ESD-TR-90-151

2

Annual Report

Knowledge-Based System Analysis and Control Defense Switched Network Task Areas

30 September 1990

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

Prepared for the Defense Communications Agency
under Air Force Contract F19628-90-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Defense Communications Engineering Center of the Defense Communications Agency under Air Force Contract F19628-90-C-0002.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

KNOWLEDGE-BASED SYSTEM ANALYSIS AND CONTROL DEFENSE SWITCHED NETWORK TASK AREAS

H.M. HEGGESTAD
Group 21

ANNUAL REPORT SUBMITTED TO
MR. TOM LAM
DCEC, DRFB
1860 WIEHLE AVENUE
RESTON, VA 22090-5500

1 OCTOBER 1989 – 30 SEPTEMBER 1990

ISSUED 17 JANUARY 1991

LEXINGTON MASSACHUSETTS

ABSTRACT

A major activity during FY90 has been the design and implementation of a network management expert system to operate in the Integrated Workstation (IW) that was developed during FY90 for use by ACOC personnel at DCA-Eur to perform DSN network management tasks. The IW was successfully tested on live and archived data, and on fault conditions deliberately induced by switch technician actions, during the period 25-28 September 1990. All parties declared that the IW features and demonstrated performance were valuable and successful. The Expert System was well received, in particular. An IW terminal has been installed on the floor of the ACOC, and the staff have been directed to familiarize themselves with its operation.

A number of changes and improvements were made in CCSIM and related programs. All were converted to run under SUN OS 4.0.3. The graphics interface program was rewritten to use the X-window system. A new document called "Using The Call-By-Call Simulator (CCSIM)" has been written, and the "CCSIM User's Manual" and the "CCSIM Software Top Level Design Document" delivered in FY90 are to be updated early in FY91.

Work was performed in expert systems development efforts for DCS transmission system control with two main components: implementation of the TRAMCON Event Generator (TEG), and participation in the Tech Control Automation Proof-of-Concept System (TCAPS). TEG had been specified during FY89, and has now been written and successfully installed and demonstrated at a number of locations. Lincoln's RADC-sponsored MITEC expert system was incorporated in a DCA-sponsored set of TCAPS field demonstrations.

Table of Contents

LIST OF FIGURES

1.0 Introduction and Summary

A major activity during FY90 has been the design and implementation of a network management expert system to operate in the Integrated Workstation (IW) that was developed during FY90 for use by ACOC personnel at DCA-Eur to perform network management tasks. The IW is the part of the Defense Switched Network (DSN) Integrated Management Support System (DIMSS) that is concerned with near-real-time monitoring and control of the network. The IW integrates on a common platform (a SUN SPARCSTATION) the network management capabilities that were demonstrated at DCA-Eur at the end of FY89. These were the Network Management Support System (NMSS), the LARS neural net anomaly recognizer, and the Network Management Expert System (NMES). The first two systems were developed by GTE. The development of NMES was a major part of our work in FY89. Development of the IW has been a joint effort between GTE and Lincoln Laboratory with GTE having primary responsibility for the design and implementation of the IW as a whole. Lincoln's primary responsibility has been the design and implementation of the IW Expert System (IWES), and we have had some interaction with GTE on the overall IW design. Chapter 2 of this report provides the details.

The IW was successfully tested on live and archived data, and on fault conditions deliberately induced by switch technician actions, during the period 25-27 September 1990. On 28 September the IW was demonstrated to the Commander of DCA-Europe (Col. Reinmann) by two of his own staff. All parties declared that the IW features and demonstrated performance were valuable and successful. The Expert System was especially well received, as explained below. An IW terminal has been installed on the operations floor of the ACOC, and the staff have been ordered to familiarize themselves with its operation and to use it continually. A full report of the DCA-Europe testing is provided in Chapter 3.

A number of changes and improvements were carried out in FY90 with respect to CCSIM and its related programs. All programs were converted to run under the new SUN Operating System 4.0.3 early in the fiscal year. The graphics interface program was rewritten to use the X-window system instead of the previously used Suntools environment which is no longer being supported. The change makes our graphics compatible with the IW environment and improves portability to other machines, but it has exacted a price in speed of response and memory requirements. Chapter 4 describes the major changes and improvements in CCSIM and experiments with CCSIM that were carried out in FY90. There are no current plans for further CCSIM development. A new document called "Using The Call-By-Call Simulator (CCSIM)" is being written, and the "CCSIM User's Manual" and the "CCSIM Software Top Level Design Document" previously written by 3S, Inc. are being brought up to date. By agreement with DCEC, delivery of these documents will take place in early FY91.

A component of DCEC FY90 tasking for Lincoln Laboratory was DRTV-funded expert systems development efforts for DCS transmission system control. This tasking is referred to as TRAMCON/DPAS alarm integration, and has had two main components in FY90: implementation of the TRAMCON Event Generator (TEG), and participation in the Tech Control Automation Proof-of-Concept System (TCAPS). TEG had been specified during FY89, as a result of substantial efforts to gather and articulate all the rules and relationships among TRAMCON equipment items, all the types of faults, and all the primary and sympathetic alarm patterns. TEG has been installed and demonstrated at a number of locations, as described in Chapter 5. The TCAPS interactions started with a modest part for Lincoln's RADC-sponsored MITEC expert system in a DCA-sponsored set of field demonstrations at Ft. Detrick, Maryland. In the course of the year the involvement grew to include two MITECs at two field sites (Ft. Detrick and the Pentagon). The details are described in Chapter 5.

A set of Appendices provide detailed documentation of specific areas of FY90 work, as follows:

APPENDIX A.	IWES System Design
APPENDIX B.	IWES Monitor & Rule Descriptions
APPENDIX C.	IWES Results File Examples
APPENDIX D.	Statistical Data Base Software Design
APPENDIX E.	CCSIM Network Management Controls

2.0 IWES Design and Implementation

A major activity during FY90 has been the design and implementation of a network management expert system to operate in the Integrated Workstation (IW) that was developed during FY90 for use by ACOC personnel at DCA-Eur to perform network management tasks. The IW is the part of the Defense Switched Network (DSN) Integrated Management Support System (DIMSS) that is concerned with near-real-time monitoring and control of the network. The IW integrates on a common platform (a SUN SPARCSTATION) the network management capabilities demonstrated at DCA-Eur at the end of FY89. These were the Network Management Support System (NMSS), the LARS neural net anomaly recognizer, and the Network Management Expert System (NMES). The first two systems were developed by GTE. The development of NMES was a major part of our work in FY89. Development of the IW has been a joint effort between GTE and Lincoln Laboratory with GTE having primary responsibility for the design and implementation. Lincoln involvement has been limited to the design and implementation of the IW Expert System (IWES) and some interaction with GTE on the overall IW design.

In this section we briefly describe the overall IW design and operating philosophy followed by a high-level description of the IWES design and functionality. Further detail on IWES can be found in Appendices A, B, C, and D.

2.1 IW System Design

Figure 2-1 shows a simplified flow diagram for the IW. The lines in the diagram represent conceptual data flows. The actual data transfers take place through a common data base that is accessed by the program modules, and messages are sent through a message dispatcher module (not shown) to inform the modules that desired data has been deposited in the data base.

The DAI (Data Acquisition Interface) polls the switches for Operational Measurement (OM) reports every five minutes. These reports are the basic input to the IW. They are reformatted by the COMM PROC module and made available to the USER INTERFACE (UI), NEURAL NET (NN), and EXPERT SYSTEM (IWES) modules. The UI provides displays of the OM data in the format similar to that used by the earlier IBM PS/2-based NMSS system. In addition, the UI module supports all other NMSS features, e.g., manual control applications, in the IW.

The NN module processes the OM reports to recognize anomalies in the switch and trunk data. It writes a file of recognized anomalies to the database and signals to the UI that icon colors in the display should be changed to indicate abnormal status of switches and/or trunks.

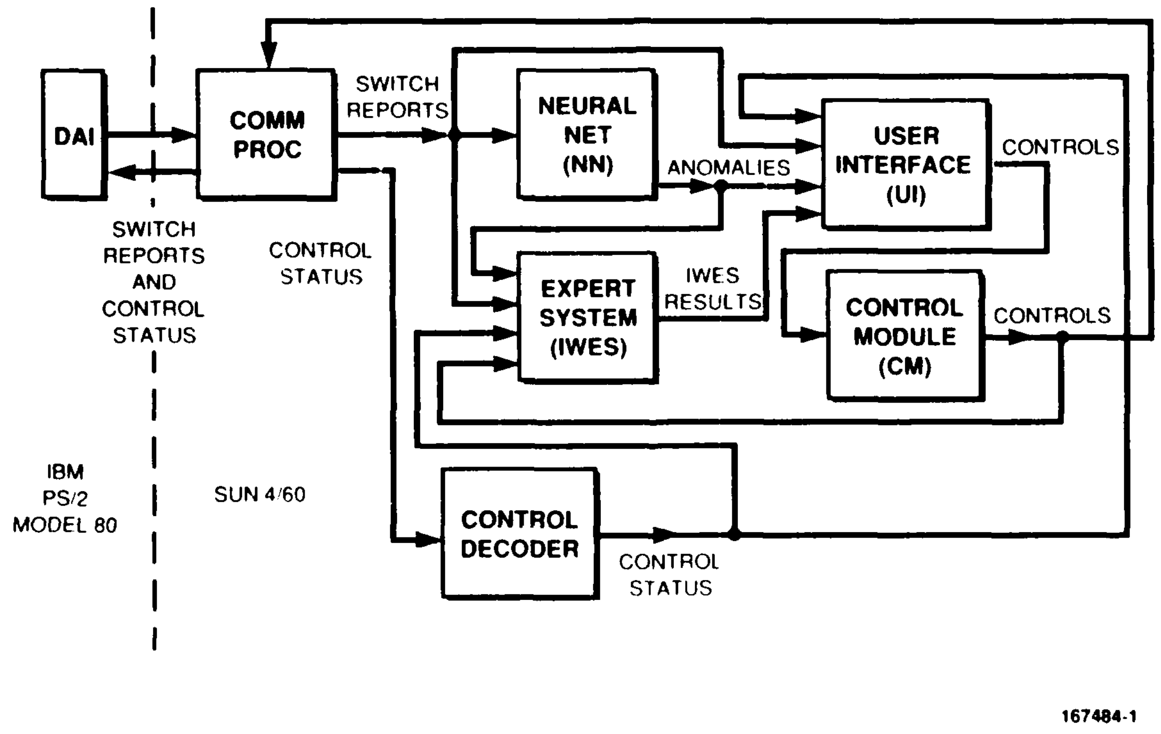

Figure 2-1
Integrated Workstation

IWES examines the file of anomalies generated by the NN and the raw OM data and generates text for presentation to the operator. The text includes a description of the problem corresponding to the recognized anomaly, recommendations for action and network management control applications (if any), and an explanation of how the problem is recognized together with the data needed to help the operator confirm or deny the reality of the problem. The text also includes a history of observations relevant to the problem.

When IWES has text for a switch or trunk problem, it signals the UI to color a segment of the appropriate switch icon to indicate to the operator that recommendations are available. Even though IWES has some capability to recognize the presence of a problem in the absence of a NN anomaly detection, it was agreed that capability would be suppressed in the IW environment. Consequently, IWES generates text only for situations in which a NN anomaly has at some time been detected. Since IWES remembers situations over time, it can generate recommendations for report intervals in which no NN anomaly was detected. For example, IWES will recommend the removal of controls when problems are observed to have gone away.

The text generated by IWES is written to a 'results' file that is read by the UI when the operator clicks the mouse button while pointing to the 'recommendations-available' icon. If the recommendations include the application of network management controls, the IW design allows the operator to approve the control application and have them sent directly to the switches through the CONTROL MODULE and the COMM PROC module. Alternatively, the operator can choose not to apply the recommended controls, or to apply others using the facilities of the UI and the CONTROL MODULE to assure that the commands to the switches will be properly formatted.

In order to facilitate the direct application of recommended controls, the IW design calls for them to appear in two forms in the 'results' file. One is text for presentation to the operator. The other is a special format meaningful to the CONTROL MODULE. The latter form was not operational at the time of the September tests.

In addition to polling the switches for OM reports, the COMM PROC module also polls for control status at the switches. The resulting reports are decoded and made available to the operator on request. Figure 2-1 shows that information also being fed to IWES, but that path is not currently operational, and its utility is questionable since the information presently being returned lacks the detail needed for use by IWES. Another path in the figure shows control application information being fed back to IWES from the CONTROL MODULE. This path is intended to provide IWES with information about the operator's control actions. It is not yet implemented, but could easily contain the requisite detail. The IW provides a logging function through its data base so that even if the operator does not use the mouse to observe problems identified by the NN and IWES, the anomalies, recommendations, etc. are preserved along with the OM report data for later analysis.

2.2 IWES System Design

In the FY89 annual report we described an independent complete NMES which processed data from the DAI, recognized patterns, confirmed problems, devised controls and actions, and displayed results to a network management operator. NMES was implemented with the expert system shell CLIPS (C Language Integrated Production System), developed by NASA/Johnson Space Center. During FY90, NMES has been transformed into an integrated part of the (IW), and is now called the Integrated Workstation Expert System (IWES).

In order to permit integration with the IW, modifications were made to the FY89 NMES structure. One major difference is that two inputs have been added to the IWES front end: statistical database values (see Section 2.4) which vary with time of day, and neural network anomalies, which are processed as potential problems to be validated by IWES. Another major difference is that actions and controls are now being written to a file for the IW to display instead of being sent to the NMES graphics. In the future, switch control actions recommended by IWES, and accepted by the operator, will be sent via the Control Module to be applied at the switch.

Briefly reviewing the IWES structure (Fig. 2-2), at the lowest level, when IWES is initialized, network representation and statistical database values are read in and stored in C structures. During each polling period, switch reports and neural network anomalies are received, processed and stored in C structures. These structures are scanned by monitors (implemented in C) to identify neural network anomalies and other interesting features and useful information. The outputs of the monitors are asserted into CLIPS as facts comprising the abstract state of the network. Problems are identified and validated by a module comprised of CLIPS rules which looks for patterns of symptoms in the abstract state of the network.

Validation of a problem announced by the neural net involves assuring that IWES monitors detected corresponding interesting features, that were abnormal for the time of day, in the raw data. Validated and unvalidated problems are confirmed over time by a confirmation module. A planning module recognizes a problem and devises actions and controls to improve or correct the situation. An observation module observes the effects of the applied controls over time, and recommends removal of the

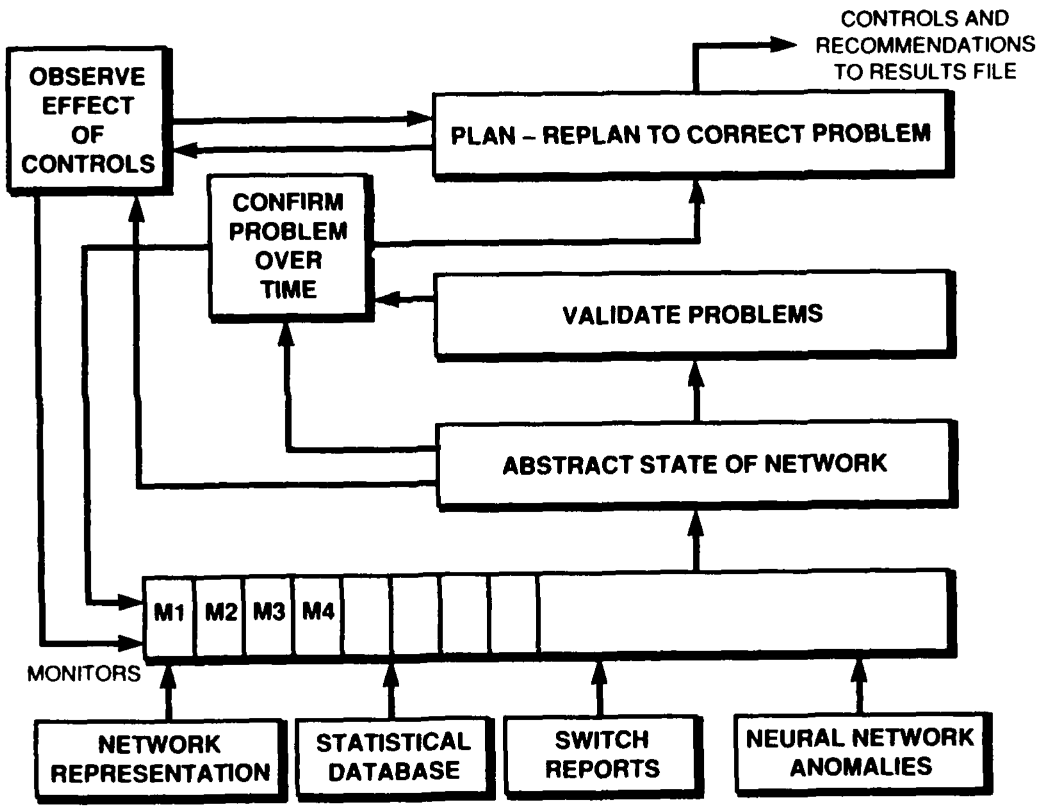

Figure 2-2
Expert System Structure

controls when the problem no longer exists. Each module adds text observations to a C structure.

For each problem detected, whether validated or unvalidated, an IWES results file is written which contains the results of the planning module, recommendations for actions and controls, an explanation of how the problem is recognized by a network manager, and the array of observations. Each polling period, a new results file is created and read in by the IW user interface to be displayed to the user.

IWES does not have the same type of smoothing-over-time component that was present in NMES. Smoothing over time is no longer applied during confirmation of anomalies, although it is still applied in some of the observations displayed to the user. Originally we had introduced time smoothing to reduce the probability of misrecognition due to noise on the lines between the switches and the DAI. When using 15 minute polling, we found that data errors did not occur frequently enough to justify the delay introduced by time smoothing. The change to 5 minute polling periods has raised this issue again. Analysis shows that call counts and holding times are very noisy when computed from 5 minute data. In the future, threshold comparisons for these and other fields that are noisy should be based on 15 minute averages of live data, to prevent frequent false alarms.

The IWES communicates with the other IW processes via the integrated workstation message dispatcher(IWMD). Socket connections are made with the IWMD at initialization time. Formatted messages are used to communicate between the two processes. The IWMD alerts the IWES when raw data, neural network results, and control status are available and IWES informs the integrated workstation user interface(IWUI) when it has results ready for display.

2.3 Anomaly Verification

In the September 1990 IW in Europe, the neural net is designated as the primary detector for network anomalies. The role of the IWES is to confirm or discount the anomaly using its own set of detectors and a data base of expected values based on time of day and trunk group or switch identity.

Section 2.3.1 describes how neural net anomalies are verified by the IWES. Section 2.3.2 describes why a statistical data base with expected values for important trunk group and switch parameters is used to help detect and verify anomalies (see section 2.4 for a description of the data in the statistical data base). Additional details and future enhancements are found in Appendices B (IWES Monitors and Rules) and D (Statistical Data Base Software Design).

2.3.1 Neural Net Anomaly Verification

The neural net anomaly is read by IWES and is asserted as a CLIPS fact. Since we did not know until virtually the week before the September tests what anomalies the IWNN would recognize, the IWES was prepared to deal at some level with all the items in the following list. An asterisk beside an anomaly indicates that the expert system has the knowledge to confirm the anomaly. There is an entry for each original LARS neural net diagnosis, those recommended in the Network Management Situation Diagnosis message (Aug.24.1990), and those recognized by the Neural Net during the September 1990 testing. The expert system was capable of validating any IWNN diagnosis made during that week.

For Switch diagnosis:

outage_remote*	rcvr-overflow-mf*
cpumismatch*	rcvr_overflow_mf*
misc_others*	perm_signal*
misc_other*	permanent_signals*
cpuinits*	facility_hits
congestion_remote	degraded_remote
congestion_reporting	degraded_reporting
cpu_simplex	internal_problem
cp_origination_problem	rcvr_problem
dist_signal_prob	line_frame_problem
trunk_system_failure	excess_overflows
exc_rcvr_out_of_service	

For Trunk Groups:

signalling_glare*	signalling_problem*
no_usage*	signalling_prob*
100%_skip*	system_busy_trks*
exc_trks_oos*	sbutrunks_oos*
exctrunks_oos*	system_busy_usage*
mbutrunks_oos*	permanent_seizure*
facility_hit*	tropofade*
degraded_md164	tg_congested
degraded_md163	dpas_failure
tg_failure	degraded_facility
failure	span_failure
continuity_failure	trunk_testing
dtc_failure	transmission_fault

Whenever one of the above anomalies is found, it is asserted as a fact in CLIPS. When CLIPS is run, the anomaly is:

1. Confirmed if supporting switch report data indicates a problem;
2. Unconfirmed if switch report data does not indicate a problem; or

3. No opinion if logic to confirm was not available or time did not allow it to be implemented.

The CLIPS rules call a "C" function to set flags and status information for the report sent to the user. When CLIPS has finished executing, the results file is created. For each anomaly detected by the neural net, the expert system supplies a description of the problem and relevant switch or trunk group data. The relevant data includes both the current switch data and corresponding normal values for that time of day, obtained from the statistical data base.

The switch anomalies that are confirmed by the expert system are as follows. (Note that these outputs can be derived from more than one input from the above lists.)

CPU mismatch	Outage remote
CPU inits occurred	Miscellaneous system failure
Permanent Signals	Mf Receiver Overflow

Similarly, the trunk anomalies that are confirmed by the expert system are:

Permanent Seizure	Excess Trunks out of Service
Facility Hit/Tropofade	Maintenance Busy Trunks Out of Service
No Usage	System Busy Trunks Out of Service
Signalling Glare	Trunk Signalling Problem

2.3.2 Runtime Statistical Database

This section describes why the statistical data base is used in the verification of neural net anomalies. See Section 2.4 for more detail on the data in the statistical data base.

2.3.2.1 Need for a Statistical Data Base

Many of the tests for network anomaly detection proscribed by expert network managers contain conditions such as:

above-average holding time
high CCB usage
high traffic
low traffic

The difficulty with these conditions lies in determining what is normal for a specific switch or trunk group at a particular time of day. Because there are often large differences between one trunk group and another, and large fluctuations with time of day, one cannot establish a single "normal" value that fits all trunk groups or all switches for all time periods. For example, neither the current operational system (NMSS) nor the neural net can detect high traffic or high overflows accurately because they have no knowledge of time of day and do not have separate thresholds for each switch or trunk group.

As a solution to determining what is normal, and thus what is abnormal, for any switch or trunk group at any given time, the IWES uses a data base with averages and standard deviations for important switch and trunk group values that fluctuate. This data is indexed by time and by trunk group or switch.

