AD-E100 003

Bolt Beranek and Newman Inc.

AD A047043

Report No. 3641

ARPANET Routing Study – Final Report

Dr. John M. McQuillan, Dr. Ira Richer, Dr. Eric Rosen

5 September 1977

BEST AVAILABLE COPY

Prepared for:
Code 535
Defense Communications Agency

DDC
RECEIVED
NOV 28 1977
B

AD No. ____
DDC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

(14) BBN ~~Report No.~~ 3641

Bolt Beranek and Newman Inc.

(6) ARPANET Routing Study.-- (9) Final Report,

(10) Dr. John M. McQuillan,
Dr. Ira Richer
Dr. Eric Rosen

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02138

(11) 5 September 1977

(12) 173p.

(15) DCA200-77-C-0616

Submitted to:

(18) SBIE

Code 535
Defense Communications Agency
Arlington, Virginia

(19) AD-E100 003

1473
060 100

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

4B

TABLE OF CONTENTS

	Page
1. Overview and Conclusions	1
2. Statement of the Problem	10
3. Measurement and Analysis of Network Disturbances	13
3.1 Introduction	13
3.2 Measurement Package and Other Tools	14
3.3 Detailed Analysis of Three Disturbances	18
3.4 Tabular Summary of Network Disturbances	35
3.5 New Network Bugs Uncovered	40
4. Explanation of Network Disturbances	51
4.1 Summary	51
4.2 The Formation of Congestion	54
4.3 The Spread of Congestion	55
4.4 Disruption of Service	57
5. Recommendations	59
Appendix I. Measurement Package Output	66

White Section
Buff Section
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. Avail. and/or SPECIAL
A

Appendix II. Message Generators Used in Experiments 79

Appendix III. Summary of Network Disturbances and Experiments 83

Appendix IV. The ARPANET Routing Algorithm 162

1. Overview and Conclusions

BBN has completed its study and diagnosis of ARPANET routing disturbances as a task under Contract DCA 200-C-616.

For several years the ARPANET has been subjected to occasional disturbances. (These have been referred to as network glitches or network disturbances.) Until the spring of 1977 these disturbances occurred with an average frequency of once every two or three days and were of short duration. In April of 1977 the frequency and duration of the disturbances increased until there were three or four a day. Subsequently, this increase was traced to a change in the IMP software governing the satellite line between SDAC and NORSAR to allow only eight packets in flight, rather than the sixteen required for full line utilization. When the software was changed back to allow sixteen packets in flight between those two IMPs, the number of disturbances decreased dramatically.

However, network disturbances are not the result of this one isolated bug. Our study has indicated that the ARPANET is subject to large-scale disturbances stemming from a variety of external causes: faulty IMP hardware, software bugs, circuit difficulties, traffic overloads (stochastic), etc. The real problem is not any particular irritant but the vulnerability of the ARPANET to congestion caused by such irritants.

The first step in our study was to define a network disturbance. We can offer three common characteristics:

1. The NCC host detects the loss of some IMP reports.

2. Some IMPs declare other IMPs in the network unreachable.

3. Users in the network see their TIP connections or host-to-host connections broken.

These events appear to be closely correlated to IMPs retransmitting packets many times to adjacent IMPs. When an IMP retransmits a packet 600 times (which takes at least 75 seconds), it declares the line down.

Determining the causes of ARPANET disturbances is a complex and difficult task given the nature of the network: the IMPs have limited memory and must communicate with the NCC by means of the same circuits that are involved in a disturbance. We have developed a modular set of measurement programs in the IMP program which allows us to take a snapshot of a given set of data (queue lengths, buffer counts, etc.) whenever a network disturbance occurs. When the disturbance has ended, a single command from the NCC causes all IMPs that have triggered to transmit their data to the NCC.

We have used this measurement package to analyze a total of 36 network disturbances which occurred in the period 5 July 1977 to 5 September 1977. Of this total 19 were spontaneously

occurring disturbances of various magnitudes and 17 were disturbances which we provoked artificially. We used the two-hour period from 7-9 a.m. on Tuesday mornings (a time reserved for ARPANET software maintenance) to conduct experiments. We used various means (making a line appear to be up in one direction only, making an IMP artificially slow, etc.) to induce congestion in one region of the network, which then led to network disturbances. The utmost caution must be used in creating such disturbances since too severe a test can readily disrupt all network service. Thus all of our experiments minimized risk through software equivalents of a dead man switch or a watch dog timer.

In the course of our investigations we discovered the following bugs:

1. The Pluribus IMP code created a free list error whenever a packet was retransmitted more than 600 times, because a garbage collection routine would attempt to free the buffer before its last retransmission.

2. The line alive/dead logic used by the IMPs to determine whether circuits are usable had an important bug which resulted in one IMP thinking that the line was up while the adjacent IMP thought it was down. This condition could persist for a minute or more, long enough for congestion to form and spread.

3. A newly discovered kind of store-and-forward lockup based on a deadly embrace among four IMPs is possible when lines in the network go up and down at a high frequency.

4. The IMPs were not processing "hello" and "I heard you" messages at a high enough priority to allow the determination of line status under high load and congestion situations.

5. IMPs with four circuits were not allocated enough store-and-forward buffers so that lockups and deadlocks could occur if they became congested.

6. A reassembly lockup was possible if the IMP program did not have enough storage to reply to certain types of control messages. It was programmed to refuse such messages which could, under certain circumstances, accumulate in the adjacent IMPs until all IMPs adjacent to a congested IMP would be full of refused control messages. At this point a reassembly lockup is possible.

7. A routing failure is possible in which information about a change in the network reaches some point far from that change by one route much faster than it reaches it by another. In this circumstance the IMPs far from the change may react incorrectly and begin to use other

routes to the affected destination instead of waiting for all the information about that destination to flow through the network.

8. Another routing failure has been observed in which packets which should be routed to other IMPs in the network are routed to stub IMPs.

9. Packets have been observed to loop among two or more IMPs in the network, although this and other routing failures do not occur in all network disturbances.

We have made progress towards solving some of these bugs, whereas others represent more fundamental structural problems in the IMP program and are therefore harder to deal with. Of course, other questions remain unanswered. The exact cause of several of the disturbances we observed is unknown. The timing of the spread of a disturbance is not fully understood. Certain bugs, including misrouting to stub IMPs, remain unexplained.

Our explanation of network disturbances goes as follows: the basic cause of the disturbances seen over the last several years in the ARPANET is that the network has no built-in protection against traffic congestion. The limit of no more than 600 retransmissions of an individual packet is much too high because it allows congestion to build up throughout the network. Congestion can build up from a particular node in several ways. Many sources may be transmitting to this destination and thus all

the paths towards it become full. Alternatively, traffic may build up in one direction and then cross traffic may be blocked from flowing through a particular node. When enough IMPs are affected, some of the IMPs begin to retransmit packets for the 600th time and to kill their lines. This is the beginning of the end of the congestion since the affected IMPs become unreachable to the rest of the network. We point out, however, that some disturbances recur in second and third "waves" of congestion.

During the build-up of congestion user service is disrupted because other traffic cannot flow through the congested regions. At the climax of the disturbance user service is further disrupted by breaking end-to-end protocol connections.

Our recommendations for corrective actions are as follows:

1. The retransmission limit has been lowered from 600 to 32 and the IMPs drop or reroute packets after retransmitting them 32 times, rather than declaring the circuit unusable.

2. The line alive/dead logic has been improved so that the IMPs do not define circuits to be up in one direction for long periods of time. However, we have observed this condition persisting for several seconds, and this may still cause disturbances. Further analysis is required in this area before the algorithm is completely correct; we recommend an explicit mechanism for ensuring

that a line is up in both directions before traffic is allowed to flow over it.

3. The retransmission of control messages should be modulated so that the network does not fill with control traffic during congestion.

4. Each IMP should give priority to traffic originating at other IMPs (perhaps by reserving some store-and-forward buffers for tandem traffic) so that high levels of host traffic do not cause store-and-forward congestion.

5. The IMP should guarantee input buffers and processing time for routing messages so that routing data is not lost during congestion.

6. The internal process which generates reply messages should act on a round-robin basis rather than first-in first-out, so that one reply does not act as a bottleneck if it cannot be delivered.

Our longer term recommendations are as follows:

1. The hold down mechanism in routing should be replaced with a better means for adaptation to changes.

2. Changes in topology should be determined much faster. Currently, the ARPANET IMPs need up to 10 seconds to detect changes, long enough for congestion to build up along the path to a dead IMP.

3. The IMP software should deal with congestion explicitly by measuring it and by metering traffic that flows through the network to avoid congestion.

4. The IMP program should deal with store-and-forward lockups explicitly so that these do not cause network disturbances.

In summary, our study has successfully explained the causes of disturbances, the mechanisms by which they grow, spread, and finally dissipate, and their disruptive effects on network service. We have solved many of the problems which lead to disturbances, and have installed changes in the IMPs to make disturbances less frequent and severe. Note, however, that these improvements were restricted to the H516/316 IMPs; we did not have the resources to install measurements or changes in the Pluribus IMPs. We wish to make clear, however, that the ARPANET is still vulnerable to congestion. As traffic loads increase, disturbances will be more likely to occur, and will be larger in scale. Therefore, it is essential that a continuing program of investigation, analysis, and improvement be carried out, as we have recommended in this report.

Our report is organized as follows:

-- Section 2 defines the problem;

-- Section 3 describes the conduct of the study;

-- Section 4 presents our explanation of network disturbances;

-- Section 5 outlines our recommendations;

-- Appendix 1 describes the measurement package;

-- Appendix 2 defines our experimental traffic patterns;

-- Appendix 3 provides detailed descriptions of 36 disturbances;

-- Appendix 4 contains a description of the ARPANET routing algorithm.

In addition to the authors of this report, the members of the team who worked on this contract were Dr. Paul J. Santos, Jr., Manager of the ARPANET, and Mr. Joel B. Levin, and Mr. James Herman, who implemented the IMP and TENEX software.

2. Statement of the Problem

For several years the ARPANET has been subjected to occasional disturbances frequently referred to as network glitches. Their existence has been reported in the daily outage summary prepared by the NCC since January 1977. What is a network disturbance? We can offer several different and consistent characteristics:

a. As seen by the NCC host: the temporary loss of some IMP reports, sometimes resulting in short node and line outages being logged.

b. As seen by the IMP subnetwork: the spurious definition of one or more IMPs as unreachable.

c. As seen by the average network user: the momentary disruption of TIP connections (or host-to-host connections).

Until the spring of 1977 the disturbances occurred with an average frequency of once every two or three days and were of short duration (usually less than three minutes as observed from the NCC). At that time, the NCC was not receiving user complaints related to the disturbances.

In April 1977 the frequency, severity, and duration of the disturbances gradually increased until there were three or four a day. The user community began to send in complaints concerning the effects of these disturbances.

In late March of 1977 we began the measurement of the number of TIP connections throughout the network on a minute-by-minute basis. We have continued this measurement as a means of determining the impact of network disturbances on communications service provided to the users. We have found that there is a sizable drop in the number of TIP connections at some or all of the TIPs during the period of a network disturbance.

In early May of 1977 we reinstated a particular software parameter on the SDAC-NORSAR satellite link. This changed the operation of that link from the ARPANET standard of eight logical channels to sixteen logical channels.

This change had originally been installed in September 1976 and was temporarily removed in early April 1977 during testing for line errors. Since the resumption of sixteen-channel operation, the NCC has observed far fewer of the disturbances. Subsequent analysis has shown, as indicated in this report, that the eight-channel operation over the satellite line was indeed a major irritant leading to network disturbances, though by no means the only one.

This study has indicated that the ARPANET is subject to large-scale disturbances stemming from a variety of external causes: faulty IMP hardware, software bugs, circuit difficulties, etc. The real problem is not any particular irritant, such as the 8-channel operation at SDAC or an IMP failure, but the vulnerability of the ARPANET to congestion

caused by such irritants. This conclusion is explained in the sections below.

3. Measurement and Analysis of Network Disturbances

3.1 Introduction

At the outset of this study we were well aware that determining the etiology of ARPANET disturbances would be a complex task. Troubleshooting problems that are intermittent and that occur in real-time systems is rarely straightforward. With a system such as the ARPANET, the difficulty is compounded because the network cannot be taken down completely for troubleshooting activities, and because conditions on the network constitute a random process with many variables. In one experiment, for example, we artificially generated a very high level of traffic through the network, hoping to cause a disturbance, but failing. We then repeated the experiment, and did cause a very large disturbance. Clearly, the only difference between the two experiments was the make-up of the non-experimental network traffic, which constituted only a small fraction of the total traffic. In other words, disturbances are often caused by the concurrence of events, each of which by itself may be innocuous.

Furthermore, we had to take the utmost precaution in performing experiments, since too severe a test can readily disrupt all network service. This reasoning led us to design our experiments so that any new program could be turned on and off at will. In fact, in some instances, we put new program modules into effect for a few minutes only. At other times, the network

was protected by a software equivalent of a dead-man switch or a watchdog timer. Finally, we attempted to minimize the effects of inevitable software bugs by careful testing, and by maintaining an "escape route" in software at all times.

3.2 Measurement Package and Other Tools

One of the first tasks that was undertaken in this study was the development of a measurement package for obtaining data on network disturbances. This software package resides in each IMP and its basic operation is as follows: every 3.2 seconds the package updates information in its memory, until certain events called triggers cause the IMP to freeze the data, in effect taking a "snapshot" of the information near the time of the trigger. For example, the package stores the number of packets queued on each of its output lines and the destinations of each of these packets. Of course, a triggering event is not synchronized with the updating times and therefore after a trigger, the measurement package holds the immediately preceding and subsequent data on the queued packets rather than the data at the exact instant of the trigger. When a disturbance has ended, a single command from the NCC causes all IMPs that have triggered to transmit their data to the NCC. Then, an off-line program is used to sort the data and generate a readable output.

The measurement package currently in use is the result of evolutionary changes made during the course of the study. As our understanding of network disturbances increased, the contents of

the package changed in order to enable us to observe phenomena that were of greater interest and of greater relevance. Since the IMP memory is limited, the size of the measurement package must be limited and compromises were made in order to conserve memory. For example, the original package allowed one trigger per IMP but each trigger recorded four snapshots, namely, the status of the IMP immediately after the trigger and for the three intervals preceding the trigger. Subsequently, this was modified to allow each IMP to trigger twice with two snapshots per trigger. More recently, the package was again modified in order to enable three triggers of two snapshots each to be recorded. In order to accommodate this last change, other data that had previously been recorded was no longer held in the IMP memory. The rationale behind this particular sequence of modifications will become apparent in the sections below where we show that disturbances often proceed in waves, and it is therefore useful to observe the sequence of triggering of IMPs in each of the several waves.

Rather than explain the various stages of evolution of the measurement package in detail, we shall only describe the contents of the present package. An itemized list of all the data that is recorded is given below; Appendix I shows a sample output that is generated by the off-line program.

-- Time and Cause of Trigger. As mentioned above, three triggers are permitted in the present package. The time

is based upon the IMP synchronization time and is accurate to approximately three seconds. Causes of triggers are currently the killing of a line, the discarding of a packet, or the rerouting of a packet.

-- Times of Snapshots. In the present version, there are two snapshots for each trigger. The snapshot time is shown as an absolute number so that the interval from the actual trigger can be determined. In addition, the interval between snapshots (approximately 3.2 seconds) is recorded and the time of the average background loop in milliseconds is also given. The subsequent items apply to each of the snapshots that is recorded.

-- Counts of the various categories of buffers that have been allocated by the IMP. In some categories, this count is broken down and the sizes of some relevant queues and subcounts are provided.

-- The status of each output line emanating from the IMP, the number of packets queued on each line, and the destinations of these packets.

In addition to the measurement package, a number of features already present in the IMP were utilized in the course of this study. One feature that was used during experimental sessions is the capability of setting up message generators in order to generate traffic within the net. In its present version each IMP

can be commanded to generate message traffic for one other destination in the network. Both the rate of traffic and the length of each message is a selectable parameter. The capability of generating artificial traffic is, of course, essential in conducting experiments. Appendix II lists the generators used during the experiments.

Some statistics recorded by the IMPs that have proved useful are cumulative counts of the numbers of packets dropped and rerouted. In the most recent tests, where packets were intentionally dropped or rerouted, this statistic proved valuable. The actual numbers are collected after a disturbance, on command from the NCC.

The NCC log itself was used in several ways. During experiments the log gave a prompt indication of the occurrence or non-occurrence of a disturbance and of the magnitude of the disturbance. During and after spontaneous disturbances, the log often provided indications as to the initial cause of the disturbance. Also, after studying the data generated during the disturbance, we often referred to the log to determine whether some secondary events that might have been related to the disturbance had indeed occurred.

Finally, for the two-hour period 7:00 a.m. to 9:00 a.m. on seven Tuesday mornings, we used the time reserved for ARPANET software development to conduct experiments. Needless to say, the ability to perform and later analyze controlled experiments

often provided the key insights that led to the level of understanding we now have. In the future, it would be convenient for us to perform certain network tests, measurements, or experiments at other times of the week, after coordination with network users and DCA.

3.3 Detailed Analysis of Three Disturbances

A. Experiment II, 26 July 1977

This experiment used the 15 Case I message generators (see Appendix II) to produce a high level of traffic through FNWC. The IMPs were set to kill the line if a packet was retransmitted 600 times. Then the IMP at FNWC was artificially slowed so that it appeared to be very busy and would be unable to handle much store and forward traffic. A timer at FNWC was set to return FNWC to normal speed after 6 minutes of artificially slowed operation.

Shortly after FNWC was slowed, a very large disturbance began, with FNWC as the focus of congestion. The causes of triggers are an IMP killing a line because of 600 retransmissions, or an IMP receiving the 599th transmission from a neighbor. The sequence of triggers is listed below:

Event	Time (sec)	Line	Event	Time	Line
1	0	LINC-DEC	27	256	CMU-RADC
2	2	SCRL-FNWC	28	261	RADC-LINC
3	13	ANL-SCOTT	29	263	BELV-CMU
4	14	GWC-ANL	30	267	NBS-ABER
5	14	DOCB-GWC	31	270	AMS16-AMS15
6	19	HARV-SCOTT	32	273	DCGL-BELV
7	28	SCOTT-FNWC	33	359	SRI2-SRI51
8	44	TYMSH-FNWC	34	368	RAND-ISI52
9	152	USC-UCLA	35	372	RCC49-BBN5
10	155	UCLA-SCRL	36	375	MIT6-WPAFB
11	166	ISI52-USC	37	375	SRI2-LLL
12	174	GWC-ANL	38	375	LLL-SRI2
13	174	HARV-SCOTT	39	375	MOFF-LBL
14	177	DOCB-GWC	40	375	DEC-HARV
15	187	CMU-ANL	41	375	LINC-DEC
16	201	ISI22-STAN	42	376	EGLIN-GUNTR

17	204	STAN-SUMEX	43	377	MOFF-AMS15
18	214	SRI2-SUMEX	44	378	WPAFB-ILL
19	215	AFWL-ISI22	45	379	LLL-LBL
20	215	ISI52-ISI22	46	380	AMS16-SRI51
21	215	RAND-ISI52	47	380	GUNTR-TEXAS
22	219	AMS15-STAN	48	381	ACCAT-RAND
23	247	SUMEX-TYMSH	49	383	NYU-HARV
24	249	ABER-BELV	50	384	AFWL-ISI22
25	251	XEROX-TYMSH	51	387	UTAH-LBL
26	255	SCRL-FNWC			

Table 1: Experiment II, 26 July -- List of Triggers

It is important to note that event times are accurate only to within about 3 seconds, so that the real order of the triggers may be slightly different from that given above.

The ARPANET map in Figure 1 illustrates these events. The circled numbers are the event numbers, and each arrow shows the direction in which the 600th retransmissions occurred. From this figure and the preceding table we see that first, traffic backed up from FNWC through SCOTT for a distance of up to four hops. Lines along these paths were killed during the first 44 seconds of the disturbance. Then the network was quiet for almost two minutes, after which traffic backed up along all other paths to FNWC. This caused the disturbance to move into the San Francisco and Los Angeles areas. This second wave of the disturbance lasted for about two minutes. Then the network was quiet for about a minute and a half, until a third wave of congestion began. The third wave was caused by user traffic from the Northeast to the D. C. area moving towards California. It lasted for about 30 seconds. Figure 2 shows these waves on a network map. The solid curve represents the first 44 seconds; the dashed curve, for t=152 to t=273 seconds is the second wave; and the dotted curve, t=387 seconds, is the final expansion.

This disturbance illustrates three phenomena which we see repeatedly during network disturbances:

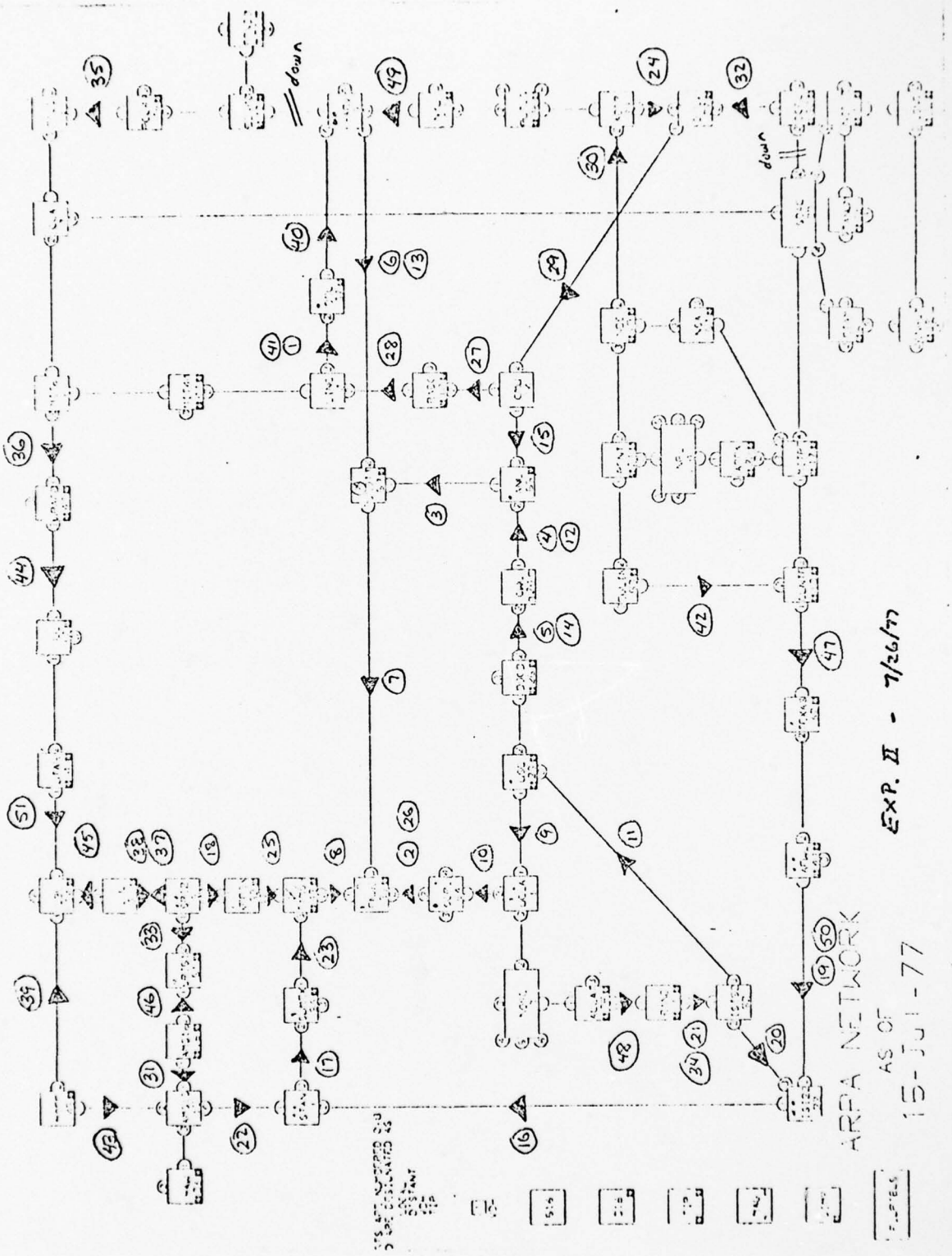


Figure 1: Experiment II, 26 July -- Map of Events

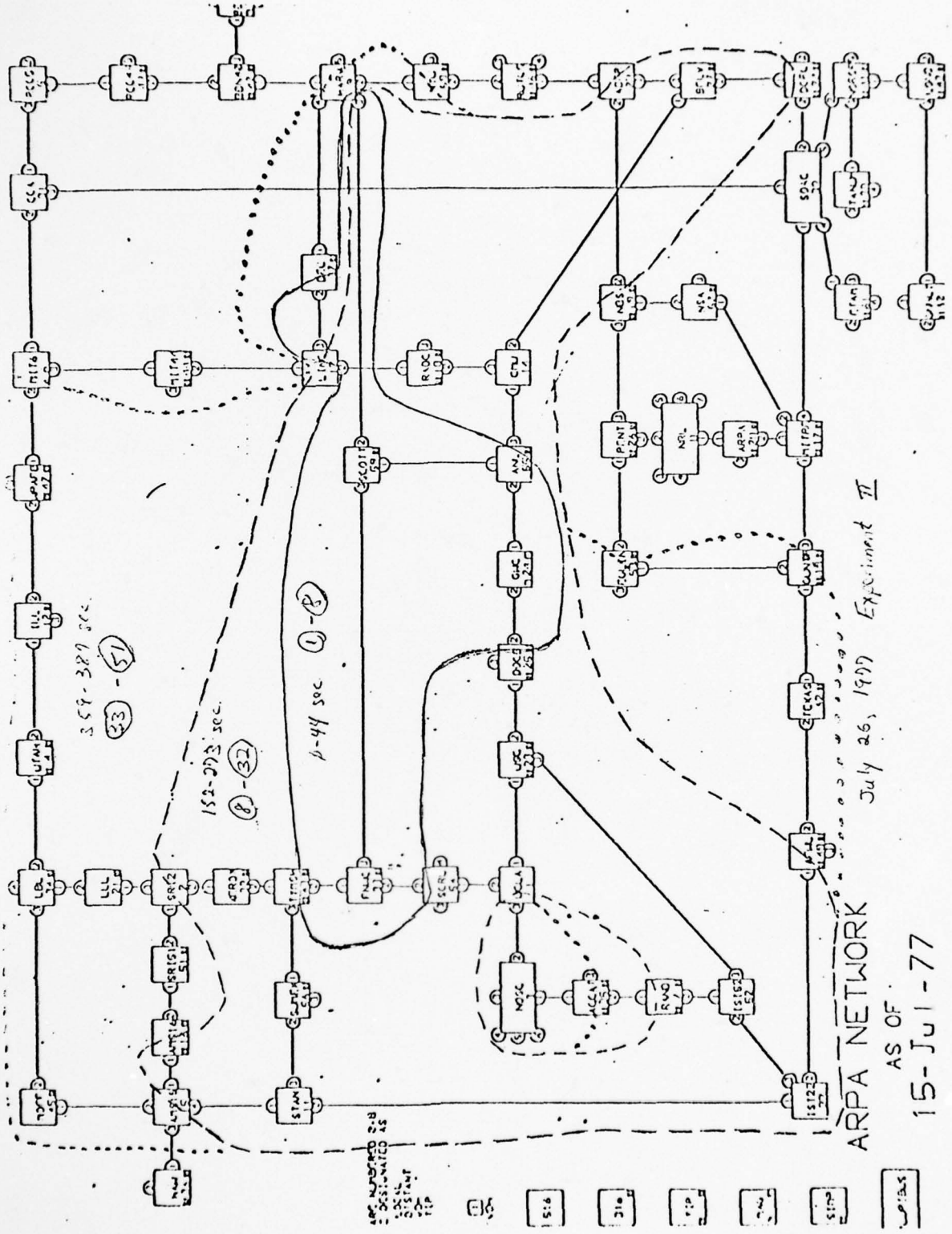


Figure 2: Experiment II, 26 July -- Waves of Congestion

1. When a disturbance forms, even relatively small amounts of user traffic heading into the area of the disturbance will cause it to spread.

2. Disturbances often seem to occur in waves, with each succeeding wave affecting a larger and larger portion of the network. There are also relatively long quiet periods between the waves. The length of these quiet periods seems to be dependent both on random factors (such as the arrival of new user traffic flows) and network timing constraints (such as the amount of time it takes to retransmit a packet 600 times).

3. Occasionally, there are retransmission failures in the opposite direction to the congestion. For example, the arrows for events 33 and 46 (near SRI51) are not in consistent directions.

This disturbance also indicated some sort of routing failure between AMS15 and HAW: during one trigger of AMS15, some traffic for ISI22 was routed toward HAW. (At its second trigger, there was no traffic for HAW.)

B. Disturbance I (Spontaneous), 18 August 1977

On this date, the network was set to kill a line whenever a packet was retransmitted 50 times on that line. The disturbance began when MIT6 crashed. Two of MIT6's neighbors (WPAFB and MIT44) reported almost simultaneously that their lines to MIT6

were down. Thirteen seconds later, IMPs in the D.C. area began dropping lines to each other (CMU was also involved). The snapshot data shows that the queues in most of these IMPs were filled with traffic destined for MIT6. This Washington area disturbance lasted for 10 seconds.

The shortest path from the D.C. area to MIT6 is via the SDAC-CCA line. However, the snapshots show that packets were not being routed to that line. This can be explained by supposing that the CCA-MIT6 and SDAC-CCA lines were also down. (Since SDAC and CCA are Pluribus IMPs which do not contain the measurement package, we have no shapshots to support this supposition, however.)

The apparent cause of the disturbance's moving into the D.C. area was that seven neighboring IMPs (EGLIN, PENT, NBS, NSA, BELV, ABER, and CMU) were filled with traffic for an IMP (MIT6) which crashed. Since it took varying amounts of time for these IMPs to learn that each possible path to MIT6 was down, routing was in a confused state, resulting in looping (see particularly event 4). Traffic for MIT6 thus had no way out of the Washington area. It must also be noted that even after one IMP sees another go down, it cannot discard traffic for that IMP until 10 seconds elapses (the IMP software uses this time to determine if any other path is available to the destination). As a result, traffic for MIT6 just piled up in the D.C. area, causing store-and-forward congestion and the resulting disturbance.

Table 2 and Figure 3 on the following pages show how the disturbance began in the MIT6 vicinity, and then suddenly sprang into existence in the Washington area. Thus the disturbance is a good example of how a disturbance can spread from one area of the network to another. It also shows the contribution of routing failure to the spread of a disturbance. Note that the two neighboring IMPs (NSA and NBS) are sending traffic destined for MIT6 to each other. In fact, each of these IMPs has a full queue to the other. We have observed similar phenomena quite often at NSA and NBS.

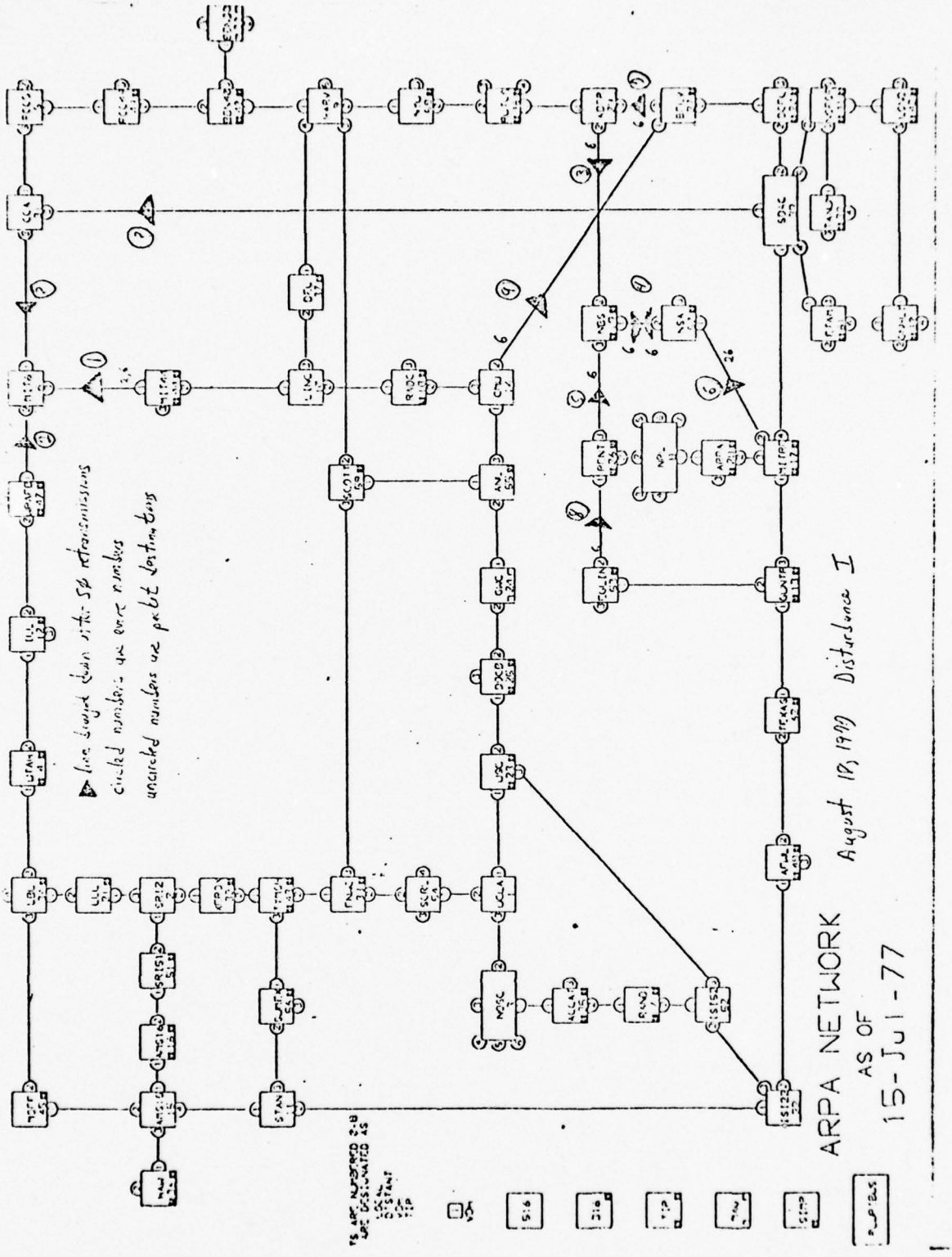


Figure 3: Disturbance I, 18 August -- Map of Events

Event	Time	Line
1	0	MIT44-MIT6
2	0	WPAFB-MIT6
3	13	ABER-NBS
4	.4	NBS-NSA, NSA-NBS
5	14	PENT-NBS
6	16	MITRE-NSA
7	18	BELV-ABER
8	21	EGLIN-PENT
9	23	CMU-BELV

Table 2: Disturbance I, 18 August -- List of Triggers

C. Experiment I, 2 August 1977

For this experiment, we used the Case I set of message generators, and slowed FNWC. The network was set to kill a line after retransmitting a packet on it 600 times. It should also be pointed out that due to an error in the measurement package, FNWC, SCOTT, and HARV were unable to trigger.

