX 11/780, as maintained by its operating system.

In addition, several adjustments were made to further improve runtime efficiency.

First, it was decided to select an initial minimum route based only upon network topology, and to only change that route using Lee's algorithm as necessary. Each gateway station maintains a compact list of the shortest paths from itself to all other loops in the network, based solely upon network topology.

Second, an improvement in Lee's algorithm, suggested in (7), was implemented. The suggestion is to perform a breadth-first-search from both the source and destination, until the two expanding regions meet, and then concatenating the two resulting paths. This provides the same quality routes as before, but, on the average, reduces the algorithm's search space.

An example of the search in both directions is given in figure 9 for the network in figure 10. The search meets at the letter I, which is the gateway pair (1,7,3),(3,8,4). The path from the source to this gateway pair passes through the tree nodes B and I. The gate routing list for this path is 3,1. The path from the destination to this gateway pair passes through the tree nodes a and f. The gateway routing list for this path is 1,3. The gate routing list for the reverse path is 1,4. Concatenation yields the gate routing list 3,1,1,4. Incidentally, for implementation in the physical model, two processors with shared memory can be used with this algorithm to further improve runtime.



Edge markings indicate gate taken from parent node

Figure 9. Trees illustrating two-way Lee's algorithm

Third, the entire simulation was rewritten in C, to increase efficiency and to produce more readable code.

The results for the random networks were that 23.4 milliseconds of VAX 11/780 CPU time were required to route the average message from source to destination. It is estimated that a microprocessor at the current level of technology is three times slower than a VAX 11/780. So, about 70 milliseconds of microprocessor CPU time would be required, which differs from the earlier result by an order of

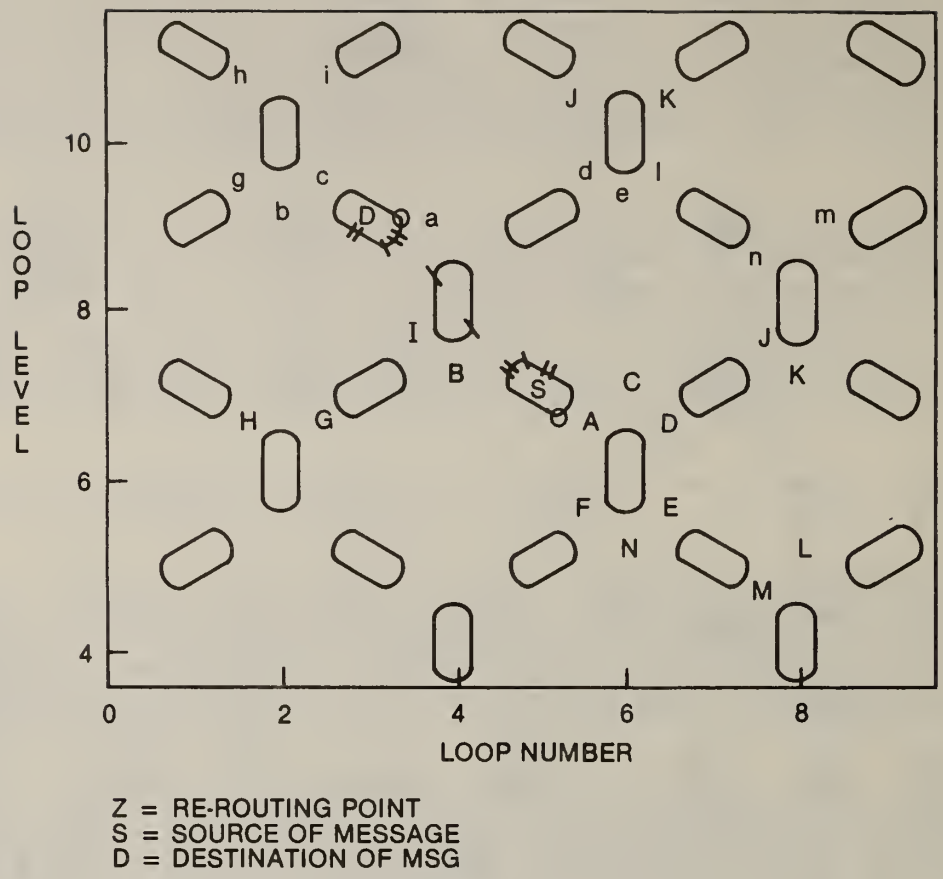


Figure 10. Network used in Fig. 9 illustration of two-way Lee's algorithm

magnitude. This change was due to the earlier inaccurate modelling and the newer, more efficient routing algorithm used.