Because the IWNN does not have a time parameter, it can misdiagnose high and low traffic situations. IWES, using the statistical data base, reports the expected values for the switch or trunk group when the NN has detected an anomaly. This information helps the operator determine if there is indeed a problem.

2.3.2.2 Runtime Option

Specifying the runtime flag, statdb, in the IWES (NMES) execution line enables the use of the stat data base. At IWES initialization time the stat values (average and standard deviation for each hour) for all switch and trunk group stats are loaded from the offline ASCII files into a hash table. The hash table provides the ES monitors and anomaly verifier with a fast and efficient random access to any statistic.

2.4 Statistical Data Base

This section describes the data in the statistical data base, including what is collected and how it is collected. The use of the statistical data base to verify anomalies is discussed in section 2.3. For a more detailed discussion of the statistical data base, see Appendix D. How the statistical data base is used by trunk group monitors is included in Appendix B.

2.4.1 Introduction

The statistical data base provides the IWES with a collection of normal values for important switch and trunk group parameters that fluctuate over time. For example, call control block (CCB) seizures at a switch are a good barometer on how busy a switch is at a given time. Large switches will use 200 to 700 CCBs in 5 minutes depending on the time of day. Small switches use only 0 to 30 CCBs during a 5 minute period. Without knowing which switch is being diagnosed and what time of day it is, it is very difficult to determine if a CCB count is normal or not.

2.4.2 Statistical Data Base Items

The statistical data base is designed to provide expected values and ranges for any important network parameter that may vary by switch or trunk group or over time. Currently 6 trunk group and 3 switch parameters are stored in the statistical data base. Each item is based on a formula of switch report fields and (normalizing) constants. The following table lists the switch report fields used in the formulas:

SWITCH REPORT FIELDS USED IN STATSDB

FORMULA ITEM	RECORD	FIELD	DESCRIPTION
MF_Use	RCVR	RCVTRU-MF	MF (interswitch) receiver usage
MF_Calls	RCVR	RCVSZRS-MF	MF receiver seizures
MF_Test_Calls	RADR	RADTESTC-MF	MF test calls
DG_Use	RCVR	RCVTRU-DGT	DG (intraswitch) receiver usage
DG_Calls	RCVR	RCVSZRS-DGT	DG receiver seizures
DG_Test_Calls	RADR	RADTESTC-DGT	DG test calls
Trunk_Use	TRK	TRU	Trunk group usage
Connects	TRK	CONNECT	Successful OUT attempts
Inservice	TRK	Trks In Service	Trunks available
Infail	TRK	INFAIL	INCOMING calls that failed
Intot	TRK	INCATOT	INCOMING calls
Ovfl	TRK	NOVFLATB	OUT attempts that overflowed
Attmpt	TRK	NATTMPT	OUT attempts
CCBs	CP	CCBSZ	Call-Control_Blocks seized

The following two tables list the statistic type, definitions, and the formulas used to calculate them.

SWITCH STATS

TYPE	DEFINITION	FORMULA
ccb	call control blocks/5 min.	CCBs*PER/12
rmf	Receiver MF holding time.	10*MF_Use/(MF_Calls - MF_Test_Calls)
rdg	Receiver DG holding time.	10*DG_Use/(DG_Calls - DG_Test_Calls)

TRUNK GROUP STATS

MNEMONIC	DEFINITION	FORMULA
aht	average holding time.	100*Trunk_Use/ (Connects+Intot-Infail)
cch	Out calls/circuit/hour.	Connects*PER/Inservice
icch	In calls/circuit/hour.	(Intot - Infail)*PER/Inservice
povfl	Percent overflows.	100*Ovfl/Attmpt
puse	Percent Trunk Group use.	100*PER*Trunk_Use/36*Inservice
tcch	Total calls/circuit/hour.	icch + cch

2.4.3 Computing Statistical Data Base Values

The statistical data base is designed to provide the IWES with an expected range of values given an item (switch or trunk group name), a statistic type, and a time of day. This is accomplished by computing the hourly average and standard deviation for each statistic based on approximately 10 days of switch report data.

Hourly averages were chosen as a reasonable, first-try tradeoff between precision and data base size. Experiments showed that averages based on less than 15 minute periods produce noisy results: the numbers constantly oscillate around averages based on a longer period. It is expected that future system testing and data analysis will determine if a shorter time period than 1 hour is needed. In any case, the statistical data base values should be interpolated when averages are requested for in-between times. For example, if 30 minute averages are stored in the statistical data base, and the ES requests a normal value for 12:50, then statistical data base interface routine should return

```
   value(12:30) + 2*(value(13:00) - value(12:30))/3.
```

Currently this is not done, even though traffic analysis shows that during work start, lunch, and work end hours, the traffic changes can be much greater than 1 standard deviation.

3.0 DCA-Eur Testing

3.1 Summary

The Integrated Workstation was successfully tested on live and archived data, and on fault conditions deliberately induced by switch technician actions, during the period 25-27 September 1990. On 28 September the IW was demonstrated to the Commander of DCA-Europe (Col. Reinmann) by two of his own staff. All parties declared that the IW features and demonstrated performance were valuable and successful. The Expert System was especially well received, as explained below. An IW terminal has been installed on the operations floor of the ACOC, and the staff have been ordered to familiarize themselves with its operation and to use it continually.

3.2 Background

In September 1989 simultaneous demonstrations were done at DCA-Europe with three separate developmental systems having complementary DSN network management functions, as reported in the FY39 Annual Report. These demonstrations clearly showed the great potential of the systems, and they also pointed the way for required effort in the ensuing months: it was obvious that the three components should be integrated into a single system. Efforts to that end began immediately, producing what came to be called the Integrated Network Management Workstation or IW. From the outset the FY90 objective was to test an initial operating capability (IOC) of the IW in late September 1990.

The three systems that had been demonstrated in September 1989 were: the DCA-Eur NMSS (Network Management Support System), a PC-based tool for gathering and displaying DSN switch data; the Lincoln NMES (Network Management Expert System); and a GTE-built LARS (Learning and Recognition System) Neural Net. The design philosophy for the FY90 Integrated Workstation was to offer the best features of all three through a single integrated user interface. The Neural Net function would do pattern recognition and symptom detection; the Expert System function would correlate and reason about the symptoms over time, and would provide operator action recommendations for each confirmed problem; and the NMSS function would provide direct access to the raw switch data for the sophisticated user addressing unusual problems.

3.3 The tests of late April 1990

From the time of the September 1989 demonstrations, there was discussion at DCEC of operational testing of the neural net and expert system installed at DCA-Europe at that time. A MITRE staff member on contract to DCEC wrote a plan for a set of on-site switch technician actions (such as applying the "restrict" and "directionalize" commands), which would supposedly replicate natural switch problem events. These tests were in fact carried out at the end of April 1990, and the NN and ES both performed badly. The reasons were:

1. There was no way to tell whether the NN and ES were capable of detecting the artificial scenarios, without advance performance of test scenarios followed by data analysis and possibly software modification, and no opportunity existed for such effort;

2. The MITRE person insisted that the NN and ES software be "frozen" before and during the demos;

3. The test scenarios produced switch report data that was markedly different from anything the NN and ES had previously been trained to detect.

Due to the marginal performance, plans were immediately initiated to do another set of tests in September 1990 and to have them be a success.

3.4 Planning for the September Tests

DCEC wrote a sizeable DSN IW Engineering Test Plan incorporating contributions from GTE and Lincoln as well as the table of test scenarios used in April. The main elements of the September tests were to be:

1. Verification that all the IW operator menus and functions work;

2. Verification that the relational database menus and options work;

3. Application of the IW to live network data;

4. Testing IW performance on archived data sets containing known network problems; and

5. Testing IW performance on problems manually induced in the actual network. This originally focused on the April MITRE test scenarios, which turned out to be prohibited by DCA-Eur authorities in September because of concern about disrupting Desert Shield communications. The ACOC staff proposed instead that switch technicians busy out certain trunk groups.

It was stipulated as part of the test plan that it would be acceptable for the IW to miss particular network problems the first time through, and that the bugs causing the misses could be found and fixed prior to retesting with similar data.

3.5 Test Results

Test categories 1 through 4 were addressed on Tuesday morning (9/25), and the results were generally good except that half a dozen bugs were found. Only one of these involved the expert system, which became sporadically "stuck" indicating that corrective action recommendations were continually available for a switch, even when no problem conditions were present. All of the bugs were found and fixed overnight, and Wednesday morning (9/26) they were retested and shown to be corrected.

Also on Wednesday morning two trunk group outages were simulated by site personnel, on the link from Donnersberg, Germany to Torrejon, Spain and on the link between Fairford and Mildenhall in the UK. The IW detected both. The test observers (including a senior NCO from the ACOC) were especially impressed with the Expert System displays, which provided concise problem descriptions, action recommendations, explanations, and relevant observations over several reporting periods. In fact, the Expert System logs often were found to have picked up pertinent observations about a fault before the Neural Net declared a detection.

On Wednesday afternoon a number of archived data sets were presented to the IW. The Neural Net detected anomalies in the data sets it had been trained on, and the Expert System made appropriate recommendations and observations. Several previously-untried data sets were presented which were known to have anomalies similar to those on which the Neural Net had been trained, and it failed to detect them. Attempts were made to retrain the Neural Net, with poor success.

On Thursday morning two more manually-induced trunk group outages were introduced, and were successfully detected by the IW. Having been trained overnight on some of the data sets it missed Wednesday, the Neural Net was able to detect anomalies in those sets Thursday. Attempts were again made to demonstrate retraining of the Neural Net, and results were still poor. The IW was operated on live network data for some time, and no major network problems were observed.

On Friday morning the system was demonstrated (on live and archived data) to Col. Reinmann and Col. Berger, the Commander and Deputy Commander of DCA-Europe. The IW was operated by Air Force Captain Paul Rovezzi, assisted by Master Sergeant Lugo, both of the ACOC staff. They were both very enthusiastic about the system, and about showing the two Colonels how well it worked. The latter were completely satisfied, and felt that the reverses of last April had been fully corrected.

3.6 Outbriefing

On Friday afternoon a briefing was presented to Col. Berger by Captain Rovezzi and MSgt Lugo. It was obvious that they had been favorably impressed by the performance of the IW. MSgt Lugo had made a series of screen dumps of the Expert System displays for his various trunk outage scenarios, and marked up the observations with Hi-Liter to point out how the system had detected all of them, and how well the annotations and explanations served his needs.

Col. Berger was told that a Sun workstation had just been installed in the ACOC proper to run the IW, and he said he would be experimenting with it over the weekend. He declared the testing exercises a success. He noted that he has looked at numerous commercial network management products since April, and the IW is the best he has seen by far, with respect to DCA needs. He was especially enthusiastic about the Expert System displays and advice.

4.0 CCSIM Development

A number of changes and improvements were carried out in FY90 with respect to CCSIM and its related programs. All programs were converted to run under the new SUN Operating System 4.0.3 early in the fiscal year. The graphics interface program was rewritten to use the X-window system instead of the previously used Suntools environment which is no longer being supported. The change makes our graphics compatible with the IW environment and improves portability to other machines, but it has exacted a price in speed of response and memory requirements.

The following sub-sections describe the major changes and improvements in CCSIM and experiments with CCSIM that were carried out in FY90. There are no current plans for further CCSIM development. A new document called "Using The Call-By-Call Simulator (CCSIM)" is being written, and the "CCSIM User's Manual" and the "CCSIM Software Top Level Design Document" previously written by 3S, Inc. are being brought up to date. By agreement with DCEC, delivery of these documents will take place in early FY91.

4.1 Common Channel Signaling

Implementation of the model for Common Channel Signaling (CCS) that was designed in FY89 was completed in the first part of FY90 and delivered to DCEC in June 1990. The model allows an experimenter to set arbitrary message sizes for call setup, takedown, preemption, etc. messages. Altogether there are twelve parameters specifying message sizes that are specified in the 'net.inval' file. From these, CCSIM calculates the CCS message traffic on each CCS link involved in a call processing event. From the average over a sampling period settable by the experimenter, it estimates the expected queuing delay on each link and uses this value in computing call setup time and the time that resources are held up when calls block. Statistics are generated showing the overall average signaling rate and the peak observed in a sampling interval settable by the experimenter.

Whether a trunk group uses CCS or In-band signaling is specified by a field in the 'net.clli' file. Two other files, 'net.baud' and 'net.ccsd', allow the experimenter to set the signaling rates and propagation delays that are to be used for each CCS link. In the absence of these files, a uniform rate for all links is assumed and set to a value specified in the 'net.inval' file.

All CCS signaling in CCSIM is assumed to be associated, i.e., CCSIM does not model an independent signaling network, and switches do not forward signaling traffic for other switches. There is no explicit model for damage to a signaling link, but the same effect can be obtained by damaging all the trunks between the associated switch pair.

4.2 New Trunk Damage Model

CCSIM now maintains the status of each trunk in a trunk group, noting whether it is in service or not in service and whether it is damaged or useable. When a trunk is damaged by command from the experimenter, any call on the trunk is now taken down. In earlier versions of CCSIM, such a call was left to hang up naturally, but the trunk could not subsequently carry a call. CCSIM now has a new command, 'SET-INSERVICE', to allow the experimenter to specify the number of in-service trunks in a group. Existing calls are not affected by changes to the in-service status of a trunk, but only trunks marked as in service can accept new calls. The in-service status of a trunk can be different at the two ends, but damage is the same at both. When a trunk group sustains partial damage or is to be considered as partially in service, the trunks to be affected are chosen randomly.

In routing a call, CCSIM searches a trunk group circularly, starting from the trunk just beyond the last one used or searched on the last attempt to use the group. It skips trunks that are marked as not in service. It stops the search on finding a free non-damaged trunk, on having searched the entire group, or on having attempted to use 'nr_line_tries' damaged trunks. 'nr_line_tries' is a 'net.inval' parameter with a default value of three. CCSIM models the time that would elapse waiting for signalling acknowledgements ("winks") that would not be received from the damaged trunks in a real network damage scenario.

4.3 New Switch Damage Model

CCSIM now takes down all calls through a switch at the time the switch is damaged. On restoral, CCSIM now generates a switch report at the next normal reporting time with values accumulated during the time between restoral and report time. Earlier versions of CCSIM left calls up until normal hang-up, and delayed the next report until a complete interval's data was available.

Trunk groups to a damaged switch can now be modeled as either 'winking' or 'non-winking' while the switch is damaged. The winking case corresponds to a situation in which receivers at the damaged switch continue to function even though the switch is unable to handle in-coming calls. The wink from the receiver causes the neighbor switch to send its signaling information and behave as though the call had been successfully routed, but since the damaged switch cannot handle the call, it will fail, i.e., the caller will give up. CCSIM now models this situation, and the experimenter can specify in the 'net.clli' file how an individual group should behave in a switch damage scenario. If marked as 'non-winking', behavior is the same as that for a damaged trunk (see Section 4.2, above).

4.4 Handling of Loops and Shuttles

The question of how CCSIM should handle loops and/or shuttles in a routing table was explored and revised. In a real network, if a call loops or shuttles between or among nodes, there is a possibility that another call will hang up during the process, freeing a line and allowing the looping call to succeed. Such a call will tie up an excessive number of trunks, and network managers consequently work hard to avoid the situation. CCSIM can not readily model this behavior because it operates on a call-by-call basis, and there is no other call that could hang up during the process. Therefore there is no possibility of a looping or shuttling call ever succeeding in CCSIM. In earlier versions, CCSIM would halt if it detected a looping or shuttling call. It has now been changed to merely print a warning message to the output file. The change helps experimenters debug new routing tables, and allows experiments involving changes in routing tables (reroutes) to get through transient conditions during which loops and/or shuttles may be unavoidable because routing tables in a set of switches cannot all be changed at the same instant.

4.5 Network Management Control Changes

During FY90 the Protective Reservation of Equipment (PRE) and the Destination Code Cancellation (DCC) controls were added to CCSIM. The formats used to invoke Cancel-To (CANT) and Cancel-From (CANF) were changed to correspond to those used by the DMS switches in DSN-Europe.

The complete CCSIM repertoire of network management controls can be found in Appendix E.

4.6 Switch Report Improvements

The switch reports generated by CCSIM have been extended to contain all of the OM reports being collected for the IW at DCA-Eur. These include five new switch status reports, and nine new fields in each trunk group report. Some of the new reports and fields can be derived from data accumulated during the simulations. Others, such as the 'glare' field in the trunk group report, are simulated by random numbers with experimenter-settable mean values. Still others, such as the number of CPU mismatches that were observed during the reporting period, are always set to zero.

Experiments with CCSIMN at DCA-Eur showed that reported seizures and holding times for Call Condense Blocks (CCBs), MF receivers, and Digitone receivers were not well matched to archived data from the real network. By adjusting the algorithm for calculating these values and tuning the local traffic intensities, we achieved a considerable improvement in the matches. Significant further improvement would require the introduction of switch-dependent parameters into the simulation.

Appendix F contains detailed information on the switch reports now available from CCSIM.

4.7 Busy-Destination Call Matrix

A new capability was added to CCSIM during FY90. In earlier versions CCSIM provided a parameter in the 'net.inval' file called 'pcbusy' that allowed an experimenter to specify that a percentage of all call attempts would be simulated as having reached a busy destination and would then retry with a probability of 'prbusy' after a random time with a mean value of 'trbusy'. Now CCSIM can accept a matrix of 'pcbusy' values contained in a file called 'net.busy' that allows an experimenter to specify the percentage of busy-destination calls independently for each source/distination pair. The intended use of this new capability is to facilitate the simulation of traffic problem scenarios such as focused overloads.

4.8 Utility Updates

All the utility programs originally associated with CCSIM have been reviewed and updated. Programs which use the link file now expect satellite trunks above the diagonal and terrestrial trunks below the diagonal. The program to generate routing tables algorithmically, 'genrt', has been updated and a minor bug in the original program has been corrected. A new variable 'longitude' has been provided to allow the user to specify the average number of statute miles to a degree of longitude in his network. The utilities which create files for CCSIM such as 'genrt', 'genfile' and 'link_to_clli' now produce files in the format expected by the current CCSIM.

4.9 Preplan Evaluation Experiments

A series of experiments was carried out to demonstrate the capability of CCSIM to aid in the evaluation of the effectiveness of the preplans being developed at DCA-Eur. The runs included two switch damage scenarios (UXB and FRD) and two switch congestion cases. The latter were realized by increasing the traffic destined for a particular switch (RTT) by factors of five and ten. The switches simulated by CCSIM are based on the NTI DMS switches in the DSN which do not display congestion symptoms because their call processing capabilities are more than adequate for the trunking used in the network, i.e., the switches can keep up with the maximum rate that call attempts can come in over the trunks. Consequently, we did not expect to see any symptoms of congestion go away in response to the preplan actions. However, the preplans have an effect on traffic in the network, and that effect can be observed.

The results showed small improvements in Grade-Of-Service (GOS) and Call-Failure-Rate (CFR) for the switch damage preplan applications. (CFR measures the fraction of calls that fail to succeed within the number of retry attempts allowed in the simulation.) For the congestion scenarios, GOS showed an improvement, but CFR worsened. The detailed results were delivered to DCEC for further evaluation.

4.10 CSSIM Integration with the IW

CCSIM now writes switch reports to files in a format compatible with the IW database. Such reports can be generated and held as archived data to be fed manually to the IW. Further work is needed in the IW and CCSIM to provide convenient interaction, control of simulation time, and proper separation between the real network and the simulation contexts to avoid confusing both the operator and IW software such as IWES. Also, a Fortran compiler for the SUN 4 machines at DCA-Eur is needed before CCSIM could be compiled there.

If it were desired to use CCSIM for training IW operators, further work on a trainer's interface would be needed to allow the training supervisor to introduce anomalies such as CPU mismatches that are not derived from the basic simulation and damage models.

5.0 TRAMCON/DPAS Alarm Integration

A component of DCEC FY90 tasking for Lincoln Laboratory is DRTV-funded expert systems development efforts for DCS transmission system control. This tasking is referred to as TRAMCON/DPAS alarm integration, and has had two main components in FY90: implementation of the TRAMCON Event Generator (TEG), and participation in the Tech Control Automation Proof-of-Concept System (TCAPS).

5.1 The TRAMCON Event Generator

The TRAMCON (TRAnsmission Monitoring and CONtrol) system polls a variety of communications and support equipment at a string of manned and unmanned microwave radio stations, gathering status and alarm information and displaying the results on a console at the TRAMCON Master site (which is typically located at a Tech Control Facility). It is commonplace for a microwave equipment failure to trigger both a primary alarm from the failed equipment and a constellation of sympathetic alarms from downstream sites that (for example) react to the loss of an incoming carrier signal. This situation can be very confusing to a human operator, because all the alarms appear to be equally valid and important, even though only one of them represents a genuine outage. Worse, there are many different variations on this theme, and they can greatly overload a human's ability to logically deduce fault causes, especially when the person is relatively junior.

Another complicating factor is partial overlap between TRAMCON alarms and those generated by DPAS (Digital Patch and Access System) facilities. Microwave sites typically handle multiple T1 carrier signals, many of which are connected to DPASs along with T1 carriers from other sources (such as land lines and satellite channels). The DPASs detect a standardized set of fault conditions and produce alarm signals that can be monitored either at the DPAS control console or at remote locations. Some of the DPAS faults are directly related to TRAMCON faults, and others are not. Clearly, access to both kinds of alarms would help a human operator in reasoning successfully to infer the locations of failure events; however, this additional dimension exacerbates the problems of the operator in comprehending all the causes, effects and variations.

The remedy that is being pursued is development of an expert system that is capable of performing the required deductive reasoning about patterns of TRAMCON and DPAS alarms. The FY89 Annual Report reproduced the result of initial efforts to that end, which was a Specification for a TRAMCON Event Generator (TEG). A major piece of work in FY90 was implementation of TEG, as described below. It should be noted that these initial efforts have focused upon the European version of TRAMCON;

generalization to the other versions is a significant but straightforward piece of work which has not yet been addressed under DCEC tasking to Lincoln Laboratory.

The motivation for creating TEG was to support knowledge engineering for the alarm interpretation expert system, given that we have essentially no prospect of access to a real TRAMCON system operating in a live DCS environment. All such real systems are in Europe, and the few examples in CONUS (e.g., at Fort Huachuca, AZ and at CCSC, Tinker AFB, OK) are very limited software maintenance systems running on canned data. Even if we could spend extended periods at European TRAMCON sites, we would clearly not be permitted to explore all the equipment failure modes on operational DCS circuits. In order to fill this need, we decided to build a software system that would produce the same alarm constellations as a real TRAMCON, for every possible equipment failure in the string of microwave stations monitored by the real TRAMCON.

