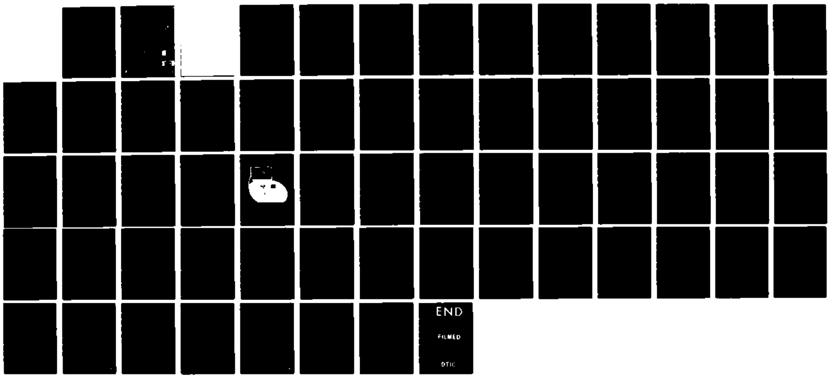
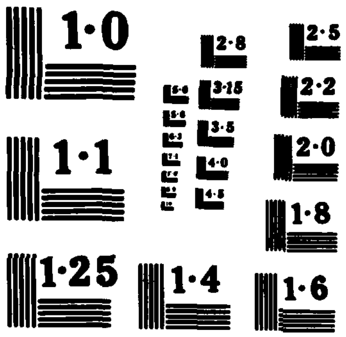

END
FILMED
DTIC


1·0
2·8
2·5
3·15
2·2
3·5
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

[illegible]

[illegible]

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

[illegible]

[illegible]

85 07 15 90 8

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

DEFENSE SWITCHED NETWORK TECHNOLOGY AND EXPERIMENTS PROGRAM

ANNUAL REPORT
TO THE
DEFENSE COMMUNICATIONS AGENCY

1 OCTOBER 1983 — 30 SEPTEMBER 1984

ISSUED 29 MAY 1985

LEXINGTON MASSACHUSETTS

ABSTRACT

This report documents work performed during FY 1984 on the DCA-sponsored Defense Switched Network Technology and Experiments Program. The areas of work reported are: (1) development and evaluation of routing and system control techniques for application in the Defense Switched Network (DSN), (2) instrumentation and integration of the Experimental Integrated Switched Network (EISN) test facility, (3) development and test of data communication techniques using DoD-standard data protocols in an integrated voice/data network, and (4) EISN system coordination and experiment planning.

TABLE OF CONTENTS

Abstract iii

List of Illustrations vii

1. INTRODUCTION AND SUMMARY 1

2. DSN ROUTING AND SYSTEM CONTROL TECHNIQUES EVALUATION 3

2.1 Introduction 3

2.2 Routing Algorithm Simulation Studies 4

2.2.1 Call-by-Call Simulator Development 4

2.2.2 Routing and Preemption Algorithm Performance Results 7

2.3 EISN Routing and Control Experiments 11

2.4 Switch Features Required With an Outboard Processor 13

2.5 Planning and Design of EISN Security Experiments 14

2.5.1 Background and System Issues 14

2.5.2 Design of an EISN EFTO Test-Bed 17

2.5.3 Test-Bed Configuration and Experiment Planning 21

2.6 Preliminary Investigation of Application of Expert Systems Techniques to System Control 21

3. EISN ROUTING AND CONTROL FACILITY INSTRUMENTATION 23

3.1 Introduction 23

3.2 RCP Hardware and Digital Switch Integration 23

3.3 RCP Software Development 25

3.4 Packet/Circuit Interface (PCI) Integration 29

3.4.1 Voice and CCS Link Between PCI and RCP 30

3.4.2 PCI Support 30

3.5 EISN Experiment Control and Support 30

4. DATA PROTOCOLS AND VOICE/DATA EXPERIMENTS 33
4.1 Summary of Experiment Results 33
4.1.1 Data Experiments 33
4.1.2 Voice/Data Integration Experiments 34
4.2 Protocol Issues 37
4.2.1 Window Size Issues 39
4.2.2 Segment Size Issues 39
4.2.3 Acknowledgment Policy Issues 41
4.2.4 Retransmission Issues 41
4.2.5 Pacing Issues 43
4.2.6 Reliable Estimation of Round-Trip Time (RTT) 47
4.3 Conclusions 49
5. EISN SYSTEM COORDINATION AND EXPERIMENT PLANNING 51
References 53
Glossary 55

LIST OF ILLUSTRATIONS

Figure No.		Page
1	Block Diagram of Call-by-Call Simulator	6
2	20-Node Test Network DSN1	9
3	Simulator Results When Satellite and 50 Percent of Voice Trunks Are Destroyed in DSN1. (a) Blocking Probability for High- and Low-Precedence Calls; (b) Number of Calls Preempted.	10
4	Block Diagram of an Outboard Routing and Preemption Processor Connected to a Commercial Switch via Attendant Console Ports	14
5	Block Diagram of a DSN Node Including a CCS Processor That Performs Link-Level Part of CCS Protocol for Both Switch and an Outboard Routing and Preemption Processor	15
6	Current System Configuration	17
7	System Configuration With EFTO	18
8	Advanced Routing and System Control Test-Bed	24
9	Current RCP Software Structure	26
10	Voice/Data Integration Experiment Configuration	34
11	Delay Histogram: Voice/Data Multiplexing Experiment	36
12	Timing Diagram for a File Transfer Starting With a Successful Burst of Four Segments	44
13	Timing Diagram for a File Transfer Starting With a Burst of Four Segments in Which One (S[2]) Is Lost in Network	45
14	Timing Diagram Showing a Network Delay Increase Large Enough to Cause Retransmission of Segment S[i] on a Time-Out. Definition of Round-Trip Time (RTT[i]) Is at Issue.	48

