Simulation of Hierarchical Routing Algorithms

C.V. Ramamoorthy, O. Nishiguchi, W. -T. Tsai

Computer Science Division

Department of Electrical Engineering and Computer Science

University of California

Berkeley, CA 94720

ABSTRACT

This document describes two routing simulators for computer communication networks. One is to simulate the new Arpanet routing algorithm [MCQ 80] and its extension to hierarchical networks [TSA 82, RAM 83], and the other one for the original Arpanet routing algorithm [MCQ 74] and its extension to hierarchical networks [MCQ 74, KAM 76, HER 82]. The novel features of the simulators are that they are highly parametrized so that they could be used for a variety of purposes. Specifically, the input network could be of arbitrary clustering structures, the reliability of links of networks could be arbitrarily chosen, the delay along each link could be arbitrarily predetermined according to the real network environments, and the patterns and flow rates of input data streams could be arbitrarily chosen. This report is intended to be used as a manual for using the simulators, thus many details of the algorithms will not be presented for the sake of brevity. The detailed algorithms could be found in [MCQ 74, MCQ 80, RAM 83].

1. Introduction

Two routing simulators for computer communication networks are constructed. One is to simulate the old Arpanet algorithm [MCQ 74] and its extension to the hierarchical networks [MCQ 74, KAM 76, HER 82]. The other one is to simulate the new Arpanet algorithm [MCQ 80] and its extension to the hierarchical networks [TSA 82, RAM 83]. By using proper parameters, both programs could also simulate the fixed routing schemes. The commands for these two kind of algorithms are **arpa1** and **arpa2** respectively. These two simulators are highly parametrized, they could be used for a variety purposes such as estimating the proper control parameters of the routing algorithms and determining the optimal clustering structures. Specifically, the input network could be of arbitrary clustering structures, the reliability of links of networks could be arbitrarily chosen, the delay along each link could be arbitrarily predetermined according to the real network environments, and the patterns and flow rates of input data packets could be arbitrarily chosen. Many other parameters could also be controlled by the user. This report is intended to be used as a user manual for the simulators, thus many details of the routing algorithms will not presented for the sake of brevity. The readers who are not familiar with the routing algorithms are asked to consult [MCQ 74, MCQ 80, RAM 83] first before reading this report. Currently, the simulators could handle up to 2 level hierarchies only.

We will first introduce the terminology, then the detailed protocols simulated, and assumptions in this section. Section two is the main body of this report, and it will contain the manual to use the simulators.

1.1. Terminology

In hierarchical networks, we cluster the nodes of the networks into several regions which we call first level clusters. Each first level cluster contains at least one node and does not overlap with any other first level clusters. All the nodes that are within the same first level cluster with respect to a given node or a given first level cluster are called **local nodes**. The nodes that have links to nodes in other clusters are called **Cluster Accessing Nodes (or CAN's)**. The CAN's are also called the border nodes or gateways in the literature.

1.2. Update Protocols Used in the Simulators

In the original Arpanet routing algorithm, the update packets are generated periodically and a technique called *hold down* is used to avoid the loop problem [MCQ 74]. Its simulator, **arpa1**, also uses exactly the same technique. In the new Arpanet case, the update packets are generated asynchronously and each node which receives an update packet will broadcast it to all its neighbors. The update packets are initiated only when the delays along the outgoing links are significantly different from the previous delays. This is accomplished by introducing the *threshold value*. The simulator for the new Arpanet, **arpa2**, can control this threshold value either as a constant, or a decreasing function of time. Thus, two versions of the new Arpanet routing scheme are offered. Details of the available options are in section 2.1. Furthermore, both programs could simulate fixed routing schemes if there is no link failure.