In developing the TEG specification, we explored all available sources of knowledge about TRAMCON and the systems it monitors. Information was obtained from expert personnel at DCEC/DRTV, at the Fort Huachuca and Tinker AFB locations mentioned above, and at the 1945th Communications Group in Feldberg, Germany. In particular, the senior Tech Control personnel at the 1945th made available to us an extensive set of training aids they had developed for local use, aimed at the two major microwave system segments they monitor with TRAMCON. All of this knowledge was organized and integrated into the TEG specification reproduced in the FY89 Annual Report.

The design of TEG in FY90 began with the notion of exploiting the forward-chaining capability of an expert system shell. Forward chaining is the process of reasoning from a set of stated conditions to the logical consequences of those conditions. The TEG knowledge base is in effect a set of rules of the general pattern, "given a system connectivity data base, and given a failure of type x on element y in that data base, the resulting TRAMCON alarms will be ..." . The expert system shell CLIPS (C Language Integrated Production System), developed by NASA/Johnson Space Center and furnished without charge to Government agencies, was chosen for the project on the basis of prior successful projects with CLIPS at Lincoln Laboratory.

As implemented in FY90, TEG is a free-standing system that will run on any machine supporting CLIPS. In particular, this includes UNIX and DOS computers and PCs. While the development was done in the context of the DEB II microwave system in Europe, TEG can be loaded with any legal TRAMCON network configuration file. It interacts with the user via a succession of menus which prompt the user to select a site, an equipment item, and a failure type to inflict. TEG then produces a complete list of the primary and sympathetic alarms that a real TRAMCON would see under similar conditions in the real world. No attempt is made to reproduce all the menus, options and user interface details in this report; the interested reader can arrange with DRTV to see the actual system.

TEG has been installed at multiple locations besides Lincoln Laboratory, including DCEC/DRTV and AFCC/CCSC, Tinker AFB. In the latter case in particular, it was successfully brought up in a few minutes on the Air Force standard minicomputer (the AT&T 3B2/600G), and was then exposed at length to scrutiny by two CCSC staff personnel who had newly arrived from European assignments where they worked daily with real TRAMCON equipment. These personnel gave their approval to TEG, noting that it produced correct results and that it would be very useful for training purposes.

An interesting application of TEG was initiated in FY90, on an RADC-sponsored project to study the problem of distributing modules of expert system intelligence throughout the DCS to achieve efficient, distributed, survivable system control. The first stage in that project is to create a computer-based representation of a user-definable network including TCFs and their internal equipment, transmission systems, TRAMCONs, DPASs, and all the circuits carried by the network. TEG is in the process of integration with this network simulation, where it will be used to generate alarms in response to user-inflicted network faults. This work will continue in FY91.

The primary FY91 purpose for TEG will be to furnish an information source and development environment for the TRAMCON alarm interpretation expert system. In the process, DPAS alarms are to be studied and understood, and to be incorporated in TEG as well. The expert system is effectively an inversion of the rule base in TEG; however, it is far from simple. For example, there is a many-to-one mapping in certain fault instances: several different faults can lead to similar alarm patterns, and the expert system will have to use collateral information and complex inference strategies in attempting to resolve such cases.

5.2 TCAPS Field Demo Participation

DCEC/DRTV became involved in FY89 with a Modular Building Block (MBB) demonstration effort, initiated by DCA Headquarters, to develop and deploy a Tech Control Automation Proof-of-concept System (TCAPS) based on available technology. The centerpiece of this project was to be a modular cabling and control console system developed earlier by Sandia National Laboratory, and indeed it was very successful; the system is still in place where it was demonstrated at the Army TCF at Fort Detrick, Maryland, where it is in daily use for centralized polling, configuration and control of a large number of AN/FCC-100 multiplexers.

DCEC/DRTV was aware of a separate Lincoln Laboratory project sponsored by Rome Air Development Center, developing a Machine Intelligent TEch Control (MITEC) expert system to automate the operation of TCFs (Annual Reports, Knowledge-Based System Analysis and Control, FY89 and FY90). The idea was advanced in FY89 of having the DCA buy and install two MITECs at Fort Detrick, together with the testbeds of modern remotely-accessible TCF equipment required by MITEC in order to achieve hands-off operation, and to integrate a demonstration of the available MITEC technology with TCAPS. This integration was to take the form of having the MBB operator alert a MITEC expert system of the occurrence of a fault in the MITEC testbeds, whereupon the MITEC would automatically isolate the fault, restore service by electronically patching spares into the circuit, and send a full report of the activity back to the MBB console.

As the planning progressed, interest developed at the Air Force 7th Comm''nications Group (a TCF in the Pentagon) in participating in the demonstrations. A decision was made to locate one of the MITECs at the 7th CG, and to provide a dedicated T1 circuit and multiple telephone circuits between the MITECS at 7th CG and Ft. Detrick. Numerous demonstrations of the systems were done at both Ft. Detrick and the Pentagon, for a variety of visitors ranging from working Tech Controllers to flag officers. More complete details are given in the FY90 Annual Report on Knowledge-Based System Analysis and Control, which was written for RADC. In fact the MITEC demos in association with TCAPS were pivotal in triggering a decision by the Air Force Communications Command to fund the technology transfer of MITEC into the working Air Force communications systems inventory. An excellent degree of synergism has been realized between the MITEC and TCAPS work in several areas, such as database design, report generation, and FCC-100 control techniques. There is also the prospect of future coupling between MITEC and the DCEC-sponsored TRAMCON expert system.

APPENDIX A. IWES System Design

The Integrated Workstation Expert System (IWES) is written in 'C' and in an expert system shell developed by the Artificial Intelligence Section of the Mission Planning and Analysis Division at NASA/Johnson Space Center. The shell is called CLIPS ('C' Language Integrated Production System). CLIPS, CLIPS updates and CLIPS documentation are available at no cost for U.S. government work. To register as a CLIPS user and receive update information call the CLIPS Users Help Desk at (713)280-2233 during the hours 8:00 a.m. to 4:45 p.m., central time, Monday through Friday. Any mail correspondences should include your name, current address and phone number to keep records up-to-date, and should be addressed to:

CLIPS Users Help Desk/M30
Computer Sciences Corporation
16511 Space Center Boulevard
Houton, Texas 77058

The IWES is the result of integrating an existing stand-alone expert system, the Network Management Expert system (NMES), into the Integrated Workstation (IW). Some NMES user options and internal code relating to the original system are obsolete with relationship to the IW, but still exist in IWES. Code still exists to communicate with the NMES graphics, a call by call simulator and a switch report translator. None of the standalone options or code are visible to the IW operator, but they will be invisible to anyone involved with programming the IWES. In the future, we would like to remove all of the old NMES code that is not in use.

A.1 Program Flow

Because most of the program logic of IWES is embodied in CLIPS rules, which are highly unprocedural, we have decided to describe IWES program flow in pseudo code. Below is a description of the major procedural routines that constitute IWES. A description of the CLIPS rules and 'C' monitors is found in Appendix B.

In the following pseudo code, we describe the programming structures and the important functions executed in each major routine. The order within the pseudo code routines is identical to the order within the real IWES code. Functions are decipherable from plain text descriptions by the () following them. When a function is called it is also followed by the section of this document it can be found in and by the name of the actual 'C' routine within which it is located.

A.1.1 IWES main routine

```
main()                                       (nmes.c)

  nmes_init() (section A.1.2)                (nmes_init.c)
  nmes_reset() (section A.1.2)               (nmes_reset.c)
  iwmdinit() (section A.1.2)                 (iwmd.c)

  loop while running
    process_pipe_input() (section A.1.3)     (process_pipe_input.c)
```

A.1.2 Initialization routines

```
nmes_init()                                  (nmes_init.c)
  init_clips()                               (Clips routine)
  load_clips_rules() (section A.1.2)         (load_clips_rules.c)
  read_node_file() (section A.1.2)           (read_node.c)
  read_clli_file() (section A.1.2)           (read_clli.c)
  open initial iwes log file
  read_ctrl_info() (section A.1.2)           (read_ctrl.c)

nmes_reset()                                 (nmes_reset.c)
  reset_clips()                              (Clips routine)
  assert_nodes() (section A.1.2)             (read_node.c)
  assert_cllis() (section A.1.2)             (read_node.c)
  load() (section A.1.2)                     (stats.c)
  initialize IWES times
  reset_switch_link() (section A.1.2)        (nmes_reset.c)

iwmdinit()                                   (iwmd.c)
  set up connection with iwmd

load_clips_rules()                           (load_clips_rules.c)
  load rules into clip

read_node_file()                             (read_node.c)
  IWGetSwitchTable()                         (IWDB routine)
  build IWES linked list switch structure

read_clli_file()                             (read_clli.c)
  IWGetNetworkTable()                        (IWDB routine)
  build IWES link and clli structures

read_ctrl_info()                             (read_ctrl.c)
  read_sitetype_file() (section A.1.2)       (read_ctrl.c)
  read_syntax_file() (section A.1.2)         (read_ctrl.c)

assert_nodes()                               (read_node.c)
  assert node information into clips

assert_cllis()                               (read_node.c)
  assert clli information into clips
```

```
load()                                              (stats.c)
  load statistical database containing normal
    values into hash table and insert facts
    into clips

reset_switch_link()                                 (nmes_reset.c)
  initialize status arrays

read_sitetype_file()                                (read_ctrl.c)
  read in the control sitetype file

read_syntax_file()                                  (read_ctrl.c)
  read in the control syntax file
```

A.1.3 Communication with the IWMD

```
process_pipe_input()                                (process_pipe_input.c)
  checksocket() (section A.1.3)                     (checksocket.c)
  if message available from message dispatcher
    processiwmd() (section A.1.3)                   (iwmd.c)

checksocket()
  check if any messages are available               (checksocket.c)

processiwmd()                                       (iwmd.c)
 do while (remaining) /* while messages still remain */
   read_iwmd() (section A.1.3)                      (iwmd.c)
   parse the message
   if bad message, return
   if polldone
process_switch_report_iwdb()(section A.1.4)
(process_switch_report.c)
      send procdone message to the iwui
      if not receiving neural network output
        send recsdone message to the iwui
    else if diagdone
      if receiving neural network output
        process_neural_net() (section A.1.6)
(process_neuralnet.c)
        send recsdone message to the iwui
    else if polledst
      print message - no other actions taken yet
    else if ctrlsent
      print message - no other actions taken yet
    else if ctrlresp
      print message - no other actions taken yet
    else if decodedone
      print message - no other actions taken yet
    else
      print don't know how to respond to message
```

```
read_iwmd()                                        (iwmd.c)
  returns a single message from the message
  dispatcher. Keeps track of any other messages
  that have been read, but not processed, so it
  can return them next time it is called
```

A.1.4 Processing switch reports

```
process_switch_report_iwdb()             (process_switch_report.c)
  IWGetTrafficTable()                    (IWDB routine)
  get time from traffic table
  if new date
    open a new log file
  for each switch a report was received from
    store data in IWES network representation
      structures
    assert fact in to clips that a switch report has
      been received
    run_switch_monitors() (section A.1.5)
                                         (switch_monitors.c)
    get_iwdb_clli() (section A.1.4)      (process_switch_report.c)
  runes() (section A.1.7)                (runes.c)

get_iwdb_clli()                          (process_switch_report.c)
  for each trunk group a report was
    received from, store data in IWES
      network representation structures
    clli_monitors() (section A.1.4)      (process_switch_report.c)
  find_cllis_without_reports() (Section A.1.5)
                                         (report_monitors.c)
```

A.1.5 Running Monitors

```
run_switch_monitors()                    (switch_monitors.c)
  if switch is reporting
    mark_switch_report_received()        (switch_monitors.c)
    find_no_mf_receiver_free()           (switch_monitors.c)
    find_no_dialing_receiver_free()      (switch_monitors.c)
    find_cp_overflows()                  (switch_monitors.c)
    find_mf_radr_overflows()             (switch_monitors.c)
    find_dial_radr_overflows()           (switch_monitors.c)
    find_cpu_overflows()                 (switch_monitors.c)
    find_trmtcm_overflows()              (switch_monitors.c)
    find_trmter_overflows()              (switch_monitors.c)
    find_trmtrs_overflows()              (switch_monitors.c)
    find_dcm_overflows()                 (switch_monitors.c)
```

Switch monitors detect anomalies in the switch report fields. For descriptions of the above 11 switch monitors see section A.7.1.1.

(Switch Monitors)

```
clli_monitors()                         (process_switch_report.c)
  check_fail_clli_report()              (report_monitors.c)
  find_stats()                          (stat_processor.c)
  if using the statistical database
    calculate_statistics()              (stat_processor.c)
  find_clli_trunks_down()               (report_monitors.c)
  find_hundred_overflow()               (report_monitors.c)
  find_zero_overflow()                  (report_monitors.c)
  find_low_holding_time()               (report_monitors.c)
  find_low_ht_100_overflow()            (report_monitors.c)
  find_low_ht_calls()                   (report_monitors.c)
  if receiving neural network output
    find_max_usage_few_calls()          (report_monitors.c)
    find_zero_usage_calls()             (report_monitors.c)
```

Clli monitors detect anomalies in the clli report fields. For descriptions of the above clli monitors see section A.7.1.2 (Trunk Group Monitors)

```
run_interval_monitors()                 (interval_monitors.c)
  for each switch
   find_nodes_not_responding()          (interval_monitors.c)
    find_no_outgoing_attempts()         (interval_monitors.c)
```

Interval monitors detect anomalies by analyzing fields in neighboring switches report fields. For descriptions of the above interval monitors see Section A.7.1.1.2 (Other Switch Monitors)

A.1.6 Processing neural network output

```
process_neural_net()                    (process_neuralnet.c)
  IWGetIwnnReadDataStream()             (IWDB routine)
  read in neural network results
  if neural network switch anomaly
    find_nn_switch_error() (section A.1.6)
                                        (process_neuralnet.c)
  else (neural network trunk group anomaly)
    nn_clli_error() (section A.1.6)     (process_neuralnet.c)
  remove old text from switch state array
  remove old text from link state array
  runes() (section A.1.7)               (runes.c)

find_nn_switch_error()                  (process_neuralnet.c)
  fill in neural network information structure
  find switch structure
  create text message
  check_nn_switch_validity() (section A.1.6)
                                        (process_neuralnet.c)
```

```
check_nn_switch_validity()                (process_neuralnet.c)
  match with known neural network switch anomaly
    add text message about anomaly to switch
    state array
    assert anomaly into clips facts

nn_clli_error                             (process_neuralnet.c)
  fill in neural network information structure.
  find clli structure
  create text message
  check_nn_clli_validity() (section A.1.6)
                                          (process_neuralnet.c)

check_nn_clli_validity()                  (process_neuralnet.c)
  match with known neural network clli anomaly
    add text messge about anomaly to link
    state array
    assert anomaly into clips facts
```

A.1.7 Running the expert system

```
runes()                                   (runes.c)
 if time_to_run_clips() (section A.1.7)(runes.c)
    if not the first time period
      run_interval_monitors() (section A.1.5)
                                          (interval_monitors.c)
      assert the current time into clips
      run(-1) /* run clips */             (clips run routine)
    create_results_file() (section A.1.8)
                                          (results.c)
    reset_report_received() (section A.1.7)
                                          (runes.c)

time_to_run_clips()                       (runes.c)
  if receiving neural network output and neural
      network output has not been received
    return(0)
  else
    return(1)

reset_report_received()                   (runes.c)
  mark reports as not received
```

A.1.8 Creating the results file

```
create_results_file()                     (results.c)
  IWGetIwesWriteDataStream()              (IWDB routine)
  count the number of switches with information
    to be put in the results file
  write header line to file
  for each switch
```

```
    If switch information is available
      add_sw_info() (section A.1.8)    (results.c)

add_sw_info()                          (results.c)
  write switch name, alarm, and number of anomalies
      to file
  add_sw_conclusions() (if any) (section A.1.8)
                                       (results.c)
  add_sw_recommend() (if any) (section A.1.9.1)
                                       (results.c)
  count the number of trunk groups with information
      to be put in the results file
  if there are trunk groups with information
    add_tg_info() (section A.1.8)      (results.c)

add_sw_conclusions()                   (results.c)
  for each problem at the switch
    get_sw_descriptions() (section A.1.8)
                                       (rec_description.c)
  write description to file
  write explanation from switch state array to file

get_sw_description()                   (rec_description.c)
  get normal values from statistical database
    for use in description
  switch on problem type
    add description for problem to conclusion
       structure

add_tg_info()                          (results.c)
  for each clli with information
    write clli name,trunk group number, src, dest,
        alarm, and number of anomalies to file
    add_ln_conclusions() (if any) (section A.1.8)
                                       (results.c)
    add_tg_recommend() (if any) (section A.1.9.2)
                                       (results.c)

add_ln_conclusions()                   (results.c)
  for each problem on the clli
    get_ln_descriptions() (section A.1.8)
                                       (rec_description.c)
  write description to file
  write explanation from link state array to file

get_ln_description()                   (rec_description.c)
  get normal values from statistical database
    for use in description
  switch on problem type
    add description for problem to conclusion
       structure
```

A.1.9 Determining Recommendations

A.1.9.1 Recommendations for Switch Problems

```
add_sw_recommend()                              (results.c)
  get_sw_recommend() (section A.1.9.1)
                                                (recommend.c)
  write alarm, problems, actions and controls
      to the file

get_sw_recommend()                              (recommend.c)
  for each problem at the switch
    get_sw_problem() (section A.1.9.1)  (rec_prob.c)
    if using preplans
      get_sw_ppln() (section A.1.9.1)   (pre_plan.c)
      if no preplans were found
        get_sw_actions() (section A.1.9.1)
                                                (rec_action.c)
        get_sw_controls() (section A.1.9.1)
                                                (rec_ctrl.c)
    else
      get_sw_actions() (section A.1.9.1)
                                                (rec_action.c)
      get_sw_controls() (section A.1.9.1)
                                                (rec_ctrl.c)

get_sw_problem()                                (rec_prob.c)
  switch on problem type
    add problem text for problem to
       recommendation structure

get_sw_ppln()                                   (pre_plan.c)
  get text for problem type
  get_sw_ppln_from_file() (section A.1.9.1)
                                                (pre_plan.c)

get_sw_ppln_from_file()                         (pre_plan.c)
  read pre_plan from file
  if no actions are in the file
    get_sw_actions() (section A.1.9.1) (rec_action.c)
  if no controls are in the file
    get_sw_controls() (section A.1.9.1)
                                                (rec_ctrl.c)

get_sw_actions()                                (rec_action.c)
  switch on problem type
    add actions for problem to recommendation
       structure
```

```
get_sw_controls()                               (rec_ctrl.c)
  switch on problem type
    add controls for problem to recommendation
       structure
```

A.1.9.2 Recommendations for clli problems

```
add_tg_recommend()                              (results.c)
  get_ln_recommend() (section A.1.9.2)
                                                (recommend.c)
  write alarm, problems, actions and controls
      to the file

get_ln_recommend()                              (recommend.c)
  for each problem on the clli
    get_ln_problem() (section A.1.9.2) (rec_prob.c)
    if using preplans
      get_ln_ppln() (section A.1.9.2)   (pre_plan.c)
      if no preplans were found
        get_ln_actions() (section A.1.9.2)
                                                (rec_action.c)
        get_ln_controls() (section A.1.9.2)
                                                (rec_ctrl.c)
    else
      get_ln_actions() (section A.1.9.2)
                                                (rec_action.c)
      get_ln_controls() (section A.1.9.2)
                                                (rec_ctrl.c)

get_ln_problem()                                (rec_prob.c)
  switch on problem type
    add problem text for problem to
       recommendation structure

get_ln_ppln()                                   (pre_plan.c)
  get text for problem type
  get_ln_ppln_from_file() (section A.1.9.2)
                                                (pre_plan.c)

get_ln_ppln_from_file()                         (pre_plan.c)
  read pre_plan from file
  if no actions are in the file
    get_ln_actions() (section A.1.9.2) (rec_action.c)
  if no controls are in the file
    get_ln_controls() (section A.1.9.2)
                                                (rec_ctrl.c)

get_ln_actions()                                (rec_action.c)
  switch on problem type
    add actions for problem to recommendation
       structure
```

```
get_ln_controls()                           (rec_ctrl.c)
  switch on problem type
    add controls for problem to recommendation
       structure
```

A.1.10 Network Status Routines Called from Clips

```
nn_switch_status()                          (nn_status.c)
  compare the status(problem) sent as a parameter
    from clips with a list of expected status
    parameters to find the correct status
    store or remove text in the switch state array
      describing the problem found.
    add the problem to or remove the problem from
      the current problems array

nn_clli_status()                            (nn_status.c)
  compare the status(problem) sent as a parameter
    from clips with a list of expected status
    parameters to find the correct status
    store or remove text in the link state array
      describing the problem found.
    add the problem to or remove the problem from
      the current problems array
```

A.2.1 Inputs to IWES

A.2.1.1. Switch Reports

Switch reports are read into the IWES, processed for anomalies and used in validating neural network results. A call to IWDB routine IWGetTraffic(timestamp) is made in IWES routine process_switch_report_iwdb(), which is located in process_switch_report.c. This call returns a pointer to a structure that contains traffic information for each switch that responded to a poll for the time period specified.

Format: Traffic information contains OMs for switches and related trunk groups. For the exact format of the structure see section 5.4 of the Integrated Workstation Programmer's Guide.

A.2.1.2. Neural Network

Neural network results are read into IWES every time interval. They are validated and used to determine network problems and make recommendations for network management actions to be taken to correct the problems. A call to IWDB routine IWGetIwnnReadDataStream(timestamp) is made in IWES routine process_neural_net(), which is located in process_neural_net.c.

This call returns a file pointer to the neural network anomaly file. IWES uses this file pointer to read in anomalies line by line.

Format: See neural network output file (section 5.7 of the Integrated Workstation Programmer's Guide)

A.2.1.3. Switch Representation

Switch representation information is read into IWES upon initiation. IWES uses this information to set up a linked list of switch structures, which are used through out all of the code. A call is made to IWDB routine IWGetSwitchTable() in IWES routine

read_node_file(), which is located in read_node.c. This call returns a pointer to a structure that contains information about the switch.

Format: See IWDB section of the Integrated Workstation Programmer's Guide(5.4).

A.2.1.4. Trunk Group Representation

Trunk representation information is read into IWES upon program initiation. Trunk information is added to the related IWES switch structures to be used throughout program execution. A call is made to IWDB routine IWGetNetworkTable() in IWES routine read_clli_file(), which is located in read_clli.c. This call returns a pointer to a structure that contains information about the trunk groups and their connectivity.

Format: See IWDB section of the Integrated Workstation Programmer's Guide(5.4).