DEFENSE SWITCHED NETWORK TECHNOLOGY AND EXPERIMENTS PROGRAM

1. INTRODUCTION AND SUMMARY

This report documents work performed during FY 1984 on the DCA-sponsored Defense Switched Network Technology and Experiments Program. The areas of work reported are: (1) development and evaluation of routing and system control techniques for application in the Defense Switched Network (DSN), (2) instrumentation and integration of the Experimental Integrated Switched Network (EISN) test facility, (3) development and test of data communication techniques using DoD-standard data protocols in an integrated voice/data network, and (4) EISN system coordination and experiment planning.

Routing algorithm development simulation studies during FY84, as described in Section 2.2, have focused on continued development of an extensive call-by-call network simulator and on the application of this simulator to evaluate the dynamic performance of new routing and Multi-Level Precedence (MLP) techniques. A major result of the simulation studies was that adaptive routing with guided preemption was shown to be as effective as the more complex strategy of precedence flooding with preemption, in providing low blocking probability for both low- and high-precedence calls after network damage. Continued development of the simulator has resulted in the addition of guided preemption, precedence flooding, call retries, and a number of new traffic-generation and statistics-collection features. The simulator was initially delivered to DCEC in February 1984; a second delivery, including all features developed in FY84, is planned for October 1984.

EISN routing and control experiments, described in Section 2.3, have focused on experimental evaluation of call control and Common-Channel Signaling (CCS) implementations in the Routing/Control Processors (RCPs) in 2- and 3-node networks. Capabilities demonstrated include: RCP control of interswitch voice calls with CCS, phantom (emulated) RCP calls with preemption, and running of multiple virtual RCPs in a single PDP-11/44. These capabilities provide the basis for multi-site RCP experiments to be conducted after the initial deliveries of RCPs to EISN sites (Ft. Huachuca and Ft. Monmouth) during the first part of FY85. In addition, a number of basic routing experiments have been conducted using the Packet/Circuit Interface (PCI) facilities at the EISN sites. In particular, a demonstration of simultaneous multi-site PCI calls involving DCEC, Ft. Monmouth, and Ft. Huachuca was shown at the April 1984 EISN Steering Committee Meeting at DCEC.

Two study efforts were carried out in conjunction with the development of DSN routing and system control techniques. A switch features requirements study (Section 2.4), undertaken at the request of the DCA to extend and generalize the RCP work, resulted in a specification of those features required in a commercial switch to support new routing and preemption algorithms with outboard processor control. Strategies for increasing the call-handling capacity and efficiency of a switch/outboard processor combination, beyond the experimental capability provided by a single operator port interface, are defined and compared. The second study effort, described in Section 2.5, addressed the planning and design of EISN security experiments. The key issue addressed is the design of an Encryption for Transmission Only (EFTO) strategy for a flexibly demand-assigned satellite system such as that used in EISN. A system design for achieving this goal has been developed and is described in Section 2.5. In addition, preliminary investigations were begun (see Section 2.6) into the application of expert systems techniques to telecommunications system control. A more extensive study of this area is planned for FY85.

The major FY84 effort in EISN instrumentation and integration, described in Section 3, has been the continuing development of the RCP hardware and software system. Hardware developments have included the integration of all RCP interface components into interface cabinets for each site. RCP software development has been a major activity which has led to a complex, real-time software structure that supports both voice-call demonstrations and phantom-call distributed simulation tests. Extensive software development and debugging were carried out to develop the switch interface handlers, call sequencer logic, link-level CCS protocol, and the user interface. In addition to the RCP efforts, development has been carried out (Section 3.4) to support voice and CCS links between a PCI and an RCP-equipped switch. These links will provide satellite trunking for the RCP-based EISN test-bed.

Efforts in the area of data protocols have continued to focus on exploration of the performance of the DoD standard protocols, TCP and IP, in an environment including voice and data traffic. During FY83 we developed measurement tools, extended our gateway capabilities, and carried out experiments that examined the behavior of TCP file transfers in competition with other data traffic. During FY84 we extended our measurements to include situations in which the data traffic contended with packet-voice traffic for network resources, and conducted a study of the issues to be faced in implementing a TCP that would aim to achieve good performance in a general network environment. In Section 4, we summarize the experiment results and present our conclusions from the study of TCP implementation issues relative to performance.

Finally, as described in Section 5, Lincoln has continued its role in EISN experiment planning and system coordination. An FY84 Project Master Plan was delivered to DCEC detailing FY84 experiments and outlining FY85 plans. With Lincoln coordination, the Wideband SATNET Task Force continued to address problems of network performance and reliability. A system problem which caused network failure when more than five streams were set up was identified and diagnosed, and modified PSAT (Pluribus Satellite Interface Message Processor) software was prepared by Bolt, Beranek, and Newman (BBN) to correct the problem. Intensified user-level network tests were run during August and September with the network set in an operational configuration; this provided a number of successful demonstrations of voice and data transmission capabilities, but uncovered some additional system problems. The channel characteristics were improved by a move from the WESTAR III to the WESTAR IV satellite in late July.