However, the handlings of link failure are different in two programs. This is true as in the original Arpanet algorithm hold down technique is used, while there is no such counterpart technique in the new Arpanet algorithm. In the original Arpanet scheme, the data packets on the failed link will have to remain in the queue until hold down is finished. Then, they are delivered to the optimal links. In the new Arpanet scheme, the data packets on the links are delivered to other optimal links at the time when nodes detect the failures or receive the news from the neighbors. Thus, **arpa1** should be used instead of **arpa2** if fixed routing scheme with link failure is desired. Because in the fixed scheme, packets which are waiting at the failed links will have to remain there until the links recover. This can be accomplished by setting the hold down period and measurement period to very large values (See 2.2 for more detail).

1.3. Assumptions

We made following assumptions for the simulators.

(1) All lines are full duplex lines of the same capacity.

(2) Update packets have higher priority over data packets.

(3) The processing time of packets at nodes is not taken into account.

(4) Except when the links fail, the transmissions of packets over lines are error free.

(5) The originating nodes of packets are always different from the destination nodes.

(6) The sizes of update packets and data packets are fixed.

(7) Retransmission of packets are not considered.

The assumptions are consistent with most the assumptions made for network simulation in the literature [RUD 76, CHU 80, CHO 81, HER 82, BIE 84].

2. Manual

2.1. Input Parameters

There are two simulation modes: the first one is the interactive mode and the second the batch mode. In the interactive mode, the user could type in the control parameters in response to the requests from the programs. While in the batch mode, it is necessary to provide a file which has all the parameters in it. Let the file name be *example*. The command to use the simulator would then be **arpa2 < example**. The user could choose the appropriate mode by setting the first parameter requested by the program. Batch mode is usually preferred for convenience.

Each node in the network will assume an unique ID, which is the serial number of the nodes. It should start with the number *1*, and end with the number *no_of_nodes_in_the_network*. For hierarchical network, each cluster should contain one or more nodes which have serial ID's.

It is the user's task to make sure that all the input parameters are consistent, otherwise unpredictable result would occur. The programs do not check the consistency of the input data. When inputing the parameters, each parameter could be separated by one or more *blanks*, or *tabs*, or *returns*. There are total twenty-seven parameters needed and they are listed below.

(1) Interactive mode indicator

Return *1* if interactive mode is desired, otherwise return *0*.

(2) Data packet delay (second)

It is important to recall that packet size is assumed to be fixed in both simulators. The data packet delay is calculated as the size of the data packet divided by the communication bandwidth of links. For example, if the capacity of the communication link is 50 kb/s, and the size of packets is 1 kb, then the data packet delay is 0.02 (= 1/50).

(3) Update packet delay (second)

The update packet delay is calculated the same way as the data packet delay except that the size of the update packet should be used instead of the data packet in the previous case.

(4) Total simulation time (second)

When the programs are started, initially there is no packet in networks. It will thus take some time for the network to be in stable condition. Both simulators are programmed so that they start to collect data after 10 seconds. Thus, it is important that the total simulation time should be sufficiently larger than 10.

(5) Measurement period (second)

Return the measurement period, which is the time interval that each node checks its delay along its outgoing links.

(6) Number of nodes in the network

(7) Number of clusters in the network

For non-hierarchical network, this value should be set to *1*.

(8) Number of CAN's in the network

For non-hierarchical network, this value should be set to *0*.

(9) Last node ID in each cluster

Return the largest node ID in each cluster.

Parameter (10) to (12) should be specified for each node.

(10) CAN indicator

Return *1* if the current node is a CAN, otherwise return *0*.

(11) Number of CAN level neighbors, and their node ID's

If the current node is not a CAN, skip this request. Otherwise first return the number of CAN level neighbors, followed by the neighbor's ID's. For example, *2 3 4* means current CAN has 2 CAN level neighbors and their node ID's are 3 and 4. Unfortunately, current version of **arpa1** can handle only one CAN level neighbor.

(12) Number of local neighbors, and their node ID's

Return the number of local neighbors, followed by their node ID's.