A.2.1.5. Preplans

Preplans allow a network management expert to tailor recommended actions and controls for specific switches and trunk groups. They are accessed directly from files in the IWES routine get_sw_ppln_from_file(), which is located in pre_plans.c. The preplan files are located in the $(IW_DATA)/iwes_config/ppln directory and have a filename format of:

```
preplan.X        where X is a three character site name
                       e.g. preplan.UXB
    or
preplan.X.Y      where X is a three character site name
                       and Y is a short clli name
                       e.g  preplan.UXB.SVN095
```

Format:

<u>Header Line</u>

```
<sw_name or clli_name> Where: sw_name = name of the switch
                              clli_name = name of the clli
```

<u>Problem recommendations</u> (repeat for all types of problems that have preplanned recommendations)

```
<problem>  Where: problem is a one of a predetermined set
           of text strings associated with IWES problems.
           These text strings are hard coded in IWES.
           Presently hard coded IWES problems include:
                   Switch_Outage
                   Switch_Congestion (although no neural
                        network output is available for
                        this problem)
                   Permanent_Seizure
                   Signalling_Glare
                   Signalling_Problem

  Actions
    <actions[1]>
       .              Where: actions[i] = a line of text
       .                     (80 chars max) ending with a
       .                     newline character. n must be
       .                     less than 20.
    <actions[n]>
  Controls
    <control[1]>      Where: control[i] = a line of text
                      for the user interface to display
                      (80 chars max) ending with a newline
                      character.  n must be less than 20.
    <controlparameters[1]>    controlparameters[i] = a line

   of  .              formatted text (80 chars max) ending
       .              with a newline character. n must be
       .              less than 75. This text will be sent
       .              on to the control module.
Presently,  .         formatting of the control parameters
       .              is not complete.
    <control[n]>
    <controlparameters[n]>
  End
```

A.2.1.6. Control Format Information

Control information is accessed by the IWES in order to verify and format recommended controls accurately. It is accessed directly from files in IWES routines read_sitetype_file() and read_syntax_file(), which are located in read_ctrl.c. The control format information files are located in the ${IW_DATA}/iwes_config directory and have filenames and contents as follows:

ctrl_types.sites	- contains a list of the reporting sites followed by their site type
ctrl_syntax.nr	- contains a list of site types, actions and controls followed by an associated syntax number.
ctrl_syntax_params	- contains a list of syntax numbers followed by a set of parameter numbers.
ctrl_syntax_labels	- contains a list of parameter numbers followed by a text label and a parameter definition.

IWES must go through 4 steps to get the complete set of control parameters. First it determines the site type, then the syntax number, then parameter numbers and finally the actual parmaters.

Presently, only ctrl_types.sites and ctrl_syntax.nr are read into IWES.

Formats:

ctrl_type.sites

<sw_name> <site_type> Where: sw_name = name of the switch site_type = an integer representing the type of the site.

(This line is repeated for each reporting switch)

ctrl_syntax.nr

<site_type> <action> <control> <syntax_number>

Where: site_type = an integer representing the type of the site.
action = action to take. One of: APPLY, LIST or REMOVE.
control = One of a group of previously specified controls.
syntax_number = A syntax number associated with the site_type, action and control.

(This line is repeated for each different set of site_type, action and control)

ctrl_syntax_params

```
<syntax_number> <parameter_numbers>

  Where: syntax_number = A syntax number associated with
            the site_type, action and control.
         parameter_numbers = a list of parameters numbers
            in the order expected by the control module.
```

(This line is repeated for every syntax number)

ctrl_syntax_labels

```
<parameter_number> <label> <parameter definition>

  Where: parameter_number = a parameter number from the
            parameter numbers listed in the control
            syntax parameters file.
         label = a predetermined label for the parameter
            number that IWES is aware of.
         parameter definition = limits of the parameter.
```

(This line is repeated for every parameter number)

A.2.1.7 Statistical Data Base Input Files

Statistical data base input files are found in the $DSNPATH/expert/stats/db directory. Specifying the runtime flag, statdb, in the IWES (NMES) execution line enables the use of the statistical data base. At IWES initialization time the stat values (average and standard deviation for each hour) for all switch and trunk group stats are loaded from the offline ASCII files into a hash table.

In addition to the actual stat files, there are two files in the stats/db directory that provide parameters for statistical data base monitor thresholds and information on trunk groups that run normally with INSERVICE set less than EQUIPPED.

A.2.1.7.1 Stat Files

Each file is an ASCII text file containing hourly stat information for a specific stat (type) for one switch or trunk group (item). Since there are up to 24 hours of switch report data available, stat files will normally contain 24 records representing hours 0 through 23. Each stat file record has the following format:

Hour Average #_Of_Samples Minimum Maximum Standard_Deviation.

For each of the 14 reporting switches in the European network, there are 3 stat files: CCB usage per 5 minutes, MF receiver holding time, and Digitone receiver holding time.

There are currently 102 trunk groups connecting the 14 reporting switches with each other and other non-reporting switches. For each of these trunk groups the statistical data base has 6 files: percent overflows, average holding time, incoming calls per circuit per hour, outgoing calls per circuit per hour, total calls per circuit per hour, and percent use of total trunk group capacity. In total, there are 654 (3 * 14 + 6 * 102) stat files in the data base.

The stat file name format describes what data was used to compile the stat values, and what stat type is being computed for what item:

```
stat-<date>+<additional_days>-<item>.<type>.
```

For example, the file "stat-900103+9-ezl.ccb" contains CCB seizure information for the switch EZL. The data was computed from 10 days of archived data beginning 900103. The file "stat-900806+8-tjs052.povfl" contains Percent Overflow data for the trunk group 52 out of TJS. The data was computed from 9 days of archived data beginning 900806.

The format of the statistical data base files allows them to be directly processed by a plotting program, XGRAPH, when displaying the averages over time. For example:

```
xgraph stat*.ccb
```

will display the CCB usage curves over 24 hours for the 14 switches.

A.2.1.7.2 Threshold Factor File

The stat-SD_FACTOR.fact file contains factors for resetting the thresholds the IWES uses to detect deviations from normal network activity. The default thresholds for trunk group stats is 1 standard deviation above or below the average value. The .fact file allows the IW operator to configure offline a different standard deviation factor for each of the 6 trunk group stat types. At IWES initialization time, if a .fact file is in the stat/db directory, the trunk group stat thresholds in the array stat_sd_factors[] will be reset. Each line in the .fact file has one integer and corresponds to one trunk group stat type. The order is implicit: percent overflows (povfl), incoming calls/circuit/hour (icch), calls/circuit/hour (cch), total calls/circuit/hour (tcch), average holding time (aht), and percent trunk group use (puse).

The current .fact file contains 6 records: 2,2,2,3,2,2, which indicates that povfl,icch, cch, aht, and puse thresholds are set at 2 times their standard deviation from average, and that tcch is set at 3 times its standard deviation. Note, that because the statistical data base contains a computed standard deviation and average for every trunk group for every hour of the day, each threshold check is specific for the trunk group and time of day.

A.2.1.7.3 Inservice File

The stat-<dates>-TGS.insrv contains trunk groups that normally run with a low inservice count. The IWES uses this information at runtime to inhibit messages warning of low inservice trunk groups to the operator. Currently there is no .insrv file because all trunk groups were running normally at their equipped level when the system was installed in September 1990.

The format of each line is:

```
<local_switch> <tg_#> <CLLI> <normal_low_inservice_#>
<destination_switch>
```

An example is:

```
ezl 90 ABE001 0 abe
```

A.2.2 Outputs from IWES

A.2.2.1. Results File

The results file is written by the IWES and read by the IWUI. A call to IWDB routine IWGetIwesWriteDataStream(timestamp) is made in IWES routine create_results_file(), which is located in results.c. This call returns a file pointer to the iwes results file. IWES uses this pointer to write into the results file line by line. The result files are located in the ${IW_DATA}/YYMMDD/iwes directory and have a file name format of:

```
YYMMDD.HHmm.iwes    where: YY = year
                           MM = month
                           DD = day
                           HH = hour
                           mm = minute
                         e.g. 900927.1540.iwes
```

Format:

<u>Header Line</u>

```
Time: <Timestamp> Switches: <num_of_switches>

    Where: timestamp is in the format YYMMDD.HHMM (described
                in section ?)
           num_of_switches = the number of switches for which
                information about the switch or any of the
                trunk groups leaving the switch is contained
                in the file.
```

<u>Switch Information</u>

```
<sw_name>

    Where: sw_name = name of the switch

  Anomaly_alarm: <alarm> Anomalies: <num_of_anomalies>
     <Anomalies[1]>
         .
         .
         .
     <Anomalies[num_of_lines]>

    Where: alarm = level of the anomaly alarm. The alarm
    could be one of 6 values:
                 NODATA       0
                 NORMAL       1
                 LEVEL2       2
                 LEVEL3       3
                 LEVEL4       4
                 DOWN         5

           As long as this alarm is greater than 1 conclusion
                and recommendation information will be
                available.
           num_of_anomalies = number of anomalies at the
                switch.
           Anomalies[i] = a line of text (80 chars max)
                ending with a newline character.

  Description: <num_of_lines>    Where: num_of_lines = the
      <Description[1]>                  number of
         .                              description lines
         .                              that follow  .
         .                        Description[i] = a line
         .                              of text (80 chars
         .                              max) ending with a
         .                              newline character.
     <Description[num_of_lines]>
```

```
      Explanation: <num_of_lines>    Where: num_of_lines = the
          <Explanation[1]>                   number of
 .                                 explanation lines
 .                                 that follow
              .                      Explanation[i] = a line of
              .                              text (80 chars max)
              .                              ending with a
              .                              newline character.
         <Explanation[num_of_lines]>
      Recommendation Alarm: <recommend_alarm_level>
        Where: recommend_alarm_level = level of the
          recommendation alarm.
               The alarm can equal either zero or one.  If it
               equals zero then no problems, actions or Controls
               will follow (see example). If it equals one
               problems, actions and/or controls will follow.

      Problem: <num_of_lines>      Where: num_of_lines = the number
          <Problem[1]>                    of problem lines that
              .                           follow.
              .                    Problem[i] = a line of text
              .                    (80 chars max) ending with
              .                    a newline character.
         <Problem[num_of_lines]>
      Actions: <num_of_lines>      Where: num_of_lines = the number
          <Actions[1]>                    of action lines that
              .                           follow.
              .                    Actions[i] = a line of text
              .                    (80 chars max) ending with
              .                    a newline character.
         <Actions[num_of_lines]>
      Controls: <num_of_controls>      Where: num_of_controls =
          <control[1]>                        the number of
                                              controls that
                                              follow.
                            control[i] = a line of text for the
                                         user interface to display
                                         (80 chars max) ending
                                         with a newline character.
                                         n must be less than 20.
<controlparameters[1]> controlparameters[i] = a line of formatted
                                         text (80 chars max)
                                         ending with a new line
                                         character. n must be less
                                         than 75.  This text will
                                         be sent on to the control
                                         module.  Presently,
                                         formatting of the control
                                         parameters is not
                                         complete.
```

```
<control[number_of_controls]>
<controlparameters[number_of_controls]>
```

Trunk_groups: <num_of_trunk_groups>

Where: num_of_trunk_groups = the number of trunk groups leaving this switch for which information is contained in the file. The following trunk group information will be repeated for each trunk group.

<u>Trunk group information</u> (repeat for all trunk groups with anomalies)

<Clli_name> Num: <trunk_group_num> Src: <src> Dest: <dest>

Where: clli_name = name of the clli.
trunk_group_num = the trunk group number.
src = the source switch.
dest = the destination switch.

```
Anomaly_alarm: <alarm> Anomalies: <num_of_anomalies>
   <Anomalies[1]>
      .
      .
      .
   <Anomalies[num_of_lines]>
```

Where: alarm = level of the anomaly alarm. The alarm could be one of 6 values:

NODATA	0
NORMAL	1
LEVEL2	2
LEVEL3	3
LEVEL4	4
DOWN	5

As long as this alarm is greater than 1 conclusion and recommendation information will be available.
num_of_anomalies = number of anomalies for the trunk group.

Anomalies[i] = a line of text (80 chars max) ending with a newline character.

```
 Description: <num_of_lines>    Where: num_of_lines = the
     <Description[1]>                  number of
         .                             description lines
         .                             that follow
         .                            Description[i] = a
         .                             line of text (80
         .                             chars max) ending
         .                             with a newline
         .                             character.
<Description[num_of_lines]>

Explanation: <num_of_lines>    Where: num_of_lines = the
     <Explanation[1]>                 number of explanation
         .                            lines that follow
         .                            Explanation[i] = a line
         .                            of text (80 chars max)
         .                            ending with a newline
         .                            character.
    <Explanation[num_of_lines]>

Recommendation Alarm: <recommend_alarm_level>
   Where: recommend_alarm_level = level of the
            recommendation Alarm
            The alarm can equal either zero or one.  If it
            equals zero then no problems, actions or
            Controls will follow (see example).  If it
            equals one then problems, actions and/or
            controls will follow.  At this point, this alarm
            is not used in the user interface.

 Problem: <num_of_lines>      Where: num_of_lines = the
                              number of
 <Problem[1]>     problem lines that follow.
       .          Problem[i] = a line of text
       .          (80 chars max) ending with
       .          a newline character.
    <Problem[num_of_lines]>

 Actions: <num_of_lines>      Where: num_of_lines = the
                              number of
    <Actions[1]>              action lines that follow.
       .          Actions[i] = a line of text
       .          (80 chars max) ending with
       .          a newline character.
    <Actions[num_of_lines]>
```

```
Controls: <num_of_controls>  Where: num_of_controls = the
                             number of controls that
                             follow.
   <control[1]>  control[i] = a line of text for the user
                             interface to display
                 (80 chars max) ending with a newline
                 character.  n must be less than 20.

   <controlparameters[1]>
      .     controlparameters[i] = a line of formatted text
      .     (80 chars max) ending with a newline
      .     character. n must be less
      .     than 75. This text will be
      .     sent on to the control
      .     module. Presently,
      .     formatting of the control
      .     parameters is not complete.

   <control[number_of_controls]>
   <controlparameters[number_of_controls]>
```

A.2.2.2. Log File

The log file contains additional observations about network status. The log file is written to directly in IWES routine write_log(), which is located in com_package.c. The log is written into the ${IWDATA]/log/iwes directory and has a file name format of:

```
        YYMMDD.iwes.log     where: YY = year
                                   MM = month
                                   DD = day
                                 e.g. 900927.iwes.log
```

Format:

```
  <time> <observation>   where: time is in the form HH:MM:SS
                                HH = hour
                                MM = minute
                                SS = seconds
                                observations = a line of text
```

(Th. is repeated for as many observations as there are during the day)

A.3 Message Dispatcher Messages

A.3.1 Received from the Message Dispatcher:

Messages are read from message dispatcher in IWES routine read_iwmd(). Read_iwmd() is called by processiwmd(). In processiwmd() IWES analyses the messages it has receives and determines what actions it should take as a result of receiving the message. Both read_iwmd() and process_iwmd() are located in iwmd.c.

A.3.1.1 Polldone

Format: <src> <dest> polldone <timestamp>

When IWES receives a polldone message it accesses raw switch reports from the IWDB, runs monitors and statistical database monitors to detect switch report anomalies and asserts switch report anomalies into CLIPS as facts.

A.3.1.2 Diagdone

Format: <src> <dest> diagdone <timestamp>

When IWES receives a diagdone message it reads the neural network output file and asserts neural network anomalies into CLIPS as facts.

A.3.1.3 Polledst

Format: <src> <dest> polledst <timestamp>

Presently, no actions are taken when IWES receives a polledst message.

A.3.1.4 Ctrlsent

Format: <src> <dest> ctrlsent <timestamp>

Presently, no actions are taken when IWES receives a ctrlsent message. In the future, IWES could read the ctrlsent file and check for controls that it had recommended.

A.3.1.5 Ctrlresp

Format: <src> <dest> ctrlresp <timestamp>

Presently, no actions are taken when IWES receives a ctrlresp message.

A.3.1.6 Decodedone

Format: <src> <dest> decodedone <timestamp>

Presently, no actions are taken when IWES receives a decodedone message.

A.3.2 Sent to the Message Dispatcher

Both of the messages described below are sent to the message dispatcher in processiwmd() which is located in iwmd.c

A.3.2.1 Procdone

Format: iwes iwui procdone <timestamp>

IWES sends a procdone message to the user interface when it has completed processing the switch reports. Presently, the user interface is not taking any action when it receives the procdone message.

A.3.2.2 Recsdone

Format: iwes iwui recsdone <timestamp>

IWES sends a recsdone message to the user interface after it has completed processing the neural network results, running clips and writing out its results. Upon receiving the recsdone message the user interface accesses the IWES results file.

APPENDIX B. IWES Monitor & Rule Descriptions

This Appendix contains descriptions of the monitors and the rules used in IWES (the version of NMES created for the IW workstation.)

The monitors described below are C routines. The rules are written in CLIPS. Monitor names are those used in IWES. The program name is followed by an English language name intended to be descriptive of the function of the monitor.

The confirmation of anomalies involves an interaction between monitors and CLIPS rules. When the switch report data is read, the fields are checked by monitors. If the contents of a switch report field is not normal, then a CLIPS fact is asserted which states this abnormality. When the rules are run, these abnormal data facts are matched with the neural net outputs to obtain either confirmed or not-confirmed anomalies. There is a set of 8 rules for each anomaly recognized by the neural net. Rather than describe all the rules, we present a description in some detail of a template for these 8 rules (Section B.2.1). The description omits some details that relate to CLIPS syntax and other internal programming issues in order to simplify the presentation without losing information relative to the functions performed by the rules. Following the description of the template, we present an English language explanation of the logic involved in the confirmation of each anomaly.

B.1 IWES Monitors

Monitors are C language routines that process switch report data and/or the outputs of other monitors and store the results in a set of C language structures representing the network. Many of them also work to help maintain (in C arrays) lists of switches and trunk groups that have anomalous (non-normal) states. The items on the lists combine the switch or trunk group with an anomaly identifier. When a monitor recognizes an anomaly, it appends an item to the appropriate list. It may also remove an item previously put on the list. There can be more than one item on the list for a particular switch or trunk group, but in such a case, the anomaly will be different. These lists are reported to the IW as "Observations."

Many of the monitors assert their findings as CLIPS facts. When operating with the Lincoln Laboratory graphics, monitors generally write messages to a scroll window so that the user can be made aware that a monitored situation has occurred. That activity does not take place when IWES is running in the IW and is omitted from the descriptions that follow.

The order in which individual monitors are run is important since some use the results generated by others. In the following sections, they are described in the order in which they run in MNES.

B.1.1 Switch Monitors

The monitors in this and the following section are run as each switch report arrives.

B.1.1.1 Switch Report Register Monitors

These monitors check the individual registers in the switch report for a non-zero value which can indicate an abnormality with the system. The checks all follow the following logic:

```
    IF the value of the register is greater than zero THEN
       assert for CLIPS the fact that switch X showed an instance
       of
       a non-zero register for this time cycle
       append a register-failure state to the switch state list
    ELSE remove any register-failure state for switch X which
might be left from a previous cycle
```

The following includes a list of the switch monitor routines and the individual registers they are checking. They are all found in the file switch_monitor.c.

Monitor - find_no_mf_receiver_free
    Check the switch report RCVR Mf Receiver Overflow register for evidence that attempts to assign a Multi-Frequency (MF) receiver found none free (overflow occurred).

Monitor - find_no_dialing_receiver_free
    Check the switch report RCVR DGT Receiver Overflow register for evidence that attempts to assign a Digitone receiver found none free (overflow occurred).

Monitor - find_cp_overflows
    Check Call Processing (CP) Registers for:
- CCB overflow during attempt to seize a Call Condense Block (CCB)
- CP failures due to illegal software conditions (CP TRAP)
- CP failures due to unexpected results detected (CP SUIC)
- Call originations denied during warm and cold restarts (INITDENY)

Monitor - find_mf_radr_overflows
    Check MF RADR Registers for:
- MF failure to get response from receiver within the lower delay threshold (RADLDLYP)
- MF failure to get response from receiver within the upper delay threshold (RADUDLYP)

Monitor - find_dial_radr_overflows
Check Digitone RADR Registers for:
DGT failure to get response from receiver within the lower delay threshold (RADLDLYP)
DGT failure to get response from receiver within the upper delay threshold (RADUDLYP)

Monitor - find_cpu_overflows
Check CPU registers for:
CPU mismatch interrupts due to interprocessor differences between the two CPU's (MTCHINIT)
CPU data store, program store, CMC link or CMC data port system busied following a mismatch or trap interrupt (CPUFLT)
CPU warm restart and cold restart (SYSWINIT, SYSCINIT)
CPU simplex -
SYNCLOSS - Processor made simplex following mismatch interrupt MSYLOSSU - Time usage of simplex mode due to manual intervention
SSYLOSSU - Time usage of simplex mode due to system action

Monitor - find_cmc_overflows
Check CMC registers for calls to diagnostics on the CMC as a result of errors and faults on the registers CMC 0 and CMC 1.

Monitor - find_trmtcm_overflows
Check TRMTCM registers for:
TCM partial digit calls occurred - i.e. one or more digits received but not enough for translation (TCMPDIL)
TCM permanent signal calls occurred - i.e. Seizure with no ensuing digits (TCMPSIG)
TCM vacant code calls - i.e. Translation occurred but no matching pattern is found (TCMVACT)

Monitor - find_trmter_overflows
Check TRMTER registers for:
TER miscellaneous system failure calls (TERSYFL)
TER reorder calls which can be caused by mutilated digits, forced release, unexpected stops, excessive digits for translation, invalid start signals, invalid translations and others.(TERRODR)

Monitor - find_trmtrs_overflows
Check TRMTRS registers for:
TRS emergency treatment 1 for deflected calls for CBK, DCC, ARC, or anything sent to EA1 (TRSEMR1)
TRS emergency treatment 2 for deflected calls for CBK, DCC, ARC, or anything sent to EA2 (TRSEMr2)

Monitor - find_dcm_overflows
Check DCM registers for:
DCM referrals of a digital trunk for diagnostics (DCMCCTDG)
DCM digital trunk diagnostic failures (DCMCCTOP)

B.1.1.2 Other Switch Monitors

These switch monitors are independent of neural net anomaly recognition. They check for receiving switch reports, not receiving switch reports, and evidence of activity at neighbor switches.