2. DSN ROUTING AND SYSTEM CONTROL TECHNIQUES EVALUATION

2.1 INTRODUCTION

Previous Lincoln efforts[1,2] have resulted in new mixed-media routing and preemption algorithms directed at the dual DSN requirements of survivability and low cost. Mixed-media routing procedures without preemption were initially evaluated using a steady-state network analysis program provided by DCEC that was modified to support new routing procedures.[3,4] More complex procedures including precedence flooding routing procedures were then evaluated using a new call-by-call simulator.[2,5]

During FY84, mixed-media routing and preemption procedures were implemented and tested in a 3-node test-bed network containing real switches and Routing Control Processors (RCPs). This marks the beginning of the final phase of routing and preemption algorithm development during which algorithms, switch logic, and protocols will be experimentally validated and demonstrated in a real network test-bed. In addition, research using the call-by-call simulator has continued and new smaller efforts were completed in the areas of data encryption and expert systems techniques for network control.

More specifically, work during FY84 has focused on the following (subsections which describe each area are indicated in parentheses):

(a) Delivery of the call-by-call simulator to DCEC and continued development of the simulator to add guided preemption, precedence flooding with preemption, internal call generation, call retries, and more efficient calculation of the average number of calls-in-progress and link occupancy (2.2.1).

(b) Use of the call-by-call simulator to compare different types of preemption, including blind preemption, guided preemption, and precedence flooding with preemption under severe network damage (2.2.2).

(c) Experimental evaluation of the logic and CCS protocol that supports spill-forward mixed-media routing in 2- and 3-node RCP test-bed networks using both real calls offered by users and phantom calls offered during distributed simulation runs (2.3).

(d) Specification of those switch features required to support new routing and preemption algorithms in a commercial switch, and analysis of the advantages and disadvantages of using an outboard processor to add these new features (2.4).

(e) Investigation of techniques that could be used to provide data encryption in the broadcast satellite component of the test-bed network (2.5).

(f) Initial investigation of the applicability of expert system techniques to system control in military circuit-switched networks (2.6).

Some major results of Lincoln's FY85 routing and system work are highlighted in the remainder of this introductory section.

One major result of adding guided preemption to the call-by-call simulator was the development of a more efficient and practical implementation of guided preemption that requires switches to pass very little additional information over the CCS network to preempt. In this implementation, the paths of all calls impinging on a switch are stored and searched when preemption is required. The search selects that lower-precedence call with the most links in common with the path of the current high-precedence call. Only a small amount of memory is required to store call paths (typically less than 4 kbytes), and the time for the extra search is minimal.

A major finding of call-by-call simulation studies was that adaptive routing with guided preemption was shown to be as effective as precedence flooding with preemption in providing low blocking probability for both high- and low-precedence calls. The one advantage of the more complex precedence flooding technique was that fewer low-precedence calls were preempted under severe network damage. This is due to the brute-force flood search which selects the one-call path that both uses the fewest links and preempts the fewest calls. Another finding of simulation studies was that guided preemption reduces the number of calls preempted by as much as 20 percent compared with blind preemption as used in AUTOVON.

Experiments using RCPs in 2- and 3-node test-bed networks validated the protocols and logic required to set up calls in these small networks and led to a number of important modifications in RCP software. They also demonstrated that it is possible to run multiple virtual RCPs within each PDP-11/44 without extensive changes to the existing operating system. This technique will make it possible to perform distributed simulations in test networks with five PDP-11/44s and 10 to 20 virtual RCPs. RCP tests were also used to debug the extensive statistics collection facilities available within RCPs and to perform initial comparisons between results obtained in 3-node RCP networks and in the call-by-call simulator.

2.2 ROUTING ALGORITHM SIMULATION STUDIES

2.2.1 Call-by-Call Simulator Development

Development of a call-by-call simulator was begun in FY82 to evaluate new complex routing and preemption procedures that could not be analyzed using steady-state queueing theory models. Extensive additions and improvements were made to the simulator during FY84. Enhancements included:

(a) Adding the ability to retry after a prespecified probabilistic interval when a call is blocked or preempted.

(b) Adding statistics to support call retries; new statistics included final blocking probability after all retries are complete, and the blocking probability on the 1st, 2nd, 3rd, . . . try.

(c) Adding internal call generation to eliminate the need for an input file of offered calls.

(d) Adding guided preemption.

(e) Adding preemption to precedence flooding such that the choice of a path after flooding depends on both path length and the number of links where calls must be preempted.

(f) Improving the efficiency of data-base access when the average number of calls in progress and link occupancy are calculated.

(g) Reducing the simulator clock interval from 1/3 to 1/10 s.

(h) Increasing the maximum simulator run time from 1 to 24 h.

A block diagram of the current simulator with the above enhancements is presented in Figure 1. In this new simulator, calls can be generated internally or calls can be read in from a call file generated by an external call generation program. The minimum input required by the simulator is now a file containing routing tables, a file describing the network topology and link capacities, a file containing the offered traffic matrix, and a file containing controls for the current run. Events within the simulator that can occur at prespecified times have been expanded to include:

(a) Damage to nodes and links in the network,

(b) Generation of a new first-offered call,

(c) Retry of a call that was offered previously and preempted, blocked, or taken down by damage, and

(d) Termination of a call.

Internal changes to the calls-in-progress simulator data base were made to record the times of internal events concerned with calls. This greatly improved the efficiency of the simulator when computing the average number of calls in progress and the average link occupancy. It reduced the simulation run time by a factor of two on test runs with large networks.