Within approximately 5 seconds, congestion backed up from FNWC to SCRL, UCLA, DOCB, and ISI52, causing these nodes to trigger and to drop lines. We may speculate that congestion also backed up to SCOTT and HARV, though we have no way to determine this.

After approximately 13 seconds, ISI22 became congested, with 23 buffers apparently stuck on its TASK reply queue. This indicates that ISI22 was actively engaged in source-destination communication with certain IMPs (as opposed to store-and-forward traffic), but was too busy to send out replies to control messages from those IMPs. When a control message from a remote IMP is received, a reply is created and placed on the Task Reply Queue. Eventually the reply will be removed from the Task Reply Queue and transmitted (as a store-and-forward packet) to the remote IMP. However, the Task Reply Queue is serviced with a very low priority. It can therefore grow quite large in an IMP which is very busy with host traffic or store-and-forward traffic. Furthermore, the Task Reply Queue is treated as a FIFO stack. If the first packet on it cannot be transmitted, say

because there is too much traffic backed up along the path to its destination, then no other packet from that queue can be transmitted. A further complication is that if a packet on the Task Reply Queue is not sent, the control packet to which it is the reply will be retransmitted from the source IMP every two seconds. This can only result in increasing any congestion which may exist.

The congestion at ISI22 caused congestion at AMS15, which filled up with traffic directed to ISI22. As a result, congestion backed up along all paths containing traffic to ISI22 or AMS15, causing lines to be dropped all the way back to the East Coast (see Figure 4).

After approximately 60 seconds, the network disturbance had completed its damage on the West Coast and was causing lines to go down on the East Coast. However, the outages in the East fell into a much less noticeable pattern.

The explanation for this disturbance seems to be as follows. During the first five seconds, traffic backed up from FNWC through SCRL, UCLA, and USC to ISI52. ISI52 at the time contained a considerable amount of traffic for IMPs in the San Francisco area. This traffic was rerouted away from USC and towards ISI22. At the time ISI22 had a very high reassembly buffer count, indicating that it was actively engaged in end-to-end conversation with some other IMPs. Indeed, the snapshots show traffic for ISI22 coming from as far away as RCC5.

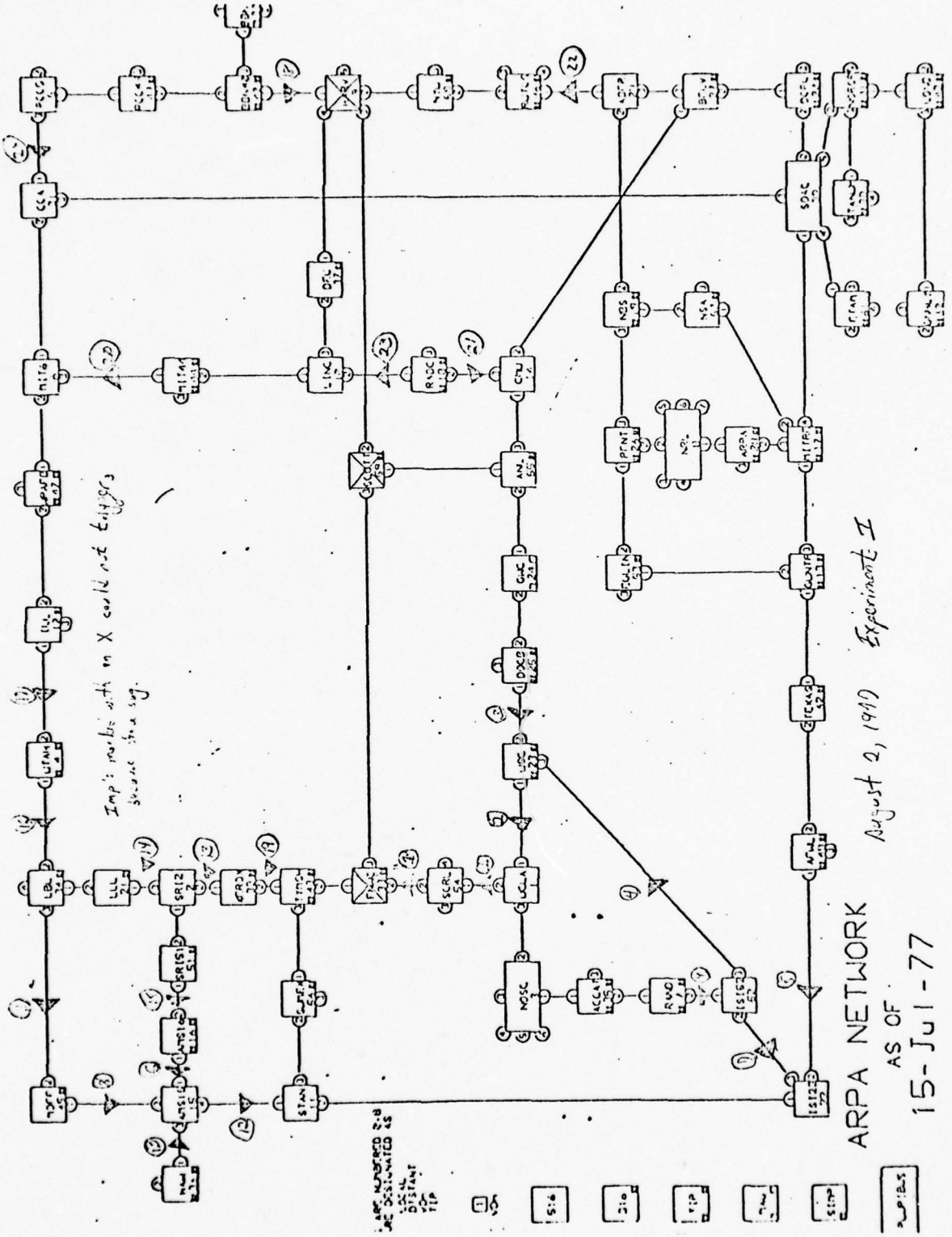


Figure 4: Experiment I, 2 August -- Map of Events

We may hypothesize that this additional store-and-forward traffic load from ISI52 was too much for ISI22 to handle in conjunction with its own traffic. Traffic began to back up from ISI22 in all directions. This caused the disturbance to spread into the San Francisco area, as IMPs in that area filled with traffic destined for IMPs in the Los Angeles area. The disturbance spread simultaneously toward the Northeast, the source of much of the traffic destined for California.

This disturbance is interesting because the major focus of congestion, ISI22, was not involved in the experiment in any way. All traffic towards it was real user traffic, and this was of a much smaller volume than the artificially generated traffic. Nevertheless, the only way in which the artificiallty generated traffic contributed to the disturbance is by removing one of the two main paths between the San Francisco area and the Los Angeles area. This caused the other path to become overloaded with real user traffic; and that is what caused the disturbance to spread.

This disturbance illustrates two important recurring phenomena:

a. The pattern of traffic plays at least as important a role in the occurrence and spread of disturbance as does the actual amount of traffic.

b. The end-to-end mechanisms in the network can interact with the store-and-forward mechanisms to increase congestion.

Event	Time	Line	Focus of Congestion
1	0	SCRL-FNWC, UCLA-SCRL	FNWC
2	2	USC-UCLA	FNWC
3	5	DOCB-USC	FNWC
4	5	ISI52-USC	FNWC
5	13	AFWL-ISI22	ISI22
6	16	AMS16-AMS15	ISI22
7	17	ISI52-ISI22	ISI22
8	21	MOFF-AMS15	ISI22
9	23	RAND-ISI52	ISI22
10	25	HAW-AMS15	ISI22
11	35	LBL-MOFF	ISI22
12	38	AMS15-STAN	ISI22
13	40	SRI2-XEROX	FNWC & ISI22
14	42	LLL-SRI2	FNWC & ISI22
15	42	SRI51-AMS16	ISI22
16	45	UTAH-LBL	ISI22

17	54	ILL-UTAH	ISI22
18	58	BBN40-HARV	ISI22 & FNWC
19	62	XEROX-TYMSH	FNWC
20	63	MIT44-MIT6	ISI22
21	64	RADC-CMU	
22	64	ABER-RUTGS	
23	71	RADC-LINC	
24	71	RCC5-CCA	

Table 3: Experiment I, 2 August -- List of Triggers

3.4 Tabular Summary of Network Disturbances

The table which follows presents a summary of the 36 network disturbances which we measured and observed. They are almost equally divided into spontaneous and experimental disturbances. We have indicated the network conditions prevailing at the time, along with the immediate cause, and the size of the disturbance. Each of these events is described in more detail in Appendix III, which also contains maps of these disturbances.

3.4 TABULAR SUMMARY OF NETWORK DISTURBANCES

Date	Identification Number	Spontaneous(S) or Experimental(E)	Network Conditions	Number of IMP's Affected	Immediate Cause	Comments
7/6	I	S	lines killed after 600 retransmissions	17	line 53 goes flakey	
7/12	I	E	lines killed after 600 retransmissions	4	a)FNWC-SCOTT line up one way b)Case I generators used	
7/12	II	E	lines killed after 600 retransmissions	12	a)FNWC-SCOTT line up one way b)Case II generators used	Shows that failure of line up/down logic can be important cause of disturbance
7/13	I	S	lines killed after 600 retransmissions	20	unknown	
7/14	I	S	lines killed after 600 retransmissions	26	bug in Pluribus IMP software at SDAC	Shows poor adaptation of routing to loss of important path
7/14	II	S	lines killed after 600 retransmissions	33	bug in Pluribus IMP software at SDAC	
7/14	III	S	lines killed after 600 retransmissions	24	unknown	a)Two foci of congestion (UTAH, ISI22) b)Poor adaptation of routing to existence of congested area
7/15	I	S	lines killed after 600 retransmissions	22	BBN40 run out of free buffers, while net contains much traffic destined to it	Overload of traffic to single destination
7/15	II	S	lines killed after 600 retransmissions	18	unknown	Two foci of congestion (in Northeast and D.C.)

Date	Identification Number	Spontaneous(S) or Experimental(E)	Network Conditions	Number of IMP's Affected	Immediate Cause	Comments
7/18	I	S	lines killed after 600 retransmissions	25	unknown	Two foci of congestion (BBN40, MITRE)
7/19	I	S	lines killed after 600 retransmissions	28	hardware trouble at MIT6	a)Net contained many packets to broken destination(MIT6) b)Cross-traffic in congested area contributes to severity of disturbance
7/19	II	S	lines killed after 600 retransmissions	35	trouble at SDAC	Complex set of traffic flows makes pattern hard to discern
7/19	III	S	lines killed after 600 retransmissions	31	unknown	No discernible pattern
7/19	IV	S	lines killed after 600 retransmissions	22	hardware trouble at MIT44	a)Network contained many packets to broken destination (MIT44) b)Traffic looping between AMS15 and HAW c)Three foci of congestion
7/19	V	E	lines killed after 600 retransmissions	28	a)slow FNWC b)Case I message generators	
7/19	VI	E	lines killed after 600 retransmissions	32	a)slow FNWC b)Case I message generators	a)Traffic looping between AMS15 and HAW b)User generated cross-traffic significantly affected disturbance
7/21	I	S	lines killed after 600 retransmissions	17	trouble on satellite line from SDAC to NORSAR	

Name	Identification Number	Spontaneous(S) or Experimental(E)	Network Conditions	Number of IMP's Affected	Immediate Cause	Comments
7/21	II	S	lines killed after 600 retransmissions	40	flakey line connecting BELV and ABER	Complex pattern, multiple foci
7/22	I	S	lines killed after 600 retransmissions	38	trouble with MIT6	Disturbance spread by routing's attempt to send traffic into disturbed area
7/26	I	E	lines killed after 100 retransmissions	6	a)FNWC slowed b)Case I message generators	
7/26	II	E	lines killed after 600 retransmissions	43	a)FNWC slowed b)Case I message generators	a)Disturbance occurred in waves b)Disturbance aggravated by user generated cross-traffic
7/26	III	E	lines killed after 100 retransmissions	14	a)FNWC slowed b)Case I message generators	
7/26	IV	E	lines killed after 100 retransmissions	40	a)FNWC slowed b)Case II message generators	Disturbance occurred in waves
7/26	V	E	lines killed after 50 retransmissions	17	a)FNWC slowed b)Case II generators	Disturbance occurred in waves
8/2	I	E	lines killed after 600 retransmissions	27	a)FNWC slowed b)Case I message generators	a)Two foci of congestion b)Disturbance aggravated by user generated cross-traffic
8/2	IB	E	lines killed after 100 retransmissions	12	a)FNWC slowed b)reduced Case I message generators	

Date	Identification Number	Spontaneous(s) or Experimental(E)	Network Conditions	Number of IMP's Affected	Immediate Cause	Comments
8/2	II	E	packets discarded after 100 retransmissions	25	a)FNWC slowed b)reduced Case I message generators	First experiment in discarding packets
8/2	III	E	packets discarded after 50 retransmissions	10	a)FNWC slowed b)reduced Case I message generators	
8/4	I	S	packets discarded after 100 retransmissions	15	lines 10, 12, 43 crash simultaneously, for no apparent reason	Only spontaneous disturbance where packets are discarded
8/18	I	S	lines brought down after 50 retransmissions	12	MIT6 crashed	Disturbance began in one area, spread to second area which contains traffic to first
8/18	II	S	lines brought down after 50 retransmissions	10	problem at ACCAT	
8/18	III	S	lines brought down after 50 retransmissions	45	BBN40 and RCC49 became isolated	Network overloaded with traffic to RCC49, which is isolated
8/23	I	E	packets rerouted or discarded after 32 retransmissions	14	a)FNWC slowed b)reduced Case I message generators	
8/23	II	E	packets rerouted or discarded after 32 retransmissions	8	a)FNWC slowed b)Case III message generators	
8/23	III	E	packets rerouted or discarded after 32 retransmissions	14	a)FNWC and GUNTR slowed b)Case IV message generators	
8/23	IV	E	packets rerouted or discarded after 32 retransmissions	6	overload FNWC then reset it	

3.5 New Network Bugs Uncovered

In this section we present a number of difficulties with IMP software which were uncovered as a result of our investigations under this contract. In some cases we were able to supply a simple solution to the problem, while in other cases we were not able to do anything more than note the existence of the problem.

1. We discovered a bug in the Pluribus IMP code which resulted in a packet buffer free list error whenever a packet was retransmitted more than 600 times. The problem was caused by the Pluribus IMP reliability package performing a garbage collection of the buffer before it had been retransmitted 600 times. The Pluribus IMP tries to insure that each packet circulates from one use in the IMP to another within a certain amount of time. The 600 retransmissions of a packet took longer than this amount of time, causing a free list error.

2. An important bug was found in the line alive/dead logic, which is used in the IMPs to determine whether circuits are usable for network traffic. Simply stated, the problem was that the IMP on one end of the line would think it was up, while the adjacent IMP would think it was down. This condition would persist for a minute or more which was long enough for congestion to form and spread, since the IMP which thought the line was up would begin immediately to use it, but it would not receive any acknowledgments over that circuit.

The line alive/dead logic has evolved over the last several years as follows:

a. The original algorithm determined that a line was up after 30 consecutive time periods passed (each 2/3 of a second in length) in which the IMP was able to send a "hello" message to the adjacent IMP and receive an "I heard you" message.

b. The algorithm was changed in 1973 so that the number of periods was increased to 60, while at the same time the requirement for consecutive successes in each of these 60 periods was eliminated. In its place an up/down counter was installed to assist in using some of the newer lines in the network which did not have the same high quality error characteristics as earlier lines. The counter would count from -60 to 0 as "I heard you's" were successfully received. It would count down towards -60 when an "I heard you" was not received successfully.

c. The next change to the algorithm was to reset the counter when it reached -80 back to the original value of -60.

d. The final change to the algorithm (an unintentional change) was to modify the reset value to be -240. Thus when a line was declared down, the IMPs on each end would start off their counters at -60, and, if no "I

heard you" messages were received, would decrement these counters down to the value of -240 and then reset them to -60 and repeat the process. The bug resulted from the fact that the two IMPs at each end of the line might not be counting their timers in synchronism since one IMP could reset its counter to -60 asynchronously with the other IMP when its program was restarted or the line in question looped towards one IMP. Once the line was unlooped or the program restarted, the timers would be out of phase by as much as 180 time periods of 2/3 of a second each. Thus it was possible when the line finally came up again for one IMP to use it for more than two minutes before the other IMP declared it up.

We implemented a partial solution to this problem by changing the reset value to be -60 again on receipt of any input. On the other hand, we have not changed the nature of the up/down counter to be a consecutive timer as it originally was since we feel that this subject requires further analysis. Therefore, at the present time it is still possible for IMPs to declare lines up in one direction only. This may happen on a line in which only some "I heard you" messages are received, or on a line which has more errors in one direction than another. The change we made simply reduces the probability of a long line up/down mismatch. The network has recently experienced mismatches lasting only a few seconds at most.

3. We identified a new kind of store-and-forward lockup which may be happening in the network from time to time. We have only circumstantial evidence for this lockup, since it is very difficult to determine precisely when or if it is happening. We do know that many network disturbances have happened without any lockups. Therefore, this can be regarded as a minor irritant for most disturbances. The lockup we observed is shown in Figure 5. The sequence of events is as follows:

a. Line 53 went down.

b. Packets from IMP 59 to IMP 33 and beyond were routed via IMP 9. Packets from IMP 33 to IMP 9 and beyond were routed via IMP 54.

c. Line 53 came up before traffic had dissipated.

d. All IMPs updated their routing and funnelled traffic back to Line 53.

e. A deadly embrace formed in which IMPs 9 and 54 had committed eight packets to their circuits towards the center of the lockup, while IMPs 33 and 59 had reached the store-and-forward limit and thus could not accept any more packets.

4. Another difficulty encountered in the IMP algorithms for determining whether a line is alive or dead concerns the processing of "I heard you" messages. Previously the IMP program

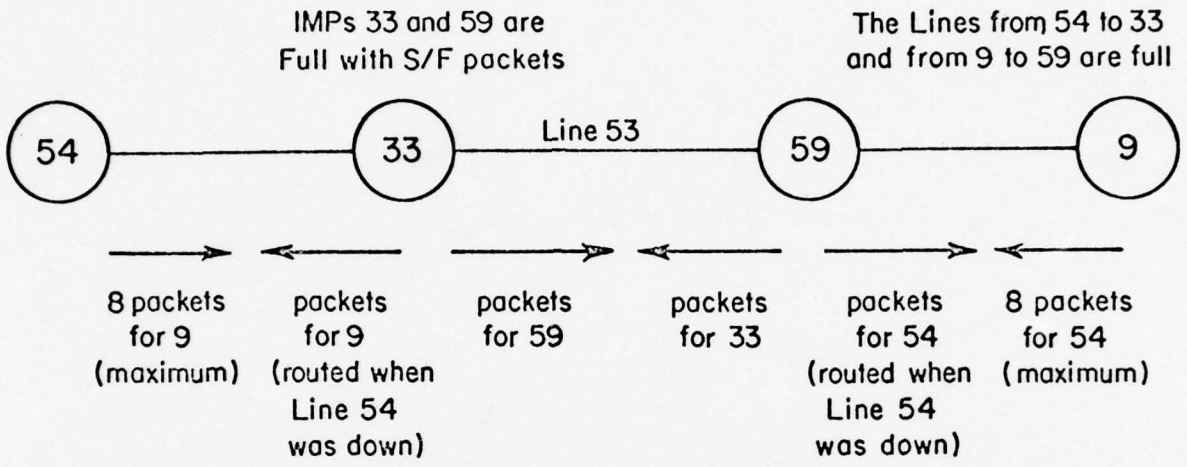


Figure 5: A Four-IMP Store-and-Forward Lockup

had dealt with these messages at the TASK interrupt level. This means that the "I heard you" control message required a packet buffer and a certain amount of TASK processing time. Thus these control messages could be lost under high load. This is very undesirable since the "I heard you" messages are necessary for determining whether the line is up or down which is especially important in the face of high load or congestion. Therefore, the program has since been changed to perform all the critical line alive/dead processing at modem input processing level rather than at TASK which is the lowest priority interrupt.

5. A minor bug which was uncovered in the IMP program which had serious impact was that the store-and-forward buffer count was not computed correctly for IMPs with four lines. This could lead to store-and-forward lockups and to congestion. We had one experience of this with IMP 9 at Harvard during the experiments on July 12. The remedy for this problem was simply to compute the store-and-forward buffer limit on the basis of the number of lines at a particular IMP.

6. Another type of lockup resulted from a path through the IMP program which is seldom used. When the IMP program receives certain types of control messages from other IMPs (such as the get-a-block, reset, or reset request messages), it is necessary for the TASK processing to return a control reply message. In order to do this, it transforms the header of the received control message into the appropriate reply and inserts the reply

on a special queue. When TASK is doing this, it must reserve a buffer in the reassembly count for the reply message, since otherwise the IMP memory could fill with unaccounted packets. There is one condition under which the IMP program cannot process such control messages: when the reassembly limit has already been reached. In this case the IMP was programmed to refuse the control message: that is, it would not send back an acknowledgment to the neighboring IMP. The neighboring IMP would simply retransmit the control message until the destination IMP accepted it. Under conditions of extreme congestion, it is possible that the congested destination IMP would be full of partially reassembled messages (awaiting other data packets in the network before they can be sent into the hosts). Further, one may suppose that control messages such as the get-a-block may be flowing through the network towards this busy destination IMP. If all of the IMPs adjacent to the destination fill with control messages for the destination, which cannot be accepted, then a reassembly lockup occurs, since the data packets which are needed to free up reassembly storage are trapped more than one IMP away in the network.

This lockup condition can easily be avoided by the expedient of dropping a control message when there is no IMP memory left over to store the reply control message. We installed this change into the IMP program and since that time we have not seen evidence for network disturbances which spread from one area of the network to another without affecting all of the intermediate

IMPs. We believe that congestion cannot spread in this manner without the action of some source-to-destination bug or lockup condition. While we think we have removed the major cause of such congestive spreading, we cannot be sure that there are no other lockup conditions present in the network.

7. In examining some of the network disturbances which seem to be based on routing failures, we have been able to hypothesize a particular type of routing failure based on a shortcoming in the hold down mechanism. Consider the network shown in Figure 6. Suppose IMPs A and B are both sending traffic to IMP D. Suppose further that traffic is flowing on the line from IMP C to IMP D and increasing so much that IMP C enters hold down. At this point information about the hold down at IMP C will propagate through both sides of the network towards IMPs A and B. If the information flows at approximately the same rate, both IMPs A and B will hear about the new routing information, will enter hold down, subsequently will leave hold down, and continue to route their traffic in the same direction. If, on the other hand, the routing update information flows much more quickly to IMP A than to IMP B, it will enter hold down because its path to IMP D got worse and will leave hold down perhaps even before IMP B has begun to enter hold down. If this occurs, then IMP A will think that its neighbor IMP B has a new, better route to IMP D and will begin to use it. Then the nodes on the path between IMP A and IMP C will one by one begin to send their traffic to IMP D by way of IMP B. Meanwhile, the correct routing information has been

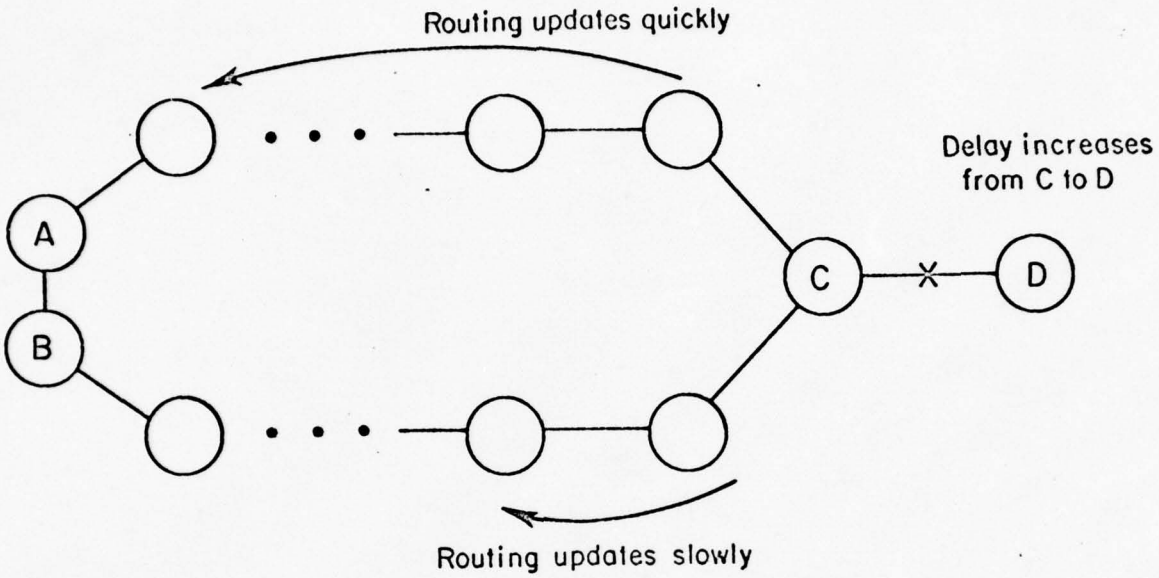


Figure 6: A Routing Hold Down Failure

flowing from IMP C to IMP B and as it reaches each node in turn, each node will correctly update its routing data and thus correct routing information will begin to flow clockwise around the loop following several nodes behind the incorrect routing information which is also flowing clockwise around the loop. This situation can persist for a long period of time. Furthermore, it can persist when the nature of the information change is that the line from IMP C to IMP D went down and thus IMP D is isolated. In this case a large part of the network may oscillate between believing that IMP D is up and believing that it is down.

8. Another routing failure which has been observed to occur is that IMP 15 at AMES occasionally sends traffic destined to other nodes in the network to IMP 36 at Hawaii. This is a clear-cut routing failure since Hawaii is a stub and can offer no better route to nodes in the Continental United States than IMP 15's neighboring IMPs. We believe that this failure occurs as a result of a failure in the hold down timer at IMP 15. Through some set of mechanisms which we do not adequately understand at this time, IMP 15 sends a routing message to IMP 36 indicating that the best paths to certain destinations have increased delay and instead of waiting long enough for IMP 36 to receive this data, perform an update, and send back new routing information indicating that it has accepted the update, IMP 15 instead accepts an update from IMP 36 based on old routing information which makes it appear as if IMP 36 has a better route to the affected destinations. One probable contributing factor to this

failure is the fact that the line between IMPs 15 and 36 is a satellite channel; the round-trip delay is 536 milliseconds. Thus the hold down timer should probably be increased for this particular channel.

9. Another routing failure which has been observed to occur is looping packets between two IMPs which have entered hold down pointing towards each other. This particular bug can happen as follows: If two IMPs are approximately the same distance from some destination IMP by different paths (like IMPs A and B in our previous example) they may sometimes route traffic by the separate paths, and sometimes to each other. As traffic levels fluctuate, there may be moments when A and B both find their paths have increased in delay, and simultaneously decide to route traffic to D via each other. If this happens, and the situation is compounded by a further increase in delay on these new paths, IMPs A and B will enter hold down to D while using routes which point at each other. This is a serious problem, since the routing updates at each IMP will show a linearly increasing delay estimate to D as long as the loop persists. This increase will cause hold down to remain in effect, and thereby cause a vicious cycle. The IMPs detect and break such loops at present; we would prefer to find a way to prevent such loops from forming.

4. Explanation of Network Disturbances

In this section we present our conclusions as to the causes of network disturbances and the detailed sequence of events which make up the network disturbances.

4.1 Summary

The basic cause of the disturbances seen over the last several years in the ARPANET is that the network has no built-in protection against traffic congestion. That is, when the offered traffic in some region of the network exceeds the network's capacity to carry that traffic, then congestion builds up throughout that region and possibly throughout the network as a whole. Eventually, the network is so full of traffic for the congested area that little or no other traffic can flow through the network. The disturbance reaches a climax when the IMP programs in the affected regions determine that they have retransmitted certain packets more than the nominal limit (which has been set at 600 retransmissions). At this point the IMPs declare the circuits to be unusable. This isolates the region of congestion from the rest of the network and permits normal operations to resume, although any user with a host-to-host protocol connection in the affected region would find his connection broken. In the case of a TIP user this would result in a "Net Trouble" message on his terminal.

In coming to this understanding we have made the following general observations:

-- A limit of no more than 600 retransmissions of an individual packet is much too high because it allows congestion to build up throughout the network.

-- Much lower values of this retransmission limit, as low as 32 retransmissions before corrective action, can be installed in the network without significant adverse effects.

-- Congestion builds in many different ways. Traffic from many different sources can be sent towards a single congested destination. Alternatively, traffic may build up in one direction and then cross-traffic may be blocked from flowing through a particular node. Finally, traffic headed in a particular direction may become blocked off, in turn blocking other traffic which is routed in the same direction.

-- The build-up of congestion in the network is a highly variable phenomenon. We were not able to repeat our experiments with any degree of accuracy nor were spontaneous disturbances seen to recur with a high degree of similarity.

-- A number of suspected causes of congestion turned out not to be significant. For instance, we sometimes

observed packets looping between IMPs during network disturbances. However, such events did not always happen and were not directly the cause of the disturbance. Also we observed network disturbances in which the Pluribus IMP program was suspected of being the focus, while at other times no Pluribus IMPs were involved.

-- Network disturbances usually ended as a result of large numbers of network lines being declared down by IMP software as a result of 600 retransmissions of a packet. When enough lines had been declared down to isolate the congested region of the network from the rest of the network, then all traffic in a congested region was either delivered, or *else* dropped due to unreachable destinations. When the lines were declared up again, normal operations resumed unless the congestive irritant was still in force.

For purposes of exposition, a network disturbance can be viewed as a three-part event:

1. Formation of congestion. Some external event causes congestion to form in the vicinity of a particular IMP or IMPs.

2. The spread of congestion. The forming congestion sets other events in motion which lead eventually to the

spread of that congestion to adjacent IMPs and ultimately to IMPs several hops removed from the original site of the problem.

3. Disruption of service. The spread of congestion causes normal network operations to be disrupted. Eventually, it also causes the congestion itself to dissipate.

We discuss each of these phases of the network disturbance in a separate section below.

4.2 The Formation of Congestion

How does congestion form in the network? There are three basic forms of traffic overload which can be viewed as the external event triggering congestion.

1. Host Overload
   a. The host becomes busy.
   b. The host goes down without warning. (In this case the IMP keeps packets for the host for 30 seconds).

Both of these conditions result in the IMP filling with packets for the host until it reaches its reassembly limit.

2. Line Overload
   a. The line becomes busy.
   b. The line goes down.
   c. A bug: running a satellite line with only eight logical channels.
   d. A bug in the allocation of store-and-forward buffers.

e. A bug: in the line alive/dead logic.

Each of these conditions causes the IMP to assign eight packets to a particular line which is the maximum allowed, or alternatively, causes the IMP to assign enough packets on all of its lines so that it reaches the store-and-forward limit.

3. Node Overload

a. The IMP memory fills with buffers.

b. The IMP CPU becomes busy.

Each of these conditions causes the IMP to fill up with packets until its free list of buffers is empty or else until it is so busy that the IMP CPU is not available for processing modem inputs.

To summarize, each of the external events listed above (an overload at a particular host, line, or IMP) causes the IMP to reach the corresponding storage limit (source-to-destination buffers, store-and-forward buffers, or total buffers). At this point, congestion forms because that IMP does not acknowledge packets that it is receiving from the adjacent IMPs. This is the key event in the formation of a network disturbance: one IMP failing to acknowledge packets from the adjacent IMP or IMPs.

4.3 The Spread of Congestion

Once congestion has formed at a particular IMP, it spreads through the network in a variety of ways:

1. If many IMPs in the network are sending to the congested IMP, their traffic backs up away from the congested IMP, forming a tree of congestion emanating from that IMP. This traffic may be data packets being transmitted to that node, RFNMs in response to data traffic, or control traffic such as get-a-block messages being retransmitted at a high rate by other IMPs in the network wishing to initiate traffic to the congested node.

2. Congestion may spread by one stream of traffic interfering with another. This happens in one of two ways. The original stream of traffic is that destined to the congested node or on paths which flow through the congested node. Secondary congestion can build up on paths flowing in the same direction; IMPs along the path are blocked from transmitting to IMPs further along the path. Alternatively, it can build up on paths which intersect the congested path; traffic begins to build up on many different paths through the network tracing a series of expanding paths away from the affected area.

3. Congestion may spread from one area directly to another area in the network without all the adjacent IMPs becoming congested. This happens when the congested IMP is filled with certain kinds of traffic for some other IMP in the network, which causes that IMP to become congested itself. This may happen, for instance, when

the original congested IMP is unable to deliver control reply traffic to some IMP in the network. This IMP may then reach its reassembly limit because all its packet buffers are filled with messages which have been sent into the network and which are awaiting responses from the congested node. At this stage the second node can become the focus of a new area of congestion.

4.4 Disruption of Service

Once the network disturbance has spread to a number of nodes in the network, including many of the nodes three or four hops removed from the original focus, most network services in the immediate region are disrupted. The reason for this disruption is simple. Each IMP can assign a maximum of eight packets to a particular circuit in the network. Once it has assigned those packets to the circuit, it continues to retransmit them until they have been successfully acknowledged. Thus no new traffic can be accepted and introduced into the congested region. Therefore, once congestion forms, it is very slow to dissipate. The ultimate response to this congestion is that an IMP retransmits a particular packet 600 times on a line and then declares the line down. At this point, service is disrupted for a second reason: the user may find his connection closed because no network path exists from his node to his intended destination.

Once lines are declared down, congestion may be eliminated because all sources of traffic to the congested area discover

that it is unreachable. All IMPs in the network will then discard traffic for the unreachable IMPs.

5. Recommendations

In this section we present our recommendations for the corrective actions to be taken in preventing future ARPANET disturbances. Some of these recommendations represent minor IMP software changes and have, therefore, been implemented in the course of the present study. Others are more difficult changes which should be made only after a careful analysis of their impact. Finally, other suggestions we make are quite difficult and will require a more extensive period of study before they can be completely specified and implemented. In the paragraphs below we present our short-term recommendations first, followed by several longer-term recommendations for improvements.

1. Our primary recommendation is that the IMPs no longer kill circuits after retransmitting packets 600 times. During the course of this study we have lowered the retransmission limit successively from 600 to 100 to 50 and finally to 32. We have reprogrammed the IMPs so that they drop or reroute packets after retransmitting them 32 times, rather than declaring the circuit unusable. If, after 32 retransmissions of a particular packet, the IMP finds that the best route to that destination has changed, the packet is rerouted in the new, correct manner. If the route is still the same, the IMP discards the packet and frees up that logical channel for other traffic. This change has worked relatively well in the short time we have observed it; however, further investigation is recommended.