Monitor - mark_switch_report_received (in switch_monitor.c)
Remove any not-reporting state information for the switch. (The 'reporting' state is not considered an anomaly and is not kept on the switch state list.)

```
IF a switch report is received from switch X
THEN remove any not-reporting state for switch X that might
be left from a previous cycle
```

Monitor - remove_switch_monitors (in switch_monitor.c)
Remove switch monitor states when node is not responding.

```
IF a switch report is not received from switch X
THEN remove any reporting state for switch X that might be
left from a previous cycle including switch report field
states
```

Monitor - find_nodes_not_responding (in interval_monitors.c)
Find the nodes that haven't responded during the last time period.

```
IF switch X is a 'porting switch
AND switch X did .ot report this cycle
THEN assert for CLIPS the node-not-responding fact for switch
X For each clli associated with the nodes:
  update link state for associated trunk groups assert for
CLIPS the fact no-clli-information-received
ELSE remove any node-not-responding states for this switch
which might be left from a previous cycle.
```

Monitor - find_no_outgoing_attempts (in interval_monitors.c)
Check for evidence of outgoing call attempts by switch X seen as incoming calls from switch X at its neighbors.

```
IF one or more neighbor switches to switch X reported
AND the total incoming attempts from switch X seen by those
neighbors was not greater than zero
THEN assert for CLIPS the fact that
neighbors-see-no-incoming-signals from switch X at this time
```

```
    append a no-outgoing-attempts state for switch X to the
    switch state list
    ELSE remove any no-outgoing-attempts state for this switch
    which might be left from a previous cycle
```

B.1.2 Trunk Group Monitors

A normal switch report contains OM reports for each of the trunk groups that connect the switch to its neighbors. Each trunk group (tg) report is identified by a group number. Communication problems during switch polling may cause the loss of any or all of the tg reports associated with the poll. The following monitors are run as the switch report is being processed. They are run for all the tg reports that actually arrive. An additional monitor is run at the end to identify trunk groups for which no report was received.

B.1.2.1 Trunk Group Register Monitors

These monitors check the individual registers in the trunk group switch report for a non-zero value which can indicate an abnormality with the system. The monitors all follow the following logic:

```
    IF the value of the register is greater than zero
    THEN assert for CLIPS the fact that trunk group X showed an
    instance of a non-zero register for this time cycle Append a
    register-failure state to the trunk group state list
    ELSE remove any register-failure state for trunk group X
    which might be left from a previous cycle
```

The following includes a list of the trunk registers checked. They are all found in a single routine in the file report_monitor.c.

Monitor - check_fail_clli_report

Check the following trunk registers for:
Maintenance Busy Usage (MBU) on trunks due to manual actions adjust adjusted_inservice variable for holding time calculations System Busy Usage (SBU) on trunks due to system problems. adjust adjusted_inservice variable for holding time calculations
Incoming failure including permanent signals, partial digits, mutilated digits, receiver problems, etc. (INFAIL)
Failure to effectively seize the chosen trunk due to hardware problems (OUTFAIL)
Machine detected glare on the selected trunk (GLARE)
Preemption attempts which were unsuccessful because there were no trunks available with a lower precedence level (PREOVFL)

B.1.2.2 Other Trunk Group Monitors

```
Monitor - find_stats  (in stat_processor.c)
    Calculate from the current switch report and store in the tg
    structures values corresponding to the statdb trunk group
    stat types:
         aht - Average holding time
         cch - Outgoing connections per circuit per hour
         icch - Incoming connections per circuit per hour
         tcch - Total connections per circuit per hour
         puse - Percent usage
         povfl - Percent overflow

Monitor - calculate_statistics (in stat_processor.c)
    Check for current statdb trunk group stat type values falling
    outside their acceptable ranges.
```

For each of the tg stat types defined in the find_stats monitor above:

Compute an acceptable_range equal to the statdb normal value plus or minus FACTOR times the statdb standard_deviation (FACTOR comes from the stat-SD_FACTOR.fact file described in section 5.8.x.1.2.)

```
    IF the current value is outside the acceptable_range
    OR the value = 100%
    THEN assert for CLIPS the fact that an abnormal value was
    detected;

    IF the abnormal condition has occurred twice in a row
    THEN append an abnormal high or low state to the trunk group
    state list
    ELSE remove any abnormal high or low state for the trunk
    group which might be left from a previous cycle

Monitor - find_low_ht_calls  (in report_monitors.c)
    Find trunks with many connections but low holding time
    IF there have been outgoing attempts on the trunk group
    AND the holding time is less than 20 seconds but more than 0
    seconds
    AND there are trunks in service
    AND there are at least 36 calls per circuit per hour
    THEN assert for CLIPS the fact that there is an instance of
    low-ht-with-calls for this time cycleaon this tg append a
    low-ht-with-calls state to the trunk group state list
    ELSE remove any low-ht-with-calls state which might be left
    from a previous cycle
```

Monitor - find_max_usage_few_calls (in report_monitors.c)
    Find trunks with high usage and few calls, i.e., more usage than the current time cycle calls would be expected to generate

    IF there are fewer calls than there are trunks inservice
    AND the maximum usage the calls could generate is less than the usage register indicates
    THEN assert for CLIPS that there is high_usage_few_calls for this time cycle append a high-usage-few-calls state to the trunk group state list
    ELSE remove any high-usage-few-calls state which might be left from a previous cycle

Monitor - find_zero_usage_calls (in report_monitors.c)
    Check for calls with no usage

    IF there is at least one outgoing or incoming call connection
    AND the TRU usage register reports 0 usage
    AND there is at least one trunk in service after accounting for MBU and SBU
    THEN assert for CLIPS the fact that there is an instance of zero_usage_calls for this time cycle on this tg append a zero-usage-with-calls state to the trunk group state list
    ELSE remove any zero-usage-with-calls state which might be left from a previous cycle

Monitor - find_clli_trunks_down (in report_monitors.c)
    Check tg report for evidence that the number of trunks in service is less than the equipped value, and/or is less than the number reported previously.

    IF the reported number of trunks in service is less than the reported equipped number
    THEN append a decreased_capacity state for this tg to the tg state list
    ELSE remove any decreased_capacity state for this tg which might be left from a previous cycle
    IF the reported number of trunks in service is less than the number reported previously
    THEN assert for CLIPS the fact that a capacity change has occurred on this tg during this cycle

Monitor - find_hundred_overflow (in report_monitors.c)
    Check tg report for the case where the overflow peg count equals the outgoing attempts peg count and the usage value is zero. This situation corresponds to one type of trunk failure.

    IF there are outgoing attempts on the trunk group
    AND the overflows equal the outgoing attempts
    AND the trunk usage is zero

```
   THEN assert for CLIPS the fact that a
   hundred-percent-overflow-zero-usage condition exists on this
   tg at this report time
```

Monitor - find_zero_overflow (in report_monitors.c)
Check tg report for the case where there are outgoing attempts, no overflows, and no usage. This situation can occur if 100% SKIP controls are put on at both ends of the tg.

```
   IF there are outgoing attempts on the trunk group
   AND the overflow peg count is zero
   AND the trunk usage is zero
   THEN assert for CLIPS the fact that a
   zero-percent-overflow-zero-usage condition exists on this tg
   at this report time
```

The following monitor is run after the monitors described above have been run for all the trunk groups reported in a switch report.

Monitor - find_cllis_without_reports (in report_monitors.c)
By going through a list of trunk groups that exist at the reporting switch and for which reports are expected, find and mark those missing.

```
    FOR all trunk groups in the C structure representing switch X
    DO
    IF the time stored in the trunk group representation does not
    equal the time of the current switch report
    THEN assert for CLIPS the fact that a
    no-tg-information-received condition exists for this tg at
    this report time append a no-clli-report state for this tg to
    the tg state list remove all other tg states for this tg
    which can no longer be believed in the absence of a tg report
    ELSE remove any no-clli-report state for this tg which might
    be left from a previous cycle
```

B.2 IWES Rules

B.2.1 General Rules for Neural Net Confirmation

The anomalies which the IW Neural Net diagnosed were specified two days before the September testing started. The IWES had to be able to handle all possible anomalies to be prepared for the testing. Therefore, a shell of anomaly rules were designed and written which handled the original neural net anomalies, the anomalies defined in the Network Management Situation Diagnosis (Aug.24.1990) and others mentioned in discussions with GTE during the August visit to Germany. The rules were all designed using the detailed anomaly template specified below. However, the initial set of anomalies did not have confirmation logic added and were designed to give a non-committal response as an anomaly confirmation. When examples of switch data representing the anomalies became available, confirmation logic was added to the rules. The IWES had confirmation knowledge for all of the Neural Net anomalies diagnosed at the September tests. The anomalies which do not have confirmation logic will still provide the user with essential data in the output to allow the user to confirm or not confirm the anomaly.

The rules used to confirm the neural net anomalies use the same eight-rule template. The eight rules are designed to:

1. Confirm first time anomaly (nn-anomaly)
2. Confirm subsequent anomaly (nn-anomaly-still)
3. unsupported first time anomaly (nn-anomaly-disagree)
4. unsupported subsequent anomaly (nn-anomaly-disagree-still)
5. Remove confirmed anomaly when no neural net anomaly present (nn-nmes-not-reconfirmed-anomaly)
6. Remove confirmed anomaly when neural net anomaly no longer supported (nn-nmes-to-nn-only-anomaly)
7. Remove unsupported anomaly when no neural net anomaly present (nn-only-not-reconfirmed-anomaly)
8. Remove unsupported anomaly when neural net anomaly is confirmed (nn-only-to-confirmed-anomaly)

B.2.1.1 Inputs to Neural Net Anomaly Rules

There are two classes of anomalies - switch anomalies and trunk group anomalies. To report on a switch anomaly, the switch name is needed along with the status message. To report on a trunk group anomaly, the clliname, name of the source switch, name of the destination switch, and the status message is required. The Source and Destination names for the trunk groups are found in the CLLI fact which is asserted when the expert system starts execution.

The rules require the following facts:

1. A Neural Net diagnosis of an anomaly for this time period. The CLIPS fact will be prefaced by a "NN" with the anomaly information contained in it. That the NN fact is from the current reporting time is assured by requiring a match between the variable ticks-now and the current time fact.

2. A "confirmed" or "not-supported" fact indicates that this is not the first time the anomaly has appeared. These facts are asserted by the neural net rules when they are fired. Absence of this fact means that this is the first time the anomaly has been confirmed or not-supported.

3. A switch report data fact (or facts) which describe non-normal conditions in the network. These facts are asserted by expert system monitors using the switch reports and the statistical data base.

4. The NET-TIME fact is the hour-minutes-second translation used by the logging procedures.

5. The CURRENT-TIME fact is the current time represented in tenths of seconds.

B.2.1.2 Outputs of Neural Net Anomaly Rules

The actions taken when an anomaly is confirmed or not supported are:

1. Retract the confirmed or not supported fact if it exists

2. Assert a fact to be used the next time period

3. Call a C routine - either nn_switch_status or nn_clli_status where the appropriate status messages are logged and flags set.

4. Call C routine clips_write_scroll to log the expert system action (if running with the LL graphics interface).

The actions taken when an anomaly no longer exists are:

1. Retract the confirmed or not supported fact

2. Call C routine nn_switch_status or nn_clli_status to remove anomaly status information.

There are two built-in fields in the call to the C status routines which are intended to be used to return related data for the user messages. The first field is a count of the number of times the expert system has seen the same problem. The second field is used when an unidentified neural net anomaly is found. The other neural net anomalies return the variable "sw-data" which is never used by the C routine.

B.2.1.3 Neural Net Rule Template

For each neural net anomaly there are the following 8 rules:

nn-anomaly: This rule is designed to confirm a first-time anomaly.

```
IF there is an NN fact for this time period
AND the relevant switch report data is there
AND this anomaly was not confirmed last time period
THEN Assert a fact stating a confirmed anomaly for over-time
evaluation
Call a C routine to flag confirmation to IW Log   conclusion
```

nn-anomaly-still: This rule confirms that a problem found in the previous time period still exists.

```
IF there is an NN fact for this time period
AND the same problem was confirmed last time period
AND the relevant switch report fact is there
THEN Retract the confirmed problem fact from previous time
period Assert a fact stating a confirmed anomaly for
over-time evaluation
Call a C routine to flag confirmation to IW Log conclusion
```

nn-anomaly-disagree: This rule finds a neural net anomaly that the data does not confirm.

```
IF there is an NN fact for this time period
AND the relevant switch report data is not there
AND this anomaly was not confirmed last time period
THEN Assert a fact stating an unsupported neural net anomaly
was found
Call a C routine to flag non-confirmation to IW Log
conclusion
```

nn-anomaly-disagree-still: This rule finds that an unsupported anomaly found in the previous time period still exists and is still unsupported.

```
IF there is an NN fact for this time period
AND the same anomaly was unsupported last time period
AND the relevant switch report fact is not there
THEN Retract the unsupported statement from previous time
period Assert a fact stating an unsupported anomaly was
found
Call a C routine to flag non-confirmation to IW Log
conclusion
```

nn-nmes-not-reconfirmed-anomaly: This rule finds a confirmed fact from the previous time period with no new NN statement and removes the confirmed fact.

```
IF there is a confirmed anomaly fact from last time period
AND there is not a new neural net fact for the anomaly
THEN Retract the confirmed fact
Call a C routine to flag that the anomaly has disappeared
```

nn-nmes-to-nn-only-anomaly: This rule removes a confirmed fact from the previous time period when the new NN anomaly is not confirmed by the switch report.

```
IF there is a confirmed anomaly fact from last time period
AND there is a new neural net fact for the anomaly
AND the relevant switch report fact is not there
THEN Retract the confirmed fact
Call a C routine to remove the confirmed conclusions
```

nn-only-not-reconfirmed-anomaly: This rule removes an unsupported fact from the previous time period when no new NN fact exists.

```
IF there is an unsupported anomaly fact from last time
period
AND there is not a new neural net fact for the anomaly
THEN Retract the confirmed fact
Call a C routine to flag the anomaly has disappeared
```

nn-only-to-confirmed-anomaly: This rule removes an unsupported fact from the previous time period when the new NN statement is confirmed by the switch report.

```
IF there is an unsupported anomaly fact from last time
period
AND there is a new neural net fact for the anomaly
AND the relevant switch report fact is there
THEN Retract the confirmed fact
Call a C routine to remove the unsupported conclusions
```

B.2.1.4 Rules for Switch-Related Neural Net Anomalies

The Neural Net and IWES in some cases use different terms (spellings) for the same anomaly. Section B.2.1.6 has a table that shows the translation between the sets of terms.

The switch anomalies that are confirmed by the expert system are:

CPU mismatch	Outage remote
CPU inits	Miscellaneous system failure
Permanent Signals	Mf Receiver Overflow

The switch anomalies for which there is no confirmation logic are:

congestion remote	degraded remote
congestion reporting	degraded reporting
cpu simplex	internal problem
cp origination problem	rcvr problem
line frame problem	distant switch signal problem
trunk system failure	excess_receiver out of service
facility_hits	

The confirmation logic for each anomaly is included in the description of the rules. For the anomalies with no confirmation logic, a discussion of possible confirmation logic is included.

B.2.1.4.1 CPU Mismatch (nn-cpu.clp)

CPU mismatch is recognized by a non zero count in the CPU MTCHINIT register. This register counts the number of mismatch interrupts due to inter-processor differences the switch has experienced in the last reporting period. This anomaly is confirmed by a fact asserted by the switch monitors when this register is not zero. If the Neural Net fact for cpumismatch on a specific switch and the IWES monitor fact of a non-zero value in the cpu mismatch register on the same switch are in the CLIPS fact base then the CPU mismatch anomaly is confirmed. If the Neural Net fact for cpumismatch on a switch is not accompanied by a IWES monitor fact, the rules decide that the neural net anomaly cpumismatch is not confirmed.

B.2.1.4.2 CPU Initialization (nn-cpu.clp)

CPU inits is recognized by a non zero count in the CPU SYSWINIT or CPU SYSCINIT registers. These registers indicate the number of warm or cold restarts the switch has experienced in the last reporting period. This anomaly is confirmed by a fact asserted by the switch monitors when one or both of these registers are not zero. If the neural net fact for cpuinits on a specific switch and the IWES monitor fact of a non-zero value in the cpu restart registers on the same switch are in the CLIPS fact base then the CPU inits anomaly is confirmed. If the Neural Net fact for cpuinits on a switch is not accompanied by an IWES monitor fact, the rules decide that the neural net anomaly cpuinits is not confirmed.

B.2.1.4.3 Permanent Signals (nn-switch-problems.clp)

Permanent signals are recognized by MF receiver usage increasing with no great increase in seizures. There is a permanent signal register in the TCM Register that indicates when permanent signal call seizures occur. The CLIPS rules require the IWES monitor fact of non-zero permanent signals to confirm the neural net permanent signal anomaly. The ratio of MF Receiver usage/seizure is also essential. This ratio is calculated by subtracting the RADR MF test count from the number of RCVR MF seizures, dividing this count of actual MF seizures into the RCVR MF usage and multiplying by 10 (to change from tenths of seconds to seconds). This information is available from the statistical data base and should be added to the rules.

B.2.1.4.4 Outage Remote (nn-switch.clp)

Machine failure or Switch Outage Remote is recognized by no incoming calls and low holding time, possibly followed by high overflow if trunks go out of service or are put out of service. The neural net anomaly is confirmed if IWES sees that the switch did not report and the neighbors of the switch did not see any calls from the switch. If the node did report or the neighbors did see calls from the switch, then the neural net anomaly is not confirmed. The data from the trunks connected to the switch with the outage should indicate the accessibility of the switch.

B.2.1.4.5 Miscellaneous System Failure (nn-switch-problems.clp)

Miscellaneous system failure calls are recognized by a non-zero count in the TRMTER Register TERSYFL. If the IWES switch monitor detects a non-zero value in this register, the neural net anomaly will be confirmed. If the value of the TERSYFL register is zero, the neural net anomaly will not be confirmed.

B.2.1.4.6 MF Receiver Overflow (nn-rcvr.clp)

Mf receiver overflow is determined by non zero values in the RCVR MF. Overflow register of the switch report. If the switch report shows overflows, then this neural net anomaly will be confirmed. The ratio of usage/seizures will probably not be in the normal range. If there were no RCVR MF overflows, the neural net anomaly will not be confirmed.

B.2.1.4.7 Congestion at Remote Switch (nn-congestion.clp)

Congested Remote Switches are difficult to diagnose. There should be high traffic flow on most of the surrounding trunk groups. The neural net anomaly that diagnosed this was removed before September 1990 testing. IWES does not have any confirmation logic for this anomaly.

B.2.1.4.8 Congestion at Reporting Switch (nn-congestion.clp)

The DMS reporting switches should not experience congestion with the trunking available in the European DSN. The neural net anomaly to diagnose this problem was removed prior to September 1990 testing. IWES does not have any confirmation logic for this anomaly. It is expected that, should congestion occur, the symptoms would include CCB and RCVR overflows. Currently, IWES would infer other problems from those symptoms alone.

B.2.1.4.9 CPU Simplex (nn-cpu.clp)

CPU simplex is recognized by a non zero count in the CPU SYNCLOSS MSYLOSSU, or SSYLOSSU registers. The CPU SYNCLOSS register counts the times the processor complex was made simplex following mismatch interrupts. The CPU MSYLOSSU register contains the usage time of simplex CPU mode due to manual intervention. The CPU SSYLOSSU contains the usage time of simplex CPU mode due to system action. IWES does not have any confirmation logic for this anomaly yet and the neural net is not trained to recognize it. However, the CLIPS rules would need to look for a fact stating CPU simplex to contain the required knowledge for confirmation.

B.2.1.4.10 CP Origination Problem (nn-switch-problems.clp)

Switch Origination Problems can be recognized by a non-zero value in the CP INITDENY Register. The CP INITDENY Register contains an estimate of call originations denied during warm and cold restarts. This neural net' anomaly was not included in the September testing and IWES does not have confirmation knowledge coded. However, to confirm the anomaly, the CLIPS rules would need to look for a fact stating CP initdeny failure.

B.2.1.4.11 Line Frame Problem (nn-switch-problems.clp)

Line Frame Problems are recognized by CCB seizures, digitone receiver seizures, and usage dropping to a minimal level. The anomaly was included in the Network Management Situation Diagnosis document but the neural net was not trained to detect it. IWES has no confirmation knowledge coded to confirm this anomaly, nor has a specific example of the problem been stated. However, to confirm the anomaly IWES could compare the actual ratio of RCVR Digitone usage/seizures to the normal value. The actual value should be lower than usual as usage should decrease according to the anomaly definition. The CCB seizures actual value should be higher then the normal value.

B.2.1.4.12 Trunk System Failure (nn-switch-problems.clp)

Trunk System Failures are recognized by high receiver and CCB peg count with no overflow and no usage. The anomaly was included in the Network Management Situation Diagnosis document but the neural net was not trained to detect it. IWES has no confirmation knowledge coded to confirm this anomaly, nor has a specific example of the problem been stated. However, to confirm the anomaly IWES could compare the Ratio of MF usage/seizure actual value to the normal value and the Ratio of DGT usage/seizure actual value to the normal value. The actual values should be much lower than the normal values. An additional confirmation would be that the switch monitors did not have find any receiver overflows.

B.2.1.4.13 Degraded Remote (nn-switch.clp)

Degraded Remote Switch is recognized by low or no incoming calls and low holding time on the surrounding trunk groups. The anomaly was included in the Network Management Situation Diagnosis document and in the original neural net training set but the neural net was not trained to detect degraded remote switches during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. However, to confirm it, the CLIPS rules would have to look for low holding time or low incoming call facts for all trunk groups which have this remote switch as a destination. More code than the standard 8-rule template would be required.

B.2.1.4.14 Degraded Reporting (nn-switch.clp)

A degraded reporting switch could be caused by several particular problems indicated by receiver and ccb problems. The anomaly was included in the original neural net training set but the neural net was not trained to detect degraded reporting switches during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. More knowledge of the symptoms of a degraded reporting switch is needed to confirm it.

B.2.1.4.15 Internal Problem (nn-switch-problems.clp)

Switch internal problems are identified by the inability to make a receiver connection or CCB connection. The anomaly was included in the Network Management Situation Diagnosis document and in the original neural net training set but the neural net was not trained to detect degraded remote switches during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. A test set of data has not been defined. More information is needed before IWES can confirm this anomaly.

B.2.1.4.16 RCVR Problem (nn-rcvr.clp)

Receiver problems are identified by RADR and/or receiver overflows without an increase in CCB seizures. The anomaly was included in the Network Management Situation Diagnosis document but the neural net was not trained to detect receiver problems during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. A specific test case has not been identified. However, to confirm it, the CLIPS rules would have to check for either MF or DGT RCVR overflows or MF or DGT RADR overflows while the ratio of usage/seizures remains normal for both MF and DGT RCVRs. The number of CCB seizures would be in the normal range too.

B.2.1.4.17 Distant Switch Signal Problem (nn-switch-problems.clp)

Distant Switch signalling problems are recognized by an increase in reorders and vacant codes with no increase in CCB or receiver seizure. The anomaly was included in the Network Management Situation Diagnosis document but the neural net was not trained to detect excessive receivers out of service during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. A specific test case has not been identified. However, to confirm it, the CLIPS rules would have to check for switch monitor facts stating that the TRMTER Register value for reorder calls, TERRODR, and TRMTCM Register value for vacant code calls, TCMVACT, are not zero. There also should be normal values for CCB seizure and the RCVR usage/seizure ratio.