Results indicate that only 14% of the tested networks were unroutable in the improved simulation, as compared with 26% in the earlier simulation. This is because the networks used in the new simulation are not as badly fractured as the earlier networks, due largely to the alternative intraloop routing capabilities of the GRIDNET and CROSSFIRE concepts. Of the unroutable networks, 55% had an outage at the destination gateway, indicating that a majority of unroutable networks occur due to an outage at the destination gateway, not due to network fragmentation.

In the same vein, the average path taken contained only 5% more loops than the shortest path, as compared with 17% in the earlier simulations involving only complete loop outages.

An addition factor affecting the processor requirements is the buffer size necessary to accomodate packet overhead. Information concerning loop status (outages) is referred to as "append information" since this information is appended to the information field of the packet as it is discovered. The combination of the append list and the gate routing list is referred to as the "rubber buffer," since it is the only portion of the packet which grows and shrinks once the packet is under way. The size of this rubber buffer was measured for the first time in this version of the simulation. Assuming two bits per gate in the routing list (values are 1-4), and three bytes per outage (L, N, and data) in the append list, the average rubber buffer size was 71 bytes, with a standard deviation of 37 bytes. These statistics are important in determining an optimal use of bandwidth, and choice of buffer size

SUBJECTS FOR FURTHER STUDY

This section contains discussions of new ideas which have been considered but not yet implemented in any model. Some implementation plans do exist, in particular for developing data on how well condensing append information performs.

Maintenance of Updated Routing Information

Although all simulations to date have used only the information contained in the packet, there is no inherent reason why a gateway station cannot glean information from each message which passes through the gateway, maintain it in some fashion, and use the accumulated information for future routing.

It has always been (and continues to be) assumed that a packet need not contain any information about loops which are intact. The append list is said to be complete if all loops which are within a double adjacency neighborhood of the path taken by the packet, and which also contain outages, are included in the list. Similarly, a routing list is said to be complete if it contains the entire list of gates from the origin to the current loop.

If the append and routing lists are both complete, then each gateway can compute a "certainty list." The certainty list consists of which gateways and links the message being examined has passed within a double adjacency neighborhood. This can be done by internally traversing the routing list up to this point, and using the certainty information to override the previous information contained at the gateway. This provides a mechanism for continuously updating the information contained at each gateway.

If a message is determined to be unroutable based upon the current information contained at the gateway, that information is purged and re-routing is attempted based only upon the append information contained in the packet. This provides a periodic cleansing of the outage information contained at the gateway.

Condensing Append Information

When a large number of outages occur, the append list can become large. Since it is undesirable to reserve a large portion of the packet for routing information, a method of reducing the size of the append list has been proposed.

The underlying idea is a tree structure. Consider the GRIDNET as a mesh, where each loop is a marble attached to the mesh according to the GRIDNET specifications. If the marble representing the origin loop is picked up, the entire mesh will hang from that marble like a tree. Since only those loops containing outages need to be included as information in the packet, the marbles representing loops without outages may be cut away, along with the surrounding mesh.

Now consider the fact that each GRIDNET gateway knows only about its double adjacency neighborhood. Therefore, the known append information is in a fairly narrow corridor. This is largely due to the small neighborhood, and partially because it is hoped that the path is fairly straight. This narrow corridor in turn implies that a large percentage of the loops which must be in the list have no neighboring loops (except for a tree parent) in the list.

Taking all of these factors into consideration, a data representation is suggested where each entry in the tree contains a link byte or a data byte or both. Link bytes are divided ito four two bit groupings. Based upon its position in the link byte, each grouping corresponds to one of the gate numbers 1-4. Each grouping consists of a child bit and a data bit. If the child bit is 1, the loop reached through this gateway has a link byte, which implies that one or more of its neighbors (not counting its parent) are in the tree. If the data bit is 1, the loop reached through this gateway has a non-zero data byte. By convention, if both bytes are present, the data byte precedes the link byte. Data (outage) bytes are in the same format as was mentioned in the earlier discussion concerning simulation model improvements.