The addition of call retries to the simulator required major changes and additions both in the way calls were generated and in techniques used to collect statistics. The retry model used allows calls that are blocked, preempted, or taken down during damage to be offered more than once. The probability of a retry, the maximum number of retries allowed, and the average time until the next retry are specified before a run. Statistics accumulated include the number of retries, the blocking probability after 1,2, . . .,n retries, and the ultimate blocking probability considering all retries. The retry model that has been implemented selects the time until the next retry using an exponentially distributed random variable with a desired mean.

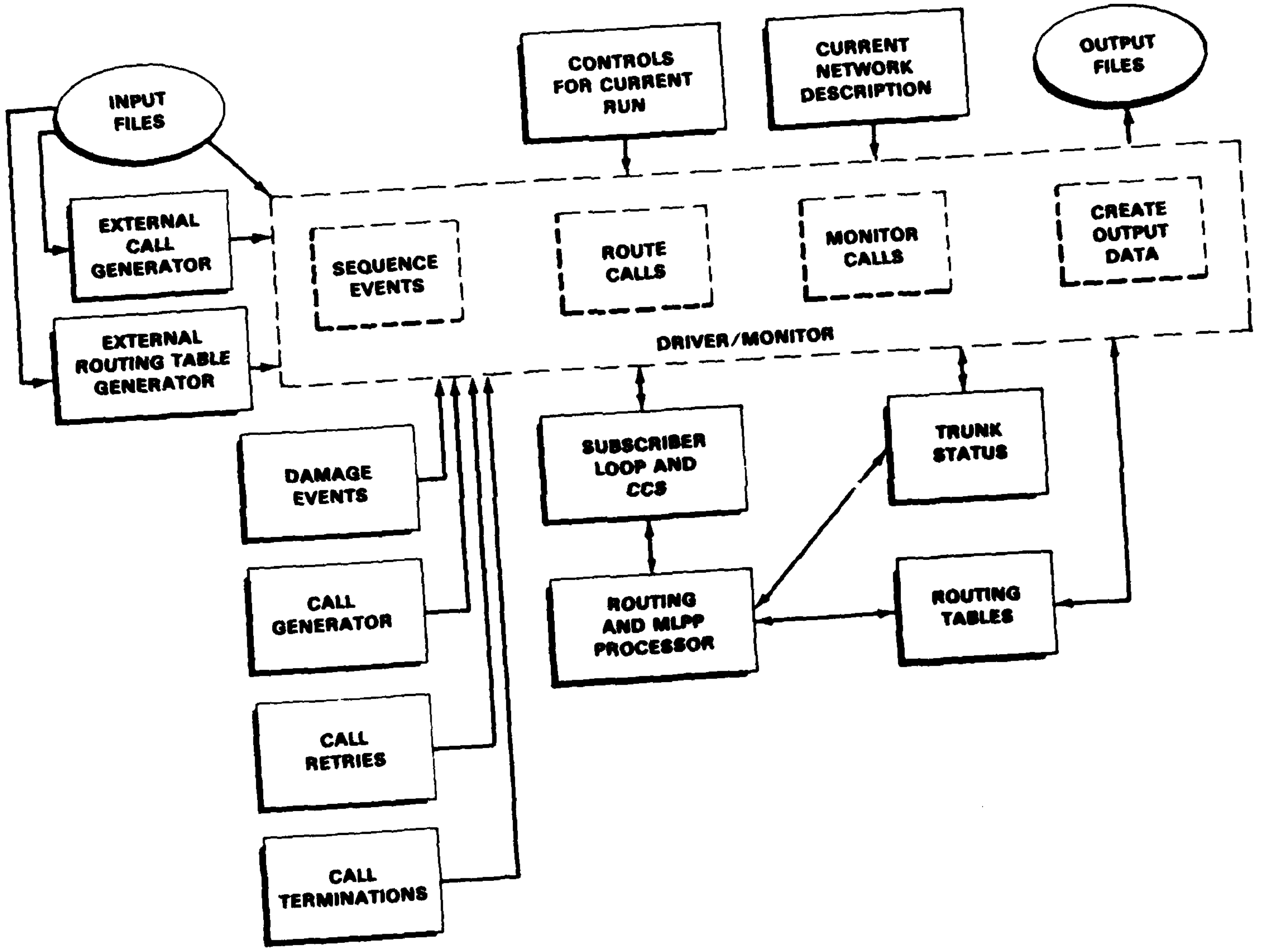


Figure 1. Block diagram of call-by-call simulator.

116977-N-06

While guided preemption was being added to the simulator, a more efficient implementation was developed that requires no additional information to be passed between switches when preemption is necessary. The paths of all calls impinging on a switch and the precedence levels of all calls are stored within each switch. When preemption is required, this data base is searched to find the lowest precedence call with the most links in common with the path of the current high-precedence call. Only a small amount of memory is required to store call paths (typically less than 4 kbytes), and the time for the extra search is minimal. Switches obtain call path information from the setup-success CCS packet that returns to the source after a path to the destination has been found. The number of extra bytes required to add call path information to this packet is small (typically an additional 2 to 3 bytes) relative to the total number of CCS bytes required for one call (typically 90 to 100 bytes).

As a result of requests from DCEC, the simulator was delivered to D. Calabrese and G. Swinsky of DCEC in February 1984 with another delivery planned for October 1984. The simulator will be used at DCEC for network performance analyses and routing research. A number of simulator enhancements were made specifically for this delivery. These included reducing internal timing to 1/10 s and extending the run time to 24 h.