B.2.1.4.18 Excess Receiver Out of Service (nn-rcvr.clp)

Excessive Receivers out of Service is identified by receiver overflow with normal level of CCS. There should be receiver system busy usage (SBU) or maintenance busy usage (MBU). The anomaly was included in the Network Management Situation Diagnosis document but the neural net was not trained to detect excessive receivers out of service during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. A specific test case has not been identified. However, to confirm it, the CLIPS rules would have to check for a no_mf_receiver_free fact and check that the SBU and MBU registers values are not zero.

B.2.1.4.19 Facility Hits (nn-switch-problems.clp)

Facility hits are recognized by permanent signals, MF receiver overflows and MF RADR delays. The anomaly was included in the Network Management Situation Diagnosis document but the neural net was not trained to detect excessive receivers out of service during the September 1990 testing. IWES has no confirmation knowledge coded to confirm this anomaly. A specific test case has not been identified. However, to confirm it, the CLIPS rules would have to check for switch monitor facts stating non-zero values were found for the TRMTCM Register TCMPSIG (permanent signal calls), RCVR Register MF RCVOVFL (receiver overflows), and either RADR MF RADLDLYP (lower threshold delay) or RADR MF RADUDLYP (upper threshold delay).

B.2.1.5 Rules for Trunk-Related Neural Net Anomalies

The Neural Net and IWES in some cases use different terms (spellings) for the same anomaly. Section B.2.1.6 has a table that shows the translation between the sets of terms.

The trunk group anomalies that are confirmed by the expert system are:

Permanent Seizure	Excess Trunks out of Service
Facility Hit/Tropofade	Maintenance Busy Trunks Out of Service
No Usage	System Busy Trunks Out of Service
Signalling Glare	Trunk Signalling Problem

The trunk group anomalies for which there is no confirmation logic are:

Degraded trunk	Congested trunk group
Dpas failure	Trunk group failure
Degraded facility	Span failure
Continuity failure	Trunk testing
Dtc failure	Transmission fault

The confirmation logic for each anomaly is included in the description of the rules. For the anomalies with no confirmation logic, a discussion of possible confirmation logic is included.

B.2.1.5.1 Permanent Seizure (nn-tg-problem.clp)

Permanent seizure, or trunks permanently held up but not with real traffic, is recognized with full usage and high holding time. The pattern continues with no supporting peg counts. There must be more usage on the trunks than the incoming and outgoing calls would be expected to generate, and there must be fewer total calls than the number of in-service trunks. These rules were used to confirm a neural net diagnosis of permanent seizure. The difficulty is recognizing when a partial set of the trunk group has been seized, thus allowing some calls to get through while blocking other calls. The number of calls, usage, number of inservice trunks, holding time and percent usage are of interest to this anomaly.

B.2.1.5.2 Facility Hit/Tropofade (nn-tg-failure.clp)

Facility Hit is recognized by a very high peg count and a very low holding time. This anomaly was called tropofade in the original neural net, LARS. There is a monitor which detects more than 3 calls per circuit with a low holding time and at least one inservice trunk group. The number of calls, usage, number of inservice trunks, holding time and percent usage are of interest to this anomaly along with the normal statistics for these fields.

B.2.1.5.3 No Usage (nn-tg-failure.clp)

The no-usage anomaly was called 100%-skip in the original neural net. There are calls but no usage and no holding time. There should also be no or very few overflows. The number of incoming and outgoing calls, the number of overflows, the percent usage, and the holding time are all used to determine this anomaly.

B.2.1.5.4 Signalling Glare (nn-tg-failure.clp)

Trunk Signalling Glare problems are identified by low holding time and glare counts in the switch report. This anomaly is confirmed by a non-zero count in the Glare register on the trunk group report. The ratio of failure is the total of outfails and glare in relationship to attempts. A high ratio of failure can indicate a trunk signalling problem instead of a signalling glare problem.

B.2.1.5.5 Excess Trunks Out of Service (nn-tg-oos.clp)

Excess Trunks Out of service requires the trunk group to have less than 75 percent of equipped trunks in service. This is a determined by the ratio of the Trunks In Service Register divided by the Trunks Equipped Register.

B.2.1.5.6 Maintenance Busy Trunks Out of Service (nn-tg-oos.clp)

Trunks Out of Service due to Maintenance Busy are identified by a count multiple of 3 for 5 minute data or 9 for 15 minute data in the MBU register. There should be overflows and some connections if some of the trunks are not out of service. The numbers of equipped and inservice trunks along with the call statistics are useful in confirming this problem.

B.2.1.5.7 System Busy Trunks Out of Service (nn-tg-oos.clp)

Trunks Out of Service because of System Busy are identified by a count multiple of 3 for 5 minute data or 9 for 15 minute data in the SBU register. There should be overflows and connections if some of the trunks are not out of service. The numbers of equipped and inservice trunks along with the call statistics are useful in confirming this problem.

B.2.1.5.8 Trunk Signalling Problem (nn-tg-failure.clp)

Trunk Signalling problems are identified by low holding times sometimes accompanied by higher peg counts. A significant portion of outgoing attempts fail or experience glare. The ratio of failure is the total of outfails and glare in relationship to the number of attempts. If this ratio is indicates more than 50% of the attempts fail, then the anomaly is confirmed.

B.2.1.5.9 Degraded Trunk (nn-tg-failure.clp)

A Degraded Trunk is difficult to confirm. It was included in the original neural net as excess traffic outgoing and excess traffic both ways. This anomaly was not included in the list of anomalies shown in September. It was felt that a more specific diagnosis was preferred to a general trunk problem diagnosis. The current trunk statistics and their normal values may be helpful in detecting this anomaly.

B.2.1.5.10 DPAS Failure (nn-tg-problem.clp)

DPAS Failure is identified by Multiple connection office receiver problems. There are general, across the network, problems. This anomaly was detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly would include comparing the trunk statistics with the normally expected values for significant decreases in holding time and increases in connections. More information would be needed in the data base to define the interconnection of the network DPAS.

B.2.1.5.11 Degraded Facility (nn-tg-problem.clp)

Trunk facility failure or degraded facility is recognized as a signalling problem for the originator. The terminating office will show high peg count (maybe), receiver overflow, permanent signals, and RADR problems. This anomaly was detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly would include the switch permanent signal register being non-zero and high incoming and outgoing attempts.

B.2.1.5.12 Continuity Failure (nn-tg-problem.clp)

Continuity failure is indicated by low holding time. This anomaly was detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly would include looking at data from the trunk reports for the source and destination switches and comparing this data to the normal expected values for holding time and connections. A test case for this anomaly needs to be identified before the rules can be written so that this anomaly can be differentiated from other anomalies identified by low holding time.

B.2.1.5.13 DTC Failure (nn-tg-problem.clp)

DTC problems or digital trunk frame problems are identified by the originating office showing trunk groups with high attempts and low holding times. This anomaly was detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly would include looking at data from the trunk reports for the source and destination switches and comparing this data to the normal expected values for holding time and outgoing attempts. A test case for this anomaly needs to be identified before the rules can be written so that this anomaly can be differentiated from other anomalies identified by low holding time.

B.2.1.5.14 Congested Trunk Group (nn-tg-failure.clp)

Trunk Congestion is difficult to confirm. It was included in the original neural net anomalies. This anomaly was not included in the list of anomalies shown in September. It was felt that a more specific diagnosis was preferred to a general trunk problem diagnosis. The current trunk statistics and their normal values may be helpful in detecting this anomaly. A test case for this anomaly needs to be identified before the rules can be written so that this anomaly can be differentiated from other anomalies.

B.2.1.5.15 Trunk Group Failure (nn-tg-failure.clp)

Trunk Failure is recognized by no incoming calls, and 100% overflow. Usage is either very low or very high. This anomaly was included in the original neural net and in the list of anomalies detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly already exists in the NMES detection of trunk failure and these neural net rules just need to be linked to the fact asserted by the existing rules.

B.2.1.5.16 Span Failure (nn-tg-problem.clp)

Span failures are indicated by low holding time. The terminating office may have multiples of 24 trunk permanent signals. There may also be referrals to diagnostics. This anomaly was detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly would include looking at data from the trunk reports for the source switch and comparing this data to the normal expected values for holding time. The permanent signal register for the switch should be non-zero as should the referrals to diagnostics register. A test case for this anomaly needs to be identified before the rules can be written so that this anomaly can be differentiated from other anomalies identified by low holding time.

B.2.1.5.17 Trunk Testing (nn-tg-problem.clp)

Trunk testing is indicated by an incoming pegcount at the terminating end with no originations on the other end. This anomaly was detailed in the Network Management Situation Diagnosis Document but was not included in the September 1990 testing. The logic to confirm this anomaly would include looking at data from the trunk reports for the source and destination switches and comparing the source incoming peg count to the destination outgoing attempts. If there are no or very few destination attempts with many incoming counts, then this anomaly would be confirmed.

B.2.1.5.18 Transmission Fault (nn-tg-problem.clp)

Trunk transmission fault is recognized as a signalling problem for the originator. The terminating office will show high peg count (maybe), receiver overflow, permanent signals, and RADR problems. This anomaly was detailed in the Engineering Test Plan, Live Network Test Scenarios but was not included in the September 1990 testing. A test case for this anomaly needs to be identified before the rules can be written so that this anomaly can be differentiated from other anomalies.

B.2.1.6 Anomaly Terminology

The neural net and IWES use different terminology and sometimes different spellings for the same anomaly. The following tables may be helpful to readers who have access to neural net as well as IWES documentation.

SWITCH ANOMALIES

Neural Net Term	IWES Term
congestion_reporting	congested reporting switch
degraded_reporting	degraded reporting switch
outage_remote	switch outage remote
degraded_remote	degraded remote switch
cpumismatch	cpu mismatch
cpuinits	cpu inits
congestion_remote	congested remote switch
rcvr-overflow-mf	mf receiver overflow
rcvr_overflow_mf	" "
facility_hits	facility hits
cpu_simplex	cpu simplex
internal_problem	switch internal problem
cp_origination_problem	switch origination problem
rcvr_problem	receiver problem
exc_rcvr_out_of_service	excess receiver out of service

permanent_signals	permanent signals
perm_signal	" "
excess_overflows	excess overflows
dist_signal_prob	distant switch signalling problems
line_frame_problem	line frame problems
trunk_system_failure	trunk system failure
misc_other	miscellaneous system failure calls
misc_others	" " "

TRUNK GROUP ANOMALIES

Neural Net Term	IWES Term
degraded_md163	degraded trunk
degraded_md164	" "
tg_congested	trunk congestion
tropofade	facility hit
facility_hit	" "
failure	trunk failure
tg_failure	" "
exctrunks_oos	excess trunks out of service
exc_trks_oos	" " " " "
sbutrunks_oos	system busy trunks out of service
system_busy_trks	" " " " " "
system_busy_usage	" " " " " "
mbutrunks_oos	maintenance busy trunks out of service
permanent_seizure	permanent seizure
100%_skip	No usage
no_usage	" "
degraded_facility	trunk facility failure
signalling_prob	trunk signalling problems
signalling_problem	" " "
signalling_glare	trunk signalling glare problems
dpas_failure	dpas failure
span_failure	span failure
continuity_failure	continuity failure
trunk_testing	trunk testing
dtc_failure	dtc problem
transmission_fault	trunk transmission fault

APPENDIX C. IWES Results File Examples

Appendix C is composed of two results file examples, one at 8:10 and the other at 9:00 on September 26, 1990. The first one contains recommendations for a single problem, excessive trunks out of service, detected at 8:10. The second example contains recommendations for permanent seizure on two separate trunk groups at 9:10.

```
Time: 900926.0810
MHL

MHLFRD115 Num: 159 Src: MHL Dest: FRD
  Description:
     Excess Trunks out of service requires the trunk to have less
     than 75 percent of equipped trunks in service
MHLFRD115 Equipped: 24 Inservice: 0 Percent Inservice:  0.0
  Observations:
     08:00:00  MHLFRD115  Low In Service
     08:10:00  MHLFRD115  24 trunks out of service - equipped: 24
          in service: 0
     08:00:00  MHLFRD115  excessive trunks out of service
     08:05:00  MHLFRD115  Switch Report shows Maintenance Busy
  Usage Count > 0
     08:10:00  MHLFRD115  Abnormal High Overflows
     08:10:00  MHLFRD115  Low Holding Time With 100 Percent
  Overflows
     08:10:00  MHLFRD115  Anomaly is exc_trks_oos 1.0
     08:10:00  MHLFRD115  possible trunk group outage
     08:10:00  MHLFRD115  Confirmed more than 25 percent of
          trunks are oos
  Problem:
     MHLFRD115 has excessive trunks out of service
     More than 25 percent trunks out of service on MHLFRD115
  Actions:
     Confirm with switch maintenance crew that MHLFRD115
     at MHL has excessive trunks out of service
     Actions not specified
  Controls:

Time: 900926.0900
TJS
TJSUXB084 Num: 66 Src: TJS Dest: UXB
  Description:
     Permanent seizure or trunks permanently held up but not with
     real traffic is recognized with full usage, and high holding
     time.  The pattern continues with no supporting peg counts.
TJSUXB084 Ins: 0 Connects: 0 Tru: 6 Sbu: 0 Mbu: 0 Inservice: 2
  Ht:  0.0 Norm ht:  2.2 Infails: 0 Percent_Use: 100.0 Norm
Usage: 82.9
```

```
  Observations:
     08:00:00  TJSUXB084  Low In Service
     08:00:00  TJSUXB084  1 trunks out of service - equipped: 3
          in service: 2
     08:00:00  TJSUXB084  excessive trunks out of service
     09:00:00  TJSUXB084  Usage is higher than calls could
          generate
     09:00:00  TJSUXB084  Anomaly is permanent_seizure 0.4
     09:00:00  TJSUXB084  Confirmed permanent seizure

Problem:
TJSUXB084 has excessive trunks out of service
   Permanent seizure on TJSUXB084
  Actions:
     Confirm with switch maintenance crew that TJSUXB084
     at TJS has excessive trunks out of service
     Call switch TJS and any associated technical control
     facilities for confirmation of a permanent seizure on
     TJSUXB084.  If conversation with site personnel indicates
     control actions should be taken, then possible control
     actions are:
     RRTE, CBK or DCC if end office, SKIP if first route to
     office, ARC if quasi-final.
  Controls:
UXB
UXBTJS084 Num: 102 Src: UXB Dest: TJS
  Description:
     Permanent seizure or trunks permanently held up but not with
     real traffic is recognized with full usage, and high holding
     time.  The pattern continues with no supporting peg counts.
UXBTJS084 Ins: 0 Connects: 1 Tru: 6 Sbu: 0 Mbu: 3 Inservice: 3
     Ht:  0.0 Norm ht:  2.3 Infails: 0 Percent_Use: 100.0 Norm
Usage: 84.3
  Observations:
     08:30:00  UXBTJS084  Switch Report shows Maintenance Busy
Usage Count > 0
     09:00:00  UXBTJS084  Usage is higher than calls could
                          generate
     09:00:00  UXBTJS084  Anomaly is permanent_seizure 0.7
     09:00:00  UXBTJS084  Confirmed permanent seizure
  Problem:
     Permanent seizure on UXBTJS084
  Actions:
     Call switch UXB and any associated technical control
     facilities for confirmation of a permanent seizure on
     UXBTJS084.  If conversation with site personnel indicates
     control actions should be taken, then possible control
     actions are:
     RRTE, CBK or DCC if end office, SKIP if first route to
     office, ARC if quasi-final.
  Controls:
```

APPENDIX D. Statistical Data Base Software Design

D.1 Statistical Data Base Input Files

Statistical data base input files are found in the $DSNPATH/expert/stats/db directory. Specifying the runtime flag, statistical data base, in the IWES (NMES) execution line enables the use of the statistical data base. At IWES initialization time the stat values (average and standard deviation for each hour) for all switch and trunk group stats are loaded from the offline ASCII files into a hash table.

In addition to the actual stat files, there are two files in the stats/db directory that provide parameters for statistical data base monitor thresholds and information on trunk groups that run normally with INSERVICE set less than EQUIPPED.

D.1.1 Stat Files

Each file is an ASCII text file containing hourly stat information for a specific stat (type) for one switch or trunk group (item). Since there are up to 24 hours of switch report data available, stat files will normally contain 24 records representing hours 0 through 23. Each stat file record has the following format:

Hour Average #_Of_Samples Minimum Maximum Standard_Deviation.

For each of the 14 reporting switches in the European network, there are 3 stat files: CCB usage per 5 minutes, MF receiver holding time, and Digitone receiver holding time.

There are currently 102 trunk groups connecting the 14 reporting switches with each other and other non-reporting switches. For each of these trunk groups the statistical data base has 6 files: percent overflows, average holding time, incoming calls per circuit per hour, outgoing calls per circuit per hour, total calls per circuit per hour, and percent use of total trunk group capacity. In total, there are 654 (3 * 14 + 6 * 102) stat files in the data base.

The stat file name format describes what data was used to compile the stat values, and what stat type is being computed for what item:

```
stat-<date>+<additional_days>-<item>.<type>.
```

For example, the file "stat-900103+9-ezl.ccb" contains CCB seizure information for the switch EZL. The data was computed from 10 days of archived data beginning 900103. The file "stat-900806+8-tjs052.povfl" contains Percent Overflow data for the trunk group 52 out of TJS. The data was computed from 9 days of archived data beginning 900806.

The format of the statistical data base files allows them to be directly processed by a plotting program, XGRAPH, when displaying the averages over time. For example:

```
     xgraph stat*.ccb
```

will display the CCB usage curves over 24 hours for the 14 switches.

D.1.2 Threshold Factor File

The stat-SD_FACTOR.fact file contains factors for resetting the thresholds the IWES uses to detect deviations from normal network activity. The default thresholds for trunk group stats is 1 standard deviation above or below the average value. The .fact file allows the IW operator to configure offline a different standard deviation factor for each of the 6 trunk group stat types. At IWES initialization time, if a .fact file is in the stat/db directory, the trunk group stat thresholds in the array stat_sd_factors[] will be reset. Each line in the .fact file has one integer and corresponds to one trunk group stat type. The order is implicit: percent overflows (povfl), incoming calls/circuit/hour (icch), calls/circuit/hour (cch), total calls/circuit/hour (tcch), average holding time (aht), and percent trunk group use (puse).

The current .fact file contains 6 records: 2,2,2,3,2,2, which indicates that povfl,icch, cch, aht, and puse thresholds are set at 2 times their standard deviation from average, and that tcch is set at 3 times its standard deviation. Note, that because the statistical data base contains a computed standard deviation and average for every trunk group for every hour of the day, each threshold check is specific for the trunk group and time of day.

D.1.3 Inservice File

The stat-<dates>-TGS.insrv contains trunk groups that normally run with a low inservice count. The IWES uses this information at runtime to inhibit messages warning of low inservice trunk groups to the operator. Currently there is no .insrv file because all trunk groups were running normally at their equipped level when the system was installed in September 1990.

The format of each line is:

```
<local_switch> <tg_#> <CLLI> <normal_low_inservice_#>
<destination_switch>
```

An example is:
ezl 90 ABE001 0 abe

D.2 Statistical Data Base Enhancements

Because the statistical data base is a recent, experimental enhancement to the ES, it is expected that there will be many desirable extensions and modifications. Known desirable new features and possible ones requiring more analysis are listed in the following two subsections.

D.2.1 Desirable New Features

Switch Parameter Threshold Checking

Currently only trunk group stat parameters are constantly monitored for abnormal values given the time of day. Monitors for the three switch stats should be added to detect when receiver holding times and CCB counts are abnormal.

Interpolating Stat Data

The statistical data base contains hourly averages. Since each hourly average is computed from switch reports from that hour, the average is, in fact, for the midpoint of the sampled time: 30 minutes past the hour. When the ES monitors and anomaly verifiers compare the value from a specific switch report with the expected value from the data base, they simply use the current hour of the switch report to retrieve a statistical data base average. Thus both the 7:00 and 7:55 switch report will be compared with the 7:30 hourly average.

Since traffic volume can double or triple in one hour (for example, work start time and after lunch), it is apparent that one average for 60 minutes does not provide very fine threshold checking during those volatile times.

The quickest solution to this problem is to (linearly) interpolate stat values to obtain good approximations between the hours. Thus, for example, a better 8:00 average can be obtained by averaging the 7:30 and 8:30 values. Data analysis comparing interpolated hourly averages with 15 minute averages shows that this solution is quite adequate.

It is important to remember that during the volatile periods the standard deviation also increases. Since ES monitor thresholds are based on both the average and the standard deviation values, any increase in error of the average value caused by rapidly changing traffic levels is offset by the increase in allowable range by the larger standard deviation.

Integrate Stat Graphics Into The IWUI

The type of display produced by the graph.com utility should be integrated into the workstation's user display options. The "trending" function, for example, currently displays only tabular data. A graphical display showing recent switch report values compared with normal values for those times would convey the needed information more efficiently. Similarly, options to display any trunk group or switch data graphically with comparisons to normal data should be provided. Normal values could be displayed from interpolated hourly averages, whereas current or archived data should be displayed using 15 minute averages since the operator will be interested in seeing timely averages instead of "smoothed" ones. Averages based on less than 15 minutes are not recommended since they are usually too noisy to be useful.

Add New Stats

Receiver seizure counts (MF and DIGITONE) are often important indicators of problems. Since they vary enormously from switch to switch and over time of day, they should be added to the statistical data base.

Calls (receiver seizures) to CCB seizures is also often a good measure on how well a switch is doing. It provides a way of checking call counts and CCB use with one index. Note, that if the statistical data base already contains averages for receiver and CCB seizures, than it does not need to compile and store the calls to CCB ratio explicitly in the data base.

Instead, it need only compute it on-the-fly when needed.

Recompute Holding Times

Currently the holding time is assumed 0 if there are no call counts to divide by. A suggested improvement based on P. McClellan's (GTE) DAI code, is to assume calls based on the maximum usage each call could provide to the usage count in the switch report. For example, if usage is 10 CCs for a 12 minute switch report and the call count was 0, assume there were in fact 4 calls: 3 each using the maximum 3 CCs per call, and 1 using the remaining CC. See DAI code or iwui/calculate_thresh_stats.c for more details.

Five minute holding time calculations are very noisy because the calculation is based on 100 second usage scans and call counts that may have been counted in the previous switch report. The stats_db program can be improved by delaying the actual calculation of usage/calls to the end of the averaging period. Instead of averaging the noisy 5 minute ratio of each switch report, simply keep running totals of usage and call counts for the period, and then divide at the end of the period to obtain the average hold time.