2. The line alive/dead logic has been improved by removing a bug in the code. This error caused one IMP to think that the line was down, while the other thought it was up, and caused this condition to persist for several minutes. We have removed the bug in which one IMP could be stuck with the very long timer before it declared the line up. On the other hand, we are still left with the problem that the line alive/dead logic uses an up/down counter, which means that every time an "I heard you" message is successfully received, the counter is incremented, and when one is not received, it is decremented. This means that if a circuit is just marginally usable the two IMPs at each end may count up at slightly different rates. In the worst case, one IMP may declare the line up, while the other may take arbitrarily long to determine the status of the line as it receives "I heard you" messages correctly and incorrectly in alternate time periods. Our recommendation is twofold. The algorithm must be changed so that there are explicit guarantees that both IMPs have determined that the line is up before traffic flows on it in either direction. Second, the alive/dead determination logic should be analyzed to determine if a consecutive counter or an up/down counter should be utilized, and to recompute the correct values of the various parameters.

3. We recommend that an analysis be performed of the correct timing parameters for retransmission of various control messages in the ARPANET. Presently, control messages such as the get-a-block, reset, and reset request messages are retransmitted

at a fairly high rate (every two seconds) which has the undesirable effect of aggravating a congested situation when one occurs. These control messages are retransmitted at a high rate, since the destination IMP is free to ignore them if it has no buffer storage for a reply. A retransmission strategy based on the urgency of the message and on the load of the network should be developed so that users get prompt service when that is possible and the network does not become overloaded during high traffic periods.

4. We recommend that each IMP should give priority to traffic originating at other IMPs (perhaps by reserving some store-and-forward packet buffers for tandem traffic). This is necessary so that host traffic originating at that node does not use all of the available store-and-forward buffers. In this case, the IMP would not be able to act as a store-and-forward node for traffic flowing through it to other destinations. Simulation studies conducted elsewhere have indicated that setting aside two store-and-forward buffers at each node for this purpose is adequate. An analysis should be made of the effect of this change on ARPANET performance.

5. The IMP should guarantee a certain amount of input buffer storage and CPU processing time for routing messages. Presently, it is possible for the IMP to lose routing data when it is very busy. Such losses must be expected and provided for, since routing messages can always be lost due to line errors or

hardware failures. On the other hand, it is very undesirable for the IMPs to lose routing information as traffic loads increase, since it is at that time that routing updates are most critical.

6. The internal background process which generates reply messages should act on a round robin basis rather than first-in first-out. Presently, this process takes the next reply message to be sent to the network and attempts to submit it to the TASK processing routine. If there is no room for this particular message to be accepted, the background process continues to resubmit it whenever possible. Thus a packet for a congested destination can serve as a bottleneck for other reply traffic which could be transmitted into the network.

If these six recommendations were implemented, we believe that routing disturbances would be less frequent and their effects would be less severe. On the other hand, traffic overloads in particular areas of the network would still lead to local congestion. Under some circumstances (if the traffic built up very quickly, or if routing was too slow) network-wide disturbances could result. Therefore, we make the following longer-term recommendations for the ultimate elimination of all network disturbances:

1. The so-called hold down mechanism in routing should be replaced by a better means for adaptation to changes. Presently, the ARPANET routing algorithm relies on the hold down mechanism for determining if a particular change in traffic or topology

should result in a change in the routing tables. Each IMP continues to use its best path to a certain destination for two seconds after that path deteriorates in order to allow the information about the deterioration of the path to propagate to the immediately adjacent IMPs. This is necessary in order to avoid the problem of using the route through the adjacent IMPs, which appears to be better than the recently deteriorated path, but which is in fact merely a reflection of the IMPs own old routing information. The hold down mechanism has been observed to fail under certain stress conditions such as multiple simultaneous routing changes in the network, long paths of different line speeds, and high traffic load conditions. It is possible that the best solution in this case is to provide a more explicit means of determining whether the adjacent IMPs have performed an update rather than relying on the simple timing mechanism which is the basis for hold down.

2. We recommend that the routing algorithm be changed so that it can determine changes in ARPANET topology within a period of one or two seconds. Presently, such determination can take 10 seconds or more, which is long enough for congestion to build up along the path to a unreachable IMP. We recommend that an investigation be carried out of line-based routing updates rather than path-based routing updates. In such systems the IMPs adjacent to a failure would send out short messages concerning the status of individual lines to which they are connected. Each IMP in the network would store a map of the complete network

topology in its memory and would accept such update messages and recalculate whether the destination IMP was still reachable in the new network topology.

3. We recommend that an analysis be conducted of explicit means for congestion prevention and control. The IMPs should deal with the formation and spread of congestion in some explicit manner. They should measure it and meter traffic flowing through the network so that it avoids congested areas. Further, the buffering strategy within the IMPs should be analyzed to insure that some sort of feedback control mechanism based on queue lengths or buffer counts could be implemented if appropriate.

4. A related recommendation is that the IMP program be analyzed to insure that all detectable lockup conditions such as store-and-forward lockup and reassembly lockup are prevented and that any unforeseen lockup conditions are detected by the IMP program in such a way as to avoid causing a network-wide disturbance. If the routing algorithm is improved to the point that it can avoid congested areas, and if packets assigned to output lines are re-examined from time to time and rerouted in different directions if appropriate, then most lockup conditions in the store-and-forward subnetwork can be avoided. Likewise, if all source-to-destination processes allocate storage for their messages before initiating traffic, then most reassembly or end-to-end lockup conditions can be avoided.

In conclusion, it is our recommendation that a number of steps be taken immediately to reduce the frequency and severity of network disturbances and to improve the overall functioning of the routing and store-and-forward processes in the IMP. On a longer-term basis we recommend a more complete re-examination of the routing process with the goals of making the routing update process faster and making the routing algorithm explicitly aware of congestion. If these steps are taken, then it is possible that the ARPANET would experience only momentary local congestion and not be subject to widespread network disturbances.

Appendix I. Measurement Package Output

Figure 7 shows a sample page of the output that is produced after a disturbance. The key in the figure describes the meaning of each entry. The output shows that Scott, IMP59, triggered 68 seconds after the start of the disturbance. (Times recorded at different IMPs are accurate to about 3 seconds.) A simple calculation, using the SYNC times, shows that the first snapshot was taken about 2.2 seconds before the trigger, and the second, about 1.0 seconds after the trigger. The trigger occurred because a packet on Line 3 was discarded. (The D on Line 4, whose neighbor is shown as IMP 0, indicates a nonexistent line.) The list below explains briefly the counters and queues whose values are shown in the snapshot:

FREE - free-buffer count

SF - store and forward count; equal to the number of packets on the reroute queue plus the total number queued on all output lines (but excluding the first packet queued on each line).

REA - reassembly count; total number of packets excluding those on the task queue and those counted in store and forward.

AL - allocate count; buffers allocated to incoming traffic.

TQ - length of task queue; buffers for packets to be processed.

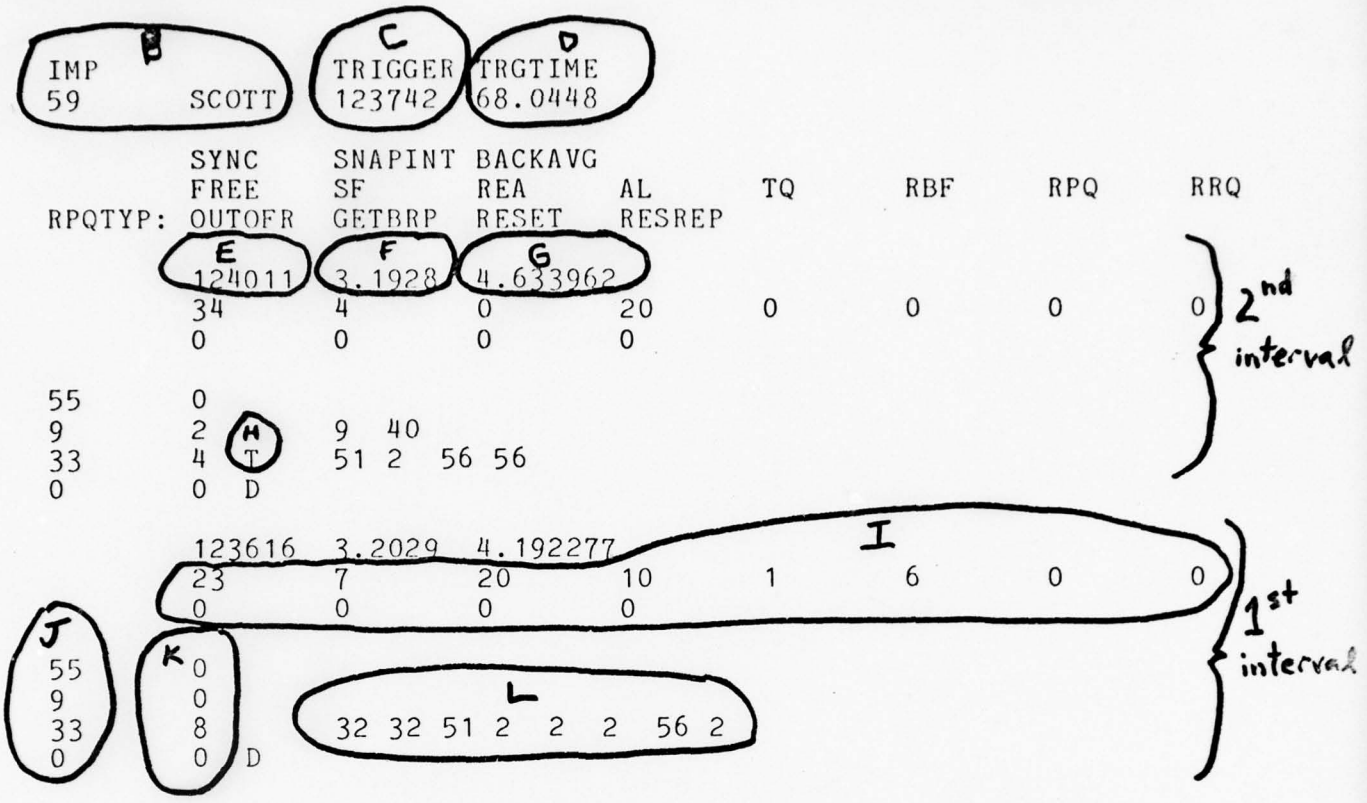
RBF - reassembly buffers; buffers for packets in reassembly, a part of the reassembly count.

RPQ - reply queue; replies generated by task.

RRQ - reroute queue; for packets that are to be rerouted. Originally rerouting took place because of channel failures; currently, exceeding the retransmission limit can also cause a packet to be rerouted.

RPQTYP - Four components of the reply queue are shown: out-of-range packets, get-a-block replies, resets, and reset replies.

The pages following Figure 9 show the complete output from Experiment I, 2 August 1977 (see section 3.3.C above).



Key

A - Date and time of disturbance.

B - IMP number and name.

C - Time of trigger in octal (16-bit, absolute)

D - Time of trigger in sec, relative to first trigger.

E - Time of snapshot, octal, absolute.

F - Elapsed time in sec from previous snapshot.

G - Duration of average background loop in msec.

H - Code for cause of trigger:

D = Down
N = Received Special Null
L = Reached Retransmission Limit
R = Rerouted Packet
T = Discarded Packet
K = Killed line; missed 5 I Heard You's

I - Octal values of the various counts and queues itemized in header (see text).

J - Neighbor IMP numbers.

K - Number of packets queued on each line.

L - Destination IMP numbers of queued packets.

(-1 means a dropped packet)

Figure 7: Sample Measurement Output

```
IMP                     TRIGGER TRGTIME
44          MIT44       5357    0

            SYNC        SNAPINT BACKAVG
            FREE        SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:     OUTOFR      GETBRP  RESET   RESREP

            5522        3.199   7.172646
            43          0       1       0       0       0       0       0
            0           0       0       0

6           0  D
10          0
0           0  D
0           0  D

            5324        3.199   7.046256
            42          0       2       0       0       0       0       0
            0           0       0       0

6           0  D
10          0
0           0  D
0           0  D

IMP                     TRIGGER TRGTIME
47          WPAFB       5374    .3328

            SYNC        SNAPINT BACKAVG
            FREE        SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:     OUTOFR      GETBRP  RESET   RESREP

            5546        3.2023  6.417435
            43          0       1       0       0       0       0       0
            0           0       0       0

6           0  D
12          0
0           0  D
0           0  D

            5350        3.2002  6.312032
            42          0       1       0       0       0       0       0
            0           0       0       0

6           0  D
12          0
0           0  D
0           0  D
```

```
IMP                     TRIGGER TRGTIME
44          MIT44       173270  1545.242

            SYNC        SNAPINT BACKAVG
            FREE        SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:     OUTOFR      GETBRP  RESET   RESREP

            173320      3.1998  7.063576
            30          6       4       0       0       0       0       0
            0           0       0       0

6           7  DL       2  2  2  2  6  6  6
10          0
0           0  D
0           0  D

            173125      3.1992  8.764932
            33          3       4       0       0       0       0       0
            0           0       0       0

6           4           2  2  2  2
10          0
0           0  D
0           0  D

IMP                     TRIGGER TRGTIME
47          WPAFB       173275  1545.37

            SYNC        SNAPINT BACKAVG
            FREE        SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:     OUTOFR      GETBRP  RESET   RESREP

            173343      3.204   6.149712
            33          5       2       0       0       0       0       0
            0           0       0       0

6           6  DL       20 20 5  5  40 40
12          0
0           0  D
0           0  D

            173146      3.1984  7.725604
            34          5       1       0       0       0       0       0
            0           0       0       0

6           6           20 20 5  5  40 40
12          0
0           0  D
0           0  D
```

```
IMP                 TRIGGER TRGTIME
29        ABER      174267  1558.323

          SYNC      SNAPINT BACKAVG
          FREE      SF      REA       AL        TQ        RBF       RPQ       RRQ
RPQTYP:   OUTOFR    GETBRP  RESET     RESREP

          174405    3.1908  1.865965
          33        7       0         0         0         0         0         0
          0         0       0         0

46        0
19        8  DL     6  6  6  6  6  6  6  6
27        0
0         0  D

          174243    3.2081  1.981532
          31        7       0         0         2         0         2         0
          0         0       0         0

46        0
19        8         6  6  6  6  6  6  6  6
27        0
0         0  D

IMP                 TRIGGER TRGTIME
19        NBS       174313  1558.835

          SYNC      SNAPINT BACKAVG
          FREE      SF      REA       AL        TQ        RBF       RPQ       RRQ
RPQTYP:   OUTOFR    GETBRP  RESET     RESREP

          174400    3.2236  6.283821
          27        7       3         0         0         0         0         0
          0         0       0         0

57        8  DL     6  6  6  6  6  6  6  6
29        1  N      40
26        0  N
0         0  D

          174202    3.1969  10.34595
          22        7       2         0         7         0         7         0
          0         0       0         0

57        8         6  6  6  6  6  6  6  6
29        0
26        0
0         0  D
```

```
IMP                 TRIGGER TRGTIME
57        NSA       174330  1559.168

          SYNC      SNAPINT BACKAVG
          FREE      SF      REA       AL        TQ        RBF       RPQ       RRQ
RPQTYP:   OUTOFR    GETBRP  RESET     RESREP

          174522    3.2     2.442748
          34        7       0         0         0         0         0         0
          0         0       0         0

17        0  N
19        8  DL     6  6  6  6  6  6  6  6
0         0  D
0         0  D

          174324    3.1998  3.137059
          34        7       0         0         0         0         0         0
          0         0       0         0

17        0
19        8  N      6  6  6  6  6  6  6  6
0         0  D
0         0  D

IMP                 TRIGGER TRGTIME
26        PENT      174335  1559.296

          SYNC      SNAPINT BACKAVG
          FREE      SF      REA       AL        TQ        RBF       RPQ       RRQ
RPQTYP:   OUTOFR    GETBRP  RESET     RESREP

          174530    3.201   5.566957
          27        7       2         0         2         0         2         0
          0         0       0         0

53        0
8         0
19        8  DL     6  6  6  6  6  6  6  6
0         0  D

          174332    3.2001  8.205385
          31        7       1         0         1         0         1         0
          0         0       0         0

53        0
8         0
19        8         6  6  6  6  6  6  6  6
0         0  D
```

```
IMP                     TRIGGER TRGTIME
17      MITRE           174506  1561.984

        SYNC    SNAPINT BACKAVG
        FREE    SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP: OUTOFR  GETBRP  RESET   RESREP

        174664  3.2023  8.404987
        36      1       0       0       2       0       2       0
        0       0       0       0

13      0
57      2  DL   26 26
28      0
39      0

        174470  3.2013  9.443363
        35      1       1       0       2       0       2       0
        0       0       0       0

13      0
57      2       26 26
28      0
39      0

IMP                     TRIGGER TRGTIME
19      NBS             174544  1562.752

        SYNC    SNAPINT BACKAVG
        FREE    SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP: OUTOFR  GETBRP  RESET   RESREP

        174575  3.2763  4.105639
        27      7       3       0       0       0       0       0
        0       0       0       0

57      8  D    6  6  6  6  6  6  6  6
29      1  DK   40
26      0  DK
0       0  D
```

```
IMP                     TRIGGER TRGTIME
27        BELV          174620  1563.878

          SYNC          SNAPINT BACKAVG
          FREE          SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:   OUTOFR        GETBRP  RESET   RESREP

          174666        3.2022  4.043182
          33            7       0       0       0       0       0       0
          0             0       0       0

14        0
20        0
29        8  DL         6  6  6  6  6  6  6  6
0         0  D

          174467        3.2002  4.289812
          32            7       0       0       1       0       1       0
          0             0       0       0

14        0
20        0
29        8            6  6  6  6  6  6  6  6
0         0  D

IMP                     TRIGGER TRGTIME
53        EGLIN         174761  1566.362

          SYNC          SNAPINT BACKAVG
          FREE          SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:   OUTOFR        GETBRP  RESET   RESREP

          175107        3.199   3.492358
          15            7       0       0       0       0       0       0
          0             0       0       0

13        0
26        8  DL         6  6  6  6  6  6  6  6
0         0  D
0         0  D

          174712        3.2013  4.078089
          15            6       0       0       1       0       1       0
          0             0       0       0

13        0
26        7            6  6  6  6  6  6  6
0         0  D
0         0  D
```

```
IMP                 TRIGGER TRGTIME
29       ABER       175014  1567.053

         SYNC       SNAPINT BACKAVG
         FREE       SF      REA     AL       TQ       RBF      RPQ      RRQ
RPQTYP:  OUTOFR     GETBRP  RESET   RESREP

         175052     3.2005  1.706933
         43         0       0       0        0        0        0        0
         0          0       0       0

46       0
19       0  D
27       0  DK
0        0  D

         174732     3.1982  1.870292
         33         10      0       0        0        0        0        0
         0          0       0       0

46       0
19       0  D
27       0  N
0        0  D

IMP                 TRIGGER TRGTIME
14       CMU        175110  1568.589

         SYNC       SNAPINT BACKAVG
         FREE       SF      REA     AL       TQ       RBF      RPQ      RRQ
RPQTYP:  OUTOFR     GETBRP  RESET   RESREP

         175161     3.1997  1.680515
         31         7       0       0        1        0        1        0
         0          0       0       0

55       0
27       8  DL      6  6  6  6  6  6  6  6
18       1          18
0        0  D

         174765     3.1987  1.729962
         30         7       0       0        3        0        3        0
         0          0       0       0

55       0
27       8          6  6  6  6  6  6  6  6
18       0
0        0  D
```

```
IMP                     TRIGGER TRGTIME
57          NSA         175114  1568.691

            SYNC        SNAPINT BACKAVG
            FREE        SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:     OUTOFR      GETBRP  RESET   RESREP

            175311      3.201   2.018285
            40          2       1       0       0       0       0       0
            0           0       0       0

17          3  DK       40 40 40
19          0  D
0           0  D
0           0  D

            175114      3.2028  2.40812
            40          2       1       0       0       0       0       0
            0           0       0       0

17          3           40 40 40
19          0  D
0           0  D
0           0  D

IMP                     TRIGGER TRGTIME
26          PENT        175203  1570.099

            SYNC        SNAPINT BACKAVG
            FREE        SF      REA     AL      TQ      RBF     RPQ     RRQ
RPQTYP:     OUTOFR      GETBRP  RESET   RESREP

            175315      3.198   5.216966
            42          0       1       0       0       0       0       0
            0           0       0       0

53          1  DK       22
8           0
19          0  D
0           0  D

            175121      3.2284  5.240909
            30          10      3       0       0       0       0       0
            0           0       0       0

53          0  N
8           0
19          0  D
0           0  D
```

```
IMP                 TRIGGER TRGTIME
27       BELV       175312  1571.917

         SYNC       SNAPINT BACKAVG
         FREE       SF      REA       AL       TQ       RBF      RPQ      RRQ
RPQTYP:  OUTOFR     GETBRP  RESET     RESREP

         175454     3.1992  2.539048
         40         2       0         0        0        0        0        0
         0          0       0         0

14       3  DK      40 40 40
20       0
29       0  D
0        0  D

         175261     3.2261  2.631403
         37         2       1         0        0        0        0        0
         0          0       0         0

14       3  N       40 40 40
20       0
29       0  D
0        0  D
```

Appendix II. Message Generators Used in Experiments

Here, for reference purposes, we list the message generators used during the experiments and referred to in Section 3.

1. Case I - with line 53, between FNWC and SCOTT, down, traffic in Groups A and B flows away from line 53. Group C adds further traffic.

IMPs		# HOPS via line 53	line 53 down	
From	To		Not via IMP33	Via IMP33
21 (LLL)	9 (HARV)	6	9	(12)
32 (XEROX)	37 (DEC)	5	10	(11)
43 (TYMS)	5 (RCC5)	6	10	(12)
54 (SCRL)	59 (SCOTT)	2	6	(>>6)
1 (UCLA)	59	3	5	(>>5)
		Group IA		
			Not via IMP59	Via IMP59
5 (RCC5)	43 (TYMSH)	6	10	(12)
18 (RADC)	43	5	9	(12)
10 (LINC)	33 (FNWL)	4	9	(10)
37 (DEC)	2 (SRI2)	6	9	(12)
9 (HARV)	2	5	10	(11)
		Group IB		
		via line 53	line 53 down	
55 (ANL)	43 (TYMSH)	3	7	
24 (GWC)	43	4	6	
14 (CMU)	43	4	8	
33 (FNWC)	59 (SCOTT)	1	7	
59 (SCOTT)	33 (FNWC)	1	7	
		Group IC		

All generators were set for long messages at maximum frequency.

2. Reduced Case I comprises the following subset of the above sources: IMPs 21, 32, 1, 5, 10, 37, 14, and 33.

3. Case II - with line 53 down, traffic in Groups A and B flows toward line 53. Group C is as in Case I.

IMPs			# HOPS	
			via line 53	line 53 down
From	To			
2 (SRI2)	59	(SCOTT)	4	10
32 (XEROX)	59		3	9
43 (TYMS)	59		2	8
54 (SCRL)	47		8	9
			(11 if not via IMP33)	
		Group IIA		
37 (DEC)	33	(FWWC)	3	9
40 (BBN)	33		3	9
9 (HARV)	33		2	8
58 (NYU)	33		3	9
49 (RCC49)	33		4	10
			(12 if not via IMP59)	
		Group IIB		
55 (ANL)	43	(TYMSH)	3	7
24 (GWC)	43		4	6
14 (CMU)	43		4	8
33 (FNWC)	59	(SCOTT)	1	7
59 (SCOTT)	33	(FNWC)	1	7
		Group IIC (same as IC)		

As in Case I, generators were set for long messages at maximum rate.

4. Case III - the Reduced Case I generators, plus the following:

From	To
43	5
54	59
18	43
9	2
55	43
24	43
59	2

These seven generators produced single packet messages at a rate of one every 6 minutes. The low rate causes connections to be broken and re-established between messages, thereby causing control messages to be generated. With seven such generators, roughly staggered in time, there is a reasonable chance that a connection change will occur during the three minutes that IMPs were slowed in the experiments.

5. Case IV - traffic normally via GUNTER, IMP 13.

IMPs		# HOPS	
From	To	via IMP 13	not via 13
57 (NSA)	48 (AFWL)	4	11
62 (TEXAS)	19 (NBS)	4	11
53 (EGLIN)	17 (MITRE)	2	4
26 (PENT)	22 (ISI)	5	10

These generators were set to send single-packet messages every 100 milliseconds.

APPENDIX III. Summary of Network Disturbances and Experiments

July 5 (Experiment)

This was the first attempt to cause a network disturbance under controlled conditions. The software in the SIMPs at SDAC and NORSAR was modified to permit only eight logical channels per satellite line rather than the usual 16. In addition, message generators were set up to direct traffic at various rates from BBN40 to NORSAR. All such traffic must travel through SDAC. However, no disturbance formed, even when the lines from SDAC were looped away.

Next, the line between BBN40 and HARV was caused to go up and down at a frequency of once every 20 seconds. This line is on the shortest path from BBN40 to NORSAR. However, no disturbance resulted, even when additional artificially generated traffic was directed to the line from both directions.

It had been noted before this experiment that network disturbances often seemed to occur after one or more lines had been going up and down rapidly for a period of time. It had also been noted that disturbances occurred more frequently when the SIMPs had eight logical channels than when they had 16. This experiment proved that neither of these conditions was sufficient to cause a disturbance, even when an attempt was made to cause congestion by artificially generating traffic.

July 6 (Spontaneous)

An IMP cannot accept a store-and-forward (S/F) packet if all its S/F buffers are full. Whenever an IMP fails to accept a packet from its neighbor, the neighbor will retransmit the packet up to some number of times, and then take some special action. At the time the software was configured so as to retransmit a packet up to 600 times. If a packet was retransmitted 600 times without being accepted by the neighbor, the transmitting IMP killed its line to the neighbor; and snapshots were taken of the IMPs on both sides of the line.

This disturbance began when line 53 (FNWC-SCOTT) went up and down several times. Traffic backed up from FNWC to SCOTT, and then from SCOTT to ANL and HARV, as all the S/F buffers in these IMPs filled up. Thus the lines connecting FNWC to SCOTT and SCOTT to ANL and HARV, were all killed after traffic backed up. Approximately one minute later, 14 other lines in the network were killed, as many other IMPs began to retransmit packets for the 600th time. The snapshots show that most of these lines were on paths directed toward the initial disturbance (FNWC, SCOTT, ANL, HARV), and that a lack of store-and-forward buffers was always responsible for the packets' not being accepted.

Thus it seems that what began as physical trouble with a single line (53) caused traffic to back up throughout the network due to store-and-forward congestion. This caused additional lines to be killed, which caused additional congestion, etc.

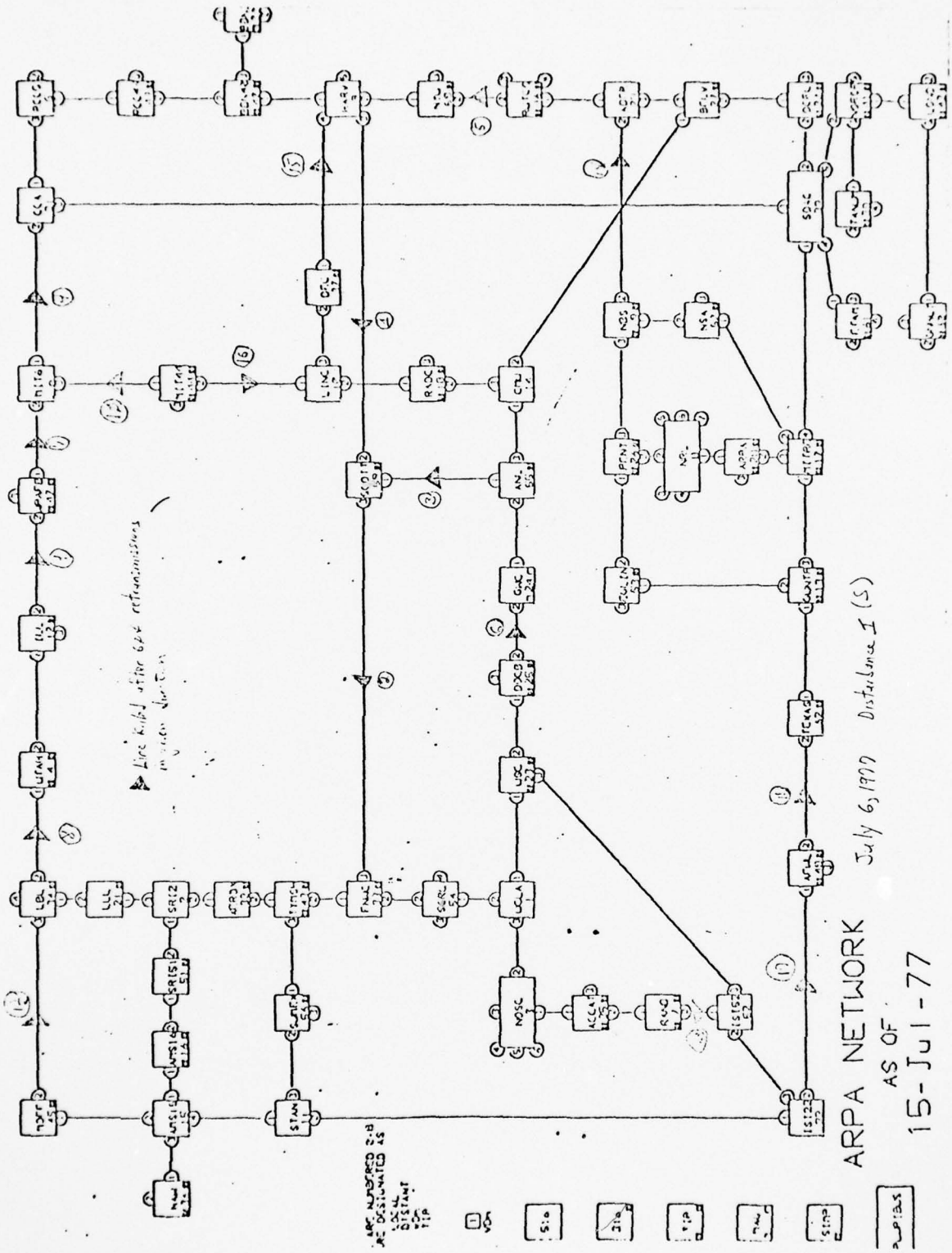


Figure 8

BEST AVAILABLE COPY

July 12 (Experiments)

It had been noted that there was a bug in the IMPs' line up/down logic, which could sometimes cause a line to appear at one end to be up while appearing at the other end to be down. The two experiments performed on this day tested the hypothesis that this problem could be a cause of network disturbances. The error was simulated by artificially causing the line from FNWC to SCOTT to appear up to FNWC and down to SCOTT for a period of 2 minutes. At the same time, message generators were set up to generate a lot of traffic which would use this line whenever it was up. Two experiments were performed, with two different sets of message generators.

Experiment I

This experiment used the Case I message generators (see Appendix II). Traffic was generated which would use the FNWC-SCOTT line when it was up, but which would be rerouted away from FNWC when that line was down. Again, 600 retransmissions of a packet caused the line on which the packet was transmitted to be killed. This experiment resulted in a very small disturbance. Only four lines were killed; none more than 3 hops from FNWC.

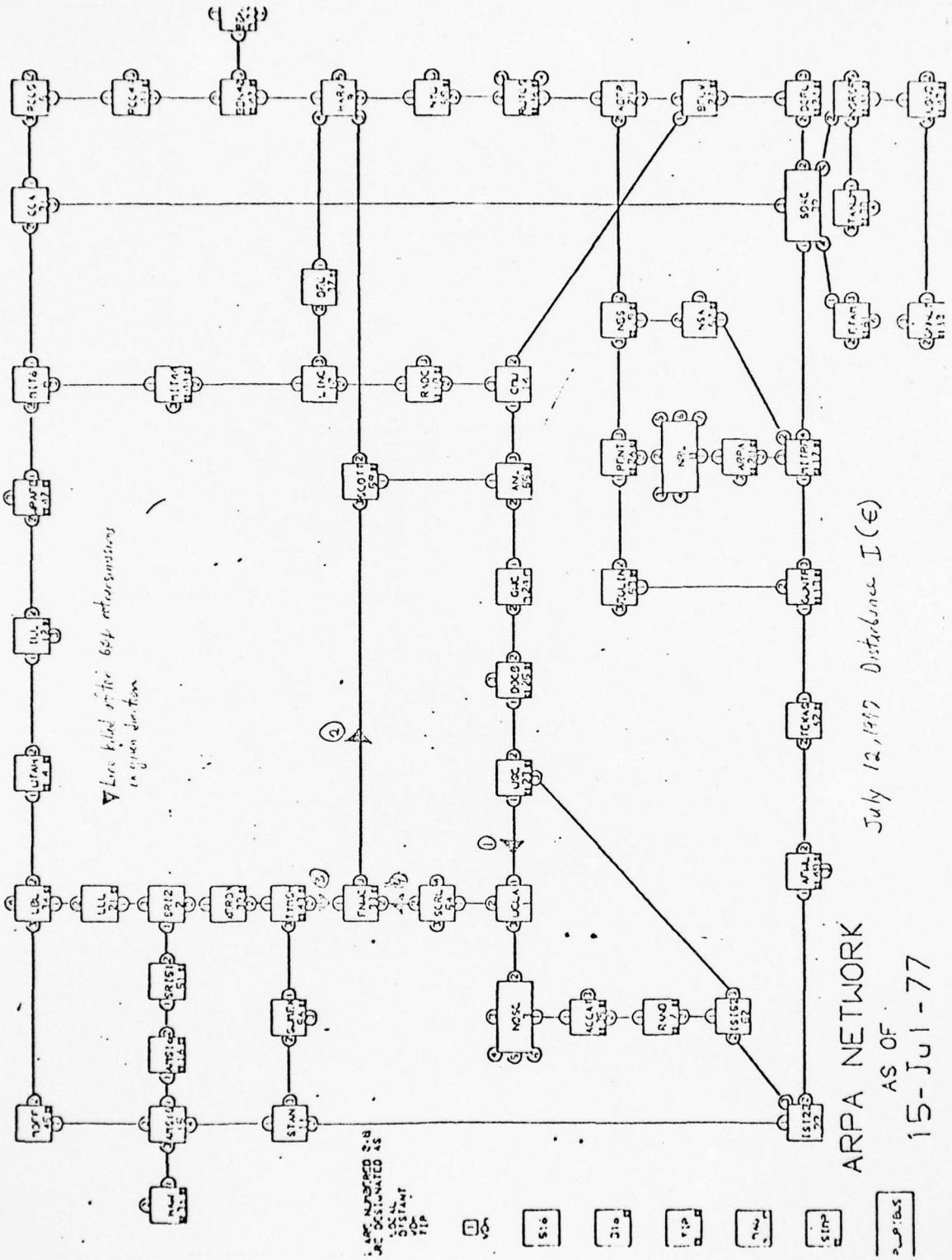


Figure 9

BEST AVAILABLE COPY

Experiment II

This was the same as experiment I, except that the Case II message generators were used (see Appendix II), causing traffic to flow towards FNWC, even when the FNWC-SCOTT line was down. This caused quite a large disturbance. Traffic backed up from SCOTT to FNWC; and then backed up from FNWC to the Los Angeles area in one direction, and in the other direction to UTAH and ILL. Within one minute, 12 lines were killed due to 600 retransmissions of a packet. As it happened, this resulted in all paths out of the San Francisco area going down, causing the network to become partitioned. There are several important conclusions to be drawn from this experiment:

1. The error in the line up/down logic could indeed be instrumental in causing a disturbance.

2. Whether or not a disturbance occurs depends at least as much on the traffic pattern as on the traffic load. There are two relevant factors here:

a. Traffic flow directly into a congested area is important in causing a disturbance.

b. As congestion spreads, IMPs kill their lines due to 600 retransmissions, and hence can become isolated. The exact pattern of IMPs which become isolated determines whether the disturbance remains local, or whether it has a more global effect, such as partition. But the pattern of isolated IMPs is itself determined by the traffic pattern.