Extensive preparations were required to deliver the simulator to DCEC both to translate software for a different operating system and a different version of FORTRAN and to provide adequate documentation. Simulator source programs at Lincoln were developed under the UNIX operating system using FORTRAN 77 and were converted to a form that can be run under the TSO operating system using IBM FORTRAN IV. A simulator users' guide was written, and all simulation documentation and comments in the source code were updated to reflect the current simulator status. The users' guide describes how to use the simulator and how to modify the simulator to add new types of routing. Other software delivered with the simulator included a program to generate routing tables, a program to generate files of offered calls, and software and documentation required to build a RATFOR interpreter to translate RATFOR to FORTRAN under the TSO operating system.

2.2.2 Routing and Preemption Algorithm Performance Results

Two series of simulator runs were performed to compare new types of routing and preemption procedures when there was extensive network damage. Four types of preemption procedures were examined. Blind preemption, as used in AUTOVON, blindly preempts on links as a call-setup CCS packet goes out from the source during the call-setup procedure. It may preempt an excessive number of calls on a multi-link path and may preempt lower-precedence calls on the first few links of a call path even if a complete call path is not established. Blind-back preemption is a slightly more complex version of blind preemption. It preempts blindly on a link as with blind preemption, but it waits until the setup-success CCS packet comes back from the destination after a complete call path has been established. Although it does not preempt calls unless a complete call path is available, it still may preempt an excessive number of calls on multi-link paths.

Guided preemption, introduced by Lincoln,[1] preempts the fewest calls on the shortest path to the destination. As described above, this is accomplished by storing the paths of all calls that impinge on a switch within each switch and searching for a lower-precedence call with the most links in common with the call being set up. In this type of preemption, no lower-precedence calls are preempted if a complete path to the destination is not available and the minimum number of lower-precedence calls is preempted on multi-link calls.

Precedence flooding preemption is the type of preemption included within precedence flooding routing. During the flood search, search-out CCS packets accumulate the number of links traversed and the number of links where preemption is required for all paths from the source to the destination. The destination switch then selects the path with the fewest links which requires preemption on the fewest links. If two paths have the same length, the path which requires no preemption or preemption on the fewest links is selected. This brute-force search will tend to provide a lower bound for the number of calls that must be preempted.

An initial series of runs was performed with severe network damage to compare mixed-media routing with blind preemption to precedence-blocked flooding with preemption. Low-precedence calls were routed using spill-forward mixed-media routing. These calls could travel one more link than the number of links in the shortest path to each destination. High-precedence calls were routed using:

(a) Spill-forward, mixed-media routing allowing three more links than the number of links in the shortest path to each destination;

(b) Same as (a) but with blind preemption;

(c) Adaptive mixed-media routing allowing three more links than in the shortest path to each destination;

(d) Precedence flooding where all high-precedence calls were routed using flooding;

(e) Precedence blocked flooding where only high-precedence calls blocked after being routed with spill-forward, mixed-media routing are routed using flooding; and

(f) Precedence flooding with preemption.

Simulator runs were made using 20-node network DSN1. Link and switch locations for this network are presented in Figure 2. Solid lines in this figure represent land links, and dashed lines represent links to one Demand-Assigned Multiple Access (DAMA) satellite. Network DSN1 is a minimum-cost network designed for a link-blocking probability of 0.1 and designed to route roughly one-third of all traffic over the satellites under normal conditions. It includes twenty switches, one DAMA satellite, and five earth stations.

Runs were performed under three damage conditions. In the first condition, only the satellite was destroyed. In the second condition, the satellite and 50 percent of all land trunks were destroyed. Those trunks on the links with the greatest numbers of trunks were destroyed first, with

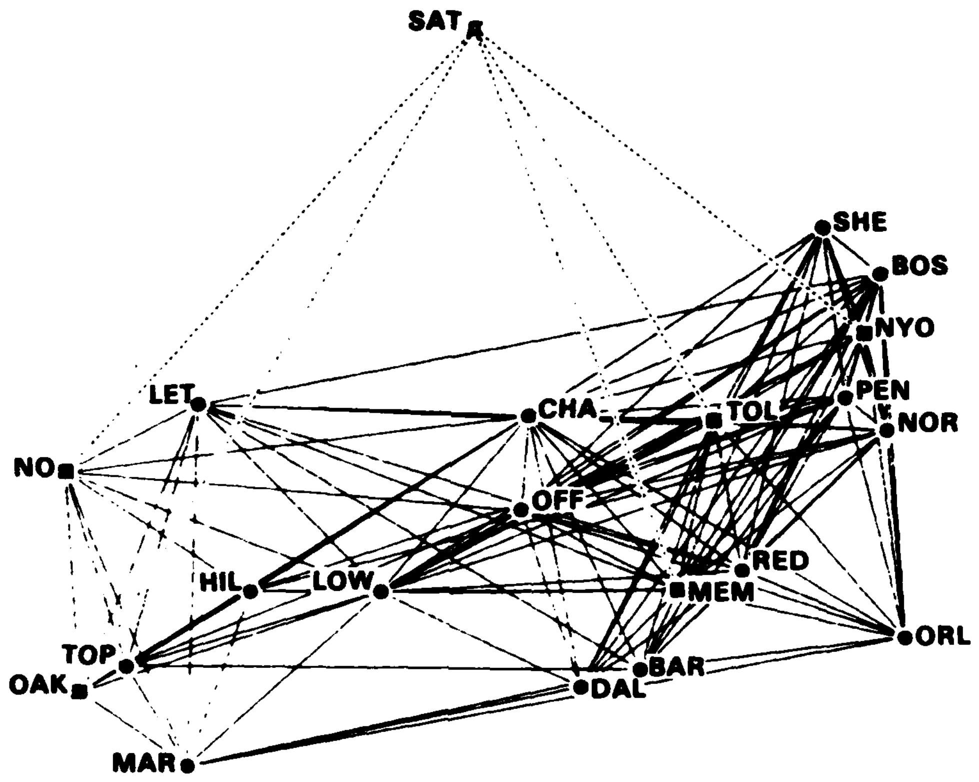


Figure 2. 20-node test network DSN1.