Another possible way to reduce the noisy holding times in the database is to replace average holding time in the statistical data base with usage. The monitors or statistical data base interface could then compute average holding time on-the-fly by dividing average usage by average total calls.

Use 15 to 20 Minute Averages When Monitoring Live Switch Reports

Analysis shows that call counts and holding times are very noisy when computed from 5 minute data. The default displays for operators of live data should all be based on 15 minute or more averages. Similarly, all threshold comparisons should be based on 15 minute averages of live data to prevent frequent false positives.

Filter Abnormal Highs And Lows From Data Base

The statistical data base averages are based on approximately 10 days of available, recent switch reports. The switch reports may contain data transmission errors (i.e., garbage) or abnormal values caused by actual network anomalies (switch and trunk group failures). Using the utility graph.avg, one can see an occasional maximum and minimum value that does not fit the curve. The stat_db.awk program should have sanity checks and specific filters for each switch and trunk group.

Another option to consider is using the "mode" average instead of the "mean" as a way to eliminate the abnormal values from the computed average.

Fine Tune Thresholds

Since the IWES in the current system is serving primarily to provide verification of (NN) suspected anomalies, rather than to detect anomalies, its trunk group thresholds are set conservatively at 2 standard deviations above/below average.

Assuming that the previously described enhancements: 15 minute real time averages, stat value interpolation, and filtering of abnormal data are done, then the monitor thresholds can and should be tightened up.

D.2.2 Possible New Features Requiring More Analysis

Multiple Statistical Data Bases Based On Weekly, Seasonal, And Holiday Variations

The current statistical data base assumes there is a normal workday traffic load and is thus based on averaging Monday

-Friday switch reports. The 1 weekend set of data we have seen indicates that traffic levels are much lower then. Holiday traffic patterns have not been examined.

Questions to be asked are:

- What problems are unique to weekends and holidays, and how are they handled?
- Are there known seasonal changes?
- Are there consistent weekly patterns? For example, does Friday traffic trail off sooner because of the upcoming weekend? If so, are these variations significant enough to require a different set of norms?

Multiple statistical data bases, each appropriate for a particular set of days, can be easily accommodated. The operator or a smart program need only move or link the appropriate statistical data base directory into the dsn/expert directory prior to (re)starting the IWES.

Self-adjusting statistical data base

If trunk group and switch traffic levels continually change slowly but significantly over time,, then a self-adjusting statistical data base may be useful. For example, new daily averages could be averaged into the statistical data base averages to produce a new statistical data base. Adjustable weight factors would control how fast the new averages affect the old average.

Integrate statistical data base into the IWDB

By integrating the statistical data base stats into the IWDB, it will be much easier for other IW functions to access the data. For example, the graphical displays comparing current values with expected could be called from the user interface as an option. An open issue is whether it is more efficient to recompute daily computed stats from the raw switch reports when displaying them, or to compute the stats once and save them in the IWDB.

Hierarchical Stats

Currently the IW monitors and threshold checks only individual trunk groups and switches. Nothing is reported about the network as a whole, nor is there any information about groups of related trunk groups such as links.

Using the statistical data base it is very easy to establish expected traffic curves for the entire network of reporting switches and their neighbors by totaling all CCBs, receiver seizures, calls, etc. for each time period. Similarly, when running, the IW could also total the individual switch and trunk group counts each reporting period. New network level monitors and display icons would provide the operator with information about the overall state of the network which could be useful in diagnosing or remedying a problem. For example, if there is congestion at one switch but network monitors indicate that overall net traffic is normal, than the operator can feel more confident that a reroute solution will help rather than worsen the problem.

Besides, network level data, composite data on related trunk group could be useful. Monitoring traffic totals for all trunk groups on 1 link could be useful in diagnosing a link problem. Similarly, if trunk-group-to-physical-circuit information is available, monitoring totals for all trunk groups on 1 physical circuit could help diagnose a physical circuit problem.

When detecting a degraded or down remote (non-reporting) switch, the ES already examines individually all trunk groups that connect to the switch. If traffic stat totals are available for remote switches based on the links connecting them, then the IW could provide additional monitors and corroborating information for those remote switches.

D.3 Statistical Data Base Utility Files

Utilities used to create and graphically display data in the statistical data base are found in the directory $DSNPATH/expert/stats/tool.

D.3.1 Gawk/AWK

The program which creates the stat files, stats_db.awk is an AWK script. Gawk is the GNU Project's implementation of the AWK programming language. It conforms to the definition and description of the language in The AWK Programming Language, by Aho, Kernighan, and Weinberger, with the additional features defined in the System V Release 4 version of UNIX AWK, and some GNU-specific extensions. The standard SUN/UNIX AWK does NOT support all the features used in the stats_db.awk script. The current Gawk binary runs only on SUN3 hardware, but source code is readily available.

D.3.2 90aug.xref

Since the switch reports do not contain the remote node id for the trunk groups, the input file "90aug.xref" is used by the stats_db.awk program to identify which trunk group numbers are interesting, i.e., connect to known switches. The name of the file indicates it was created from the 1990 August set of NMSS xref tables. When interesting trunk group numbers change, then this file should be updated. Its format is:

```
<node_tg> <remote_node> <clli> <tg_type>.
```

Only the first two fields are used by the stats_db.awk program.

D.3.3 Stats_db.awk

Stats_db.awk is an (G)AWK script that takes a set of switch reports for 1 node and produces a corresponding set of stat files. For example:

```
 gawk -f stats_db.awk $DSNPATH/expert/stats/tools/90aug.xref \
              $IWDATA/9009??/switch/*.tjs
```

will process all archived switch reports in September, 1990 for the switch TJS. The output will be 3 switch stat files (ccb, rdg, rmf), and 6 trunk group stat files (povfl, puse, icch, cch, tcch, aht) for each TJS trunk group that connects to a known switch. The 90aug.xref file will be read to identify the interesting trunk groups.

A "TG=" option is available that restricts the program to processing only the specified trunk group instead of all interesting trunk groups for the particular switch. By *specifying* "TG=999" (or any other unknown trunk group number), the program will only produce the 3 switch stat files.

The program currently only processes 1 switch at a time because of the SHELL limitation on the number and size of parameters. Processing 10 days of 5 minute switch reports at one time for all 14 switches will usually exceed the limit. I believe that the program itself has no limitation on the number of switches it processes at once, and did process all of them at one time in earlier versions when there were only 4 switch reports per hour. However the program has evolved considerably since, and the current version has not been tested with multiple switches.

D.3.4 Xgraph

Xgraph is a public domain program that draws a graph on an X display given data read from either data files or from standard input if no files are specified. It can display up to 64 independent data sets using different colors and/or line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels, and a legend. There are options to control the appearance of most components of the graph. See the man pages for more information.

Xgraph is useful for displaying information in the offline ASCII statistical data base files, or other files created by stats_db.awk. The current bin file runs on SUN3 hardware but the complete source is available in the xgraph subdirectory in stat/tools/.

D.3.5 Xgraphrc

The file xgraphrc is used to reset xgraph color and line choices. By aliasing xgraph to

```
 'xgraph <xgraph options> \!*
$DSNPATH/expert/stats/tools/xgraphrc'
```

xgraph will use the colors and line styles you have specified in the xgraphrc file. See the xgraph man pages for more information on the options.

D.3.6 Graph.avg <item> <stat_type>

Graph.avg is a shell script that given an item name and stat type, will display graphically their average, minimum, and maximum values versus time of day. For example:

```
     graph.avg tjs ccb
```

will display average CCB ranges for switch TJS. The script extracts the average, minimum, and maximum values from the appropriate statistical data base file, adds identifier strings, and then calls xgraph to display it.

D.3.7 Graph.com <item> <stat_type> <date>

Graph.com will graphically compare archived data and statistical data base averages. The user supplies the item name, stat type, and date to compare. The shell script program will then compute on the fly using stats_db.awk the averages seen each hour on the specified day, and then using xgraph display the normal averages from the statistical data base with the archived data averages. For example:

```
     graph.com tjs057 aht 900927
```

will graphically compare normal (statistical data base) holding time averages for trunk group 57 out of TJS with the values seen on September 27,1990.

D.4 Code

Code supporting the stat data base is found in 4 function (.c) files and 2 header (.h) files.

D.4.1 Header Files

stats_if.h	defines all items needed to interface to the runtime stat data base. Normally, an IWES module would only need to include this file in order to use stat data.
stats.h	defines the global DEFs and data structures used internally by the stat data base.

D.4.2 Function Files

nmes_reset.c	if the statistical data base flag is set, loads the stat data base files into a hash table.
stat_processor.c	provides new trunk group IWES monitors based on the stat data base values. Each time a trunk group report is read, overflows, holding time, trunk capacity usage, incoming, outgoing, and total traffic is checked to see if they are within the normal range for that trunk group at the current hour of the day.
rec_description.c	provides corroborating switch and trunk group information from the stat_data base when an anomaly is detected.
stats.c	contains the stat support functions for loading the data base into the hash table, and reading stat data at runtime.

APPENDIX E. CCSIM Network Management Controls

Three types of controls are implemented in CCSIM. They are listed in the following table. Pre-route controls are applied at a switch before an attempt is made to route a call out of the switch. Pre-hunt controls are applied before an attempt is made to find a free trunk in the trunk group to which the control is applied. Post-hunt controls are applied after the call has overflowed the trunk group. Post-route controls are applied after a call has failed to find a route out of the switch. Within each group the controls are listed in the order in which they are applied. The term 'DMS' in the right-hand column indicates that the control is implemented in accordance with Northern Telecom Practices for DMS switches. The term '490L' after the DRZ control indicates that this is the directionalize control as implemented in the AUTOVON 490L switches. The term 'EXP' after the CAN-EOC-OVF indicates a control introduced for experimental purposes that does not correspond to a control to be found in any real switch. A blank entry in the third column indicates that the control is implemented according to our interpretation of the DCA generic switch specification.

Pre-route Controls:		
1. Code Block	(CB)	DMS
2. Call Gap	(GAP)	
3. Alternate Route Cancel (B)	(ARC-B)	
4. Destination Code Cancellation	(DCC)	DMS
Pre-hunt Controls:		
1. Alternate Route Cancel (A)	(ARC-A)	
2. Directional Reservation of Equipment	(DRE)	DMS
3. Protective Reservation of Equipment	(PRE)	DMS
4. Directionalization	(DRZ)	490L
5. Cancel To (Percent)	(CANTP)	DMS
6. Cancel To (Rate)	(CANT-RATE)	
7. Skip	(SK)	DMS
Post-hunt Control:		
1. Cancel From	(CANF)	DMS
Post-route Control:		
1. Cancel End-of-Chain Overflows	(CAN-EOC-OVF)	EXP

CCSIM does not have different commands for applying and removing NM controls as do real switches. Instead, it uses the same command both to apply and remove a control. Removal occurs when a particular parameter (often a percent) is set to a particular value (zero for the percent parameter). In the following descriptions of the individual NM controls, the percent-equal-to-zero-for-removal convention is assumed for all controls having a percent parameter. For other controls, the parameter values for removal are specified explicitly.

Controls that cancel calls have parameters specifying the kind of announcement message to which a canceled call should be connected. There are three such announcements. In CCSIM, specifying Emergency Announcement 1 (EA1) or Emergency Announcement 2 (EA2) will cause an affected call to be counted as failed and not to be retried. The No Circuit Available (NCA) announcement causes the call to be handled just as it would have been if it had blocked due to the lack of a trunk out of the switch at which the control was applied. In that case, CCSIM will retry the call subject to the retry parameters applicable to the simulation run.

CCSIM does not have tables to translate between codes (telephone numbers) and switch (node) names. Consequently, controls such as Code Block (CB) use destination node names instead of codes to specify the calls that are to be blocked. In this respect, such controls differ in syntax from those in real switches, but the effect on traffic is the same as would occur if all office codes at a switch were specified in a real network application of a code control.

In the following descriptions the word 'ALL' is permissible as a value for some parameters. When used, its affect is the same as would be achieved by issuing a sequence of controls, one for each of the allowable values of the parameter. For example, it allows a CB control to be applied at all switches with a single command. There is no corresponding capability in a real telephone network.

Pre-Route Controls:

CB - Code Block

The CB control is put on at a switch and applies to originating calls only. It blocks a specified percentage of the traffic to a destination switch from entering the network. When the control is applied at less than 100%, only routine calls are affected. At 100%, CB blocks calls of all precedences. Blocked calls are handled according to the announcement type specified in the control.

In a real network, the CB control would apply to codes and could be used to block calls to individual telephone numbers. In CCSIM it can be used only for all the codes identified with a particular switch.

Usage: CB node1 node2 percent ann
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is the three-letter node name for the destination switch to which calls are to be blocked (or ALL)
percent is the percentage (0-100) of calls to block
ann is the announcement type (NCA, EA1 or EA2)

Example: CB ALL UXB 50 EA2
Cancels 50% of the routine calls from all nodes to Uxbridge. The canceled calls will not be retried.

GAP - Call Gap

The GAP control is put on at a switch and applies to originating calls only. It determines the rate at which traffic to a particular destination switch is allowed to enter the network. After an attempt to route a call to the specified destination has been allowed by the GAP control, subsequent calls to that destination are blocked for a period of time designated as the "gap interval." After the expiration of the gap interval, the next call to that destination will be allowed to attempt to find a route. The gap interval is chosen from the interval set 0 (no-control), 0.10, 0.25, 0.50, 1, 2, 5, 10, 15, 30, 60, 120, 300, 600 seconds, and infinity. An infinite interval prohibits all attempts. Blocked calls are handled according to the announcement type specified in the control. At intervals 0 through 600, only routine calls are affected. At the infinite interval, calls of all precedences are affected.

In a real network, the GAP control would apply to codes and could be used to gap calls to individual telephone numbers. In CCSIM it can be used only for all the codes identified with a particular switch.

Usage: GAP node1 node2 index ann
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is the three-letter node name for the destination switch to which calls are to be gapped
index is a pointer into the following table of gap intervals (1 removes the control)
ann is the announcement type (NCA, EA1 or EA2)

Index (Seconds/Call)	Gap Interval	Calls per Minute
1	0	All
2	0.1	600
3	0.2	240
4	0.50	120
5	1	60
6	2	30
7	5	12
8	10	6
9	15	4
10	30	2
11	60	1
12	120	1/2
13	300	1/5
14	600	1/10
15	Infinity	None

Example: GAP ALL UXB 11 NCA
Allows only one call per minute to enter the network from each of the other switches. Blocked calls are allowed to retry.

ARC-B - Alternate Route Cancellation (Type B)

When the ARC-B control is put on at a switch, it applies to both tandem calls and originating calls. All calls to the specified final destination are allowed to use only the direct route out of the switch. A call for which a free or preemptible trunk cannot be found on the primary route will be blocked. The control can be specified to apply to either routine-only or all-precedence traffic.

Usage: ARC-B node1 node2 precedence
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is a three letter node name for the final destination switch for the call (or ALL)
precedence is either R (routine-only), AP (all-precedences) or NONE (remove the control)

Example: ARC-B TJS UXB R
Deny alternate routes for routine traffic between Torrejon and Uxbridge

DCC - Destination Code Cancellation

The DCC control is put on at a switch and applies to all traffic, both originating and tandem calls. It blocks a specified percentage of the traffic to a destination switch and all end offices reachable only from that switch from entering the network. When the control is applied at less than 100%, only routine calls are affected. At 100%, DCC blocks calls of all precedences. Blocked calls are handled according to the announcement type specified in the control.

```
    Usage:   DCC  node1  node2  percent  ann
             node1 is the three-letter node name for the switch
              at which the control is to be applied (or ALL)
             node2 is the three-letter node name for the
              destination switch to which calls are to be
              blocked (or ALL)
             percent is the percentage (0-100) of calls to block
             ann is the announcement type (NCA, EA1 or EA2)

  Example:   DCC  ALL  UXB  50  EA2
             Cancels 50% of the routine calls from all nodes to
              Uxbridge and any end offices reachable only
              through Uxbridge. The canceled calls will not be
              retried.
```

Pre-Hunt Controls:

ARC-A - Alternate Route Cancellation (Type A)

The ARC-A control is applied to a link, i.e., one or more trunk groups between a pair of switches. It causes traffic which would normally use the link as an alternate route to skip to the next route, if any, in the routing table. Its effect is to give preference to traffic that would use the link as a direct route. It can be applied to affect routine-only or all-precedence traffic. Calls affected by these controls are not counted as attempts on the link.

```
    Usage:   ARC-A  node1  node2  precedence
             node1 is the three-letter node name for the switch
              at which the control is to be applied (or ALL)
             node2 is a three letter node name of the switch at
              the remote end of the link to which the control is
              to be applied
             precedence is either R (routine-only), AP (all-
              precedences) or NONE (remove the control)

  Example:   ARC-A  TJS  UXB  AP
             Allow only direct routed traffic to access the link
              between Torrejon and Uxbridge at Torrejon
```

DRE - Directional Reservation of Equipment

The DRE control is applied at a switch to a trunk group. It gives priority to incoming traffic by reserving a number of idle trunks in the group. When the number of idle trunks is equal to or less than the number of reserved trunks, all traffic (direct- and alternate-routed) is skip-routed.

Calls of all precedences are affected by the DRE control. Calls affected by these controls are not counted as attempts on the link.

```
    Usage:   DRE  node1  clli  reserve
             node1 is the three-letter node name for the switch
              at which the control is to be applied
             clli is the clli name of the trunk group to which
              the control is to be applied
             reserve is the number of trunks to be reserved
              (reserve = zero removes the control)

  Example:   DRE  UXB  UXBTJS084   1
             UXB must have more than one free trunk in the group
              'UXBTJS084' before it can use that group for
              routing a call of any precedence via Torrejon.
```

PRE - Protective Reservation of Equipment

The PRE control is applied at a switch to a trunk group. It gives priority to incoming traffic by reserving a number of idle trunks in the group. When the number of idle trunks is equal to or less than the number of reserved trunks, all alternate-routed traffic is skip-routed. Calls of all precedences are affected by the DRE control. Calls affected by this control are not counted as attempts on the link.

```
    Usage:   PRE  node1  clli  reserve
             node1 is the three-letter node name for the switch
              at which the control is to be applied
             clli is the clli name of the trunk group to which
              the control is to be applied
             reserve is the number of trunks to be reserved
              (reserve = zero removes the control)

  Example:   PRE  UXB  UXBTJS084   1
             UXB must have more than one free trunk in the group
              'UXBTJS084' before it can use that group as an
              alternate route for a call of any precedence.
```

DRZ - Directionalization

The DRZ control is applied at a switch to a trunk group. It places a limit on the number of trunks which may be used for outgoing calls on the trunk group. The limit ranges from 1 to the maximum capacity of the trunk group. Setting the limit to zero removes the control. Traffic (direct and alternate-routed) which would exceed the limit is skip-routed.

Calls of all precedences are affected by the DRZ control. Calls affected by these controls are not counted as attempts on the link.

Usage: DRZ node1 clli limit
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name of the trunk group to which the control is to be applied
limit is the number of trunks that may be used by outgoing calls

Example: DRZ DVN DVNTJS080 3
Donnersburg may use no more than 3 trunks in the group 'DVNTJS080' for outgoing calls to be routed via Torrejon.

CANTP - Cancel To (Percent)

This control is applied to a trunk group. It cancels percentages of the direct- and alternate-routed routine traffic offered to the group. Calls affected by this control are not counted as attempts on the link. Canceled calls are handled according to the announcement type specified in the control.

Usage: CANTP node1 clli percent1 percent2 ann
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name for the trunk group to which the control is to be applied
percent1 is the percentage (0-100) of direct-routed calls to cancel
percent2 is the percentage (0-100) of alternate-routed
calls to cancel
ann is the announcement type (NCA, EA1 or EA2)

Example: CANTP UXB UXBTJS084 0 80 EA1
Cancels 80% of the routine alternate-routed calls offered to the trunk group 'UXBTJS084' from Uxbridge to Torrejon. The canceled calls will not be retried. Direct-routed calls offered to the trunk group are not affected.

CANT-DIRECT-RATE/CANT-ALTER-RATE - Cancel To (Rate)

These controls are applied to a trunk group. They control the rate at which calls are allowed to access the group. After one call is allowed access to the group, subsequent calls that otherwise attempt to use the group are canceled until a period of time (the 'gap interval') has elapsed. The first call to arrive after the gap interval will escape cancellation and be allowed to access the group. The gap interval is chosen from the interval set 0 (no-control), 0.10, 0.25, 0.50, 1, 2, 5, 10, 15, 30, 60, 120, 300, 600 seconds and infinity. An infinite interval cancels all calls attempting to access the group. CANT-DIRECT-RATE affects only direct routed traffic. CANT-ALTER-RATE affects only alternate routed traffic. When the interval is less than infinity, only routine calls are canceled. An infinite interval cancels calls of all precedences.

Usage: CANT-DIRECT-RATE nodel clli index
or CANT-ALTER-RATE nodel clli index
nodel is the three-letter node name for the switch at which the control is to be applied
clli is the clli name of the trunk group to which the control is to be applied
index is a pointer into the following table of gap intervals (1 removes the control
ann is the announcement type (NCA, EA1 or EA2)

Index	Gap Interval (Seconds/Call)	Calls per Minute
1	0	All
2	0.1	600
3	0.2	240
4	0.50	120
5	1	60
6	2	30
7	5	12
8	10	6
9	15	4
10	30	2
11	60	1
12	120	1/2
13	300	1/5
14	600	1/10
15	Infinity	None

Example: CANT-DIRECT-RATE TJS TJSUXB084 10 EA2
Allows at most 2 direct-routed routine calls per minute to search trunk group 'TJSUXB084' for a trunk to Uxbridge from Torrejon. All other direct- routed, routine traffic that would otherwise search the trunk group will be canceled and not retried.