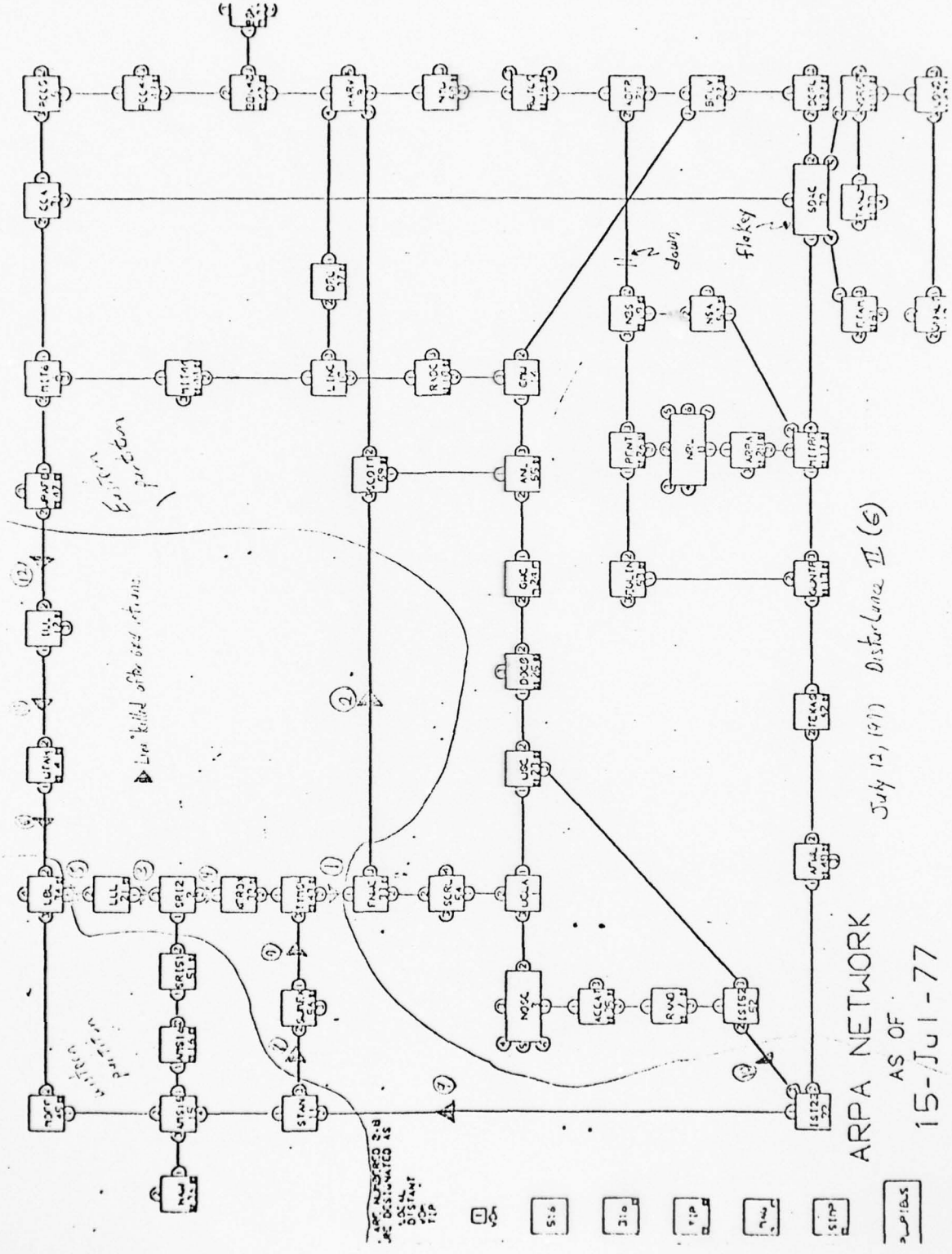


Figure 10

BEST AVAILABLE COPY

July 13 (Spontaneous)

Quite a significant spontaneous disturbance occurred, for no particular reason that we can identify. Although 15 lines were killed, the whole disturbance lasted only 25 seconds. Snapshots showed congestion backing up from MITRE in all directions. But they contained no information to help us to further our understanding of the phenomenon of network disturbances.

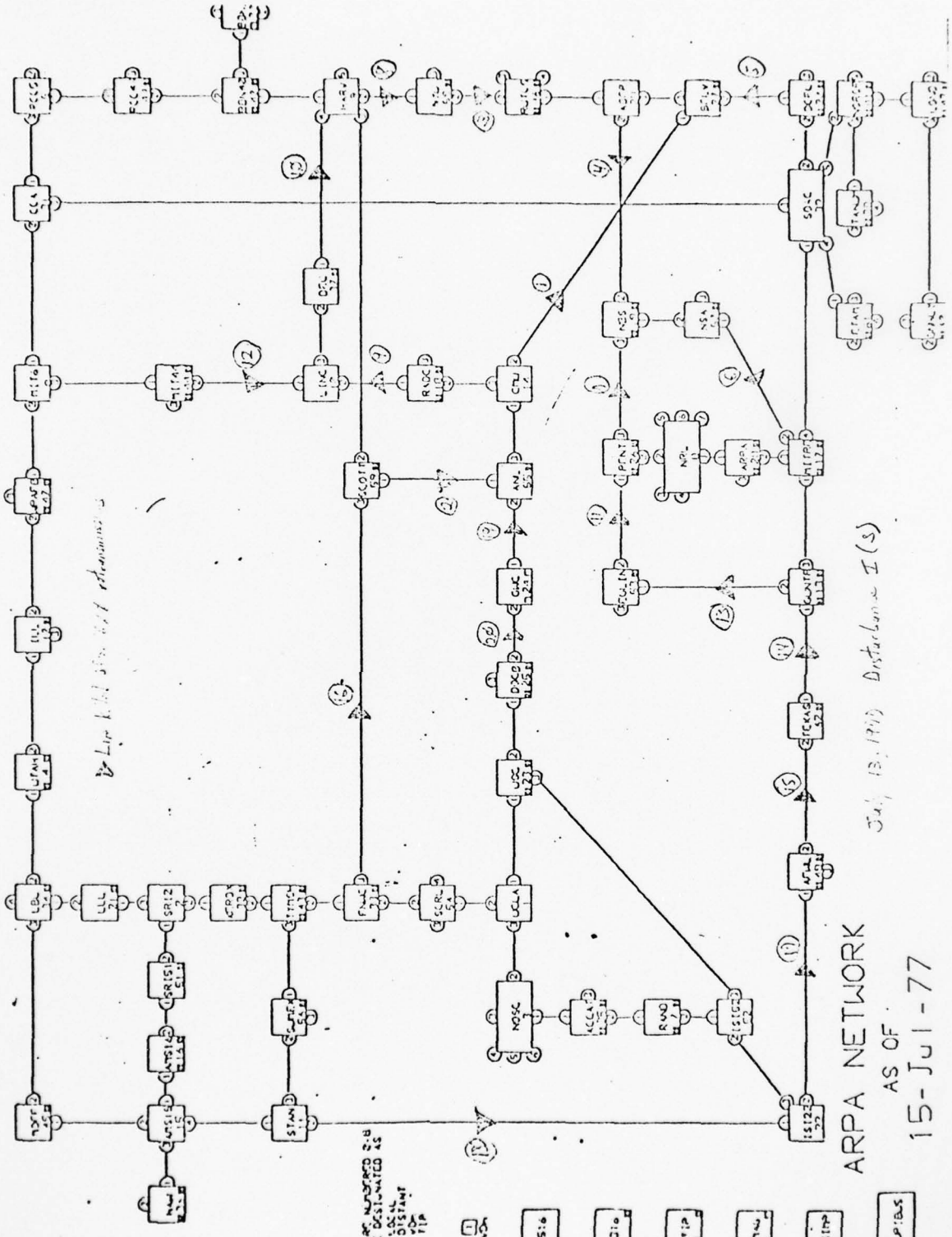


Figure 11

BEST AVAILABLE COPY

July 14 (Spontaneous)

Two spontaneous disturbances occurred, apparently as a result of software problems in the Pluribus IMP at SDAC. There was a bug which caused SDAC to believe erroneously that it had no free buffers. In both cases, this caused traffic to back up into the D.C. area.

A third spontaneous disturbance occurred, for unknown reasons.

Disturbance I

Traffic first backed up in the D.C. Area. Congestion was focused on NSA, and lines began to go down within 2 hops of NSA. After a lapse of about 20 seconds, congestion began to build in the Los Angeles area, backing up into the San Francisco area, causing many lines in those areas to go down. Approximately 30 seconds after the disturbance reached California, it spread to the Northeast, focused at BBN.

At this time our snapshots did not tell us packet destinations, so it is hard to say why or how the disturbance spread in the way it did. A plausible explanation is that IMPs in each area of the disturbance were engaged in communication with IMPs in the other areas. As lines began to drop off, traffic from one area could not get to others. Hence traffic began to build up in the area of its origin, as routing attempted to find a path to the destination.

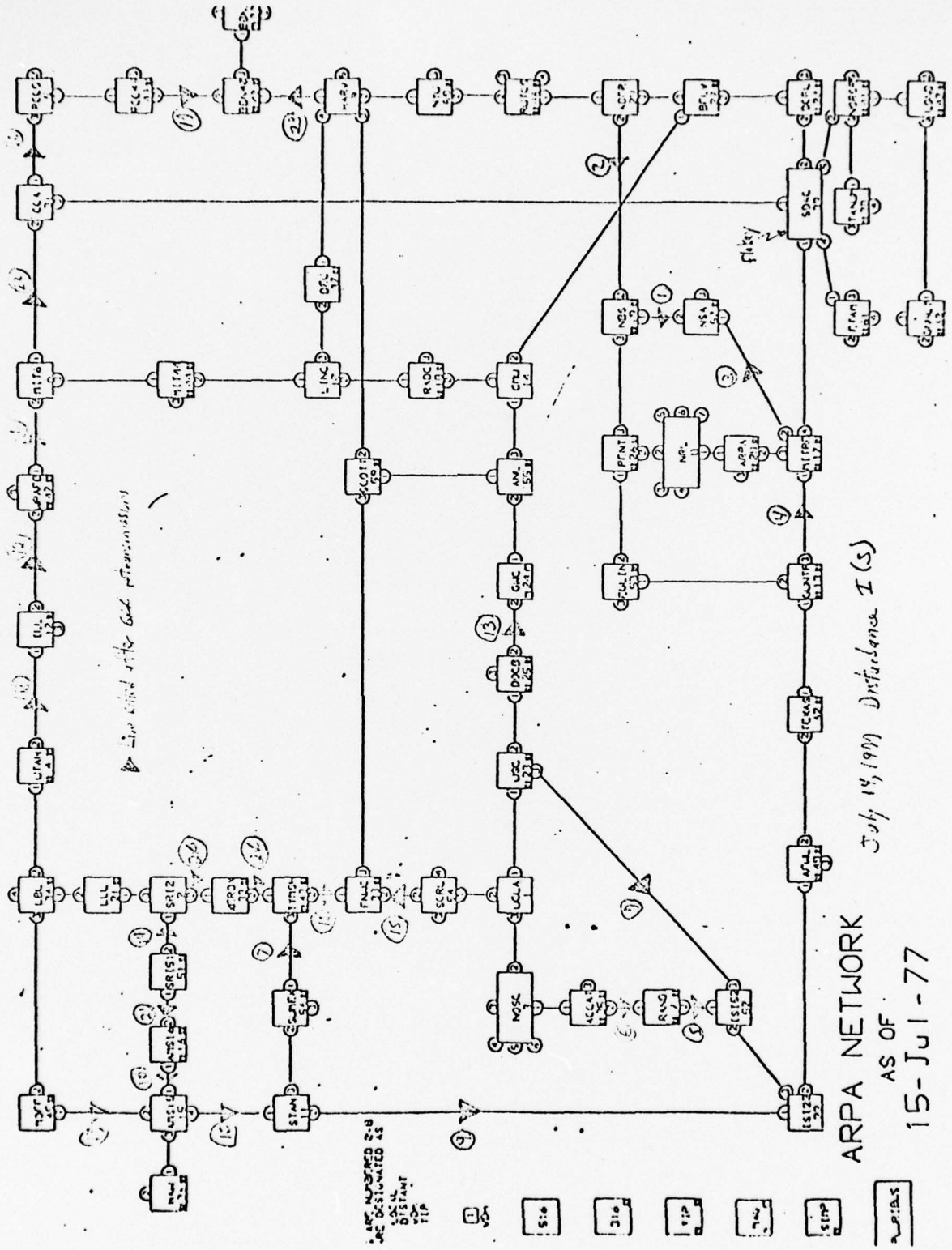


Figure 12

BEST AVAILABLE COPY

Disturbance II

In this disturbance, traffic backed up from SDAC into part of the Northeast, as well as into Washington and through to Los Angeles. As lines went down in these areas, traffic in the San Francisco area apparently had trouble getting out, causing the disturbance to spread into that area. It seems that some traffic from S.F. did manage to get to the unaffected portion of the Northeast, where it could not proceed further. This caused lines to go down in the rest of the Northeast, as well as the rest of the San Francisco area.

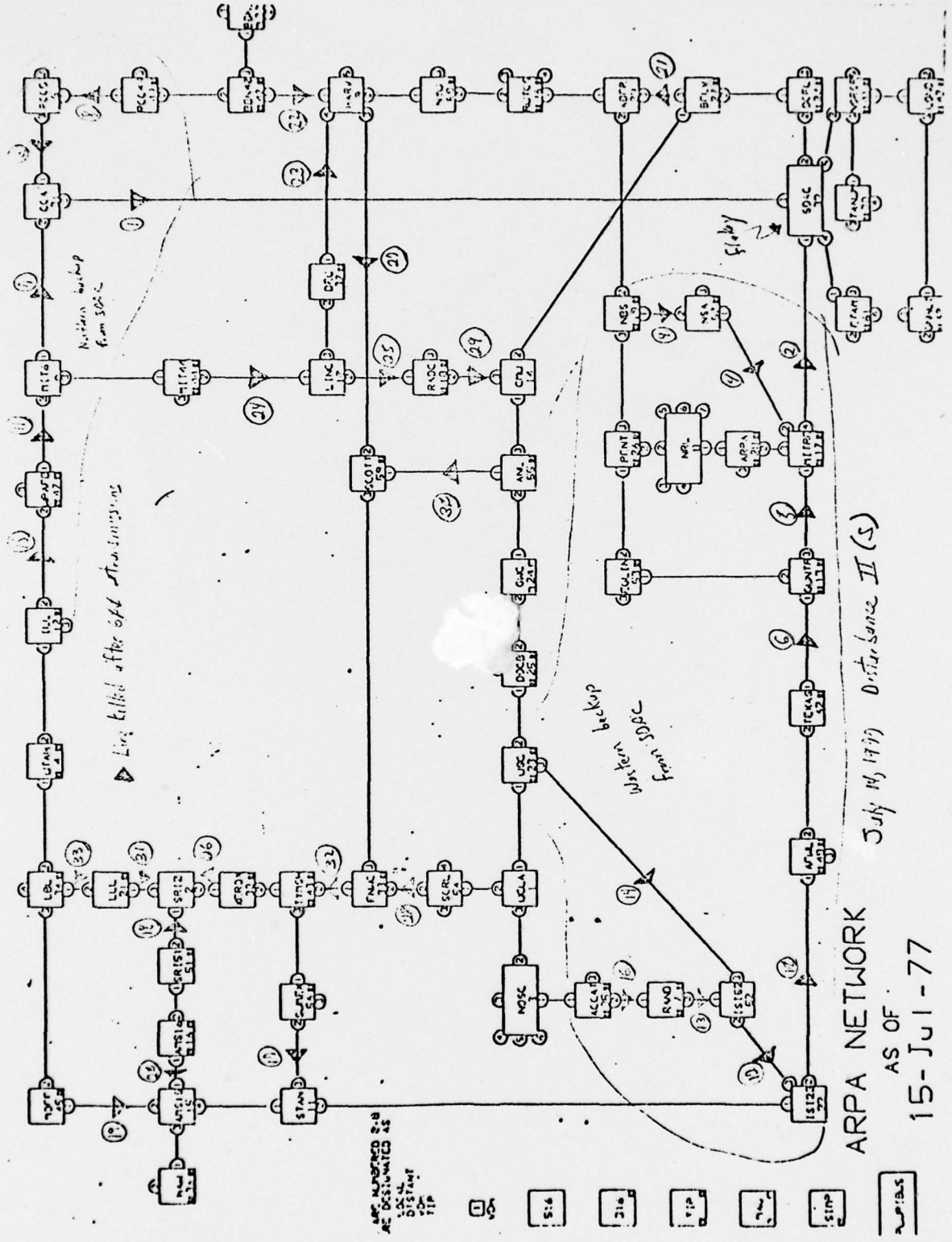


Figure 13

BEST AVAILABLE COPY

Disturbance III

A third disturbance occurred on July 14 which did not begin with congestion in the D.C. area; rather, congestion began to build along the path from MIT6 to UTAH. A minute later, the congestion had backed up all through the Northeast and around to San Francisco, causing many lines to drop out. Twenty seconds after the disturbance hit San Francisco, it began in the Los Angeles area.

A plausible explanation is the following. After the disturbance hit the Northeast, packets from San Francisco to the Northeast attempted to get there via Los Angeles. They joined with packets originating in Los Angeles which were already trying to get to the Northeast via Washington. All this traffic funneled out through ISI22, which became a second focus of congestion (the first being UTAH). However, the disturbance in the Northeast had completely isolated it, and this apparently caused traffic to stagnate in the Los Angeles area. If this hypothesis is true, then routing is incorrectly picking alternate paths which are no better than the original path.

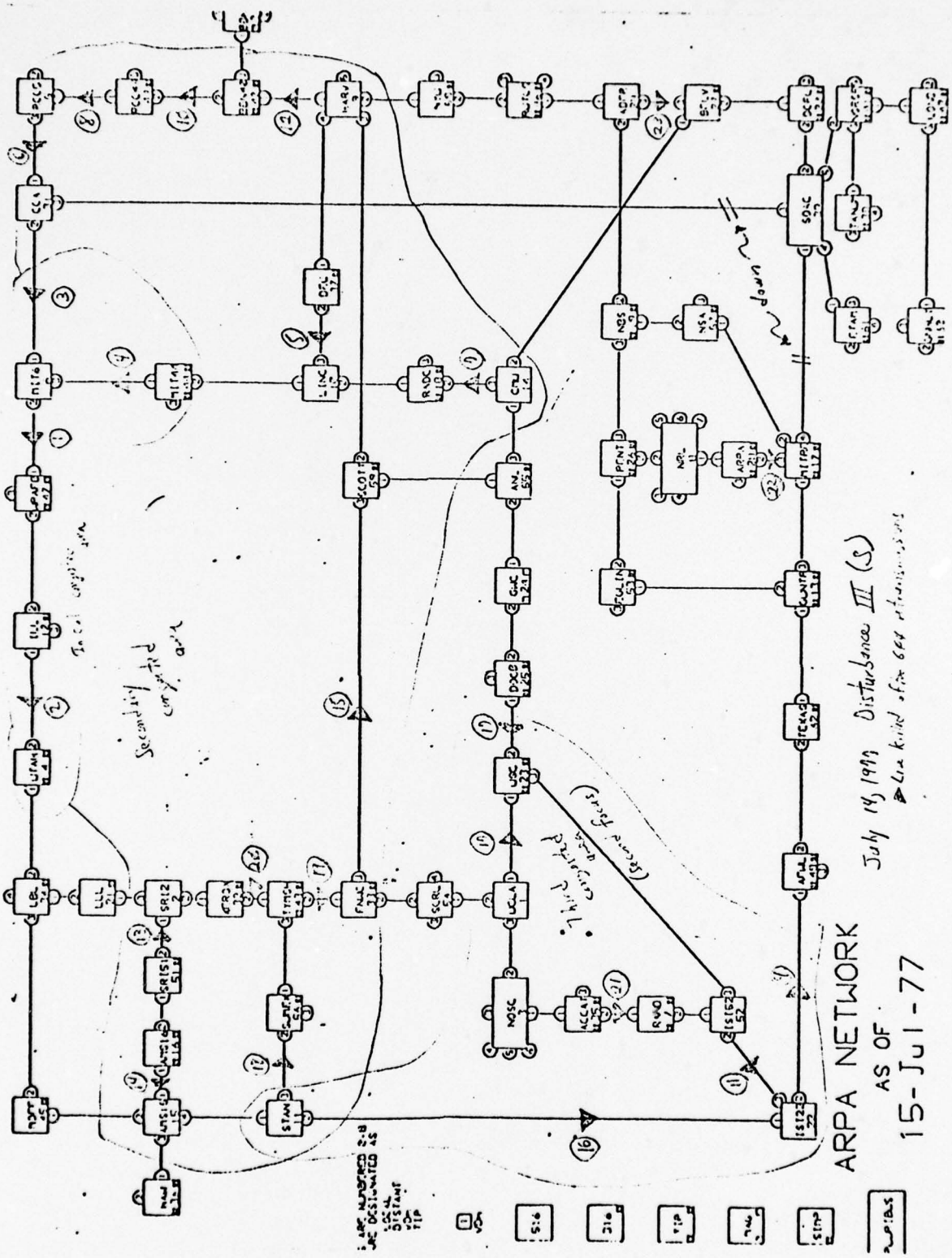


Figure 14

BEST AVAILABLE COPY

July 15 (Spontaneous)

There were two spontaneous disturbances.

Disturbance I

Traffic backed up from BBN40 along all the best paths to the San Francisco and Los Angeles areas. BBN40 had no free buffers at that time (this may have been caused by the sudden failure of a busy host at BBN, which would have left the IMP full of undeliverable packets). This makes it plausible to suppose that most of the IMPs along these paths were filled with traffic to BBN40. Congestion arose because this was much more traffic than BBN40 could handle.

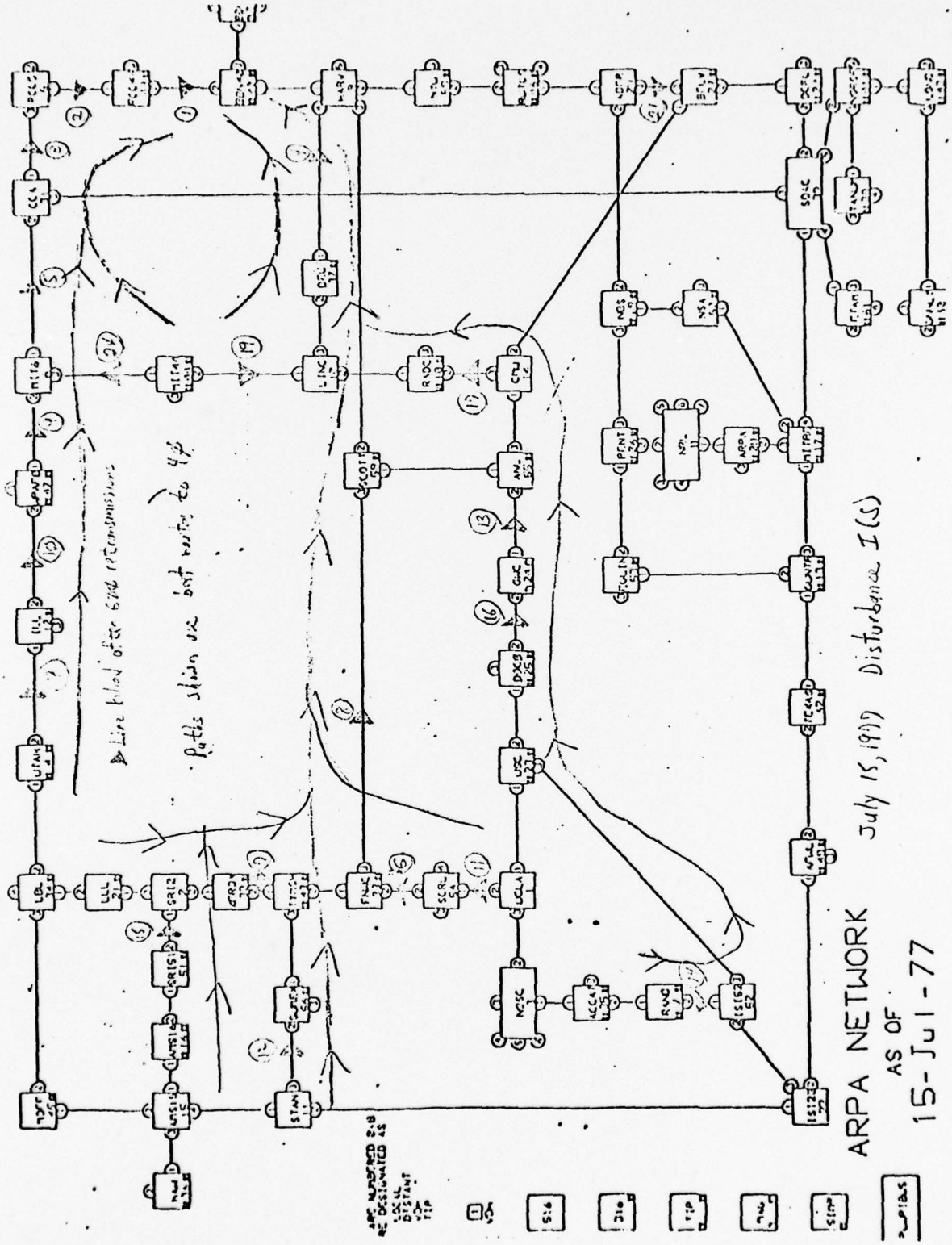


Figure 15

BEST AVAILABLE COPY

Disturbance II

Here there were two congested areas. There was traffic backed up from BBN to MIT to WPAFB, and there was congestion in the Washington area. The line connecting NBS to NSA was congested in both directions. This is a phenomenon which we began to see extremely often in later disturbances.

Perhaps traffic from each area was trying unsuccessfully to find a route into the other.

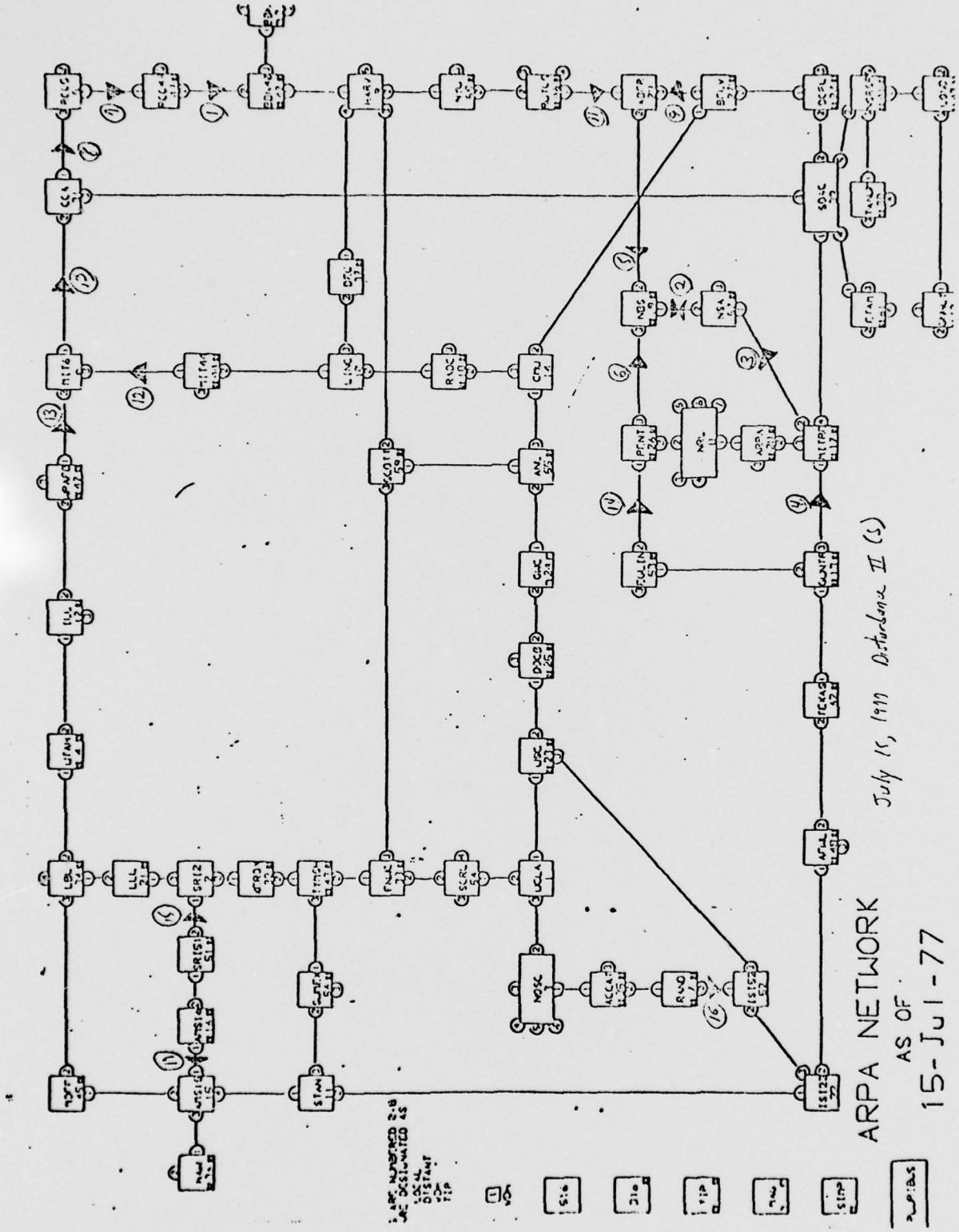


Figure 16

BEST AVAILABLE COPY

July 18 (Spontaneous)

This disturbance began with a software bug in the experimental IMP 30, connected to BBN40. It soon spread widely, with two foci of congestion. Traffic backed up from BBN40 through the rest of the Northeast to San Francisco. At the same time, traffic backed up from MITRE through the D.C. area to Los Angeles. Our data on the disturbance are not sufficient to suggest a good explanation of why it spread in this manner.

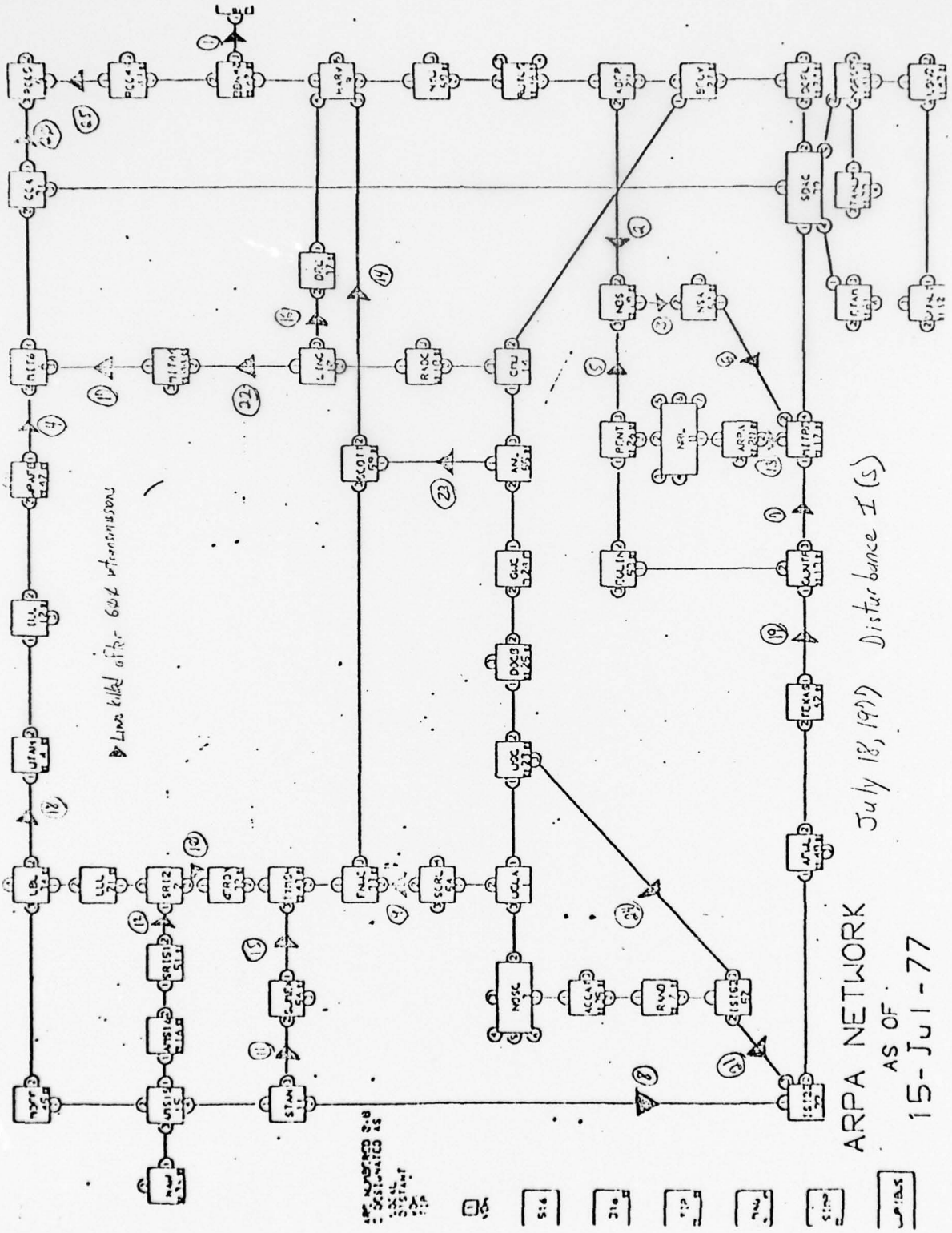


Figure 17

BEST AVAILABLE COPY

July 19 (Spontaneous Disturbance I)

From this date on, the snapshot of an IMP tells us the destinations of the packets contained in the IMP.