the result that after damage 86 out of 118 land links remained. In the third condition, the satellite was destroyed and only the three geographically shortest links on each node were preserved. After damage, only 41 out of 118 land links remained

Offered traffic was identical for all runs and consisted of 22,000 calls offered over an hour of simulated time. Roughly 20 percent of these calls were priority calls, and the remaining 80 percent were routine. Runs were performed on a VAX computer and used from 16 to 70 min. of CPU time.

Results for the case when the satellite and 50 percent of the voice trunks were destroyed are presented in Figure 3(a-b). The average point-to-point blocking probability for high- and low-precedence calls is contained in Figure 3(a), and the number of calls preempted is shown in (b). As can be seen, best performance is provided by high-precedence flooding with preemption (FLOOD HP + PREEMPT) and by adaptive routing with blind preemption (PREEMPT + ADAPT). The blocking probability for high-precedence calls was low and almost identical for these two procedures, while the blocking probability for low-precedence calls was near that provided by other algorithms. In addition, flooding with preemption performed slightly better than

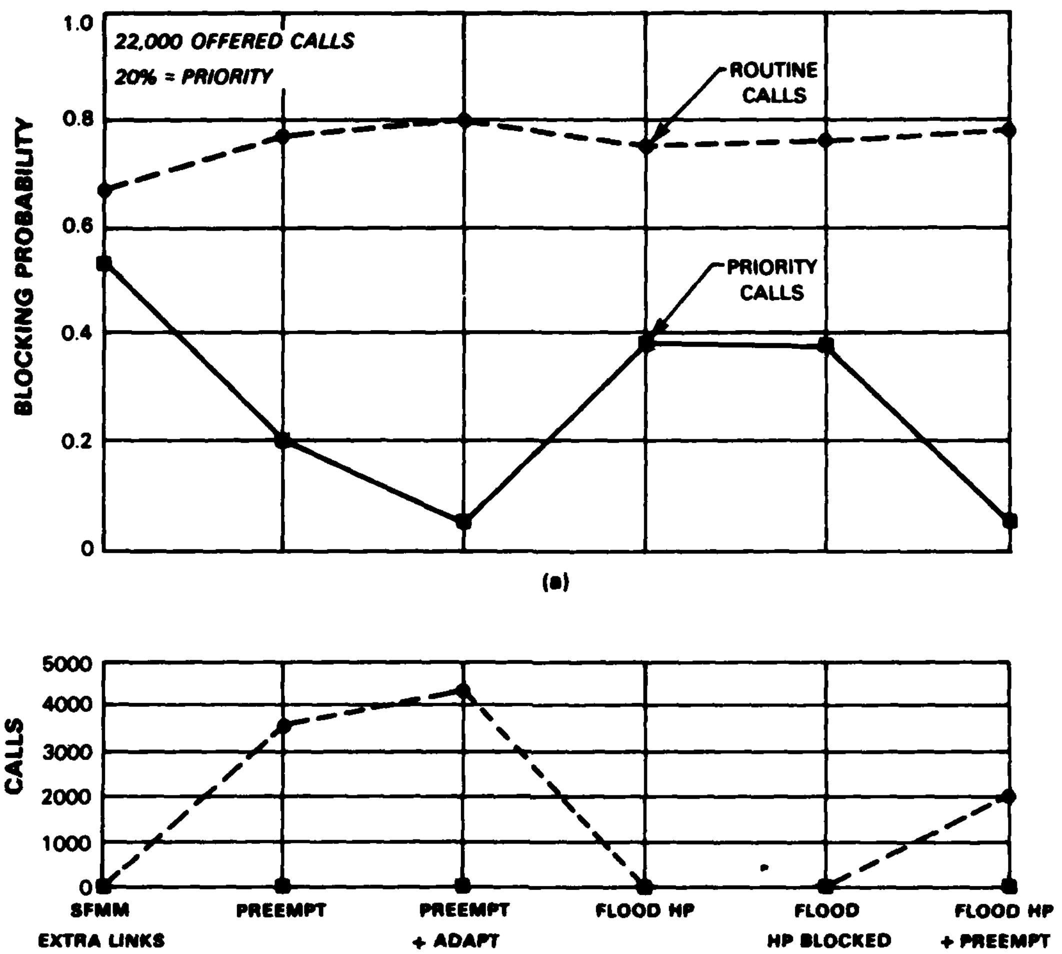


Figure 3. Simulator results when satellite and 50 percent of voice trunks are destroyed in DSN1. (a) Blocking probability for high- and low-precedence calls; (b) number of calls preempted.

adaptive mixed-media routing with blind preemption because fewer calls were preempted (2100 calls with flooding vs 4400 calls with adaptive routing and blind preemption). Results also demonstrated that high-precedence flooding does not require excessive CCS bandwidth as is sometimes assumed for flooding. The average CCS transmission rate was only 115 bps compared with 64 bps using adaptive mixed-media routing with blind preemption. Similar results were obtained for the other types of damage, although the differences in calls preempted with the two best types of routing were typically less.