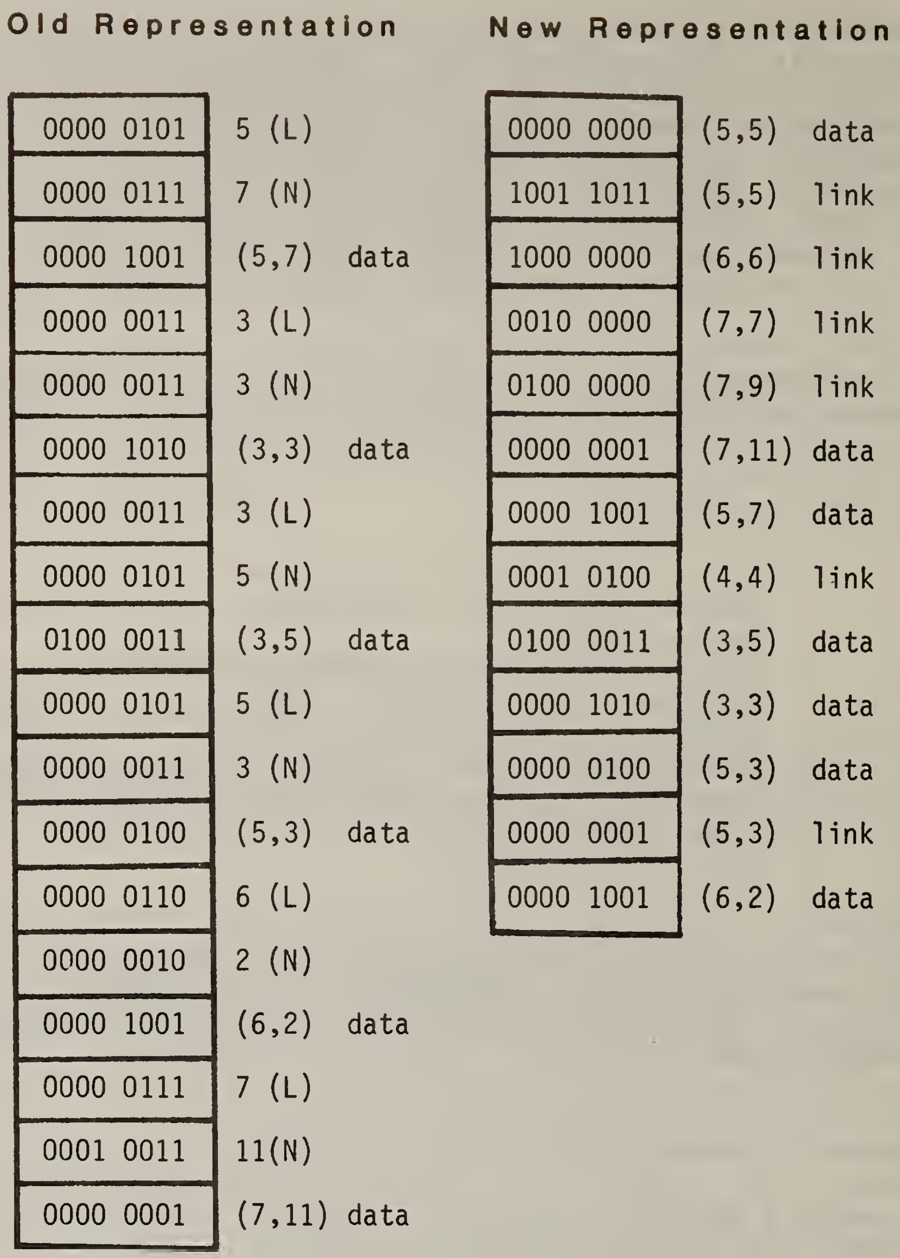
The following listing indicates the significance of each possible value of the two bit groupings in the link bytes.

Child	Data	Discussion
0	0	No link through this gateway for the tree. Child loop not contained in the tree or parent loop.
0	1	Data only. This is used for leaf nodes, i.e. entries with non-zero data but no tree children.
1	0	Child only. This is used for loops which are used only to preserve the connectivity of the tree.
1	1	Data and children.

The tree is listed in preorder, i.e., the entry for a node precedes those of its children. This ordered property allows the list to be read and written properly, without including any address fields.

Figure 11, which refers to the network in figure 12, illustrates the value of this data compression. The old method, using 3 bytes per outage, requires 18 bytes to store the information about the six partially inoperative loops, while the new method performs the same task using only 13 bytes.

It is guessed that the implementation of this tree structure would reduce the mean length of the append list by about 25% for networks similar to those used in the latest simulation. Furthermore, it is guessed that the standard deviation for the length of the append list would be reduced by about 35% for those same networks. This is due to the fact that when a large number of outages occur, the concentration of loops with outages increases. This in turn



Old Representation

0000 0101	5 (L)
0000 0111	7 (N)
0000 1001	(5,7) data
0000 0011	3 (L)
0000 0011	3 (N)
0000 1010	(3,3) data
0000 0011	3 (L)
0000 0101	5 (N)
0100 0011	(3,5) data
0000 0101	5 (L)
0000 0011	3 (N)
0000 0100	(5,3) data
0000 0110	6 (L)
0000 0010	2 (N)
0000 1001	(6,2) data
0000 0111	7 (L)
0001 0011	11(N)
0000 0001	(7,11) data

New Representation

0000 0000	(5,5) data
1001 1011	(5,5) link
1000 0000	(6,6) link
0010 0000	(7,7) link
0100 0000	(7,9) link
0000 0001	(7,11) data
0000 1001	(5,7) data
0001 0100	(4,4) link
0100 0011	(3,5) data
0000 1010	(3,3) data
0000 0100	(5,3) data
0000 0001	(5,3) link
0000 1001	(6,2) data

Figure 11. Comparison of append data representations

implies that relatively few tree nodes will be needed to preserve the connectivity of the tree.