(13) Input traffic pattern and its density

The following options are available:

D = periodic
interval (data packet injection interval from each node)

p = poisson
lambda (the average packets per second injected from each node)

r = random
range (time between two successively injected packets is randomly distributed between 0.0 and that of *range* value.)

s = poisson with surge
lambda + percent of surge traffic
Surge traffic is decided in the following manner. Let lambda *l* and surge percent *p*, then the average packets per second from each node are

(100.0 - p) % ... l
(p / 2) % ... l * 5.0 (short period)
(p / 2) % ... l / 5.0 (long period)

For example *s 2.0 30.0* means 30.0 percent of data packets are generated in

either a very short period or in a long period, and the other 70.0 percent will be determined by the poisson distribution (average 2 packets per second will be generated).

(14) Locality (percent)

Return the percentage of the packets that should be sent to the nodes in the same cluster. Return *101.0* if the packets are sent uniformly to nodes throughout the network. Locality effects have been observed in the real network [KLE 76]. This parameter is meaningless for non-hierarchical networks.

(15) Buffer size

Return the buffer size of each link. If the buffer is full, any incoming packets will be discarded and recorded.

(16) Link failure indicator

Return *0* if the link will not fail at all, otherwise return *1*. If *0* is returned, parameter (17) to (22) should be ignored.

(17) Failure type

Return *0* if both direction of the link fail at the same time, otherwise return *1*.

(18) Unreliable link indicator

Return *s* if only specific links could fail, return *u* if all links have the same reliability. If *u* is specified, (19) should be skipped.

(19) Unreliable links specification

First return the number of unreliable links, followed by the specification of the unreliable links. For example, *2 1 3 5 6* means that 2 links are unreliable and the links are between node 1 and 3, and between node 5 and 6.

(20) Link failure pattern

Return *p* if the failures of the links are periodical, return *r* if the failures are randomly distributed.

(21) Up time (second)

Return the period of up time if *p* is returned in (20), otherwise return the average up time as *r* is specified by (20).

(22) Down time (second)

Return the period of down time if *p* is returned in (20), otherwise return the average down time as *r* is specified by (20).

The following parameters are different for **arpa1** and **arpa2**. For **arpa1**, only (23) should be specified. For **arpa2**, parameter (24) to (27) should be specified.

(23) Hold down period (second)

(24) Local level threshold value (second)

Return the threshold value of the local links.

(25) CAN level threshold value (second)

Return the threshold value of the cluster access links.

(26) CAN-cluster threshold (second)

Return the threshold value for CAN to other clusters.

(27) Threshold value control interval (second)

At each measurement time, all three kinds of threshold values are decreased by *(initial value) * (measurement period) / (interval)* seconds. If any of them reaches to zero, it is initialized again. This is the actual protocol used in the Arpanet [MCQ 80].

2.2. Examples of input parameters

This section describes some input examples. Comments are added to each parameters for readability. The comments should not be there when actually running the simulators.

(1) **arpa2**, poisson input, 3 nodes, non-hierarchical, specific links fail, fixed threshold

```
0           ;need no guidance
0.0192      ;data packet delay is 0.0192 second
0.004       ;update packet delay is 0.004 second
100.0       ;total simulation time is 100.0 seconds
0.667       ;delay is measured every 0.667 second
3           ;total number of nodes in the network is 3
1           ;non-hierarchical
0           ;therefore no CAN's
3           ;last node in the cluster is 3
0           ;node 1 is not a CAN
2  2 3      ;node 1 has 2 neighbors and their node ID's are 2 and 3
0           ;node 2 is not a CAN
2  1 3      ;node 2 has 2 neighbors and their node ID's are 1 and 3
0           ;node 3 is not a CAN
2  1 2      ;node 3 has 2 neighbors and their node ID's are 1 and 2
p           ;poisson input
2.0         ;lambda is 2.0. That means average 2 packets are generated
            ; for each second from each node.
101.0       ;this has no meaning for non-hierarchical network
100         ;buffer size is 100
1           ;link failure exist
0           ;both directions fail at the same time
s           ;specific link fails
1  2 3      ;link between node 2 and 3 would fail
r           ;link failure occurs randomly
100.0       ;mean alive time is 100.0 seconds
1.0         ;mean dead time is 1.0 second
0.1         ;threshold value for local link
0.1         ;threshold value for cluster access link
0.1         ;threshold value to other cluster
1000000.0   ;threshold value is almost fixed because this value
            ;is large enough comparing to the total simulation
            ;time
```