A second series of runs was made to determine whether adding guided preemption to adaptive mixed-media routing would reduce the number of calls preempted. Blind-back preemption was included in these runs to determine which feature of guided preemption contributed most to reducing the number of calls preempted.

Simulation runs were again made using 20-node test network DSN1 when the satellite and 50 percent of the land trunks were destroyed. All runs lasted 1 h of simulated time and used a mix of traffic where 20 percent of the offered calls were priority and 80 percent were routine. Probability of blocking for both routine and priority calls varied little for the three types of preemption. The number of calls preempted, however, decreased from 4410 with blind preemption to 4270 with blind-back preemption to 4100 with guided preemption. Guided preemption thus reduced the number of calls preempted by a significant amount (310 calls, or 7 percent) relative to the additional complexity required. In addition, these results demonstrated that, under the conditions studied, the reduction in calls preempted is caused equally by preempting only after a complete call path is available and by optimally selecting low-precedence calls to preempt. Preempting after a complete call path was available with blind-back preemption reduced the number of calls preempted by 140 calls. Moving to guided preemption to select low-precedence calls optimally reduced the number of calls preempted by an additional 170 calls.

Current plans for the simulator are to add polygrid routing which is used in AUTOVON. This was not possible until recently when a polygrid routing program was provided by DCEC. It has been requested in the next simulator delivery to DCEC. Baseline studies will be performed to compare polygrid routing with adaptive routing with guided preemption, and precedence flooding routing with preemption. Further simulation runs are also planned to further explore the use of adaptive mixed-media routing and guided preemption in networks with limited connectivity with different types of damage. In addition, the simulator will continue to be used as a basis for implementing routing algorithms and CCS communication protocols in the RCP, and to provide comparison results for distributed simulation runs made using RCPs.

2.3 EISN ROUTING AND CONTROL EXPERIMENTS

Networking experiments on the EISN test-bed have divided naturally into two categories, keyed to the installed experimental capabilities. The interim EISN equipment currently in place at all sites has supported an initial set of routing and system control (RSC) experiments, as outlined in the EISN Experiment Plan (submitted to DCEC in November 1981). These initial experiments provide demonstration of the basic operations of setting up and conducting multiple satellite and terrestrial calls among EISN sites, with rudimentary alternate routing and call preemption. The advanced experimental facility based on the PDP-11 Routing/Control Processor (RCP), developed at Lincoln and scheduled for field installation at all EISN sites in FY85, provides a distributed network test-bed for validation of advanced routing and preemption concepts for the Defense Switched Network.

Basic RSC experiments have been carried out in FY83 and FY84 at Lincoln, at each site, as part of PCI installation and checkout, and at EISN Steering Group meetings. These have included RSC-1 (multi-media call setup capability), RSC-2 (satellite/terrestrial alternate routing), RSC-3 (call preemption), and RSC-4 (overflow to an outside net). A demonstration of simultaneous multi-site PCI calling was conducted at the EISN Steering Group Meeting at DCEC in April 1984, involving the DCEC, Ft. Monmouth, and Ft. Huachuca sites. The Army has expressed interest in repeating a coordinated series of these basic experiments specifically involving the Ft. Monmouth and Ft. Huachuca sites, and providing training and familiarization for site personnel and contractors. Lincoln has provided support toward these objectives in the form of planning and advance consultation, and will participate in the experiments to be conducted in FY85.

The initial thrust in experiments with the RCPs has been to validate and extend earlier results (obtained by means of steady-state analysis and call-by-call simulation) for the new routing and preemption algorithms. For a period of time prior to commencement of deliveries of RCPs to the sites, a series of in-house multi-RCP networking experiments is being carried out at Lincoln Laboratory as the first stage of this validation. The sequence of RCP experiments leading up to this activity has been keyed to the step-by-step implementation of the required capabilities. The following experiments have been performed in FY84:

(1) *RCP control of interswitch calls with CCS* (a basic functional requirement of the outboard RCP concept): The call originator goes off-hook on the Lincoln switch and dials "7"; the Lincoln RCP gives a second dial tone, and accepts the remainder of the DTMF digits requesting a destination on the DCEC switch (currently located at Lincoln). The Lincoln RCP then passes the digits by CCS to the DCEC RCP, which dials the digits and completes the call path.

(2) *Phantom calls with preemption between two RCPs*: The Lincoln and DCEC RCPs each generated "phantom-call" traffic directed toward the other, according to preselected probability distributions of call frequency and duration. Each phantom call initiates the same RCP call processing and CCS message traffic as a real call; the only differences are that no real trunks are seized, and that traffic levels can be far higher than would be possible with a few trunks and live callers. The objectives include measurement of CCS message traffic and RCP processing loads, and validation of simulation results.

(3) *RCP control of a three-switch network with tandem routing*: In FY84, phantom traffic experiments with tandem routing were conducted in a network of three RCPs at Lincoln, joined by a CCS network. The essential preparatory step for 3-node networking of real calls in-house at Lincoln was also completed: one of the switches was partitioned, by manipulation of the switch data base, to emulate two independent switches. This work has built toward completion of the 3-node in-house networking of real calls in FY85, which will be followed by three-site networking experiments.

(4) *Multi-RCP phantom traffic networking*: Experiments were run in which two independent RCP processes were running in the same physical PDP-11/44 computer. This is a critical prerequisite for FY85 experiments which will involve up to ten RCPs running in three computers in-house at Lincoln, as well as subsequent FY85 experiments involving all five RCPs after deployment.