If these guesses are reliable, consider the number of bytes which must be reserved in order to assure that packet overflow only occurs 1% of the time. For the statistics computed in the latest simulation results, 167 bytes must be reserved for append information to assure that packet

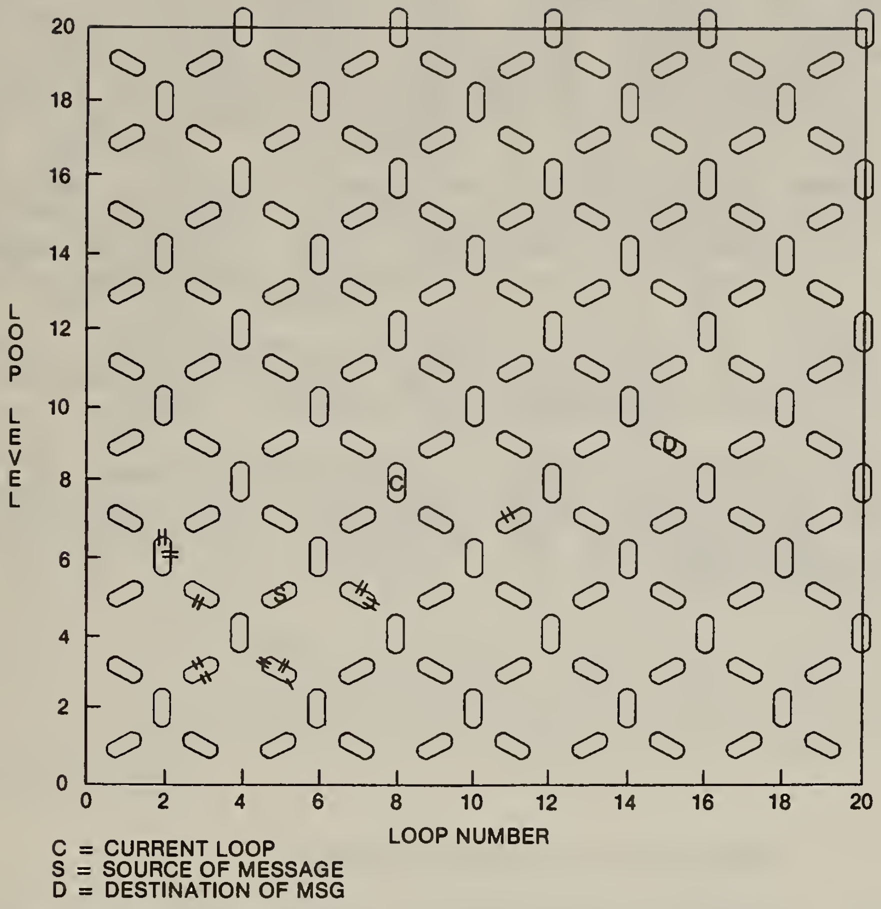


Figure 12. Sample network illustrating append list

overflow occurs only 1% of the time. If the guesses listed above are correct, the same condition can be met with the new data structure by reserving only 116 bytes.

The worst case for the tree structure occurs when a large number of entries are needed to preserve tree connectivity. If this turns out to be a major problem, the approach stated here could be modified to include an arbitrary number of trees.

It is undesirable to build this tree once per loop, so the following scheme for updating the tree to a limited degree is suggested:

The tree is obtained from the packet by the gateway, which compares its double adjacency knowledge to see whether any new nodes must be added to the tree. If there are new nodes to be added, an attempt must be made to minimize the number of network nodes which are used to preserve the connectivity of the tree. Each gateway station maintains a list of the shortest paths from itself to all other nodes in the network, based solely upon network topology. Another copy of this list can be maintained, sorted by the number of hops from the current loop to every other loop in the network. This latter list is sequentially traversed until a loop which is already in the tree is found. When this occurs, the list of gates from that loop back to the current loop is traversed. At each loop a test is made to see whether this loop is within a small neighborhood (say double) of each loop which is to be added to the tree. If this is the case, the list is modified accordingly. This approach does not always provide the smallest number of tree entries, but it should perform reasonably well.

Note that the minimum tree for the network in figure 12 requires 12 bytes rather than 13. The tree indicated in figure 11 was built on the assumption that the just-stated algorithm is used.

Resolving Packet Overflow

When a complicated routing is required, the "rubber buffer," which is carried along with the fixed protocol and the message text, may overflow the packet. This possiblity can only be averted by reserving a disproportionate amount of bandwidth for the rubber buffer.