(2) **arpa2**, poisson with surge input, 5 nodes, hierarchical network, all links fail, decreasing threshold

```
0           ;need no guidance
```

```
0.0192          ;data packet delay is 0.0192 second
0.004           ;update packet delay is 0.004 second
100.0           ;total simulation time is 100.0 seconds
1.0             ;delay is measured every 1.0 second
5               ;total number of nodes is 5
2               ;hierarchical and 2 clusters in the network
2               ;there are 2 CAN's
2 5             ;last node ID's in each cluster are 2 and 5
1               ;node 1 is a CAN
1 3             ;node 1 has 1 CAN level neighbor and its node ID is 3
1 2             ;1 local neighbor; its ID is 2
0               ;node 2 is  not a CAN
1 1             ;1 neighbor and its ID is 1
1               ;node 3 is a CAN
1 1             ;its CAN level neighbor is 1
2 4 5           ;local level neighbors are 4 and 5
0               ;node 4 is not a CAN
2 3 5           ;node 4 has 2 neighbors and their ID's are 3 and 5
0               ;node 5 is not a CAN
2 3 4           ;node 5 has 2 neighbors and their ID's are 3 and 4
s               ;poisson with surge input
2.0             ;lambda is 2.0
30.0            ;30 % of packets are surge traffic
80.0            ;80 % of packets are local ones
100             ;buffer size is 100
1               ;link failure exist
0               ;both directions fail at the same time
u               ;all links fail
p               ;link failure occurs periodically
100.0           ;alive time is 100.0 seconds
1.0             ;dead time is 1.0 second
0.1             ;threshold value for local link
0.1             ;threshold value for cluster access link
0.1             ;threshold value to other cluster
5.0             ;threshold value decreases by one fifth at every 1.0 (measurement period) second
```

(3) **arpa1**, no link fails

```
0               ;need no guidance
0.0192          ;data packet delay is 0.0192 second
0.004           ;update packet delay is 0.004 second
100.0           ;total simulation time is 100.0 seconds
0.1             ;measurement period is 0.1 second
3               ;total number of nodes is 3
1               ;non-hierarchical
0               ;therefore no CAN's
3               ;last node in the cluster is 3
0               ;node 1 is not a CAN
2  2 3          ;node 1 has 2 neighbors and their node ID's are 2 and 3
0               ;node 2 is not a CAN
2  1 3          ;node 2 has 2 neighbors and their node ID's are 1 and 3
0               ;node 3 is not a CAN
2  1 2          ;node 3 has 2 neighbors and their node ID's are 1 and 2
p               ;poisson input
20.0            ;lambda is 20.0. That means 20 packets are generated for
```

```
                ; each second.
101.0           ;this has no meaning for non-hierarchical network
100             ;buffer size is 100
0               ;link failure doesn't exist
0.1             ;holddwon period is 0.1 second
```