The disturbance began when there was hardware trouble at MIT6. Traffic for MIT6 backed up to BBN and to ISI52 (through the CCA-SDAC-MITRE-...path). This disturbance spread next to the San Francisco area, where STAN seemed to be a secondary focus of congestion. Shortly thereafter, traffic for MIT6 was backed up from MIT44 to CMU. The only difficult thing to understand here is why STAN became a secondary focus of congestion. The answer seems to be something like the following. Several IMPs in the San Francisco area (viz., TYMSH, XEROX & STAN) had high reassembly buffer counts, indicating that they were actively engaged in end-end communication. Also, some of the packets in each of these IMPs were destined for the others. In addition, AMS15 was trying to route packets to MIT through STAN, even as traffic was starting to pile up on the best path from STAN to MIT. This placed a very heavy load on STAN, leaving it with very few free S/F or reassembly buffers, thereby causing congestion.

After a lapse of four minutes, traffic to MIT6 was seen backing up along the WPAFB-ILL-UTAH path to San Francisco. However, as many other IMPs had used up their triggers already, it is impossible to know what was happening in the rest of the network at this time.

This disturbance is similar to the first one of July 15, in that it was (partially) caused by the fact that the network contained many packets to a destination which, for some reason or other, could not accept them. However, this disturbance was complicated by the presence of cross-traffic in the San Francisco area. A second focus of congestion arose when this cross-traffic competed for buffer space with the traffic which was backed from MIT.

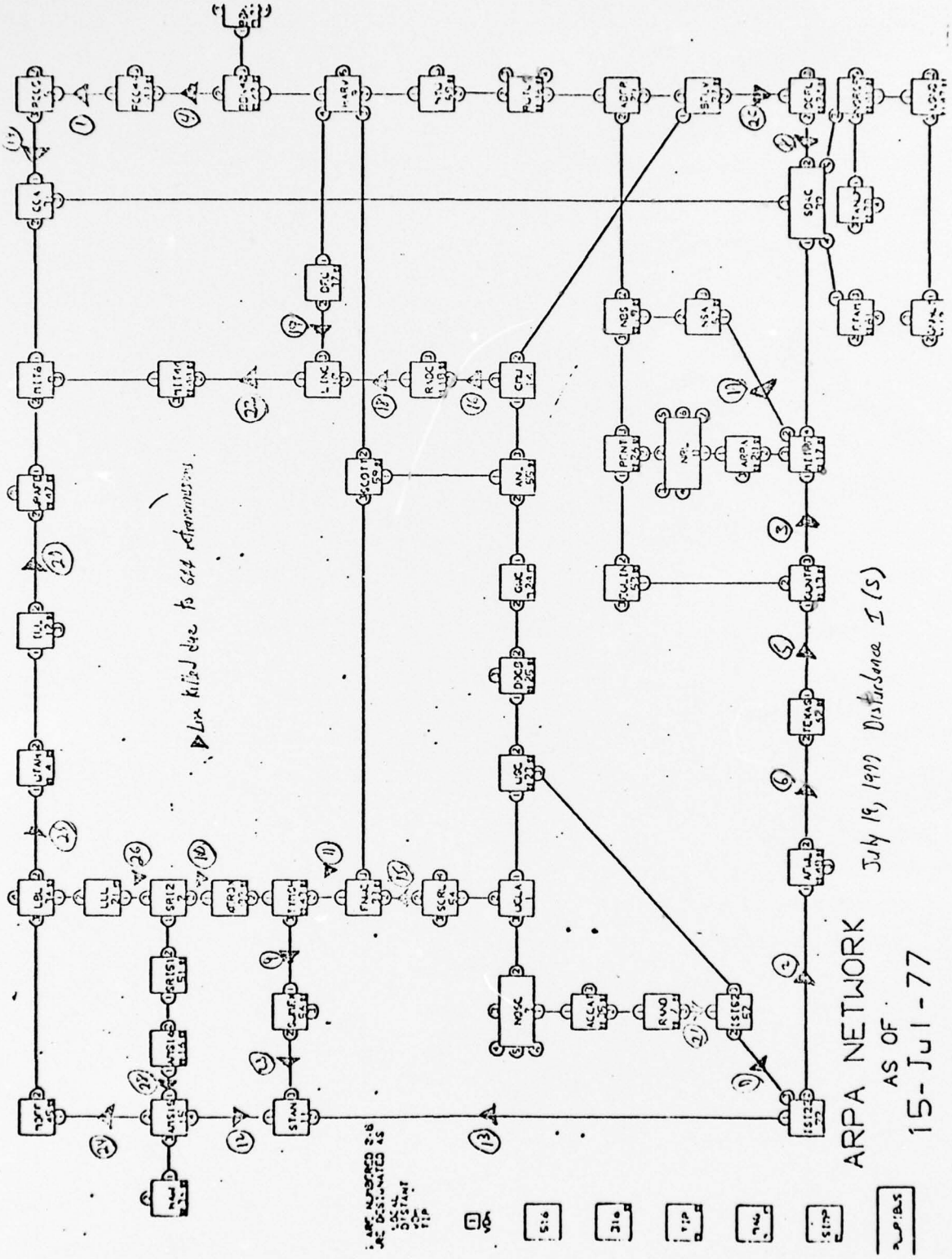


Figure 18

BEST AVAILABLE COPY

July 19 (Spontaneous Disturbance II)

This disturbance began when SDAC apparently stopped accepting packets, causing traffic to back up in all directions. However, as the disturbance spread, it became harder to discern any pattern to it. There seemed to be several foci of congestion (SDAC, ILL, ISI22, RUTGRS, and SCOTT).

The complexity of the traffic pattern makes it difficult to understand the mechanism of the spread of the disturbance.

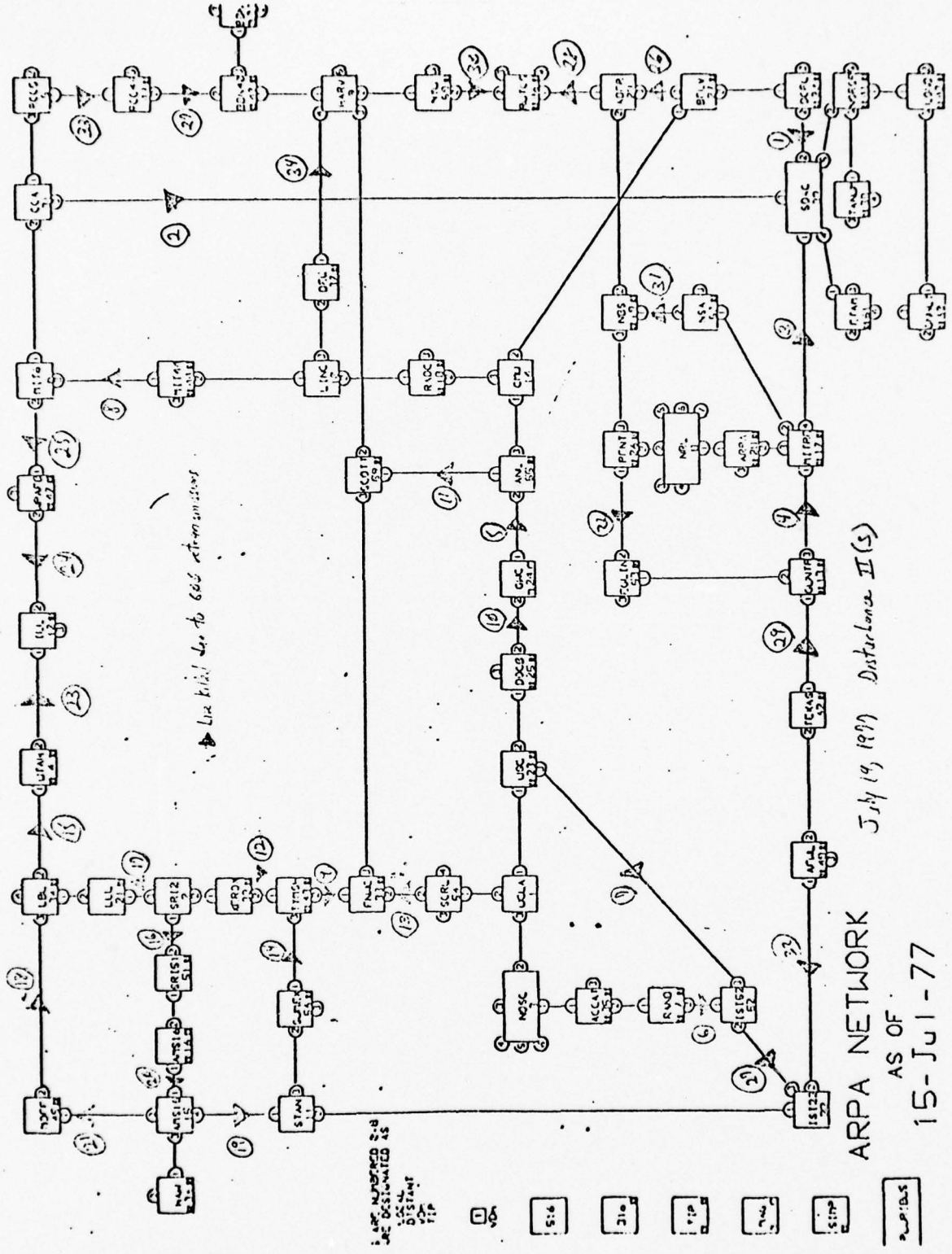


Figure 19

BEST AVAILABLE COPY

July 19 (Spontaneous Disturbance III)

There is no discernible pattern to the spread of this disturbance. Its initial cause is unknown.

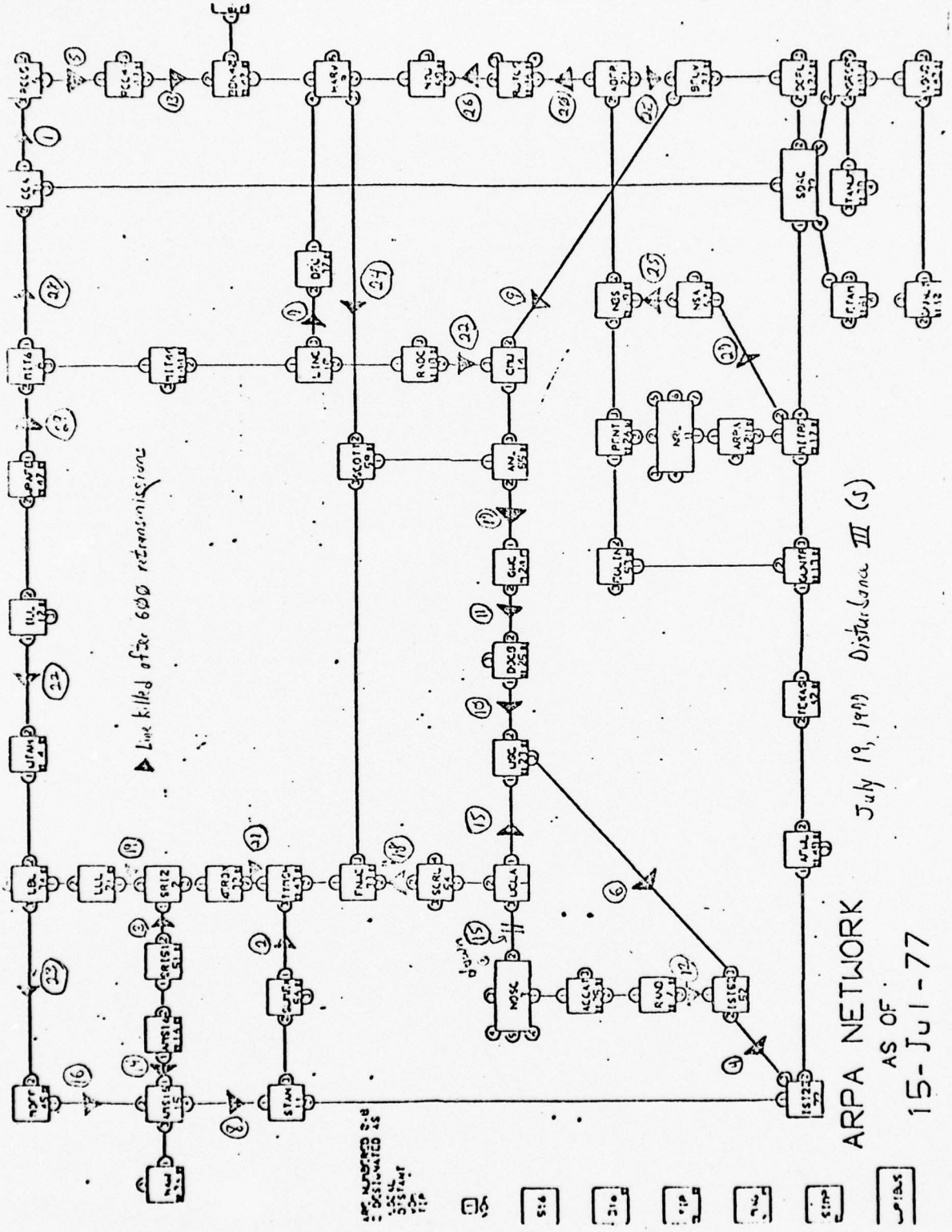


Figure 20

BEST AVAILABLE COPY

July 19 (Spontaneous Disturbance IV)

This disturbance began when there was trouble with MIT44. The network had a lot of traffic destined for MIT44, which backed up from it in all directions, particularly into the San Francisco area. Thus this disturbance seems similar to the first disturbances of July 15 and July 19. However, two further small areas of congestion developed. One involved ACCAT, RAND, USC, and ISI52. Another involved SRI51, SRI2, and XEROX. While these IMPs did not contain traffic for MIT44 (at least, no such traffic is recorded in the snapshots), several of them in each area have high reassembly buffer counts. This suggests that they were engaging in end-end communication with IMPs which became isolated by the disturbance, and that this somehow caused them to congest. The mechanism of this is not clear, however.

During this disturbance, traffic to MIT44 was looping between AMS15 and HAW, indicating a failure of the hold down mechanism.

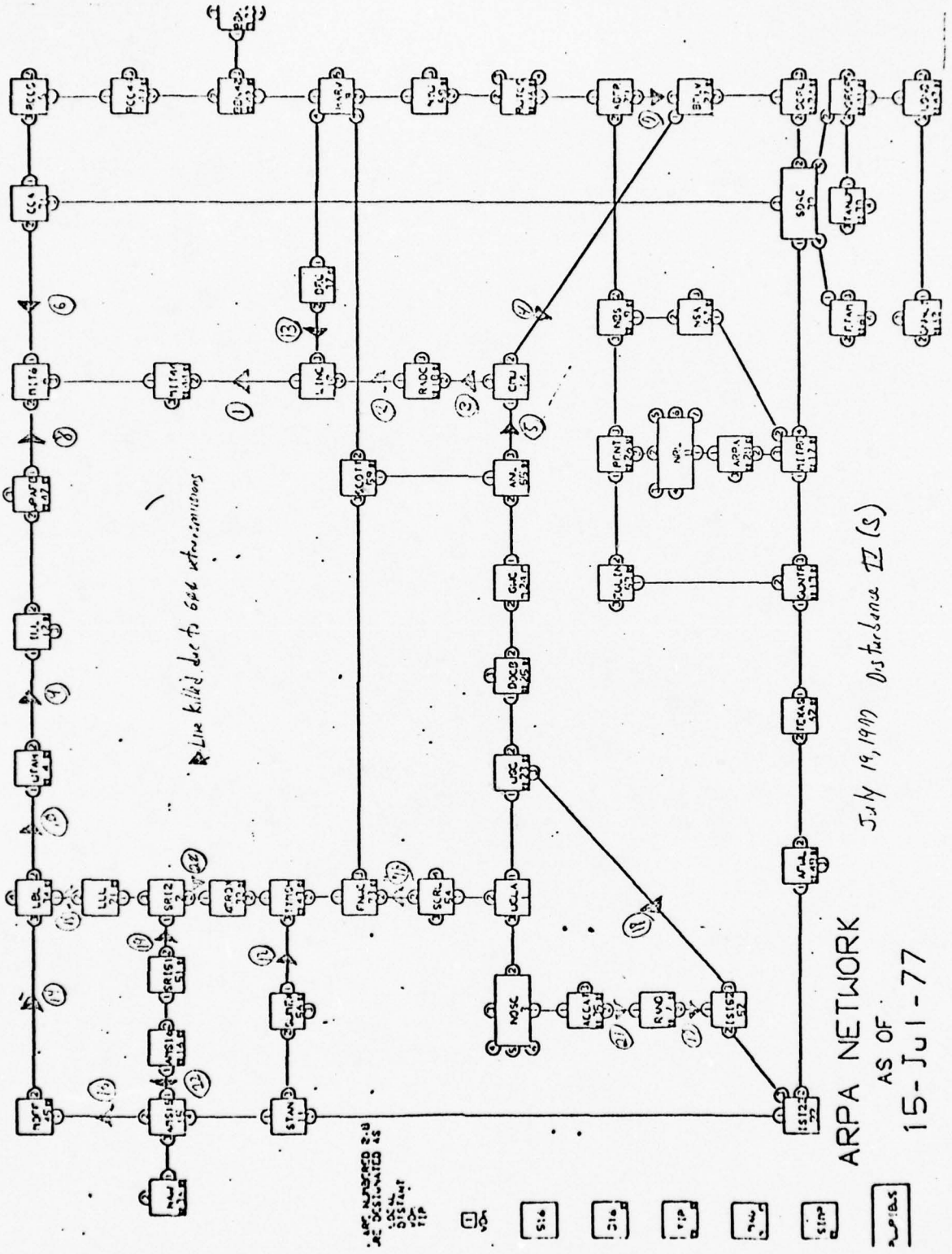


Figure 21

BEST AVAILABLE COPY

July 19 (Experiments)

Two successful attempts at causing a disturbance were made. The Case I message generators were turned on, and FNWC was slowed down. That is, the task processing loop of the FNWC IMP was set to run much more slowly than normal. Our intention was that FNWC would be unable to accept packets, except very slowly, and therefore that congestion would back up from FNWC. That turned out to be the case.

Disturbance V

Traffic backed up from FNWC in all three directions. In fact, the disturbance radiated out in straight line as far as 11 hops from FNWC. However, there was little data of interest in this disturbance.

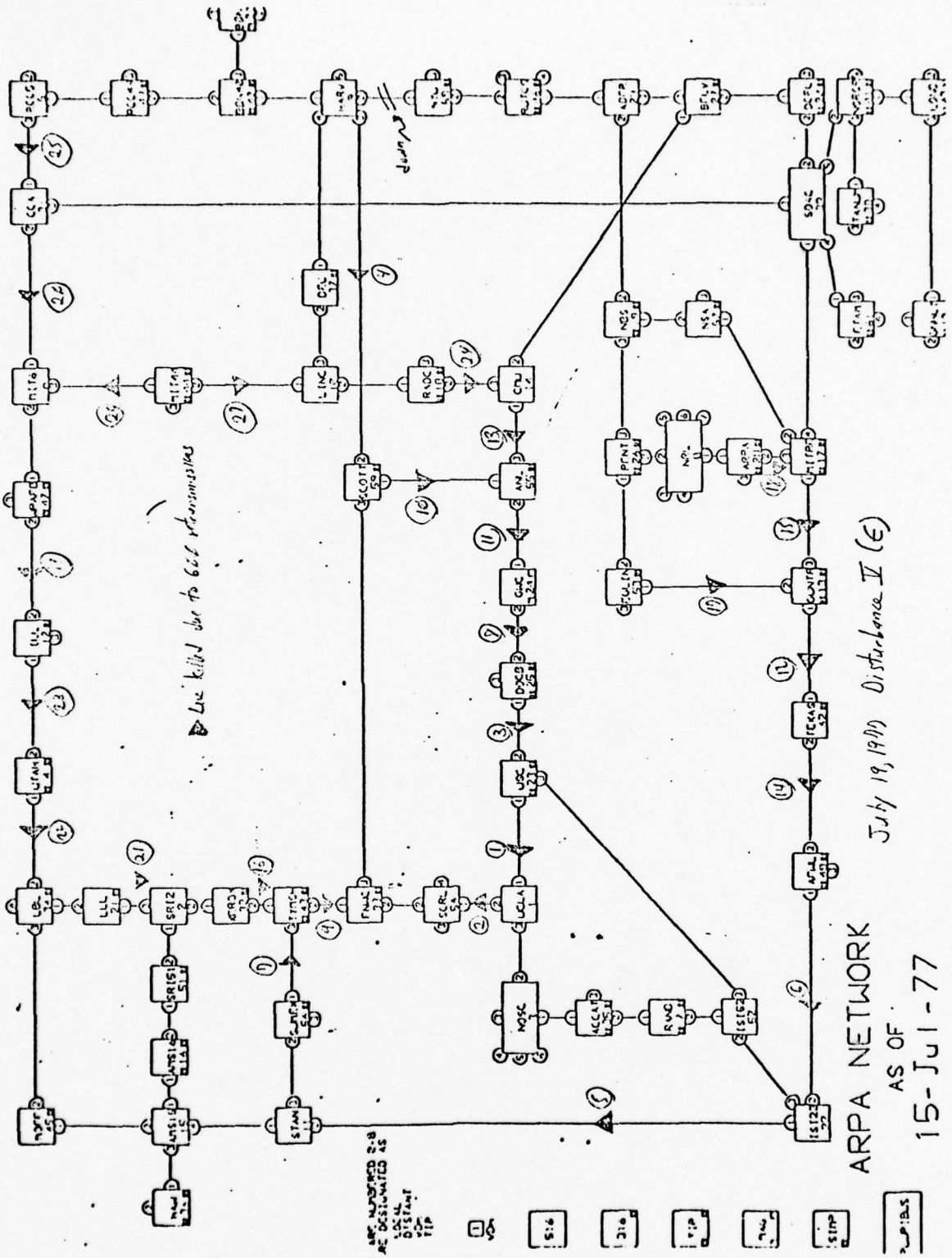


Figure 22

BEST AVAILABLE COPY

Disturbance VI

The same experiment was repeated. However, the disturbance was somewhat larger and more difficult to understand. Traffic was observed looping between HAW and AMS15. Furthermore, some packets were being retransmitted 600 times on paths leading away from FNWC.

Whereas in the previous experiment, no IMPs were observed to be short of reassembly buffers, in the current experiment there are high reassembly buffer counts at SCOTT, HARV, RCC49, and RAND. Examination of the snapshots indicates that there was traffic between SCOTT and RCC49, as well as between AMES and RCC49. This additional cross-traffic, which was real user traffic, had to travel near the area of the disturbance, and it ran somewhat against the grain of the artificially generated traffic. Although the cross-traffic placed a much lighter load on the network than did the artificially generated traffic, it was enough to significantly affect the characteristics of the disturbance.

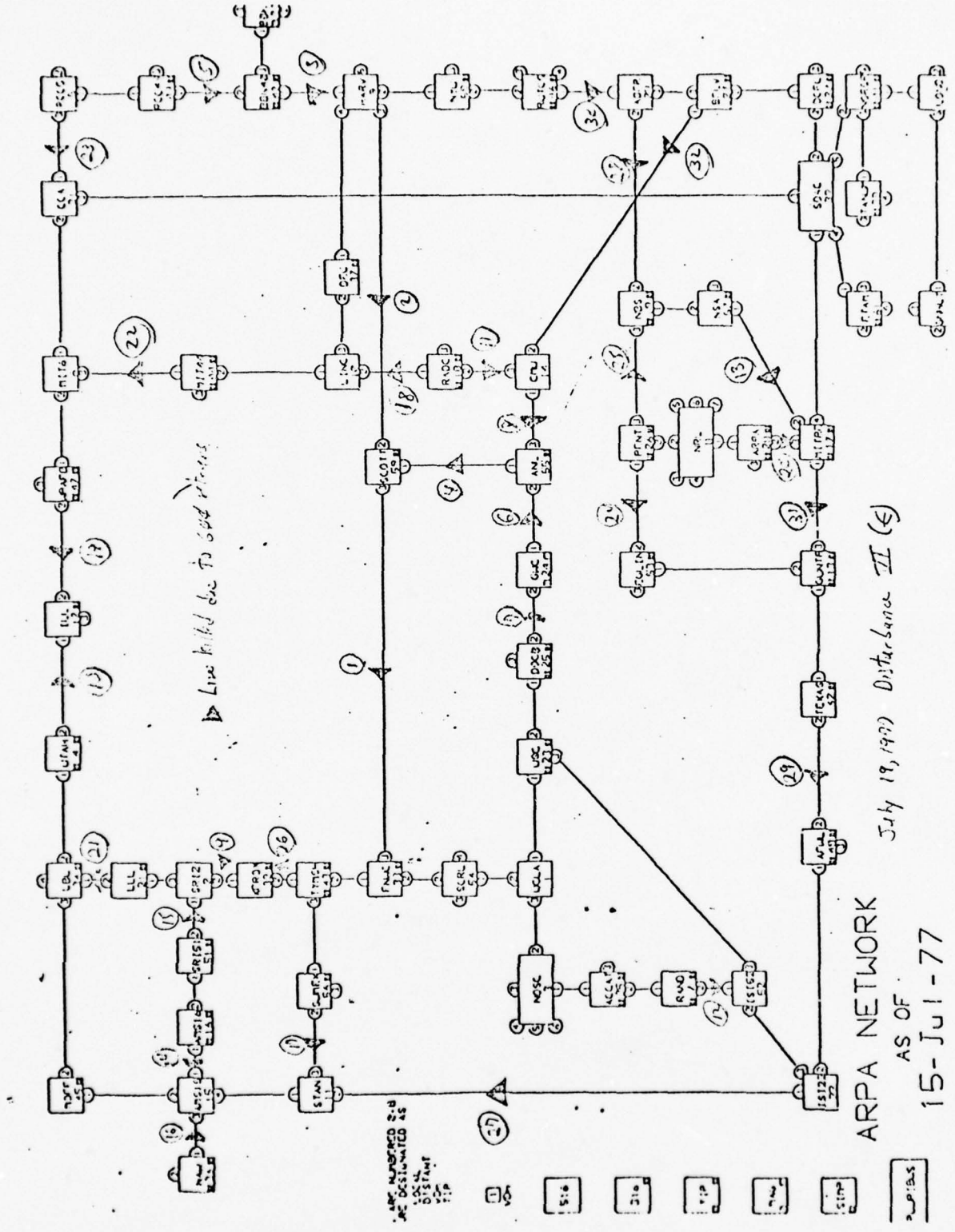

ARPA NETWORK
AS OF
15-Jul-77
July 19, 1977 Distribution II (4)
BEST AVAILABLE COPY

Figure 23

July 21 (Spontaneous Disturbance I)

This disturbance began when there was trouble on the satellite line from SDAC to NORSAR. This caused traffic to back up from SDAC in a straightforward pattern. The network contained a good deal of traffic from the Los Angeles area to Europe, as well as traffic from the Northeast to D.C. These traffic patterns made the loss of SDAC difficult to recover from.

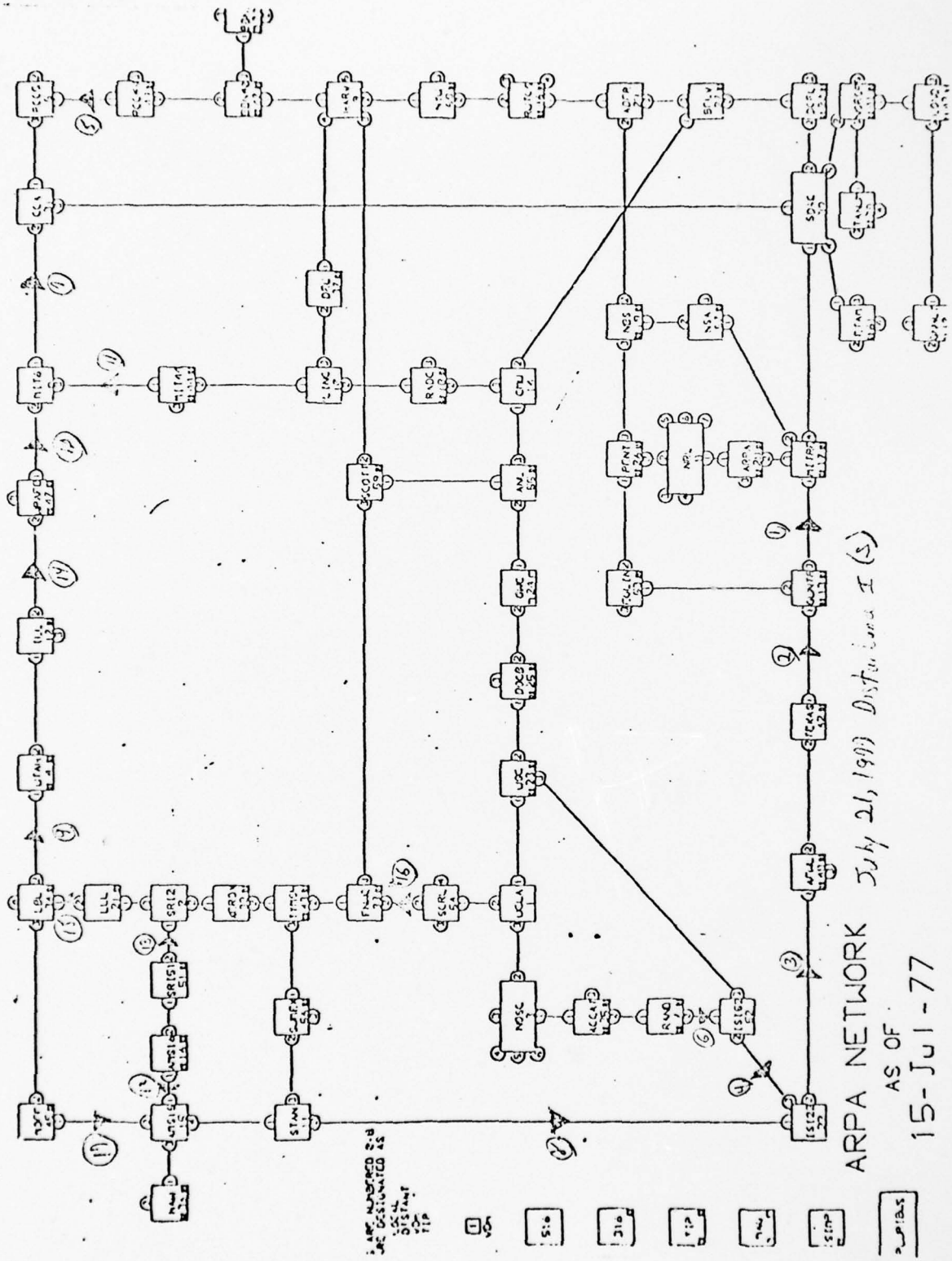


Figure 24

BEST AVAILABLE COPY

July 21 (Spontaneous Disturbance II)

This disturbance began when a faulty circuit between BELV and ABER caused traffic to back up from BELV in all directions. This caused a disturbance to spread to the Northeast, the Midwest, and the D.C. area. The disturbance then appeared in the California area, as IMPs in this area tried to direct traffic into the area of the disturbance. However, the traffic pattern is complex, and the mechanism of the spread is hard to specify.

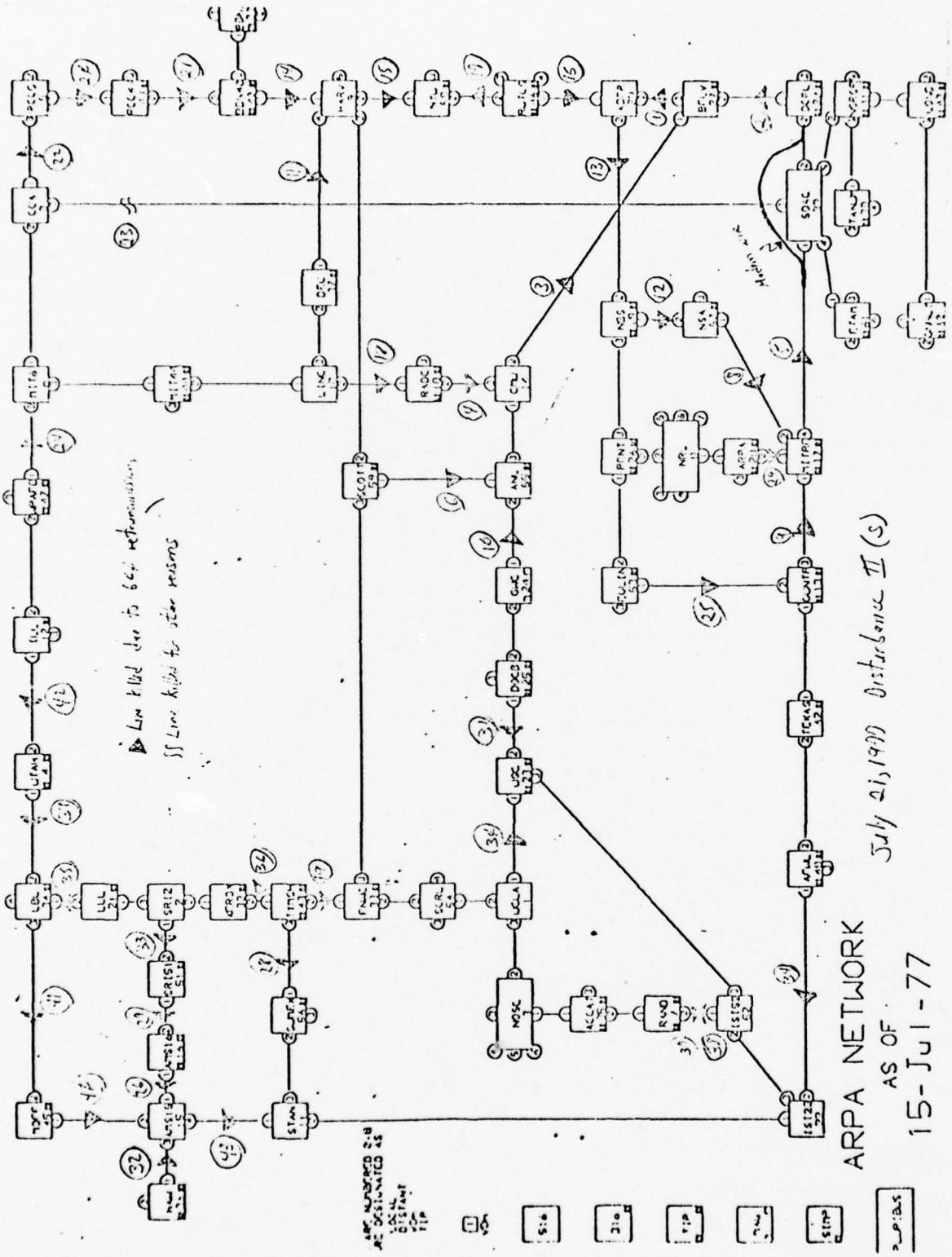


Figure 25

BEST AVAILABLE COPY

July 22 (Spontaneous Disturbance)

This disturbance was caused by problems with MIT6. Many of the IMPs in the Northeast filled with traffic to MIT6, initiating the disturbance. Many IMPs in the San Francisco area had traffic destined for IMPs in the Cambridge area. This traffic seems to have been rerouted via the Los Angeles area. Several IMPs in the Washington area also had traffic for the Cambridge area, and rerouted it via the Los Angeles area. All this rerouted traffic joined the already heavy DC-LA and SF-LA traffic flow, causing a disturbance in the Los Angeles area, which then propagated back to the San Francisco and DC areas. Then ILL started sending traffic to MIT via San Francisco. Of course, this only caused the disturbance to spread from San Fancisco to ILL.