Three methods of resolving this difficulty were considered:

The first method suggested was to discard the message text, and send the rest of the packet back to the source gate to inform it that the message had not been delivered.

The second method suggested was to split the message in half, marking each of the two messages with a binary string describing which part of the message it is, and attempt to route both messages to the destination. The receiving station, or one of the gateways on its loop, would then have to reassemble the pieces, which do not necessarily arrive in order. The problem of reassembling a message which has been split several times can be solved with a binary tree. In addition, reassembling the message at the destination requires more intelligence and memory at each secondary station than is really desirable. If reassembly is done at a gateway on the destination loop, it becomes difficult to decide which gateway and impossible to guarantee message delivery. This also introduces the requirement that there be some sort of dynamic storage allocator available.

The third method suggested was to discard some of the append information in a first-in-first-out (FIFO) fashion. This takes away the completeness property of the rubber buffer which was discussed above.

The first method was discarded since it involves too high a cost in either bandwidth or in the number of undelivered messages. The second method was discarded since it causes the problems listed earlier and can create a great deal of traffic if a message is split several times. The third suggestion has been adopted as least objectionable. A special flag can be included in the message header indicating whether append information has been destroyed, and thus instructing gateways in the routing path as to which algorithm to use in computing their own append tables. If the append data is stored using trees, then leaf nodes which are opposite in direction from the current loop, relative to the origin, are discarded first, as these data probably will not be needed again.

Network Topology

It is proposed that each gateway should have complete knowledge as to which loops are actually present within the network. This allows a gateway to make more intelligent routing decisions, independent of its append information. That is, it will not attempt to route messages into loops which do not exist, or past the edges of the network. This requires two sorts of topological information: boundaries and holes.

The requirement for the boundaries of GRIDNETs is that they be horizontally convex (thinking of L coordinates as the ordinate and N coordinates as the abscissa). This helps in the internal data representation of the append list.

Holes, or missing loops within or on the boundaries of the network, can be stored in a table with one bit per loop within the boundaries of the network.

Note that holes may occur on the boundaries of the network. This means that there is no real restriction as to the shape of the network--the property of horizontal convexity is only used to save memory at the gateway stations.

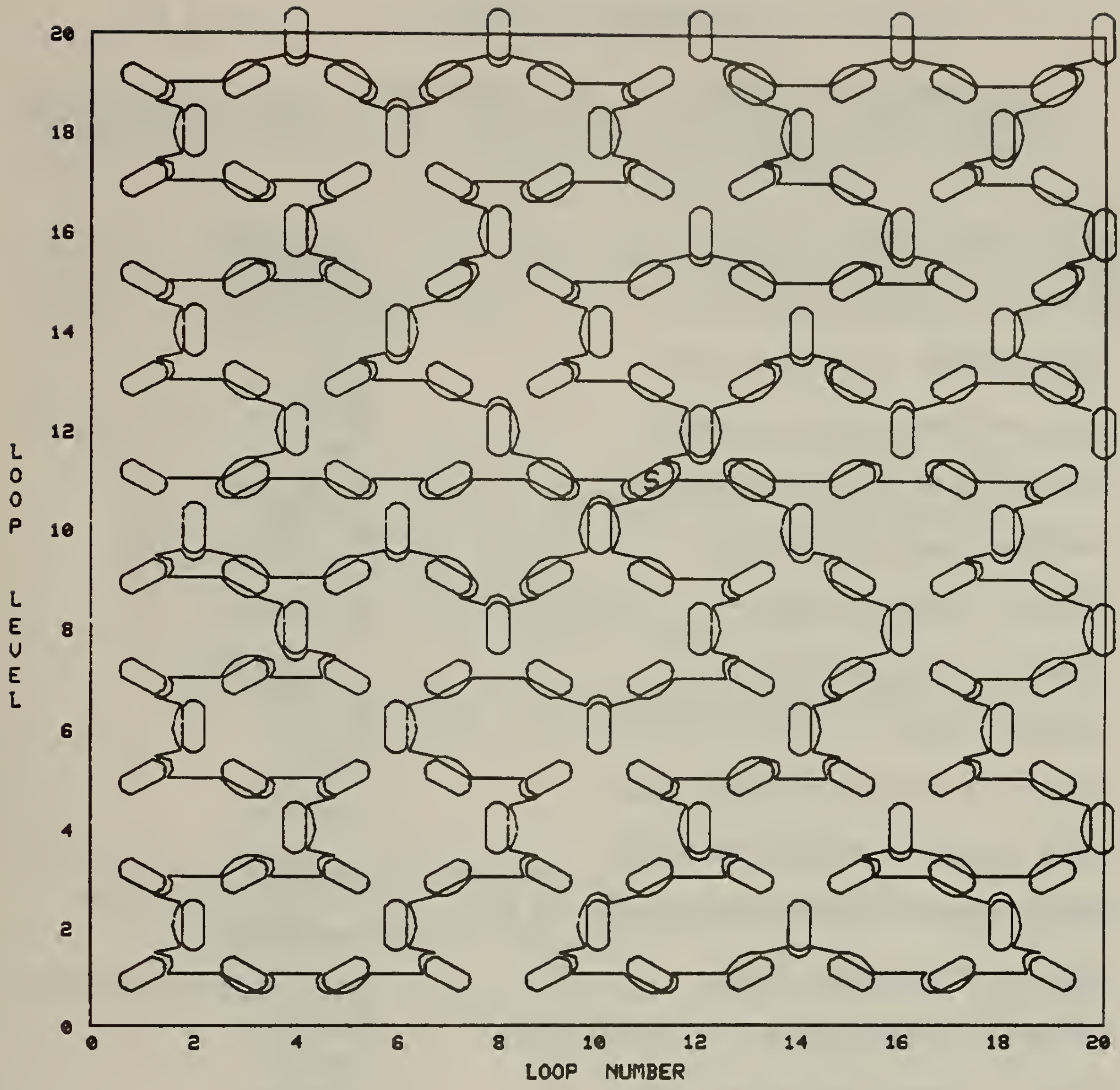
The topological information may be shared as follows:

When the system comes up, a predesignated maintenance station will know the network topology. Say that this station has GRIDNET coordinates S,L1,N1. Then it will send out four messages (one through each of the gateways on loop L1,N1) with the topological information and a pre-selected roundabout route such that each loop in the network receives the information with little or no duplication. The source and destination for each message would be S,L1,N1. The messages would have to be in a special "share message and acknowledge" mode where only single loop lookahead is used for re-routing, and re-routing must be to the next possible loop in the predesignated route. Each loop which receives the information must mark a bit in the message field which corresponds to that loop (based on the toplogical information) to acknowledge receipt.

The gateway on each loop which receives the topological information must share it with the other three gateways on its loop, using its local CROSSFIRE links. If local CROSSFIRE routing is not possible, the gateway must try to send the information the long way, through the network.

This approach is believed to be better than spreading the topological information in all directions, since it informs the maintenance station as to which loops can be contacted. It also creates considerably less traffic, although that would not be a major problem when the network first comes up.

Figure 13 shows a possible set of routes for the concept suggested above. The routes for the four messages were arrived at by hand, since, to date, limited attempts to develop a good algorithm for performing the walk have failed.



S - SOURCE OF MESSAGE

Figure 13. Example of a partition for sharing topological information

If a gateway station comes up after the rest of the network, it can begin querying its neighbors on the topology after certain of its timers have expired.

Content of Information Field

A suggested format for the information field (4) for an interloop message follows, based upon the topics discussed throughout this paper.

Size of packet in bytes	(2 bytes) {PKTSIZ}
Size of message in bytes	(2 bytes) {MSGSIZ}
Recommended delivery gate	(2 bits) {RDG}
Destination station number	(6 bits) {SD}
Destination L,N coordinates	(2 bytes) {LD,ND}
Recommended return gate	(2 bits) {RRG}
Origin station number	(6 bits) {SO}
Origin L,N coordinates	(2 bytes) {LO,NO}
Origin flag	(1 bit) {ORGF}
End-to-end acknowledge flag	(1 bit) {ETEACK}
Some appends destroyed flag	(1 bit) {SAPPDF}
Share message and acknowledge flag	(1 bit) {SMACKF}
Return to sender flag	(1 bit) {RTSF}
Gateways acceptable for delivery	(4 bits) {OKGTS}
Size of administrative data	(1 byte) {ADMSIZ}
Administrative data	(ADMSIZ bytes) {ADMIND}
Message text	(MSGSIZ bytes) {MSGTXT}
Number of append bytes	(2 bytes) {NUMAPP}
Outage information	(NUMAPP bytes) {APP}
Number of routing records	(2 bytes) {NUMRUT}
Pointer to current routing record	(2 bytes) {CURRUT}
List of gates for routing	(2 bits x NUMRUT) {RUTLST}