SK-DIRECT/SK-ALTER - Skip

The SKIP control is applied to a trunk group. It skip-routes traffic to the next trunk group in the routing chain. SK- DIRECT affects only direct routed calls. SK-ALTER affects only alternate routed calls. Calls of all precedences are affected by the SKIP control. Traffic that is skip-routed is not counted in the statistics as attempts on the link.

```
    Usage:  SK-DIRECT  node1  clli  percent
     or     SK-ALTER  node1  clli  percent
            node1 is the three-letter node name for the switch
             at which the control is to be applied
            clli is the clli name for the trunk group to which
             the control is to be applied
            percent is the percentage (0-100) of calls to be
             skip-routed

  Example:  SK-DIRECT  UXB  UXBTJS084  70
            Skip-routes 70 percent of the direct-routed calls
             that attempt to use trunk group 'UXBTJS084' from
             Uxbridge to Torrejon.
```

Post-Hunt Controls:

CANF - Cancel From

The CANF control is applied to a trunk group. It cancels specified percentages of the direct- and alternate-routed traffic overflowing from the group and prevents it from continuing to the next group in the routing chain. Canceled calls are handled according to their announcement type. Only routine calls are affected by this control. Calls affected by the CANF control are counted as attempts on the link.

```
    Usage:   CANF  node1  clli  percent1  percent2  ann
            node1 is the three-letter node name for the switch
             at which the control is to be applied
            clli is the clli name of the trunk group to which
             the control is to be applied
            percent1 is the percentage (0-100) of direct-routed
             calls to cancel
            percent2 is the percentage (0-100) of
             alternate-routed
            calls to cancel
             ann is the announcement type (NCA, EA1 or EA2)

  Example:  CANF  UXB  UXBTJS084  10  100  EA2
            Cancels 10% of the direct-routed and 100% of the
            alternate-routed traffic that overflows trunk group
            'UXBTJS084' from Uxbridge to Torrejon.  The
            canceled calls will not be retried.
```

Post-Route Controls:

CAN-EOC-OVF - Cancel End-of-Chain Overflows

The CAN-EOC-OVF control is applied at a switch. It cancels a percentage of routine calls that overflow the last trunk group in the routing chain for a specified destination switch. The canceled calls are not retried. They are counted as attempts and overflows for all trunk groups in the chain, and they are counted in a special statistic associated with the control. This control was added to CCSIM for experimental purposes. It is not found in real switches.

```
  Usage:  CAN-EOC-OVF  node1  node2  percent
          node1 is the three-letter node name for the switch
           at which the control is to be applied (or ALL)
          node2 is the three-letter node name for the
           destination switch for which overflowing calls are
           to be canceled (or ALL)
          percent is the percentage (0-100) of calls to
           cancel

Example:  CAN-EOC-OVF  TJS  ALL  50
          Cancels 50% of the routine calls at Torrejon that
           fail to find a route out of the switch, no matter
           what their destination.  The canceled calls will
           not be retried.
```

APPENDIX F. CCSIM Switch Reports

CCSIM generates two different kinds of switch reports; a full report and an IWDB report. The full report contains information known to CCSIM about such things as call handling in the switches and precedence call occupancy of trunks that is not contained in the NTI DMS switch Operational Measurement (OM) reports gathered by the IW. The full report is generated by CCSIM as an array of integers and passed in that form to the CCSIM graphics interface and to earlier versions of the expert system (NMES), but not to IWES. CCSIM commands are available to cause the full reports to be saved in ASCII files in either a compact or verbose format. The verbose format includes labels for the report lines, and is intended for a human reader. It also has a summary of total trunk activity printed at the end of each switch report. The compact format has just the report data with minimal labeling of switch names, trunk identifiers, and report times. The saved full reports are large (ten printed pages for a report from UXB for one reporting period). The commands allow the experimenter to select switches and report times to help keep the quantity of data manageable. The files accumulate the reports for all switches requested and all report times.

The IWDB reports are also written to ASCII files, but there is an individual file generated for each switch and each report time. The reports are smaller than the full reports because they contain lessinformation and are not formatted for easy reading. An IWDB-defined algorithm for generating file names is followed so that the files can be readily accessed using IW conventions. The values in the IWDB reports are calculated using values from the full reports as described below.

In the following sections we first describe the full report and then the IWDB report. Both reports are made up of a switch part that has information about the switch itself and a trunk group part for each trunk group at the switch. In the real network situation, the switch polls can specify which trunk groups are to be reported and some requested reports may be lost due to communication line problems, but in CCSIM, reports always have trunk group parts for all groups. The presentation makes use of line numbers using the notation 'Sn' for line 'n' of the switch part and 'Tn' for line 'n' of the trunk group part of the full report. Since lines can be missing from the IWDB report, line numbers there have no significance. The lines in that report are identified by OM report names and numbers.

F1.0 The Full Switch Report

F1.1 The Switch Part

The switch part is made up of a header line, ten lines of call statistics, and ten lines of simulated OM reports. The call statistics lines each have peg counts for all of the five precedence levels, with the lowest precedence (routine) counts first. All peg counts are cleared at the start of the report period. The report thus presents the number of the tallied events that occurred during the report interval. CCSIM generates reports every five minutes of simulated time, but after a switch has been restored following an outage, the first report may have counts for a shorter time interval.

In describing the simulated OM reports from the DMS switches, we make use of terminology taken from NTI Practice 297-1001-114 to identify the fields of the OM reports. When there is no notation to the contrary, it may be assumed that a field of an OM report is calculated by CCSIM as a part of the simulation. Otherwise a notation will indicate when CCSIM uses its random number generator to approximate a field or defaults it to zero.

The individual lines of the switch part are:

S1. Header. This line has the three-letter code for the reporting switch name followed by the simulation time of the report in 'hh:mm:ss' notation.

S2. Originating Calls. This line has peg counts for call attempts that originated at this switch. They include calls for which this switch is also the destination (local calls).

S3. Tandem Calls. This line has peg counts for calls coming into the switch from other switches in the network and which are destined for other switches.

S4. Outgoing Calls. This line has peg counts for calls that attempt to enter the network at the switch. It includes all calls in S3 plus the calls counted in S2 that are not local calls.

S5. Originating-Outgoing Calls NC. This line counts calls originating at the switch that blocked on attempting to enter the network. ('NC' stands for 'incomplete'.)

S6. Tandem Calls NC. This line counts calls that blocked at the switch on attempting to re-enter the network.

S7. Total Line Busy. This line counts calls that reached busy destinations at the switch. Such calls are treated as successful in other counts.

S8. Incoming Calls. This line counts all calls coming into the switch from the network. The counts are the sum of tandem (S3) and terminating (S9) calls.

S9. Terminating Calls. This line counts calls that terminate at the switch. The counts are the sum of incoming (S8) less tandem (S3) plus local (part of S2) calls.

S10. Code Cancel Calls. This line counts calls that are deflected by the CB, GAP, and DCC network management controls. Such calls are not counted as incomplete in S5 or S6.

S11. ARC Calls. This line counts calls that could not be routed out of the switch because of action of the ARC-B control. Such calls are not counted as incomplete in S5 or S6.

S12. CP. This line corresponds to the Call Processing (CP) OM report from the DMS switches. It has the following fields:

CCBSZ - Count of number of times a Call Condense Block (CCB) was seized to process a call.
CCBOVFL - Count of calls lost because there were no free CCBs.
CPTRAP - Count of failures due to illegal software conditions in the switch. (Not simulated, default to zero.)
CPSUIC - Count of failures due to 'unexpected results detected'. (Not simulated, default to zero.)
INITDENY - Estimate of call originations denied during warm or cold restarts. (Not simulated, default to zero.)

S13. CPU. This line corresponds to the CPU OM report from the DMS switches. It has the following fields:

MTCHINIT - Count of mismatch interrupts due to inter-processor differences between the two processors in the switch. (Not simulated, default to zero.)
CPUFLT - Count of times a CPU or other resource was system-busied following a mismatch or trap interrupt. (Not simulated, default to zero.)
SYSWINIT - Warm restart. (Not simulated, default to zero.)
SYSCINIT - Cold restart. (Not simulated, default to zero.)
SYNCLOSS - Count of times processor complex was made simplex following an interrupt. (Not simulated, default to zero.)
MSYLOSSU - Usage time in simplex mode due to manual action. (Not simulated, default to zero.)
SSYLOSSU - Usage time in simplex mode due to system action. (Not simulated, default to zero.)

S14. CMC. This line corresponds to the Central Message Controller

(CMC) OM report from the DMS switches. The report has two fields that show counts of the number of times the system called diagnostics for the CMC. Both fields are set to zero by CCSIM.

S15 and S16. RCVR. These two lines correspond to the Receiver (RCVR)

OM report from the DMS switches. This OM report shows activity relating to both MF (S15) and Digitone (S16) receivers in the switch. Each line has the following fields:

RCVSZRS - Count of number of times a receiver was assigned in response to a request (including tests tallied in the RADR report (S17).
RCVOVFL - Count of number of times no receiver was available.
RCVTRU - Receiver usage (accumulated count of number of receivers busy with call processing when scanned at 0-second intervals.
RCVSBU - Receiver usage because of system action. (Not simulated, default to zero.)
RCVMBU - Receiver usage because of manual action. (Not simulated, default to zero.)

S17. RADR. This line corresponds to the Receiver Attachment Delay Recorder (RADR) OM report from the DMS switches. It has three fields for each of the two receiver types (MF first). They are:

RADTESTC - Count of test attempts.
RADLDLYP - Count of test attempts that did not get a receiver assigned within a lower delay threshold. (Not simulated, default to zero.)
RADUDLYP - Count of test attempts that did not get a receiver assigned within an upper delay threshold. (Not simulated, default to zero.)

S18. TCM. This line corresponds to the TRMTCM OM report from the DMS switches. It has the following fields:

TCMPDIL - Partial digit calls. Not enough digits received for translation. CCSIM generates a random value with a mean of 'mtcmpdil' percent of the total offered call count. The total offered calls are the sum of S2 and S8. 'mtcmpdil' is settable by the experimenter in the 'net.inval' file.
TCMPSIG - Permanent signal calls. Receiver seizure occurred, but no digits were received. CCSIM generates a random value with a mean of 'mtcmpsig' percent of

the total offered call count. 'mtcmpsig' is settable by the experimenter in the 'net.inval' file.

TCMPVACT - Vacant code calls. Translation occurred, but no matching pattern was found. CCSIM generates a random value with a mean of 'mtcmvact' percent of the total offered call count. 'mtcmvact' is settable by the experimenter in the 'net.inval' file.

S19. TER. This line corresponds to the TRMTER OM report from the DMS switches. It has the following fields:

TERSYFL - Miscellaneous system failure calls. CCSIM generates a random value with a mean of 'mtersyfl' percent of the total offered call count. 'mtersyfl' is settable by the experimenter in the 'net.inval' file.

TERRODR - Reorder calls. Causes include mutilated digits, excess digits, and others. CCSIM generates a random value with a mean of 'mterrodr' percent of the total offered call count. 'mterrodr' is settable by the experimenter in the 'net.inval' file.

S20. TRS. This line corresponds to the TRMTRS OM report from the DMS switches. It has the following fields:

TRSEMR1 - Count of calls receiving EA1 treatment as a result of network management control action.

TRSEMR2 - Count of calls receiving EA2 treatment as a result of network management control action.

TRSNCRT - Count of calls receiving NCA treatment as a result of network management control action.

TRSGNCT - General No Circuit. A count of calls which block because no circuit is available.

S21. DCM. This line corresponds to the DCM OM report from the DMS switches. It has two fields that record the number of times digital trunks were referred to diagnostics. Both fields are set to zero by CCSIM.

F1.2 The Trunk Group Part

The lines in the trunk group part of the full report are ordered in a fashion similar to that of the fields of the trunk group report from the DMS switches. However, the meaning of the numbers on each line does not always match that of the corresponding field. Section F2.0 describes the algorithms used to create the DMS fields from the lines of the full report. Also, it should be noted that the lines of the trunk group part of the full report (except for T1) have fields for each of the five precedence levels.

A trunk group part is generated for each trunk group at a switch. It represents activity seen on the trunk group by the switch.

The switch at the other end of the group may also generate a report for the group. Numbers in the two reports for the same group will not always match due to differences in report times, numbers of trunks in service at the two ends, level of damage, etc.

The individual lines of the trunk group part are:

T1. Header. This line has four fields. They are:

Distant End Switch Name - The three-letter for the switch at the far end of the trunk group.
In-Service - The number of trunks marked as in service at the end of the reporting period.
Equipped - The nominal number of trunks in the group.
Capacity - The number of working (undamaged) trunks at the end of the report period.

T2. Incoming Attempts. This line has peg counts for incoming calls.

T3. Incoming Failures. This line has peg counts for incoming signaling failures detected by the switch. These are not currently modeled by CCSIM and are set to zero.

T4. Outgoing Attempts. This line has peg counts that record the number of times the switch attempted to use the group to route outgoing calls. It includes calls that were deflected by

CANT and SKIP network management controls.

T5. Overflows. This line has peg counts for calls that did not find free trunks when attempting to access the groups. Calls that may ultimately be carried as a result of preemption are counted as overflows in T5. Thus the counts here do not correspond to the 'NOVFLAT' field in the DMS switch report or the overflows displayed in the CCSIM graphics windows. Calls deflected by the CANT, SKIP, ARC-A, and CANF controls are not counted in T5.

T6. Glare. This line has peg counts for calls for which a 'glare' condition was detected by the switch. To simulate glare, CCSIM generates a random value with a mean of 'mglare' percent of the lesser of the sums of incoming attempts (T2) or connections (T12) for all precedences. It places the simulated glare value in the first (routine precedence) field and sets the remaining four fields of T6 to zero. The 'mglare' parameter is settable by the experimenter in the 'net.inval' file.

T7. Outgoing Failures. This line is intended to have peg counts of failures to effectively seize a chosen trunk due to various hardware problems. CCSIM generates counts in this field when it attempts to use trunks that have been marked as damaged.

T8. Deflected Calls. This line has peg counts for calls that were deflected by the CANT and SKIP network management controls.

T9. Traffic Usage. This line accumulates the results of sampling the trunk status of each trunk in the group at 100 second intervals. If the trunk is carrying a call at sample time, a count is registered in the appropriate precedence field.

T10. System Busy Usage. This line is intended to accumulate usage in the same way as T9, but of trunks made busy as a result of system action. CCSIM does not currently model such action and sets all values to zero.

T11. Manual Busy Usage. This line is intended to accumulate usage in the same way as T9, but of trunks made busy as a result of manual action. CCSIM does not currently model such action and sets all values to zero.

T12. Connections. This line has peg counts for successful attempts at seizing trunks in the group.

T13. Tandem Calls. This line has peg counts for incoming calls on the trunk group that turn out to be tandem calls.

T14. Preemptions. This line has peg counts for calls of each precedence on the trunk group that were preempted during the report period.

T15. Overflows After Preemption. This line has peg counts for call attempts that overflowed the trunk group after having failed to find either a free or preemptible trunk. Since routine calls can never preempt, the first field of T15 is always the same as the first field of T5.

T16. End-of-Chain Overflows. This line has peg counts for overflows from the trunk group when the group is the last group in the routing chain for the call destination. The counted calls blocked at the switch.

T17. Canceled by CANT. This line has peg counts for calls that were canceled by the CANT network management control.

T18. Deflected by SKIP. This line has peg counts for calls that skipped over this trunk group as a result of the SKIP control.

T19. Canceled by CANF. This line has peg counts for calls that were canceled by the CANF network management control.

F2.0 The IWDB Switch Report

The IWDB switch report is an ASCII file containing a header line with identification and date/time information followed by ten OM reports. If the flag 'toiwdb' in the 'net.inval' file is set to one, an IWDB switch report file is written for each non-damaged switch at every report interval. Each OM report is a line of fields separated by commas except for the last report which has a header line (TRK,16) followed by a line for each reporting trunk group at the switch. Section F2.1 has an example of such a report generated by CCSIM.

The OM reports are identified by name and report number, e.g. 'TRK,16', the numbers and names being those used in the DMS switches. In the reports collected for the IW at DCA-Eur, different report numbers occur for the same report from different switches due to different software versions in the switches, but CCSIM always generates the same numbers for the reports independent of which switch is reporting.

The OM report lines are calculated as follows:

CP,1	The other five fields of this line are copied from the fields of line S12 of the full report.
CPU,2	The other seven fields of this line are copied from the fields of line S13 of the full report.
CMC,3	The other two fields of this line are copied from the fields of line S14 of the full report.
RCVR,10	The other ten fields of this line are copied from the fields of lines S15 and S16 of the full report.
RADR,11	The other six fields of this line are copied from the fields of line S17 of the full report.
TCM,6	The other three fields of this line are copied from the fields of line S18 of the full report.
TER,7	The other two fields of this line are copied from the fields of line S19 of the full report.
TRS,8	The other four fields of this line are copied from the fields of line S20 of the full report.
DCM,9	The other two fields of this line are copied from the fields of line S21 of the full report.

TRK,16 This OM report consists of one line for each reporting trunk group. The fields of each line are as follows:

Group Number - The group number for the trunk is obtained from the 'net.clli' file. In preparing the report, the lines are ordered according to ascending trunk group number.

Equipped - The nominal number of trunks in the group.

In-Service - The number in service at the end of the reporting interval.

INCATOT - The number of incoming seizures. Calculated by summing the fields of line T2 of the full report.

INFAIL - The number of incoming failures. Calculated by summing the fields of line T3 of the full report.

NATTMPT - The number of outgoing attempts. Calculated by summing the fields of line T4 of the full report.

NOVFLATB - The number of outgoing overflows. Calculated by summing the fields of line T15 of the full report.

GLARE - The number of glare events detected. Calculated by summing the fields of line T6 of the full report.

OUTFAIL - The number of outgoing failures. Calculated by summing the fields of line T7 of the full report.

DEFLECTED - The number of calls deflected by network management controls. Calculated by summing the fields of line T8 of the full report. This value does not correspond to the same field in the reports from the real switches. We have been told that there is a bug in the switch software, and that the field is currently being ignored by the IW. We believe that CCSIM's value is appropriate for the intended meaning of the field.

TRU - Usage on traffic-busy trunks. Calculated by summing the fields of line T9 of the full report.

SBU - Usage on system busied-out trunks. Calculated by summing the fields of line T10 of the full report.

MBU - Usage on manually busied-out trunks. Calculated by summing the fields of line T11 of the full report.

CONNECT - The number of apparently successful attempts at trunk seizure. Calculated by summing the fields of line T12 of the full report.

TANDEM - The number of incoming calls initially routed to an outgoing trunk group. Calculated by summing the fields of line T13 of the full report.

PREEX - The number of preemptions exercised on the trunk group. Calculated by summing the fields of line T14 of the full report.

PREOVFL - The number of precedence calls that overflowed. Calculated by summing the last four fields of line T15 of the full report.

F2.1 Example IWDB Switch Report

The following is the contents of an IWDB switch report written to a file named '901205.0055b.uxb' according to IWDB conventions.

```
02:1990/12/05 00:55:00 WED;  ccsimUXB: 1990/12/05 01:00:00 WED;
CP,1,350,0,0,0,0
CPU,2,0,0,0,0,0,0,0
CMC,3,0,0
RCVR,10,324,0,71,0,0,112,0,20,0,0
RADR,11,75,0,0,75,0,0
TCM,6,4,0,16
TER,7,0,0
TRS,8,0,0,0,41
DCM,9,0,0
TRK,16
72,7,7,9,0,1,0,0,0,0,0,1,0,0,0,0
87,8,8,3,0,12,6,0,0,6,6,0,0,0,0,0
88,7,7,4,0,12,6,0,0,6,6,0,0,0,0,0
89,6,6,4,0,9,3,0,0,3,3,6,0,0,0,1
90,8,8,0,0,0,0,0,0,0,0,0,0,0,0,0
91,7,7,0,0,8,1,0,0,1,1,7,0,0,0,0
92,7,7,4,0,0,0,0,0,0,0,0,0,0,0,0
93,24,24,9,0,1,0,0,0,0,0,1,0,0,0,0
95,10,10,17,0,12,1,0,0,1,1,11,0,0,0,0
96,3,3,1,0,14,9,0,0,9,9,5,0,0,0,0
97,1,1,1,0,1,1,0,0,1,1,0,0,0,0,1
98,10,10,1,0,40,32,0,0,32,32,8,0,0,0,0
99,7,7,4,0,2,0,0,0,0,0,2,0,0,0,0
101,48,48,10,0,13,0,0,0,0,0,13,0,0,0,0
102,5,5,4,0,5,1,0,0,1,1,4,0,0,0,0
103,3,3,3,0,16,12,0,0,12,12,4,0,0,0,0
104,8,8,6,0,32,26,0,0,26,26,6,0,0,0,6
105,6,6,9,0,31,24,0,0,24,24,7,0,0,0,0
108,24,24,10,0,24,0,0,0,0,0,24,0,0,0,0
112,24,24,6,0,8,0,0,0,0,0,8,0,0,0,0
```

```
114,4,4,1,0,9,7,0,0,7,7,2,0,0,0,0
120,4,4,8,0,1,0,0,0,0,0,1,0,0,0,0
127,24,24,10,0,4,0,0,0,0,0,4,0,0,0,0
128,8,8,12,0,6,4,0,0,4,4,0,0,0,0,0
137,1,1,2,0,2,1,0,0,1,1,1,0,0,0,1
138,24,24,25,0,2,0,0,0,0,0,2,0,0,0,0
140,24,24,16,0,6,0,0,0,0,0,6,0,0,0,0
169,10,10,6,0,1,0,0,0,0,0,1,0,0,0,0
```

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE 30 September 1990	3. REPORT TYPE AND DATES COVERED Annual Report, 1 October 1989 – 30 September 1990

4. TITLE AND SUBTITLE Knowledge-Based System Analysis and Control Defense Switched Network Task Areas	5. FUNDING NUMBERS C — F19628-90-C-0002 PE — 62702F
6. AUTHOR(S) Harold M. Heggestad	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lincoln Laboratory, MIT P.O. Box 73 Lexington, MA 02173-9108	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DCA Engineering Group 1860 Wiehle Avenue Reston, VA 22090-5500	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESD-TR-90-151
11. SUPPLEMENTARY NOTES None	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A major activity during FY90 has been the design and implementation of a network management expert system to operate in the Integrated Workstation (IW) that was developed during FY90 for use by ACOC personnel at DCA-Eur to perform DSN network management tasks. The IW was successfully tested on live and archived data, and on fault conditions deliberately induced by switch technician actions, during the period 25-28 September 1990. All parties declared that the IW features and demonstrated performance were valuable and successful. The Expert System was well received, in particular. An IW terminal has been installed on the floor of the ACOC, and the staff have been directed to familiarize themselves with its operation.

A number of changes and improvements were made in CCSIM and related programs. All were converted to run under SUN OS 4.0.3. The graphics interface program was rewritten to use the X-window system. A new document called "Using The Call-By-Call Simulator (CCSIM)" has been written, and the "CCSIM User's Manual" and the "CCSIM Software Top Level Design Document" delivered in FY90 are to be updated early in FY91.

Work was performed in expert systems development efforts for DCS transmission system control with two main components: implementation of the TRAMCON Event Generator (TEG), and participation in the Tech Control Automation Proof-of-Concept System (TCAPS). TEG had been specified during FY89, and has now been written and successfully installed and demonstrated at a number of locations. Lincoln's RADC-sponsored MITEC expert system was incorporated in a DCA-sponsored set of TCAPS field demonstrations.

14. SUBJECT TERMS			15. NUMBER OF PAGES 116
expert systems communications control	system control network management	Defense Switched Network (DSN) communication network simulation	16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unclassified