This is an example of a disturbance which may have been caused by inaccurate routing. The disturbance might have been confined to the Northeast if routing had not sent traffic into the disturbance via alternate routes. This suggests that a routing scheme with explicit information on traffic levels or available capacity would have performed much better in this instance.

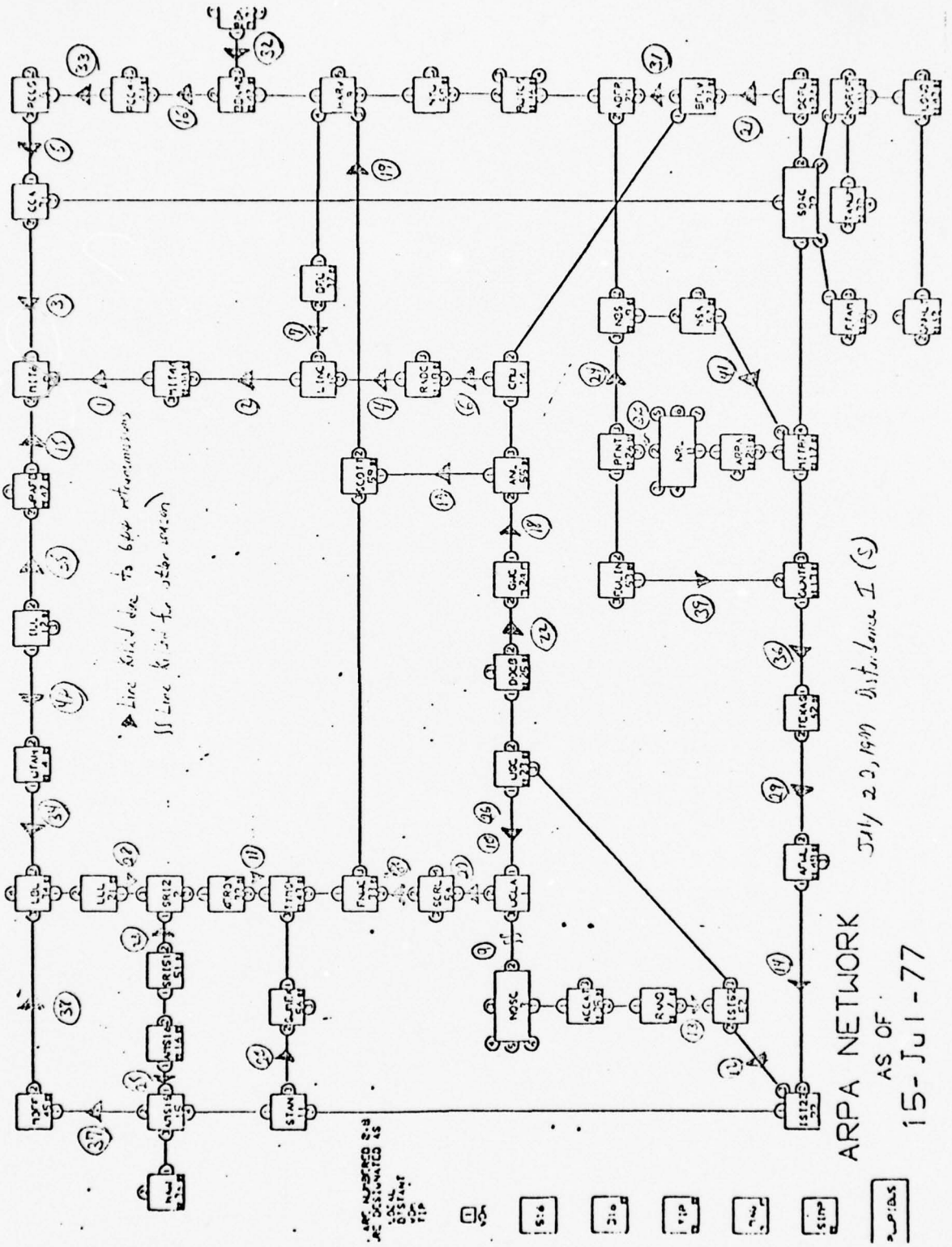


Figure 26

BEST AVAILABLE COPY

July 26 (Experiment)

These experiments tested the hypothesis that network disturbances would be less severe if packets were retransmitted considerably less than 600 times before killing the line on which they were retransmitted. Five experiments were performed. In each case, the disturbance was initiated by slowing FNWC, as in the experiment of July 19. Variables were the traffic flow (case I or case II message generators) and the number of times a packet would be retransmitted before the line on which it was being transmitted would be killed (600, 100, or 50). Due to circumstances beyond our control, the line from BBN40 to HARV was down during all the experiments.

Experiment I

Case I message generators were used, and lines were killed after 100 retransmissions of a packet.

Although congestion spread as far from FNWC as STAN and HARV (i.e. no more than 3 hops), causing five lines to go down, the effects of the congestion were localized. No real disturbance occurred.

Experiment II

Case I generators were used, and lines were killed after 600 retransmissions. A very large disturbance resulted, with FNWC as the focus of congestion. First, traffic backed up from FNWC through SCOTT for a distance of up to 4 hops. Lines along these paths were killed during the first 44 seconds of the disturbance. Then the network was quiet for almost two minutes, after which traffic backed up along all other paths to FNWC. This caused the disturbance to move into the San Francisco and Los Angeles areas. This second wave of the disturbance lasted for about two minutes. Then the network was quiet for about a minute and a half, until a third wave of congestion began. The third wave was caused by user traffic from the Northeast and the D.C. areas moving towards California. It lasted for about 30 seconds.

This disturbance illustrates two phenomena which we see again and again during network disturbances:

1. When a disturbance forms, even relatively small amounts of user traffic heading into the area of the disturbance will cause it to spread.

2. Disturbances often seem to occur in waves, with each succeeding wave affecting a larger and larger portion of the network. There are also relatively long quiet periods between the waves. The length of these quiet periods seems to be dependent both on random factors (such as the arrival of new user

traffic flows) and network timing constraints (such as the amount of time it takes to retransmit a packet 600 times).

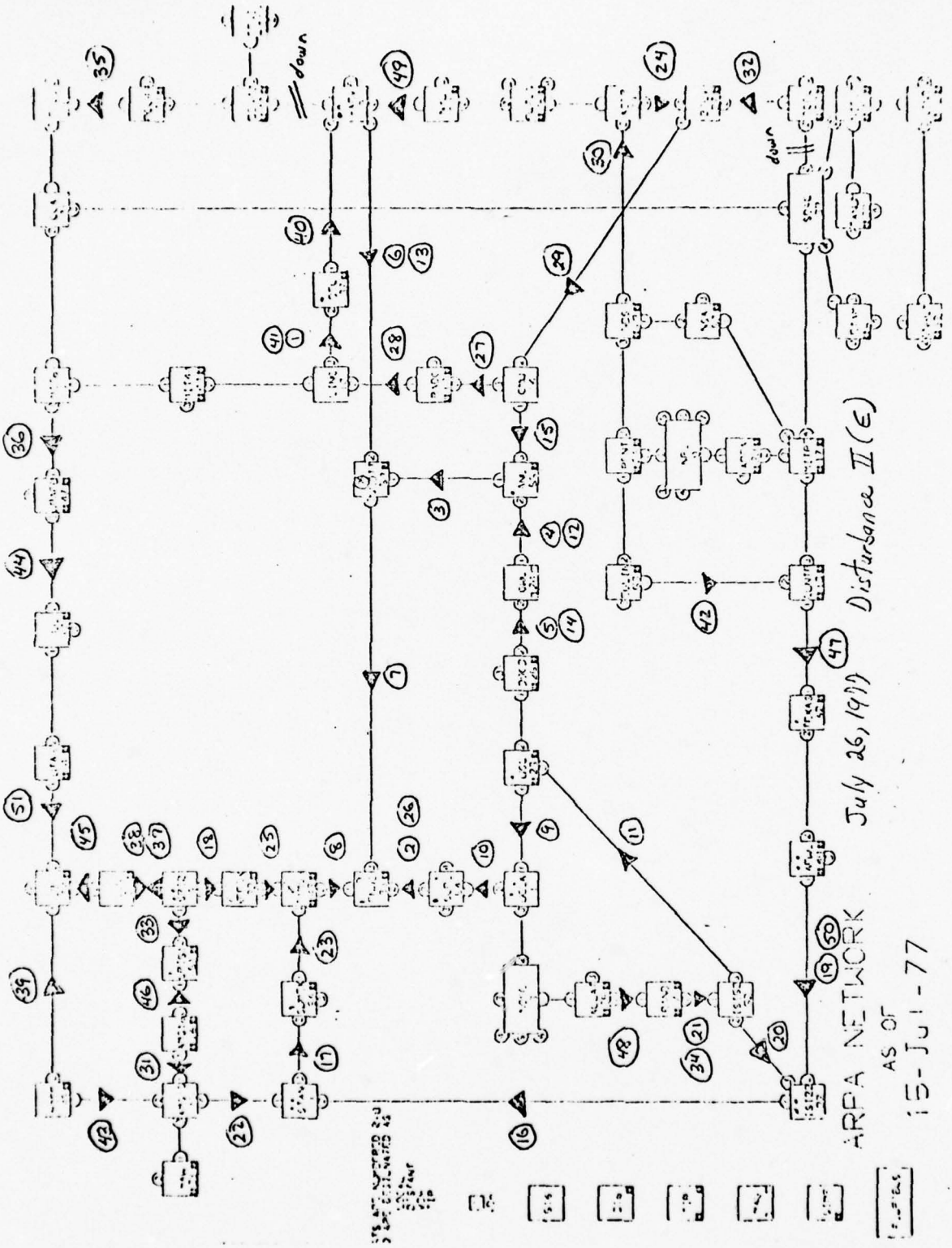


Figure 27

BEST AVAILABLE COPY

Experiment III

This was just a repeat of experiment I (case I generators, 100 retransmissions before killing a line), with similar effect. No significant disturbance resulted.

Experiment IV

For this experiment, case II generators were used, and lines were killed after 100 retransmissions. Traffic backed up from FNWC in all directions, in some cases by as much as 9 hops. As in experiment II, the disturbance formed in three expanding waves, with quiet periods of approximately 75 seconds separating them. No other particularly interesting phenomena were noted.

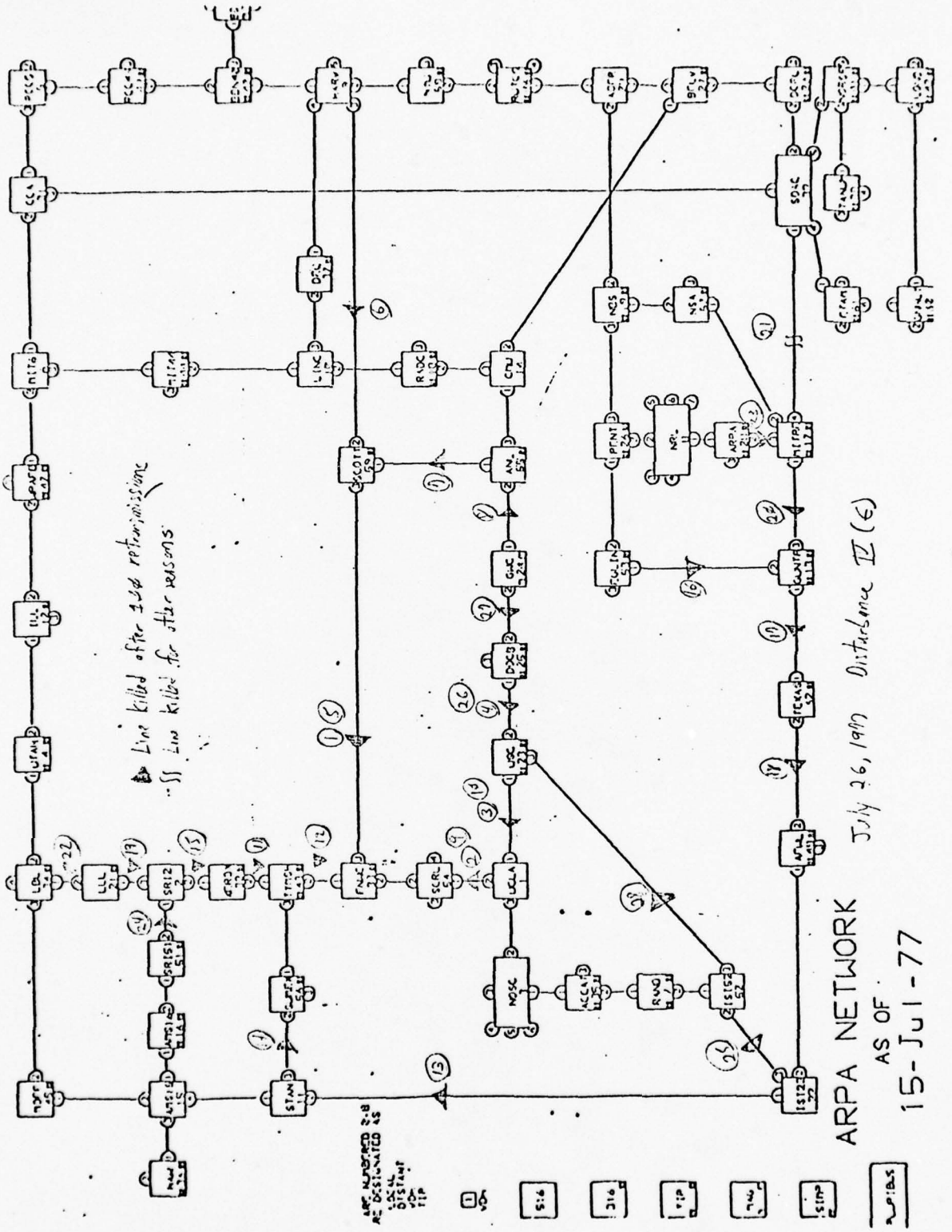


Figure 28

BEST AVAILABLE COPY

Experiment V

For this experiment, case II generators were used, and lines were killed after only 50 retransmissions of a packet.

Again, traffic backed up from FNWC in a straightforward and easily understood pattern. However, the disturbance spread very slowly and not very far. It spread in five waves, each of which lasted only for several seconds. The first three waves were separated from each other by quiet periods of approximately one minute. During these waves, the disturbance never spread more than 2 hops from FNWC. The fourth wave began about 80 seconds after the third ended, and spread up to 6 hops from FNWC. The fifth wave began about 80 seconds after the fourth ended; but it had even less spread than the fourth.

These five experiments do support the hypothesis that, for a given traffic pattern, the extent of a disturbance is directly proportional to the number of times a packet is retransmitted before killing the line on which it is being sent.

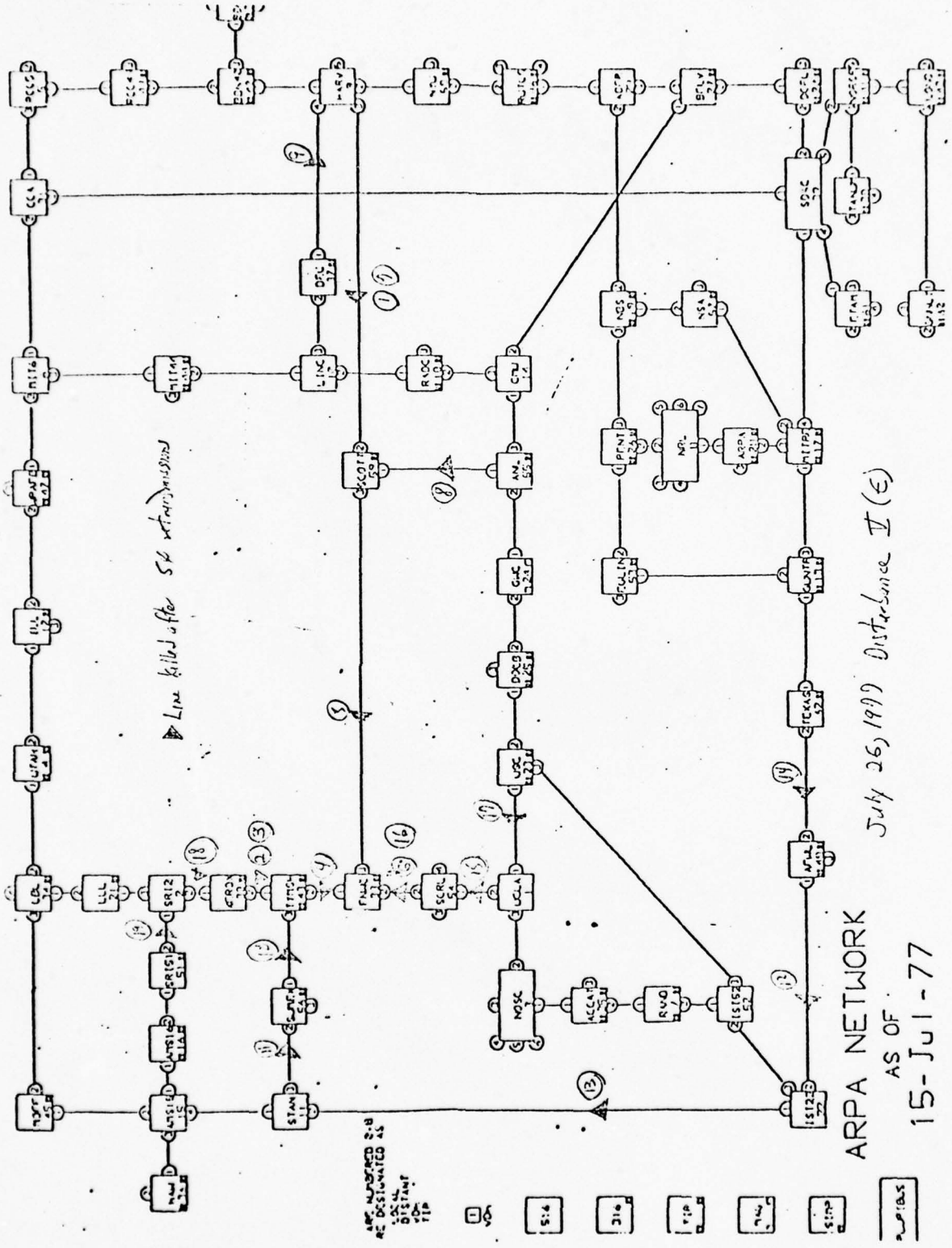

ARPA NETWORK
AS OF
15-Jul-77
July 26, 1979 Distribution II (E)
BEST AVAILABLE COPY

Figure 29

July 26 - August 2

During this week, the network was patched to kill lines after 100 retransmissions. No spontaneous disturbances occurred at all.

August 2 (Experiments)

This time we performed five experiments. The first three repeated experiments performed the previous week. The last two tested a new hypothesis, viz., that disturbances will spread less and do less damage if packets which are retransmitted many times are simply discarded, while the lines on which they are being sent remain up.

Experiment 0

For this experiment, we used the case I set of generators, slowed FNWC, and killed lines after 600 retransmissions. However, no disturbance occurred at all.

Experiment I

We just repeated the previous experiment, with all the same parameters. This time we observed quite a significant disturbance, with two distinct foci of congestion.

During the first five seconds of the disturbance, traffic was seen backed up from FNWC to SCRL, UCLA, USC, and ISI52. ISI52 had a considerable amount of traffic directed to the San

Francisco area. This traffic had to be rerouted via ISI22. At this time ISI22 had a very high reassembly buffer count, indicating that it was actively engaged in end-to-end connection with some other IMPs. Indeed, the snapshots show traffic for ISI22 coming from as far away as RCC5. Apparently, this additional S/F traffic load from ISI52 was too much for ISI22 to handle in conjuction with its own traffic. Traffic began to back up from ISI22 in all directions. This caused the disturbance to spread into the San Francisco area, as IMPs in that area filled with traffic destined for IMPs in the Los Angeles area. The disturbance spread simultaneously toward the Northeast, where much traffic destined for California was originating.

This disturbance is interesting because the major focus of congestion, ISI22, was not involved in the experiment in any way. All traffic towards it was real user traffic, and this was of a much smaller volume than the artificially generated traffic. Nevertheless, the only way in which the artificially generated traffic contributed to the disturbance was by removing one of the two main paths between the San Francisco area and the Los Angeles area. This caused the other path to become overloaded with real user traffic, and that is what caused the disturbance to spread.

This experiment illustrates once again the predominant role played by traffic pattern in the occurrence and spread of network disturbances.

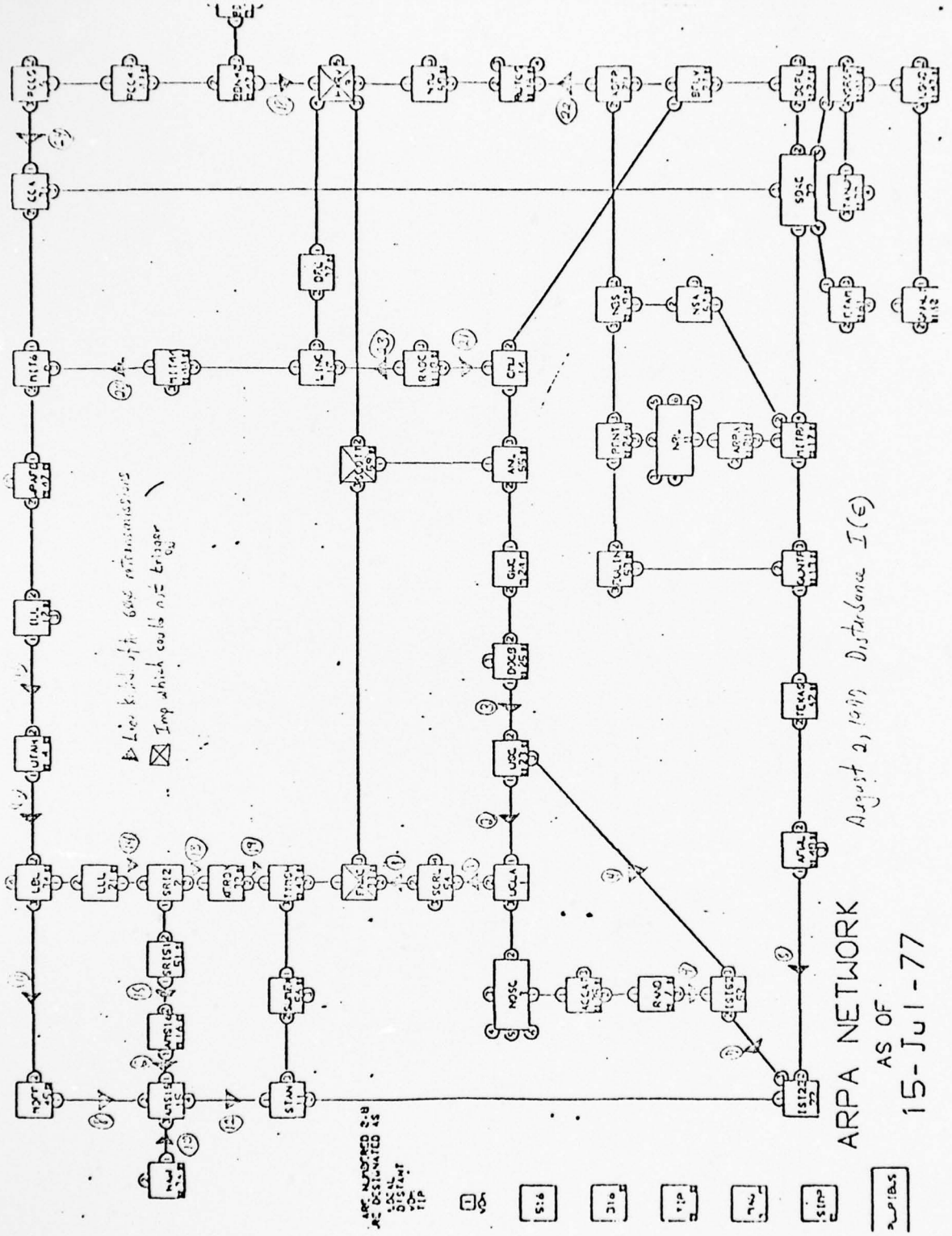


Figure 30

BEST AVAILABLE COPY

Experiment IB

For this experiment, we used the reduced case I generators, slowed FNWC and brought down lines after 100 retransmissions. This caused congestion to back up as much as 5 hops from FNWC. However, this was only a minor disturbance, with nothing particularly interesting or informative happening.

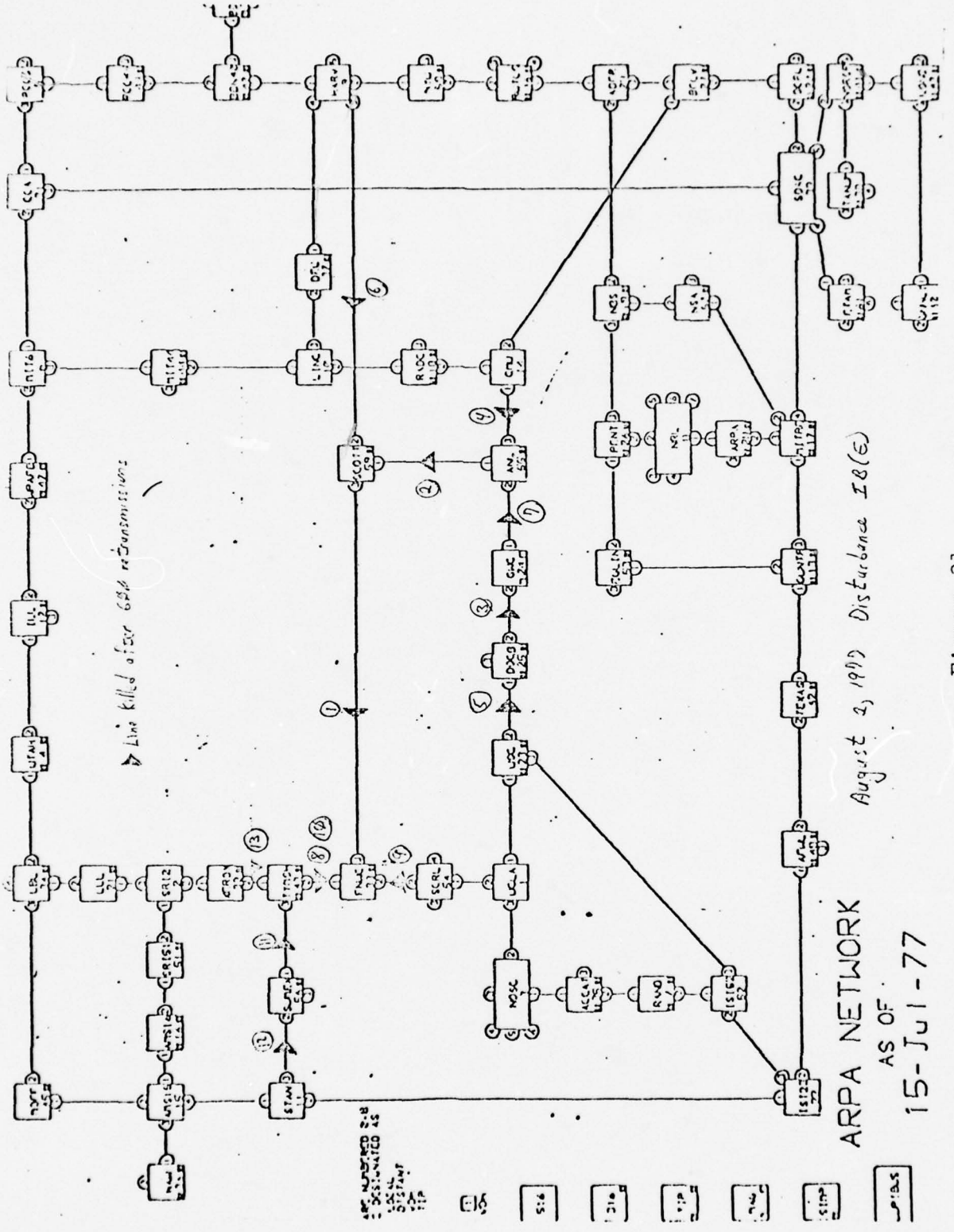


Figure 31

BEST AVAILABLE COPY

Experiment II

For this experiment, we used the reduced case I generators, slowed FNWC, and dropped (discarded) a packet after 100 retransmissions.

Congestion backed up from FNWC as much as 6 hops, and almost 600 packets were discarded. The number of packets discarded on a line was directly proportional to that line's proximity to the focus of the congestion, FNWC. Of the approximately 600 packets dropped, 350 were dropped within 1 hop of FNWC, and another 210 were dropped within 2 hops of FNWC. The congestion did not cause any lines to go down.

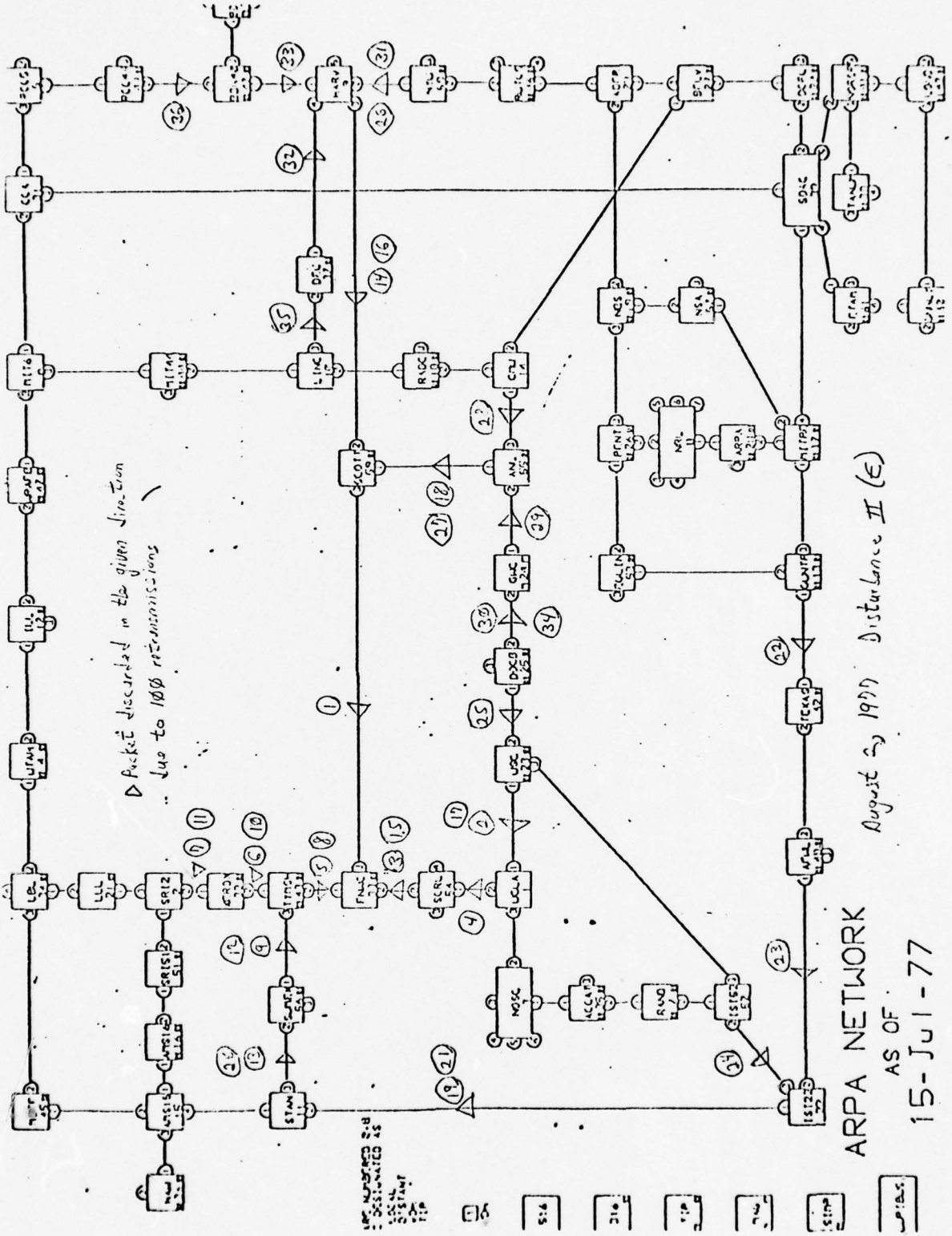


Figure 32

BEST AVAILABLE COPY

Experiment III

For this experiment, we slowed FNWC, used the reduced case I generators, and dropped packets after 50 retransmissions. This caused a small amount of congestion focused at FNWC. Packets were dropped by all nodes within 2 hops of FNWC, and by some nodes 3 hops away. A total of 238 packets was dropped, of which 169 were dropped within 1 hop of FNWC. Again, no lines were lost due to the congestion.

Experiments II and III support the hypothesis that dropping packets in a congested area is an effective way of preventing the congestion from resulting in a network-wide disturbance, especially if the number of times a packet may be retransmitted is kept fairly low.