The following procedure is suggested for updating the packet, including new routing instructions:

```
Fetch append tree from buffer to main memory "new appends",
   building the main memory append tree simultaneously.
Fetch routing list from buffer to main memory.
Logically traverse routing list up to this point and compare
   it with the new append data, creating the certainty list.
Based upon certainty list, merge new appends with old appends.
Modify main memory old append list to include local knowledge.
Modify internal append tree.

Check routing list.
If (routing blocked) {
   Attempt to compute re-routing using main memory append list.
   If (re-routing failed) {
      Overwrite main memory appends with "new appends."
      Modify main memory append list to include local routing.
      Attempt re-routing again.
      If (new re-routing failed) {
         Swap origin with destination.
         Set RTS.
         Attempt to compute a route back to origin.
         If (routing back to origin failed) kill the message.
      }
   }
}

If (out of room in the packet) {
   Select which append data is not to be included, and remove
      it from the append tree.
}
Load the append tree and the routing list back into the packet.
```

CONCLUSIONS

The results from these simulations show that, for networks with a significant number of outages, Lee's algorithm delivers all messages in relatively few loops with acceptable end-to-end processing time.

Problems still remain in limiting the total CPU processing time, while maintaining the tree structure suggested for representing outages. Much of this time can be further reduced by the use of parallel processors. In particular, the aforementioned tree can be constructed while, in parallel, a test is performed to determine whether re-routing needs to be performed. The presence of multiple microprocessors at each gateway pair, will keep total runtime at acceptable levels.

While it is hoped that they are well founded, any simulation makes a number of assumptions about the process being modelled. Therefore, it is necessary to build a feasibility demonstration model for GRIDNET in order to properly test the ideas developed and to remove any window of uncertainty.

APPENDICES

A. Discussion of why routing heuristic cannot guarantee delivery

The heuristic algorithm developed by G. E. under contract NB80SBCA0477 belongs to a class of routing algorithms known as line-searchers. This particular algorithm attempts to create a route by always selecting the adjacent loop with the shortest Euclidean distance from the destination loop. Once a path is completed, it is optimized. To avoid moving in circles, a link between the two halves of a gateway may be traversed in each direction no more than once.

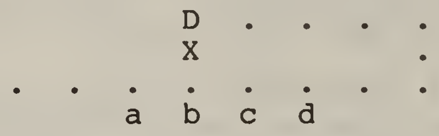
The danger of a line-search algorithm is that there is a strong possibility that two adjacent nodes (loops) will have each other as the smallest value of the measure function, which can cause a back-and-forth motion. This means that one must mark paths to make sure that a link is not traversed twice in the same direction. The problem with this, however, is that one can construct routable networks for which the router starts off in the correct direction, turns around because it learns about some outages which cause it to think that the other direction is better, only to find that the latter path is blocked. If the twice-traversed (forwards and backwards) link is the only way out, the routing will be unable to proceed despite the fact that the network is routable.

A suggested alternative was to make traversed paths "conditionally blocked," and to allow them to be tried once all other attempts have been exhausted. This will not work either, though, since one can constuct an unroutable network with the sort of back-and-forth scenario as described above.

B. Proof of why routing heuristic cannot guarantee delivery

To show that routing algorithms like the one delivered by G. E. for NB80SBCA0477 cannot guarantee delivery in a GRIDNET, consider the following:

The four nodes (loops) a,b,c, and d are connected to each other in that order. Nodes b and c have no other neighboring nodes. Node d has at least one other neighboring node, which has a path to the destination D. Node a also has at least one other neighboring node, but no path to the destination.



The attempt is to show that if b is the source, if the initial direction selected is c, and if, for some reason, node b is selected over node d, both directions of the link between nodes b and c have been traversed, so that link can never be traversed again. Since the only possible path to the destination passes through node d, routing can never be completed even though a route exists. This argument is only valid since a gateway in a GRIDNET does not have global network knowledge, and can fall into this trap due to initial ignorance.

Now it remains to show that such a situation can always come about. Consider a function f(p,D), where f is a "good" measure which becomes small as p approaches D. If f(a,D) >= f(c,D) and f(d,D) > f(b,D), the condition is satisfied. Fixing the first condition, it remains to be shown that f(d,D) > f(b,D) is possible. This condition can be imposed, based upon the definition of a "good" measure, if b is closer to D than d is, except for the outage in between.

So, since it lacks global knowledge, the routing algorithm can fail to deliver a packet in a network for which a path exists. Note that the figure above can be generalized to a GRIDNET, or many other network configurations.

REFERENCES

1. R. T. Moore, R. J. Carpenter, A. W. Holt, A. L. Koenig and R. B. J. Warnar, "Phase II Final Report Computerized Site Security Monitor and Reponse System", National Bureau of Standards, NBSIR 79-1725, PB 294 343, NTIS, Springfield, VA 22161.

2. R. T. Moore, "GRIDNET", National Bureau of Standards, NBSIR 80-2149, PB 81-144370, NTIS Springfield, VA 22161.

3. R. T. Moore, "HYBRID GRIDNET Packet and Circuit Switching in a Single Network", NBSIR 82-2588, NTIS Springfield, VA 22161.