(4) fixed scheme, interactive mode, unreliable link

```
                                ;unreliable network should be simulated
                                ;by arpal
Need a guidance?            1   ;interactive mode
data packet delay:          0.0192 ;
update packet delay:        0.004 ;
final time:                 100.0 ;
update increment:           1000.0 ;should be larger than final time
no of nodes:                3   ;
no of clusters:             1   ;
no of CAN's:                0   ;
last node of each cluster:  3   ;
node 1 is CAN:              0   ;
no of intra cluster neighbors: 2 ;
local neighbor ID:          2   ;
local neighbor ID:          3   ;
node 1 is CAN:              0   ;
no of intra cluster neighbors: 2 ;
local neighbor ID:          1   ;
local neighbor ID:          3   ;
node 2 is CAN:              0   ;
no of intra cluster neighbors: 2 ;
local neighbor ID:          1   ;
local neighbor ID:          2   ;
data dist. type:            p   ;
lambda:                     10.0 ;
locality:                   101.0 ;
maximum buffer size:        100 ;
link fail exist:        1   ;
fail type:                  1   ;one way of the link would fail.
uniform(u)/ specific(s): s  ;
no of unreliable link:      1   ;
unreliable link(pair of nodes) 1 2 ;link from 1 to 2 would fail
                                ;but link from 2 to 1 would not fail
periodic or random:         p   ;
up time:                    50.0 ;
down time:                  0.5 ;
hold down period:           1000.0 ;should be larger than final time
```

2.3. Output Example

Here is a sample output. First half shows the summary of input parameters, then followed by the simulation results.

```
******* STATISTICS *******

        Total number of nodes ... 10   no of cluster ... 1
```

Time between updates .... 6.6700e-01 seconds
Data distribution: POISSON lambda = 1.0000e+ 00
d_p_delay ... 1.9200e-02 u_p_delay 4.0000e-03
threshold ... 1.0000e-01 CAN threshold .. 1.0000e-01
des_CAN_threshold .. 1.0000e-01 thresh_inter .. 1.0000e+ 01
locality ... 1.0100e+ 02 total time span = 4.0010e+ 01
fail type = both way
all links fail
fail pattern = RANDOM up = 1.0000e+ 02 down = 1.0000e+ 00

total no of data packets =	297
total no of data packets dropped =	0
total no of discarded msgs =	102
total delay of data packets =	1.7149e+ 01
total number of hops by data packets =	877
total no of update packets =	8062
type1= 8062 type2= 0 type3= 0	
initiate of update:	
local= 674 CAN= 0 des_CAN= 0	
total delay of the update packets =	3.7814e+ 01
average delay of data packets =	5.7740e-02
average hops of data packets =	2.9529e+ 00
average delay of the control packets =	4.6905e-03
type1= 4.6905e-03	
channel utilization:	
local channel=	4.7131e+ 00
CAN channel=	0.0000e+ 00

3. Sample Result

Figure 2 shows a sample result of the simulators which checks the effect of hold down period. Network structure of this example is shown at figure 1. Other parameters which are used in this example are in section 2.2 example (3). Each simulation takes about 5 minutes of CPU time on a VAX 11/780.

4. Acknowledgement

We gratefully acknowledge Dr. Carl Davis, Mr. Charles Graff, for numerous helpful discussion and comments. This work was sponsored by the Ballistic Missile Advanced Technology Center under grant DASG-60-81C-0025; U.S. Army under grant DAAG-29-79-C-0171; and Defense Advance Research Projects Agency (DOD), Arpa Order No. 4031, Monitored by Naval Electronic System Command under Contract No. N00039-C-0235. The chart 1 was prepared by G. W. Goddard, and was taken from his master report.

5. References

[BIE 84] Biersack, E. W., "Performance Evaluation of Distributed Routing Algorithms," 1st Int'l Conf. on Computers and Applications, Jun. 1984.

[CHO 81] Chou, W., et al., "The Need for Adaptive Routing in the Chaotic and Unbalanced Traffic Environment," IEEE Transactions on Communications, Apr. 1981.

[CHU 80] Chu, W., et al., "A Hierarchical Routing and Flow Control Policy (HRFC) for Packet Switched Networks," IEEE Transactions on Computers, Nov. 1980.