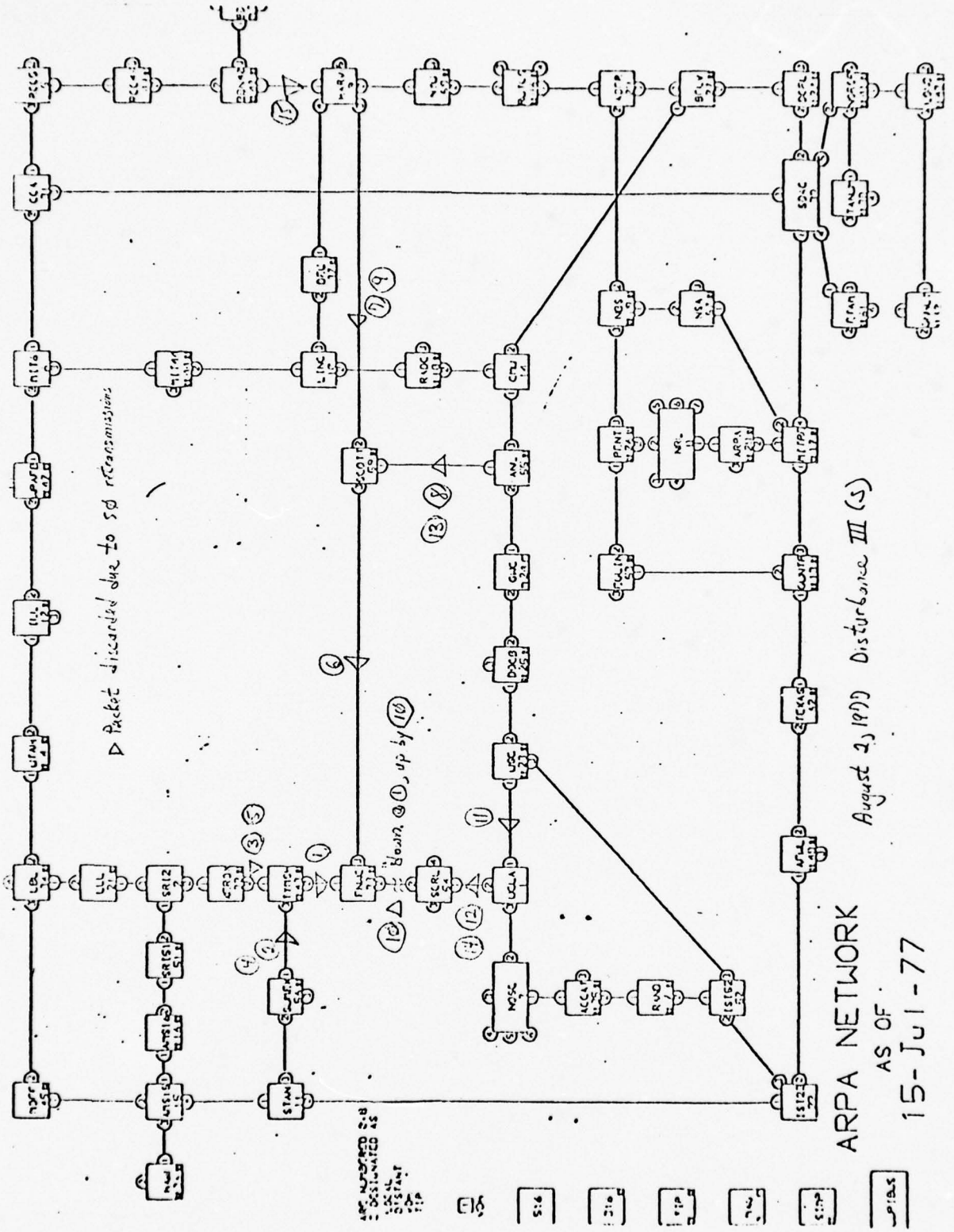


Figure 33

BEST AVAILABLE COPY

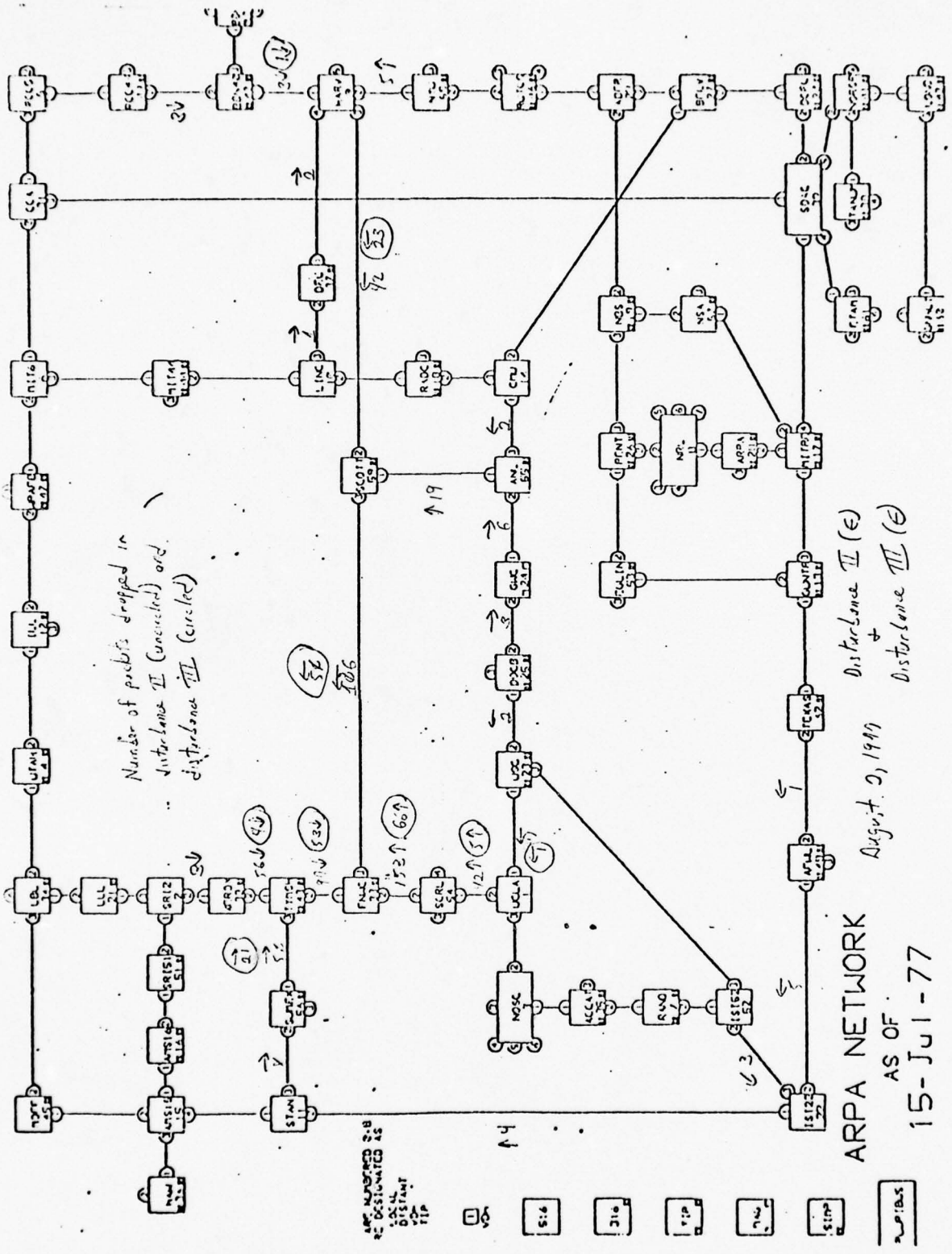


Figure 34

BEST AVAILABLE COPY

August 4 (Spontaneous)

At this time the network was set to discard packets after 100 retransmissions. The new Pluribus at CCA was not operating properly, and may have contributed to the disturbance.

The circumstances initiating this disturbance were rather peculiar. Most of the IMPs in the San Francisco area were filled with traffic for the Los Angeles area. For some reason, the lines from STAN to ISI22, TYMSH to FNWC, and LLL to SRI2 went down virtually simultaneously. We have no explanation for this, as the telephone company assures us that these three lines do not have any equipment in common. The result was that the only way for traffic to get out of the San Francisco area was through AMS15. Therefore, all traffic from the area converged on AMS15, making it the focal point of the congestion. This congestion backed up from AMS15 in all directions. Packets were also seen looping between AMS15 and HAW. After about 30 seconds, the three dead lines mysteriously came back up. However, congestion continued for another 30 seconds in the San Francisco area, and began to back up into the Los Angeles area.

A total of 69 packets were dropped: 62 in the San Francisco area, and 7 in the Los Angeles area. Congestion did not spread to any other part of the network. If the network had been killing lines instead of dropping packets, virtually all of the IMPs in the San Francisco area would have been isolated, resulting in a much greater loss of packets and user service.

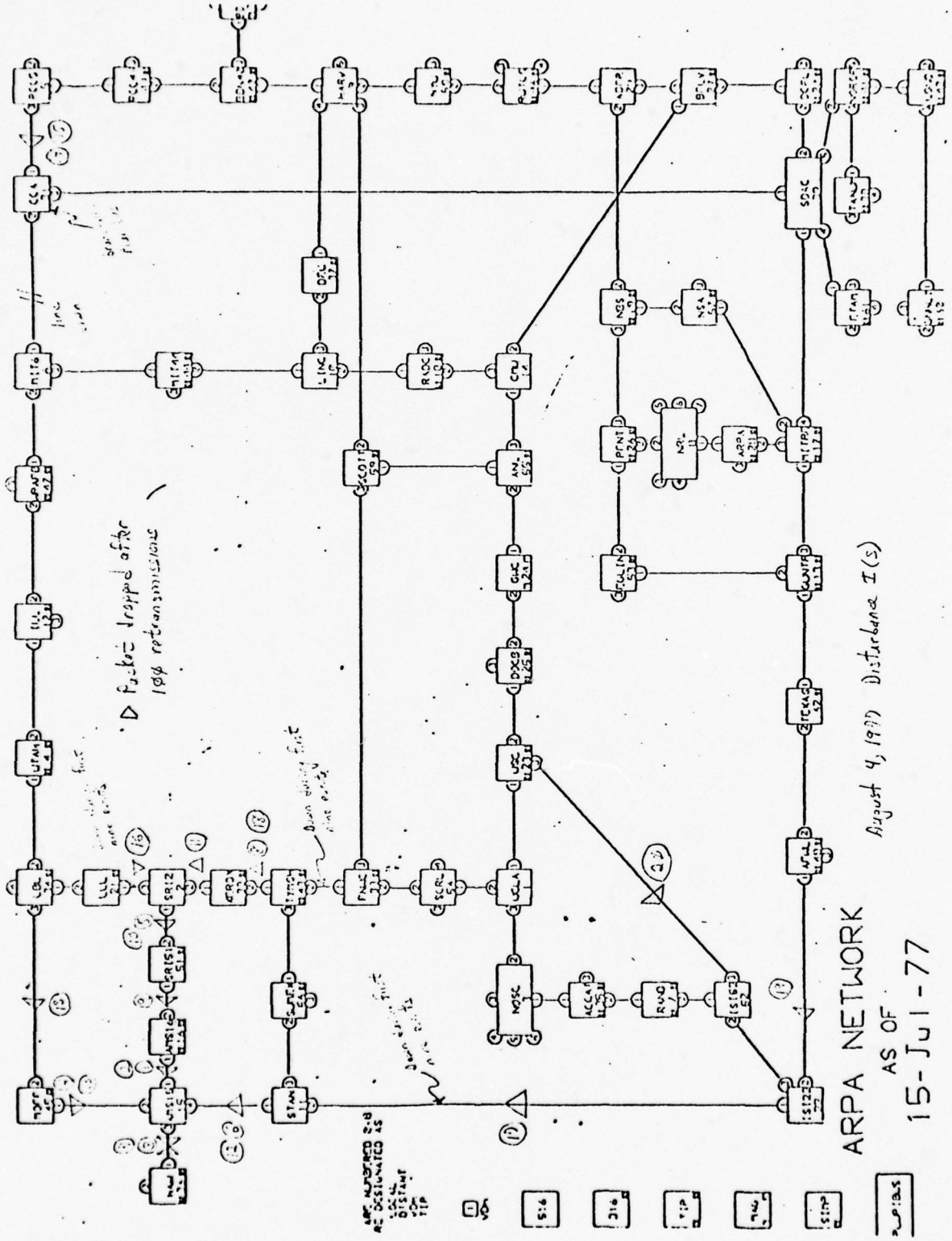


Figure 35

BEST AVAILABLE COPY

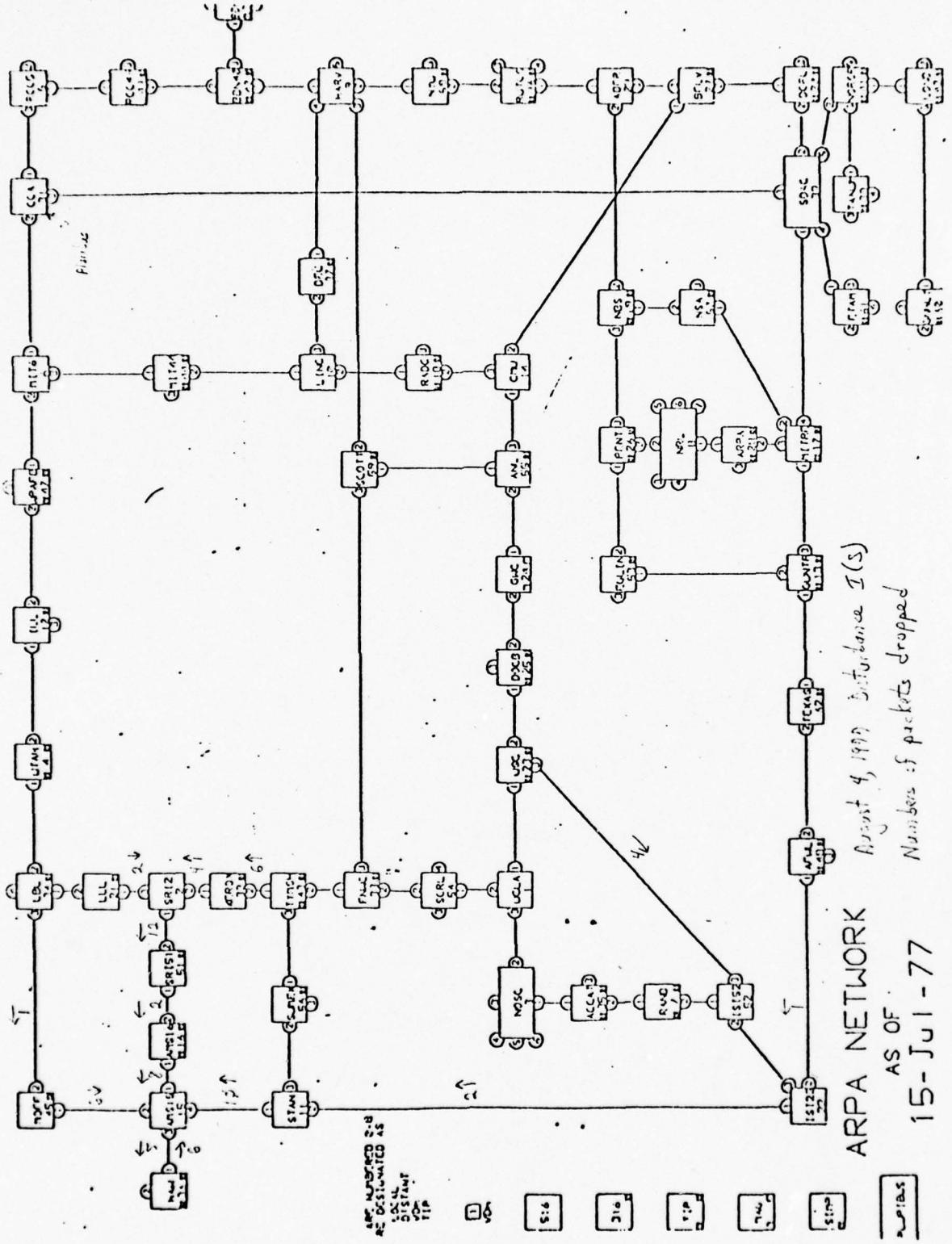


Figure 36

BEST AVAILABLE COPY

August 18 (Spontaneous)

At this time, the network was set to bring down a line after retransmitting a packet on it fifty times.

Disturbance I

This disturbance began when MIT6 crashed. At the same time, many of the IMPs in the D.C. area were filled with traffic destined to MIT6. Since it took varying amounts of time for these IMPs to learn that each path to MIT6 was unavailable, routing was in a confused state, resulting in looping. In particular, NSA and NBS, which are neighbors, had full queues to each other which contained only traffic to MIT6. This traffic was unable to get out of the D.C. area, and thus serious congestion resulted.

This disturbance is another example of a pattern repeated many times. The disturbance began in one area (around MIT6), and then spread to a different area (D.C.), because the second area contained traffic destined for the first.

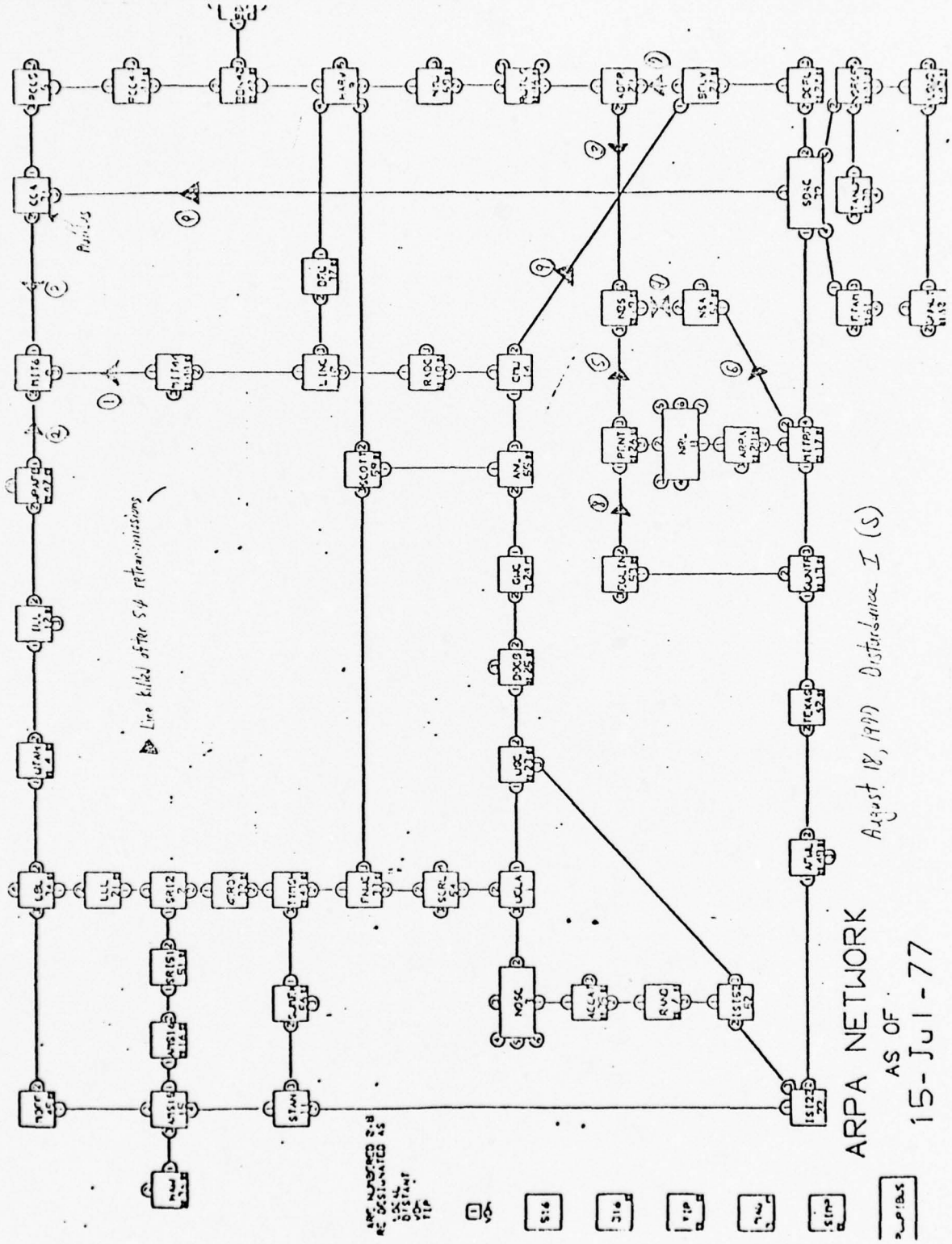


Figure 37

BEST AVAILABLE COPY

Disturbance II

This disturbance was a very simple and straightforward one. There was some sort of problem at ACCAT (We do not know its exact nature), and traffic backed up for a distance of 5 hops.

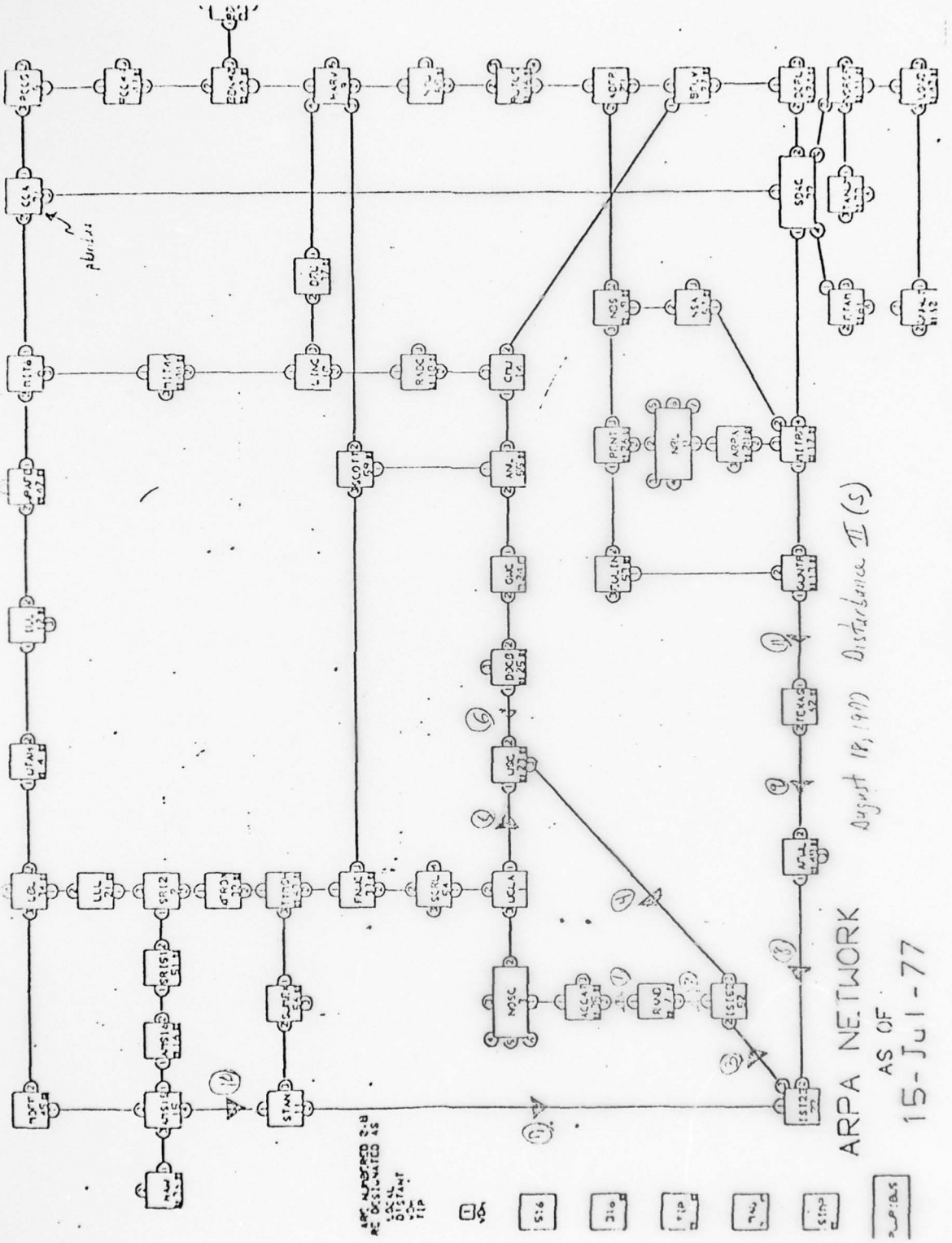


Figure 38

BEST AVAILABLE COPY

Disturbance III

Just prior to the beginning of this disturbance, a new software patch was being tested at BBN30. Traffic to BBN30 existed at BBN40, RCC49, and HARV. The disturbance began when BBN30 stopped accepting packets from BBN40, causing the line between them to go down. Then BBN40 refused to accept packets for BBN30, causing the lines to RCC49 and HARV to go down. RCC5 had traffic to BBN40, and since RCC49 was no longer accepting it, the line from RCC5 to RCC49 went down. This isolated RCC49 from the rest of the network. At this time, the network was full of traffic to RCC49, traffic which could no longer reach its destination. There were 107 buffers of such traffic spread throughout 16 IMPs. This traffic filled up many of the queues in the network, causing congestion to back up (and lines to go down) all the way back along the paths to California. This caused many IMPs to become isolated, which, of course, resulted in further congestion and in more IMPs becoming isolated.

This is another example of a disturbance caused by the presence in the network of many packets to an unreachable destination.

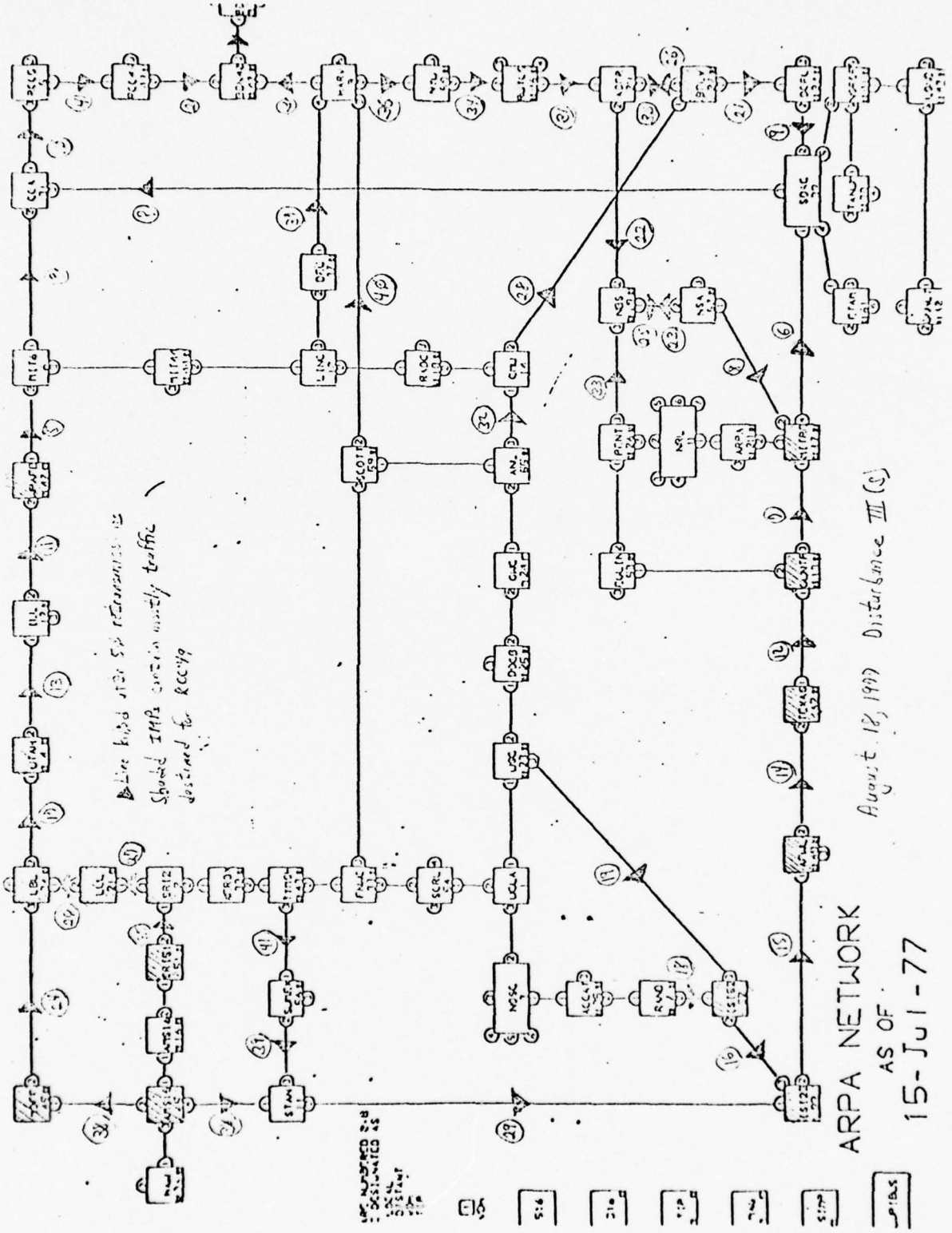


Figure 39

BEST AVAILABLE COPY

August 23 (Experiments)

During these experiments we tested out a more sophisticated packet discarding scheme. After retransmitting a packet 32 times, the IMP checks the packet to see whether it is a control packet or a data packet (including RFNMs). If it is a control packet, it is simply discarded. If it is a data packet, the IMP checks to see whether a new route has been computed for it. If not (i.e., if the route to its destination has not changed since the last time it was routed), it is discarded. If there is a new route, it is queued for it. If the packet is retransmitted another 32 times on the new route, then it is discarded.

Experiment I

We used the reduced case I message generators and we slowed FNWC. We got a typical small disturbance, backing up no more than 4 hops from FNWC. A total of 42 packets was dropped, of which 5 were control packets, and a total of 18 packets was rerouted. The disturbance occurred in three waves, with about one minute of quiet time between them. However, whereas previous disturbances showed additional spread with each succeeding wave, this disturbance did not spread in that way. Rather, each wave had about the same spread. We do not understand the significance of the 60 second quiet periods.

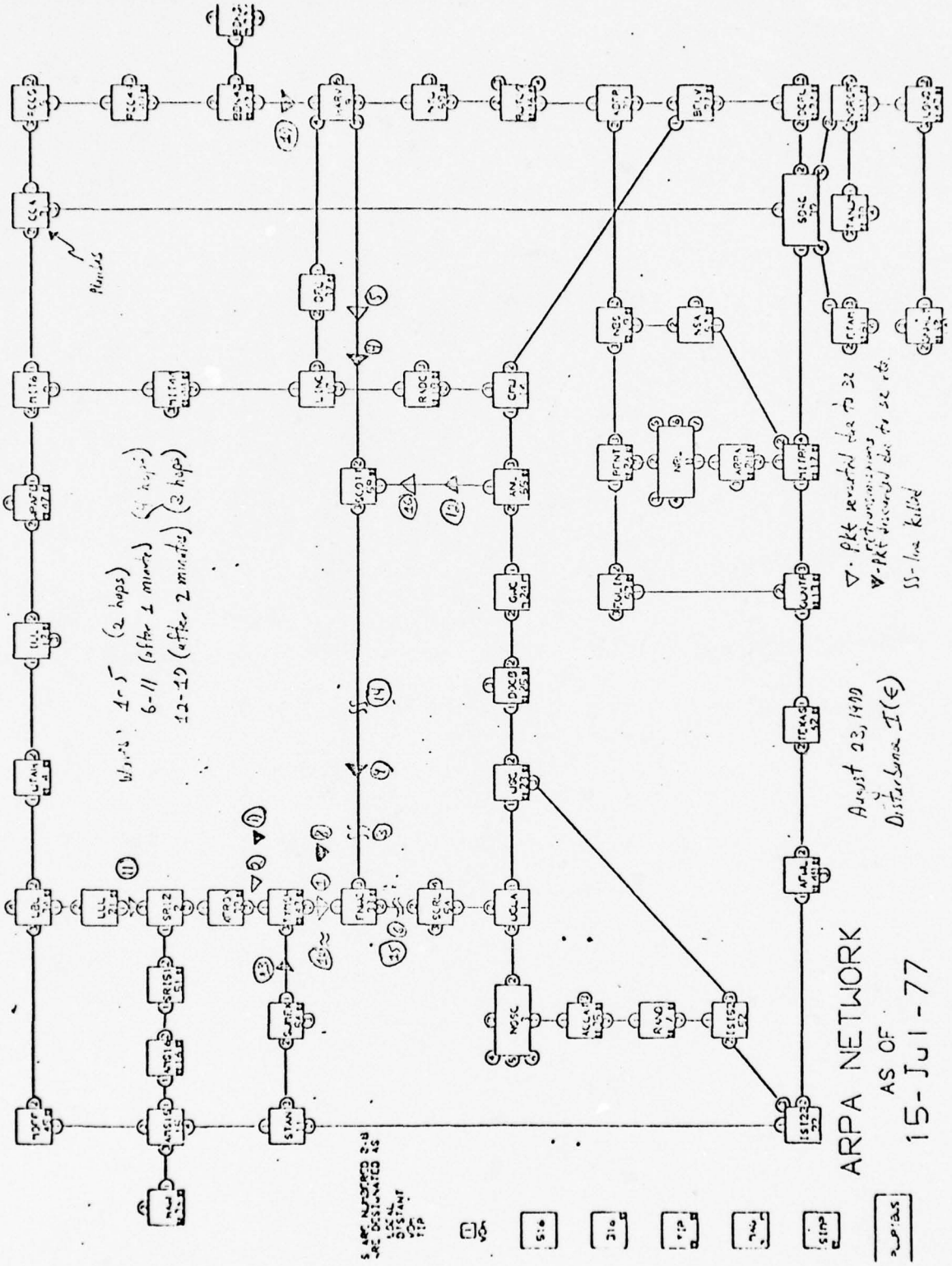


Figure 40

BEST AVAILABLE COPY

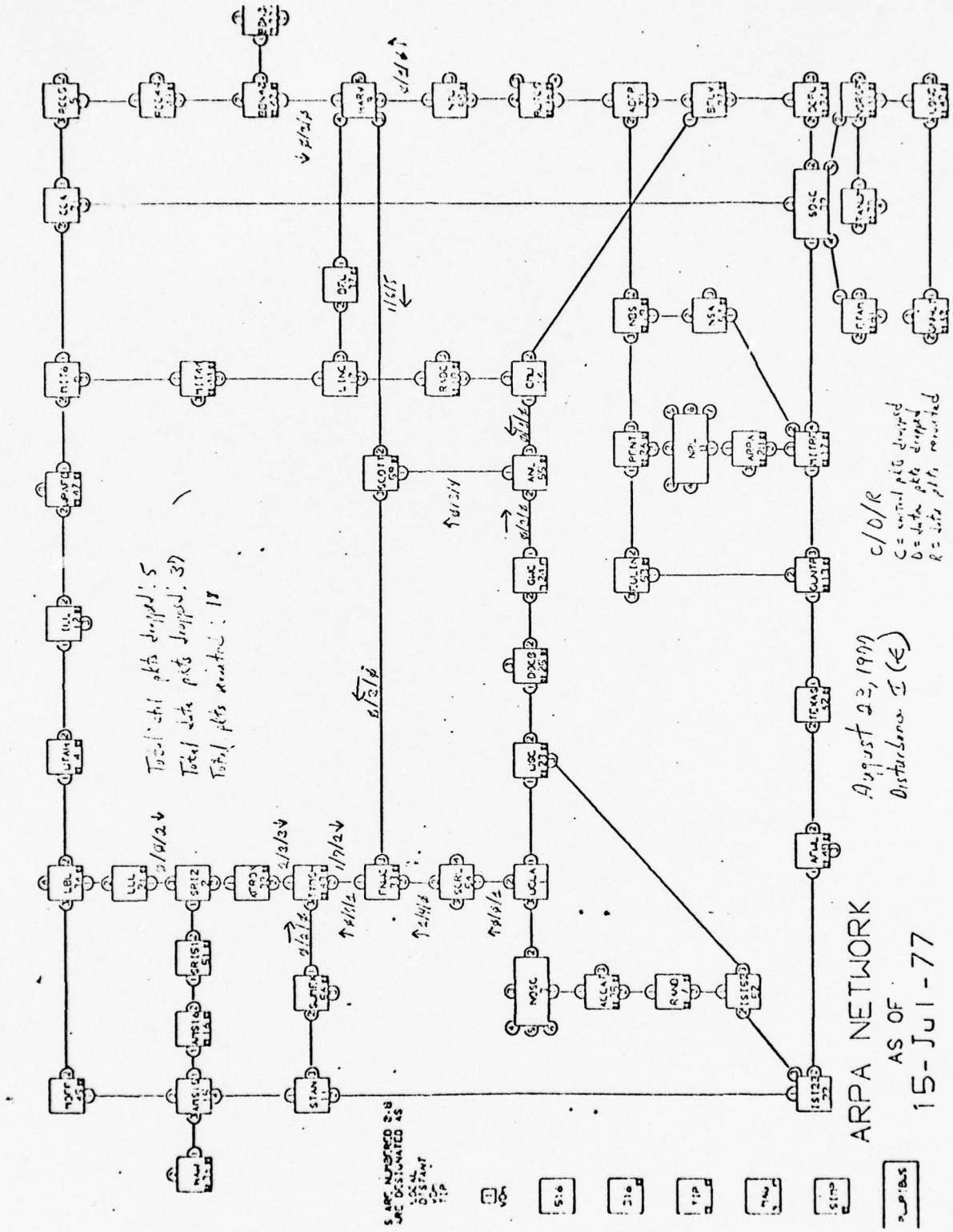


Figure 41

BEST AVAILABLE COPY

Experiment II

We used the case III message generators (see Appendix II), and slowed FNWC. The case III and reduced case I message generators create approximately the same traffic pattern, but the load is increased due to control messages being generated every two seconds. We obtained a disturbance very similar to that of experiment I, with congestion backing up short distances from FNWC. We also observed the three non-spreading waves again, with one minute quiet periods between them. A total of 43 packets was dropped, of which 6 were control packets, and a total of 5 packets was rerouted.