4. R. T. Moore, "A Discussion of Gridnet Simulation Results", NBSIR 81-2414, PB 82-142894, NTIS Springfield, VA 22161.

5. H. A. Graf et al., GRIDNET Simulation Final Report, Volume 1, System Description and Results, General Electric Report No. 81HV008, 7 August 1981, PB 82-116 013, NTIS, Springfield, VA 22161.

6. H. A. Graf et al., GRIDNET Simulation Final Report, Volume 2, Model Descriptions and Operating Instructions, General Electric Report No. 81HV008, 7 August 1981, PB 82-116 013, NTIS, Springfield, VA 22161.

7. F. Rubin, "The Lee Path Connection Algorithm", IEEE Transactions on Computers, C-23, September 1974, pp. 907-914.

8. ANSI X3.66-1979, American National Standard for Advanced Data Communication Control Procedures (ADCCP), American National Standards Institute, Inc. 1430 Broadway, New York, N. Y. 10018.

9. Kernighan, B. W. and Ritchie, D. M., "The C Programming Language", Prentince-Hall Inc., Englewood Cliffs, NJ, 1978.

GLOSSARY OF TERMS

Breadth-first-search. Starting at vertex v and marking it as visited, in a Breadth-first-search all unvisited vertices adjacent to v are visited next. Then unvisited vertices adjacent to these vertices are visited and so on. This is analogous to an ever widening circle, similar to a wave caused by throwing a stone into water.

Fixed Overhead. Fixed Overhead refers to the portion of the packet which is reserved for various control functions.

Gateway. A Gateway is the facility that permits a message to be transferred from one loop to another. It consists of two stations that are each able to access a common buffer memory. One of the stations is on one loop and the other station is on an adjacent loop. Each of the stations is capable of functioning both as a primary station and as a secondary station, and each of them are considered to be half of a gateway. When a message is to be transferred from one loop to the other, the sending half-gateway places it in the common buffer memory and notifies the receiving half-gateway. The receiving half-gateway removes the message from the common buffer memory and forwards it along the path toward its destination. (4)

Horizontally Convex. Horizontally Convex describes a region for which all horizontal lines between two arbitrary points (loops) {N1,L1} and {N2,L1} always remain within the boundaries of the region. For example, region (a) is horizontally convex, but region (b) is not.

(a)

(b)

Loop. A Loop is a closed, circular, data communications configuration. In the GRIDNET, dual Loops are used and the same data is transferred around one of the pair in a counter-clockwise direction. A Primary Station is connected with one or more Secondary stations. Gateway stations interconnect Loops and a message originating at a station on one loop may be routed through several intermediate Loops before reaching its destination station on a distant Loop. (4)

Packet. In GRIDNET, a Packet is based upon the ADCCP protocol, and consists of a flag, a station address, a control field, a variable length information field which may include control functions and addresses, a frame check sequence, and a trailing flag. (8)

Primary Station. A Primary Station issues commands to the Secondary Stations(s). The Primary Station controls the movement of message traffic. In GRIDNET the role of Primary Station is rotated among the Gateways. These may number from one to four depending upon the configuration of the network. (4)

Route. Route is the path or sequence of Loops that a message may traverse in going from its source to its destination. (4)

Secondary Station. A Secondary Station responds to commands that are issued by the Primary Station. In GRIDNET, a Gateway functions as a Secondary Station whenever it is not the acting Primary Station. A Loop may have up to perhaps 20 additional Secondary Stations that are not Gateways. (4)

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET *(See instructions)*	1. PUBLICATION OR REPORT NO. NBSIR 83-2660	2. Performing Organ. Report No.	3. Publication Date February 1983

4. TITLE AND SUBTITLE

A Discussion of Gridnet Algorithms and Simulation Results

5. AUTHOR(S)

J. A. Epstein

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Interim, 5/17/82-11/12/82

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Defense Nuclear Agency
Washington, D. C. 20305

10. SUPPLEMENTARY NOTES

□ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

This report is an evaluation of the results of computer simulation of GRIDNET conducted during the period from 17 May 1982 to 12 November 1982.

This report describes the testing and modification of algorithms which permit messages in a GRIDNET to be routed from any source to any destination, in a network having thousands of nodes, and to accomplish this routing in an efficient manner using only limited local knowledge of network operability status. Estimates were developed for algorithm performance and runtime efficiency. Additional studies were made concerning network connectivity, reducing packet overhead, network topology, and resolving packet overflow.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

alternate routing; communications networks; distributed control; network connectivity; packet overhead; packet switching; survivability

13. AVAILABILITY

☒ Unlimited

□ For Official Distribution. Do Not Release to NTIS

□ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

39

15. Price

$8.50