[GOD 83] Goddard, G. W., "Comparison Study of Adaptive Routing Algorithms," MS Report, Computer Science Division, Dept. of EECS, University of California, Berkeley, California, 94720. Sep. 1983.

[HER 82] Heritsch, R. R., "A Distributed Routing Protocol for a Packet Radio Network," S.M. Thesis, Naval Postgraduate School, March 1982.

[KAM 76] Kamoun, F., "Design Consideration for Large Computer Communication Network," UCLA-ENG-7642, 1976.

[KLE 76] Kleinrock, L., *Queueing Systems, Vol. 2, Computer Applications*, John Wiley, 1976.

[MCQ 74] McQuillan, J., "Adaptive Routing Algorithms for Distributed Computer Networks," Bolt Beranek and Newman Inc., Cambridge, MA, Report No. 2831, May 1974.

[MCQ 80] McQuillan, J., et al., "The New Routing Algorithm for the Arpanet," IEEE Transactions on Communications, May 1980.

[RAM 83] Ramamoorthy, C. V., and Tsai, W. -T., "An Adaptive Hierarchical Routing Algorithm," COMPSAC 83, Nov. 1983.

[RUD 76] Rudin, H., "On Routing and Delta Routing: A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks," IEEE Transactions on Communication, Jan. 1976.

[TSA 82] Tsai, W. -T., "Routing Techniques for Dynamic Computer Networks," MS Report, Computer Science Division, Dept. of EECS, University of California, Berkeley, California, 94720. Aug. 1982.

6. Appendix A: Description of the Simulators

The simulators run on UNIX (Trademark of Bell laboratories) on VAX 11/780's. They are totally written in programming language C and consist of about 8000 source codes. The programs are completely transportable to any machines that support programming language C. They require 123 kilobytes of static memory and about 70 kilobytes of dynamic memory to run. If more memory is needed, the program will automatically request more memory space from the operating system.

Chart 1 shows the overall view of simulation programs. The first step, is to read the input and initialize the data structures and global variables. Some initial events must be put in the event list to get things started. For each node in the network an event to generate a data packet is placed in the event list. The time that each event to occur is determined by the data packet distribution function as specified by the user. The simulator then runs by selecting successive events on the event list and perform the indicated actions until it reaches to the final time.

7. Appendix B: Short History

This simulation project started in January, 1983, when W. -T. Tsai designed the program and data structures. Four months later, G. W. Goddard implemented the old Arpanet simulator. Later, he implemented the hierarchical version of that. At the same time, J. Feng and B. Lin implemented the new Arpanet simulator concurrently with Goddard. In Sept. 1983, O. Nishiguchi started to evolve the new Arpanet routing simulator to work for hierarchical networks. Finally, he

added link failure modules to both **arpa1** and **arpa2** programs.

8. Appendix C: Correctness of the Programs

Since the programs are fairly large (about 8K lines) and complex, the correctness of the programs is not easily ensured. The following techniques are used to check the correctness of the programs:

(1) Comparing the results with other simulators: we have compared our results with those simulation results in [BIE 84]. He has constructed his own simulation programs and the assumptions are almost the same with ours. However, he assumes that the packet sizes are distributed according the poisson process, and in our case the packet sizes are the same. Even with such difference, the results obtained are remarkably close to each other when other parameters are the same.

(2) Comparing the results of two simulators: since both programs could simulate the fixed routing schemes, the results obtained from both programs should be consistent. This is indeed the case for the two programs.

(3) Extensive debugging: we used extensive test cases for debugging.

(4) Independent inspections: both programs were inspected by at least two people independently.

(5) Extensive built-in fault tolerant checking: many built-in checking statements as sanity check are inserted throughout the entire programs. If the data does not satisfy any of the checking conditions, the programs would exit abruptly. The checking statements are inserted at the beginning of each module as well as after any important operations on the data.