We observed no significant difference between the effect of the reduced case I message generators and the case III generators.

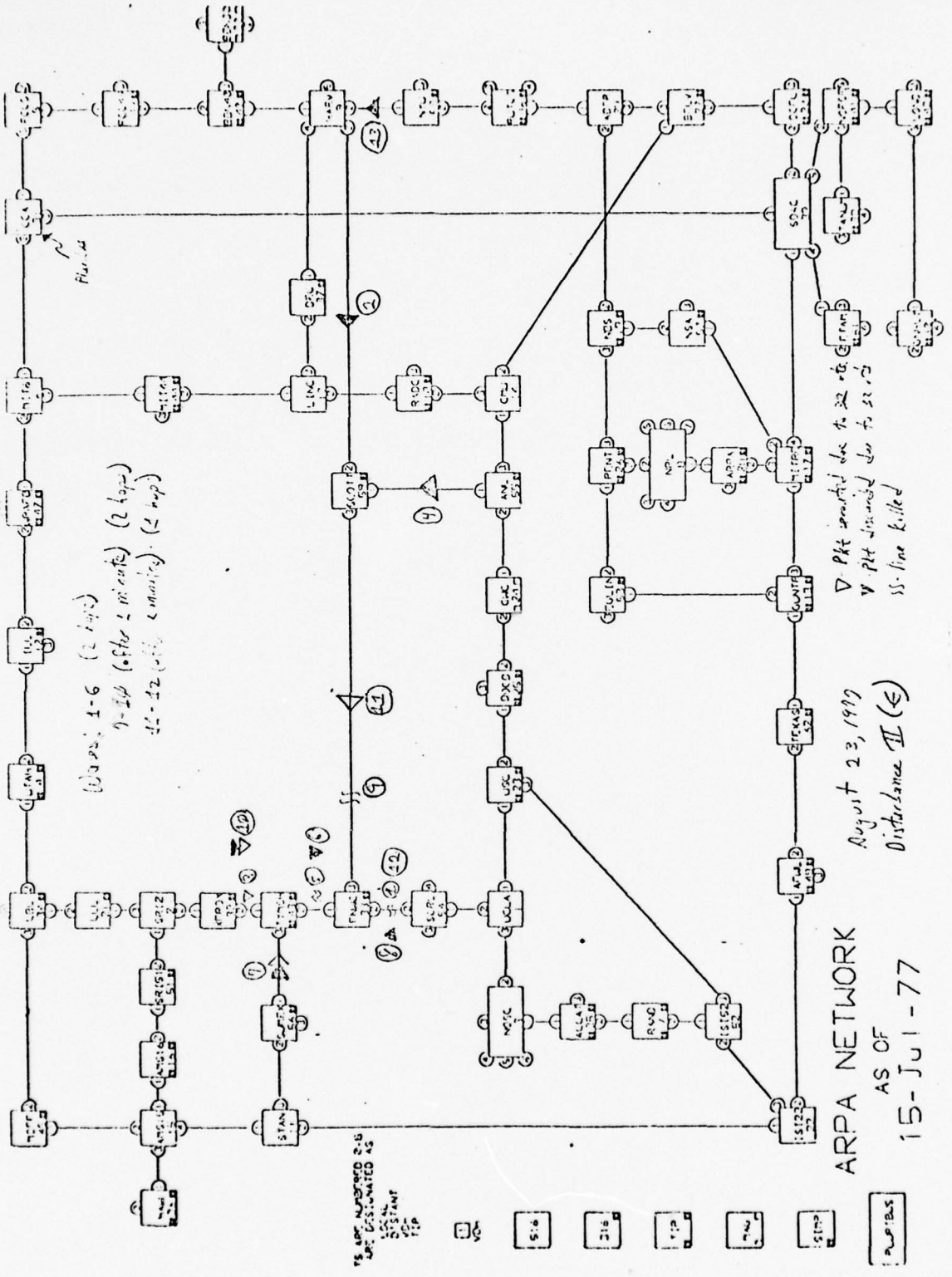


Figure 42

BEST AVAILABLE COPY

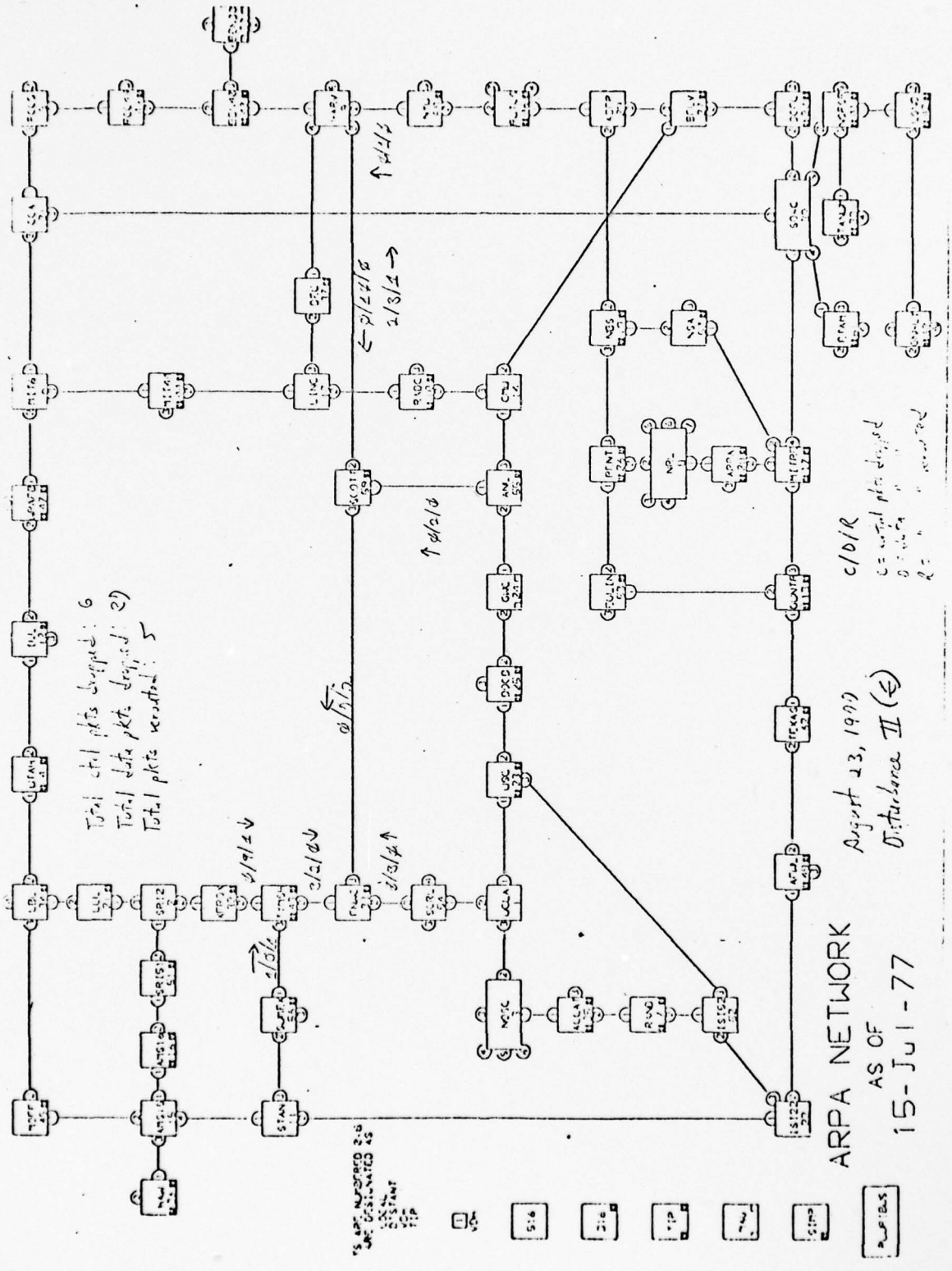


Figure 43

BEST AVAILABLE COPY

Experiment III

We used the case III and case IV message generators, which generated large amounts of data traffic and control messages in the vicinity of both FNWC and GUNTR. We also slowed both FNWC and GUNTR. We hoped thereby to create a two-focus disturbance. We did create two small disturbances, with IMPs up to 3 hops from GUNTR or FNWC discarding or rerouting packets. However, we did not create a real two-focus disturbance, since there was no apparent interaction between the disturbance focused on GUNTR and that focused on FNWC.

A total of 93 packets was discarded, 32 of which were control packets, and 14 packets were rerouted. Again, the disturbance occurred in waves, with about a minute of quiet time between them.

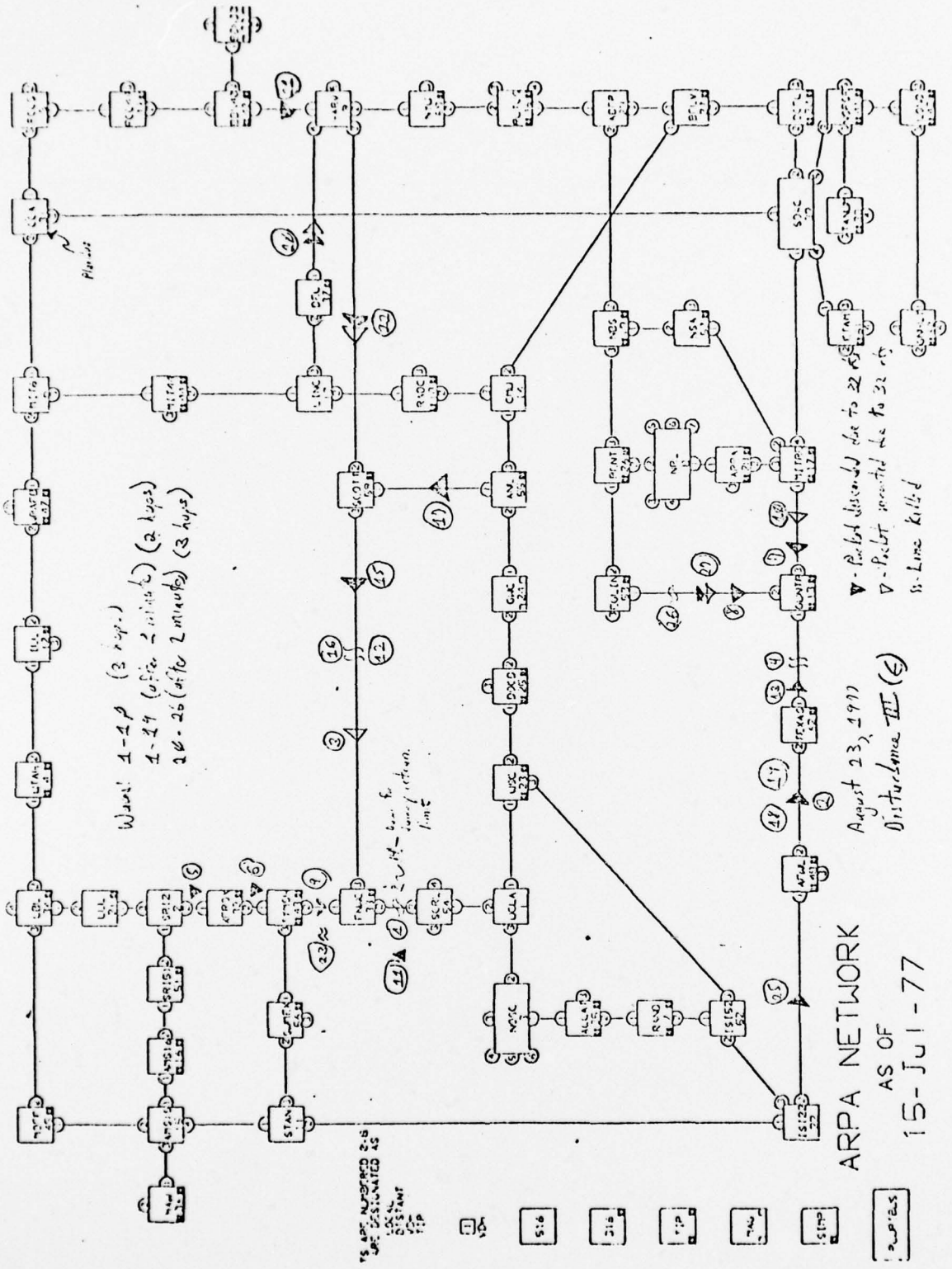


Figure 44

BEST AVAILABLE COPY

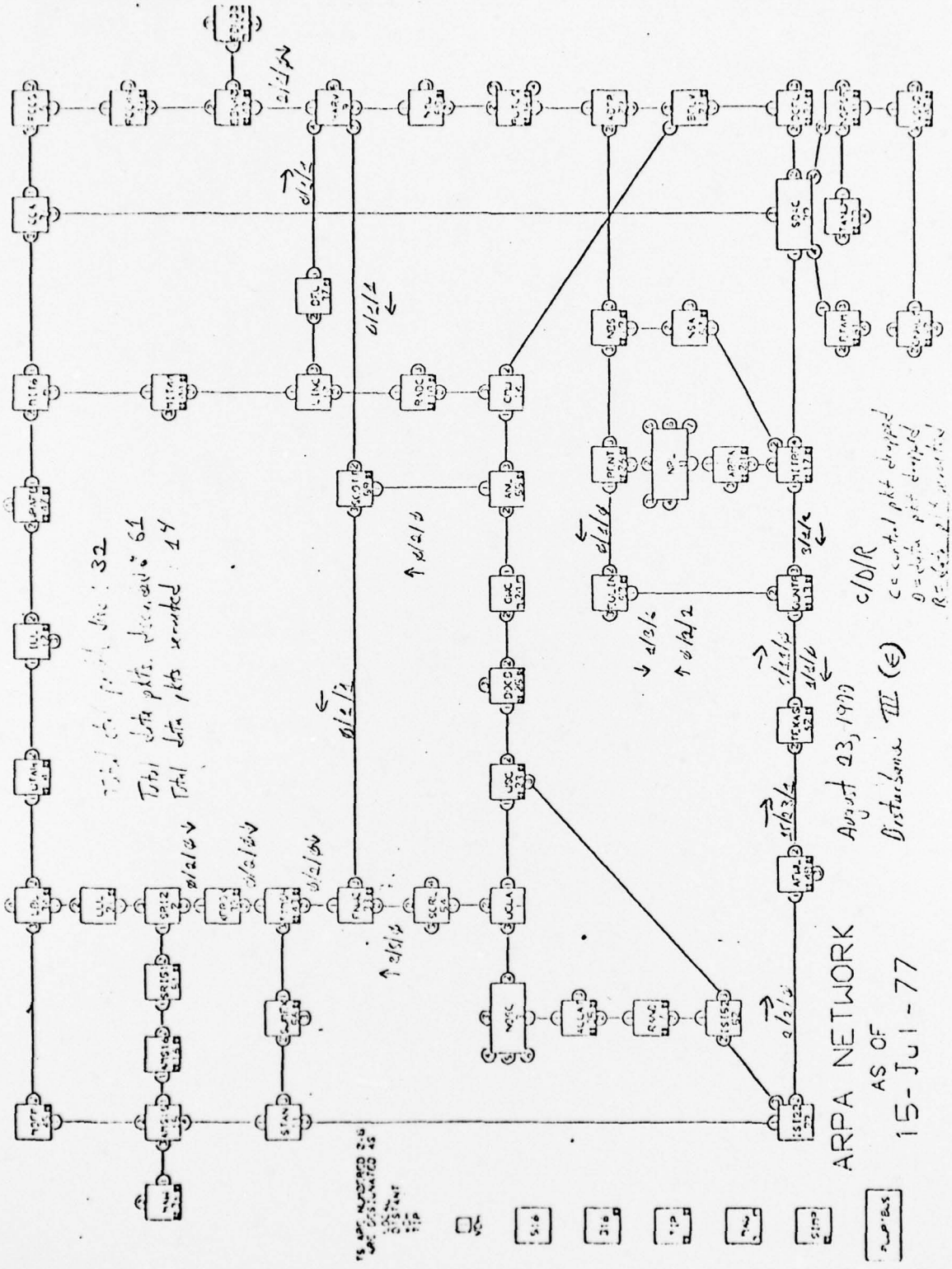


Figure 45

BEST AVAILABLE COPY

Experiment IV

For this experiment, we altered the reduced case I message generators so that they were all sending traffic to FNWC at the maximum rate. We then gave FNWC a restart command, causing it to cleanly bring itself down and then up again. In this way we hoped to simulate the condition wherein the network fills up with traffic destined for an IMP which cannot receive it.

Unfortunately, we did not realize that an IMP which is being restarted continues to accept (and discard) traffic for itself. Thus we failed to bring about the hoped-for condition. We caused only a very short (lasting only 5 seconds) and minor disturbance, with packets being discarded up to 2 hops from FNWC. A total of 34 packets was discarded, 9 of them control packets, and no packets were rerouted.

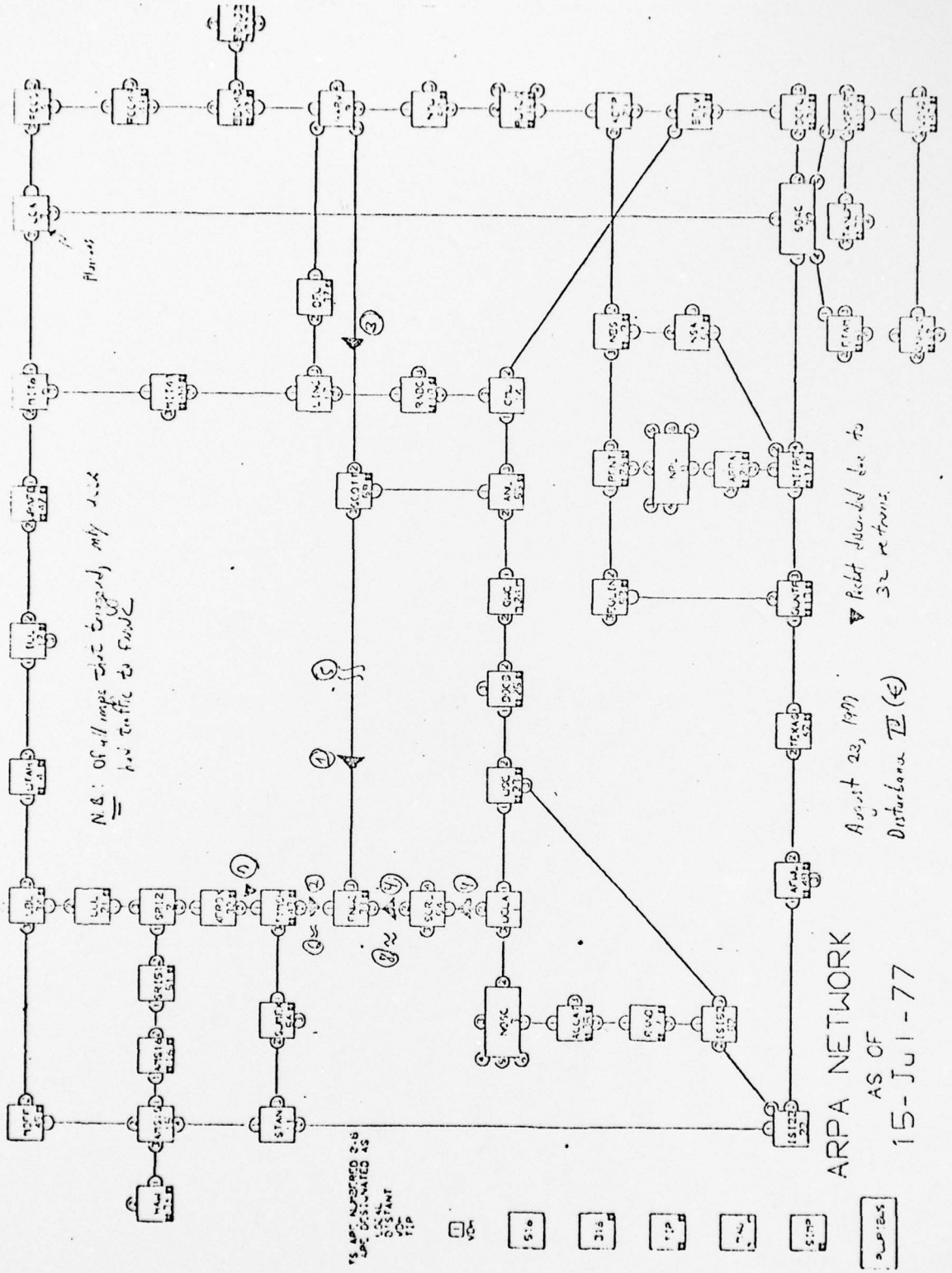


Figure 46

BEST AVAILABLE COPY

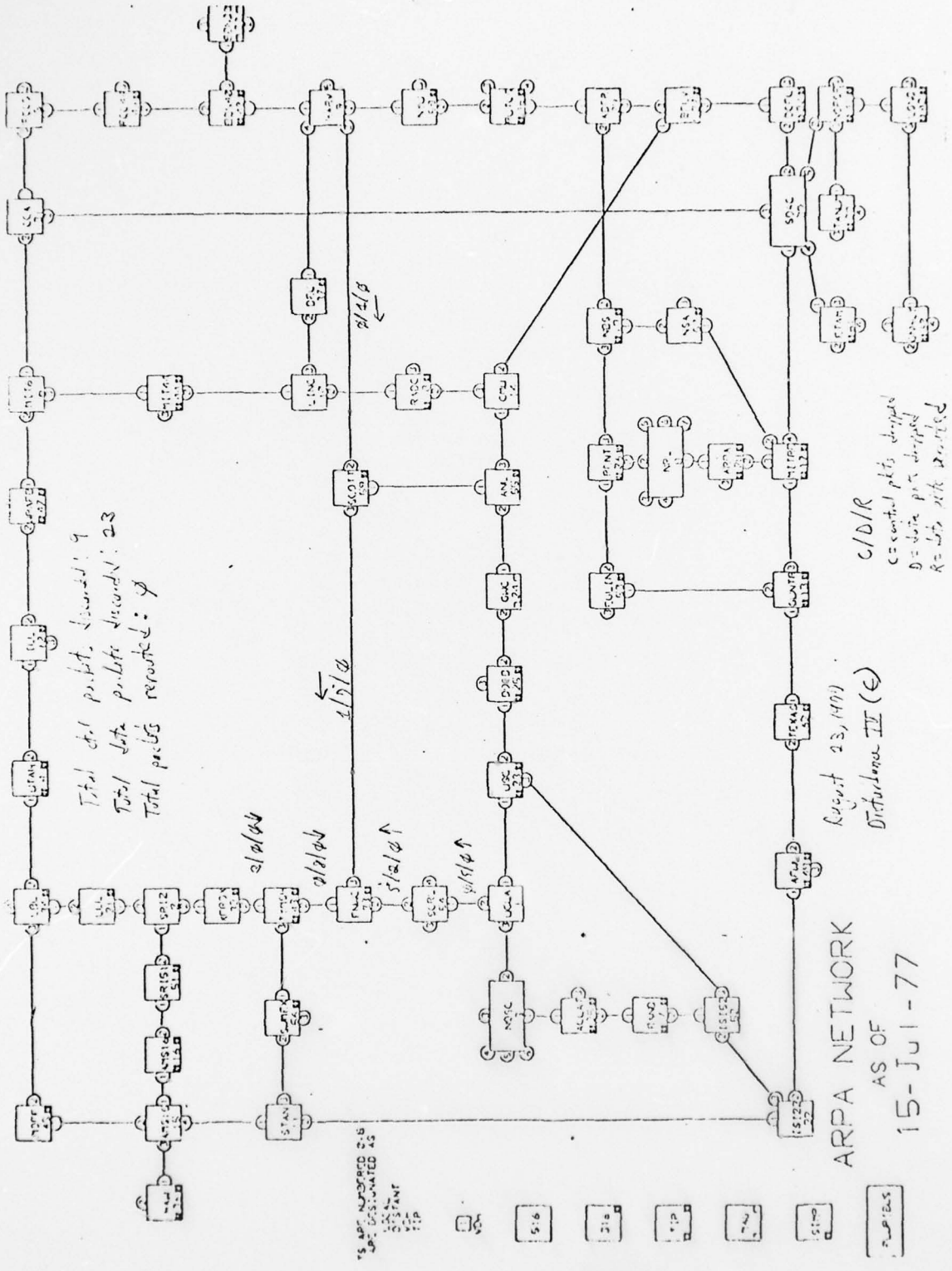


Figure 47

BEST AVAILABLE COPY

Appendix IV. The ARPANET Routing Algorithm

Terminology. We begin by explaining the terminology used below to describe the ARPANET routing algorithm. The actual IMP program is written in assembly language and does not use the terms employed here; we have re-written it below in a PL/I-like language. The algorithm performs two basic functions: determining whether each other IMP is reachable, and calculating the path of least delay to those IMPs which are reachable. The program maintains a number of tables indexed by IMP; these all have one entry for each of the NN IMPs in the network. All these tables carry checksums which are a function of the NN entries, and which the program periodically verifies in order to minimize and localize the effects of any hardware or software failures. ROUTE is the directory giving the output line corresponding to the shortest hop path to all IMPS; DROUTE is the analogous minimum delay directory. HOP and DEL are the tables containing the number of hops and estimated delay on these paths. The maximum values in these tables are denoted by HMAX and DMAX respectively. HOPIN and DELIN are the two parallel tables which together constitute the routing message received at an IMP. HOPOUT and DELOUT are the tables constructed by an IMP to form its next routing message to be transmitted, and to be used as the new HOP and DEL tables. The algorithm uses a technique called hold down to ensure that an IMP will not base its routing decisions on old information sent to it by its neighbors. The technique consists of using a timer, HOLDT for the best hop path

and DHOLDT for the minimum delay path, to wait a short interval before deciding to change to a new path. This interval is long enough for all the neighbors of the IMP to learn about the new information, so that the IMP can make its decision accurately, rather than on the basis of "echos" of its old information. This interval is set to HTMAX. In order to determine if hold down is necessary on the minimum delay path, the program keeps two counters, DELOLD and DELTHS, which accumulate the increase in delay in the previous update period, and in the current update period respectively. Finally, the program inserts a transition delay so that all IMPs in the network learn about the transition before it becomes final. GODOWN is a table of timers which are set to GDMAX when an IMP begins to go down; COMEUP is a similar set of timers set to CUMAX when an IMP is first detected as reachable.

Routing Message Input Processing. When a routing update message is received, the input module first verifies the checksum on the message. If it is incorrect, the program reports an inter-IMP failure and ignores the message. Otherwise, it puts the message on a queue for a lower-priority routine. This routine then takes the following actions:

1. Take the routing message off the input queue.

2. Verify the checksum on the following code before executing it.

3. Abort if the checksum is incorrect.

4. Ignore the message if the input line is down, or if it is looped, or if the serial number on the routing message is the same as that on the last message received, or if the format compatibility number does not match the number used by the receiver.

5. Initialize the checksum on the output message to be generated, put in the sender's node number, compatibility number, and the incremented serial number.

6. Perform the basic routing computation as follows:

```
For I<-1 to NN do;
  if I=SELF then HOPOUT(I),DELOUT(I),ROUTE(I),DROUTE(I)<-0;
    else do;
      if ROUTE(I)=RMAX
        then /* I has been declared unreachable */
          if HOPIN(I)>HMAX
            then /* I is still not reachable via J */
              do; DELOUT(J)<-DMAX; HOPOUT(I)<-HMAX;
              end;
            else /* a new path to I has been discovered */
              do; COMEUP(I)<-CUMAX; /* Set Come-up timer */
                  ROUTE(I)<-J;
                  HOPOUT(I)<-HOPIN(I)+1;
              end;
        else /* I has not been declared unreachable */

          if ROUTE(I)=J
            then /* Current min-hop path is via J */
              do; if HOPIN(I)+1>HOP(I)
                     then /* Best path got worse, hold it down */
                       HOLDT(I)<-HTMAX;
                  if HOPIN(I)+1<HMAX
                     then /* Best path still exists */
                       do; GODOWN(I)<-0;
                           /* Make sure godown timer is off */
                           HOPOUT(I)<-HOPIN(I)+1;
                       end;
                     else /* I is no longer reachable through J */
```

```
                        do; if GODOWN(I)=0 then GODOWN(I)<-GDMAX;
                            HOPOUT(I)<-HMAX;
                        end;
                end;
              else /* current min-hop path is not through J */
                if HOLDT(I) not = 0
                  then /* I is being held down, make no changes */
                    HOPOUT (I)<- HOP(I);
                  else if HOPIN(I)+I>HOP(I)
                    then /* No reason to switch paths */
                      HOPOUT(I)<-HOP(I);
                    else /* New min-hop path is via J */
                      do; ROUTE(I)<-J; GODOWN(I)<-0;
                          HOPOUT(I)<-HOPIN(I)+1;
                      end;

            if DROUTE(I)=J
              then /* current least-delay path is through J */
                do; D <-DELIN(I)+DELLOC-DEL(I);
                    /* D<-change in delay since last update */
                    DELOLD(I)<-MAX (DELOLD(I)+D, 0);
                    DELTHS(I)<-MAX (DELTHS(I)+D, 0);
                    if DELOLD(I)>8 or DELTHS(I)>8
                       then /* delay has gotten worse - hold down */
                         DHOLDT(I)<-HTMAX;
                    DELOUT(I)<-DELIN(I)+DELLOC;
                end;
              else /* current least delay path is not through J */
                if DHOLDT(I) not = 0
                  then /* I is being held down, make no changes */
                    DELOUT(I)<-DEL(I);
                  else if DELIN(I)+DELLOC>DEL(I)
                    then /* No reason to switch paths */
                      DELOUT(I)<-DEL(I);
                    else /* New least delay path is via J */
                      do; DROUTE(I)<-J;
                          DELOLD(I), DELTHS(I)<-0;
                          DELOUT(I)<-DELIN(I)+DELLOC;
                      end;
        end;
      /* update checksums on any tables with changed entries */
end;
```

7. Finalize the routing message checksum and put it in the message buffer.

8. If the current routing message buffer is not being transmitted on any line, circulate the triple buffer system as follows:

1. assign the new message to the output buffer;
2. assign the output buffer as the free buffer area;
3. assign the free buffer area as the buffer in which the next message will be built.

Routing Message Output Processing. When the program receives the completion interrupt on one of the IMP's lines, it first checks to see if the send-routing flag is set. Routing transmissions take priority over all others. If the flag is set, the program takes the following steps:

1. Verify the checksum on the following code before executing it.
2. Abort if the checksum is incorrect.
3. Locate the next routing message buffer from which to transmit, and verify that the message has a correct checksum.
4. Report an intra-IMP failure if the checksum is in error, and resynchronize the line protocol before continuing.
5. Otherwise, initiate the output and dismiss the interrupt.

Periodic Routing Processing. Every 640 milliseconds the program takes the follow steps:

1. Verify the checksum on the following code before executing it.

2. Abort if the checksum is incorrect.

```
For I<-1 to NN do;
  if HOLDT(I)>0
    then /* decrement hold down timer for hops: 1.28-1.92 secs */
    HOLDT(I)<-HOLDT(I)-1;
  if DHOLDT(I)>0
    then /* decrement hold down timer for delay: 1.28-1.92 secs */
    DHOLDT(I)<-DHOLDT(I)-1;
  DELOLD(I)<-DELTHS(I); DELTHS(I)<-0;
  if GODOWN(I)>0
    then /* decrement timer for IMPs going down: 7.7-11.5 secs */
    do; GODOWN(I)<-GODOWN(I)-1;
        if GODOWN(I)=0
          then /* IMP I just died */
            do; ROUTE(I)<-RMAX; DROUTE(I)<-RMAX;
            end;
    end;
  if COMEUP(I)>0
    then /* decrement timer for IMPs coming up: 34.6-40.3 secs */
    COMEUP(I)<-COMEUP(I)-1;
  /* update checksums on any tables with changed entries */
end;
```

3. Verify the checksums on the various tables: ROUTE, DROUTE, HOLDT, DHOLDT, GODOWN, COMEUP.

Every 25 milliseconds, the program takes the following steps:

1. Set the send-routing flag for the appropriate lines according to their line speed and line utilization. The rule used is that the overhead due to routing messages should range from 3% of the line bandwidth on busy lines to 15% of the line bandwidth on idle lines.

2. Attempt to circulate the routing message buffers as in step 8 in the section above on Routing Message Input Processing.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN report no. 3641	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): ARPANET routing study	5. TYPE OF REPORT & PERIOD COVERED: Final report
6. PERFORMING ORG. REPORT NUMBER: BBN report no. 3641	
7. AUTHOR(s): John M. McQuillan, Ira Richer, Eric Rosen	8. CONTRACT OR GRANT NUMBER(s): DCA200-77-C-0616
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman, Inc. 50 Moulton Street Cambridge, Ma 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communications Agency Code 535 Arlington, Va	12. REPORT DATE: 5 September 1977 13. NUMBER OF PAGES: 170 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Communications networks, On line systems, Routing, Systems engineering, ARPANET

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

For several years ARPANET has been subjected to occasional disturbances (referred to as network glitches or network disturbances) Frequency of these disturbances increased from once every 2 or 3 days to 3 or 4 a day in April 1977. This increase was traced to to a change in the IMP software governing the satellite line between SDAC and NORSAR reducing the number of packets per flight. When the software was changed back to allow 16 packets per flight the

next page

number of disturbances decreased dramatically.

However network disturbances are not the result of this one isolated bug. Other causes include: faulty IMP hardware, software bugs, circuit difficulties, traffic overloads, etc. The real problem is not any particular irritant but the vulnerability of the ARPANET to congestion caused by such irritants.