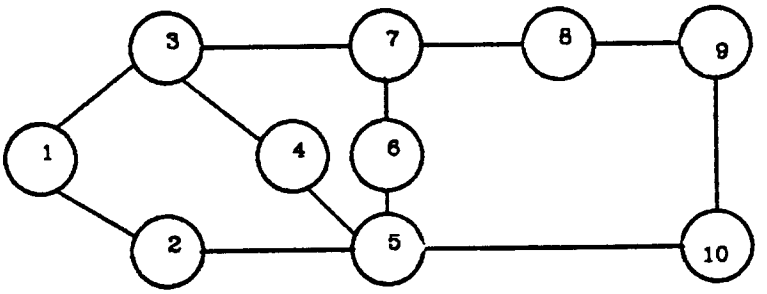
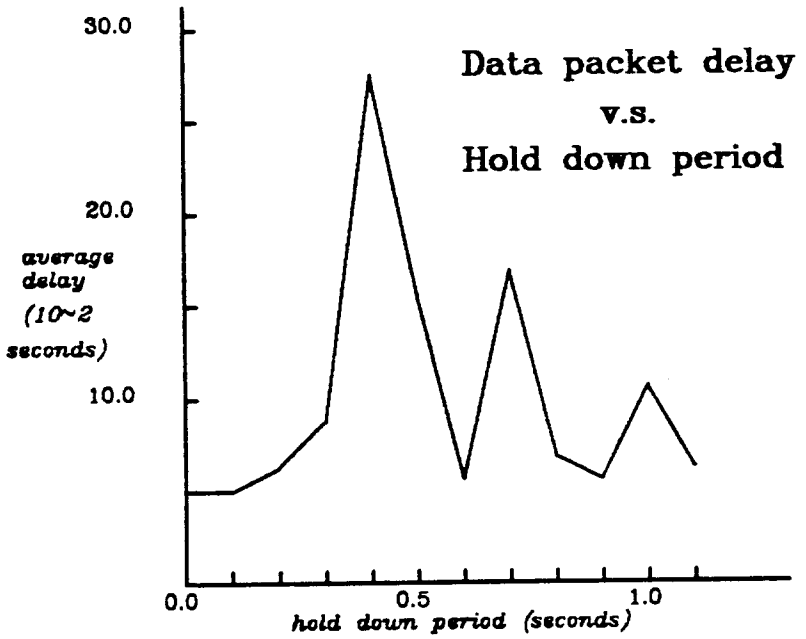


figure 1. Network Structure



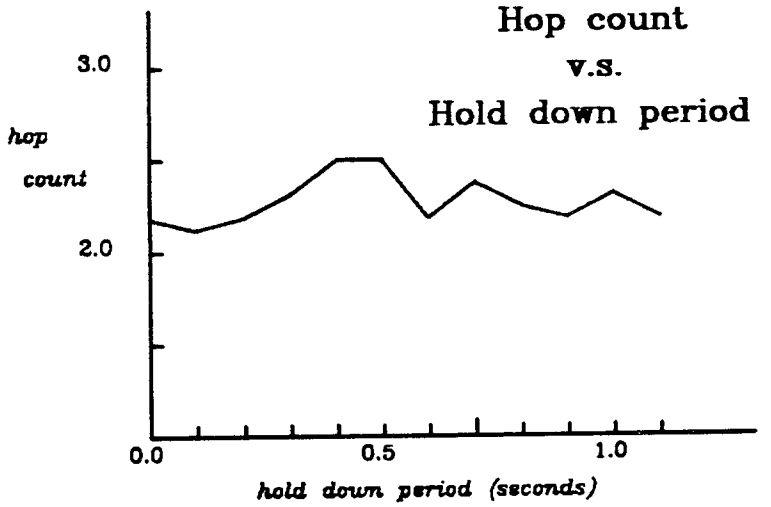


figure 2. Effect of Hold down period

CONTROL FLOW FOR ROUTING SIMULATION

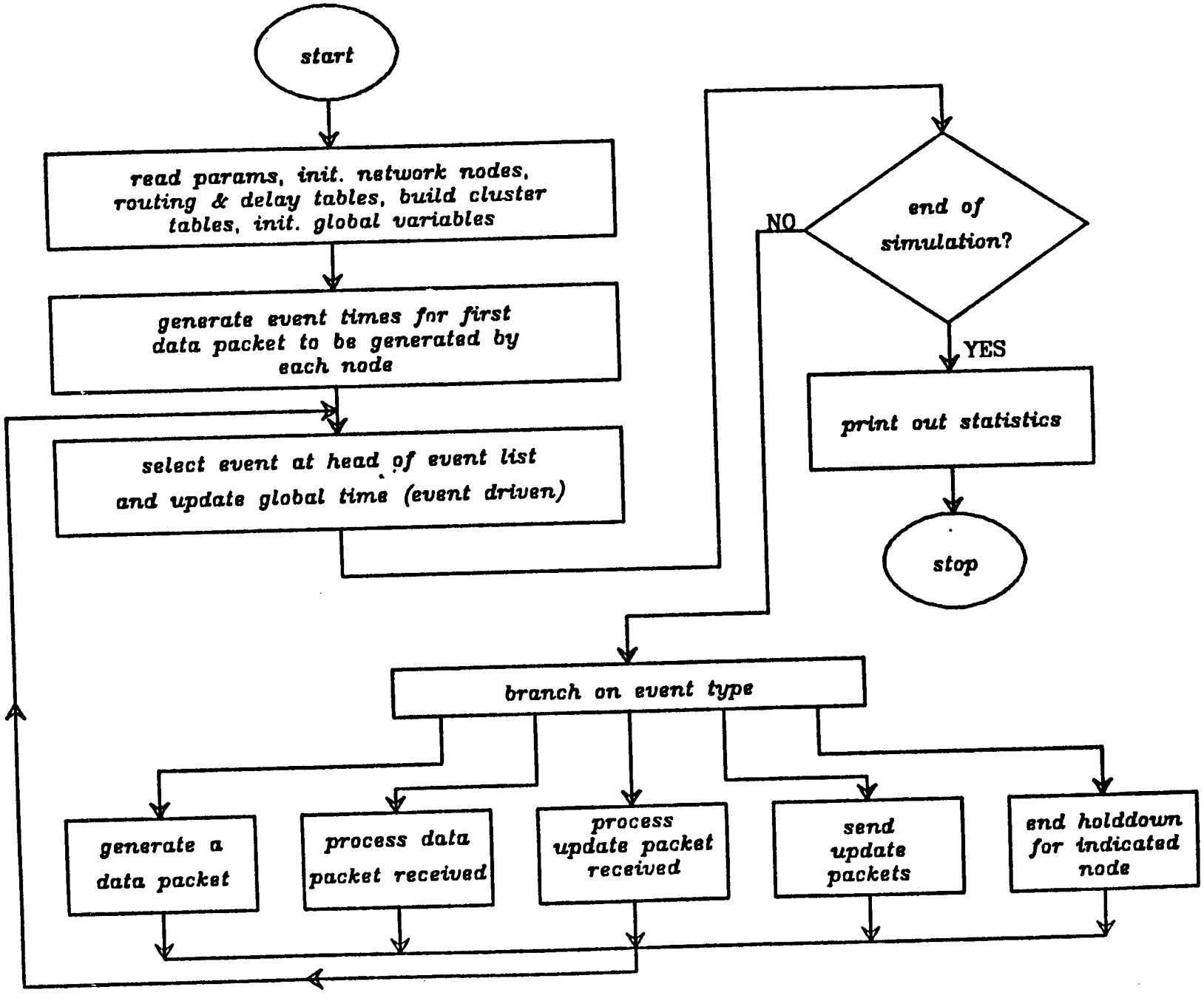


CHART 1