The FY84 hardware and software implementation efforts building upon the work of previous years, and the experiments carried out with these facilities, have accomplished two major goals. The first is validation of the basic RCP concept of utilizing a programmable outboard processor to control switch call-handling functions (including preemption) through an attendant console port. The basic strategy and implementation of a critical element of this concept, namely common-channel signaling (CCS) among RCPs, has also been shown to be valid. The second category of major accomplishments in FY84 was completion of essential preparations for the FY85 experimental activity, as noted in items (3) and (4) above. This FY85 work will complete Lincoln's overall objectives in the program, and will serve as a springboard for follow-on applications of the EISN test-bed by the MILDEPs and by DCA.

2.4 SWITCH FEATURES REQUIRED WITH AN OUTBOARD PROCESSOR

In the course of developing the outboard processor and attendant console interface architecture for the EISN test-bed, as described in earlier Annual Reports, the capabilities and feature sets of commercially available switches were reviewed and compared. The RCP and interface designs were based upon the conclusions of this review, as were the ongoing simulations, routing and control experiments, and hardware/software development efforts. Early in FY84, the DCA requested some amplification and supplementary study of switch feature questions in a broader context than EISN test-bed design, namely the feasibility of implementing outboard processors to realize special military functions such as new routing and preemption procedures in a real network incorporating various types of off-the-shelf commercial switches.

Under this concept, outboard processors could control calls going out over DSN trunks by means of the monitoring and control capabilities available through the attendant console ports. The outboard processors could also communicate with each other over a CCS network, as shown in Figure 4. A small study effort, described in Reference 6, was performed to determine and enumerate the switch features required to support new routing and preemption algorithms with an outboard processor.

We found that a modern class 4/5 switch designed for use with CCS appears to have all of the 22 features required to support an outboard processor. The call-handling capacity of a switch outboard-processor combination is limited by the bandwidth of the switch-to-outboard-processor connection and by the ability of the switch's central processor to handle the unexpected demands placed on it by outboard processors. The bandwidth limitation can be overcome by interfacing the outboard processor to the switch using multiple attendant console ports or by using a custom higher-speed interface. The switch-processor limitation can be overcome if the switch processor is sized appropriately and software tasks associated with outboard processors are given high priority.

The outboard-processor concept is being used by Lincoln in the EISN network primarily as a technique for convenient experimental validation of new routing and preemption algorithms. The RCP development also demonstrates the feasibility of implementing these algorithms via outboard control of commercially available digital switches. The other alternative of contracting

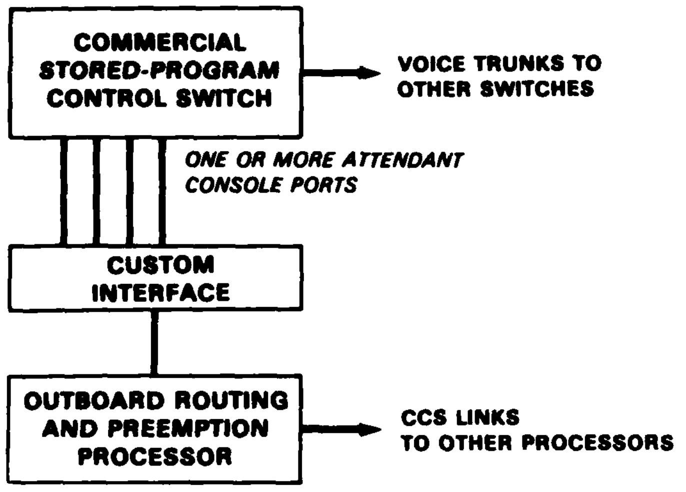

COMMERCIAL STORED-PROGRAM CONTROL SWITCH
VOICE TRUNKS TO OTHER SWITCHES
ONE OR MORE ATTENDANT CONSOLE PORTS
CUSTOM INTERFACE
OUTBOARD ROUTING AND PREEMPTION PROCESSOR
CCS LINKS TO OTHER PROCESSORS
138862-N

Figure 4. Block diagram of an outboard routing and preemption processor connected to a commercial switch via attendant console ports.

with a commercial switch manufacturer to add new routing and preemption algorithms, statistics collection, and network control facilities to a commercial switch suffers from high cost and lack of experimental flexibility, and is not suitable for EISN. In the EISN network, outboard processors (RCPs) control routing and preemption for all calls and implement a CCS packet-switched network. In the DSN, the call-handling capacity of an outboard processor could be increased if calls with routine precedence were controlled by the switch and only higher-precedence calls were controlled by the outboard processor. In addition, the outboard processor could be simplified if it did not have to implement the Message Transfer Part of the CCS protocol but used a separate CCS processor. Such a configuration is shown in Figure 5 where both the switch and the outboard processor have high-bandwidth, standardized connections to the CCS processor. The CCS processor is connected to other CCS processors via CCS links with data rates of up to 64,000 bps. This processor implements a new version of the Message Transfer Part of CCITT No. 7 specialized for the DSN. The outboard processor implements a modified version of the Telephone Users Part of CCITT No. 7 that supports new types of routing and preemption. One such modified version of this protocol is currently being used in the RCPs.

2.5 PLANNING AND DESIGN OF EISN SECURITY EXPERIMENTS

2.5.1 Background and System Issues

The Department of Defense requires that communication systems carrying official traffic must be encrypted if their transmissions are subject to interception by unauthorized parties. For classified traffic, the requirement is met as appropriate within DoD security standards. Unclassified, official traffic can be protected with Encryption for Transmission Only (EFTO) using the

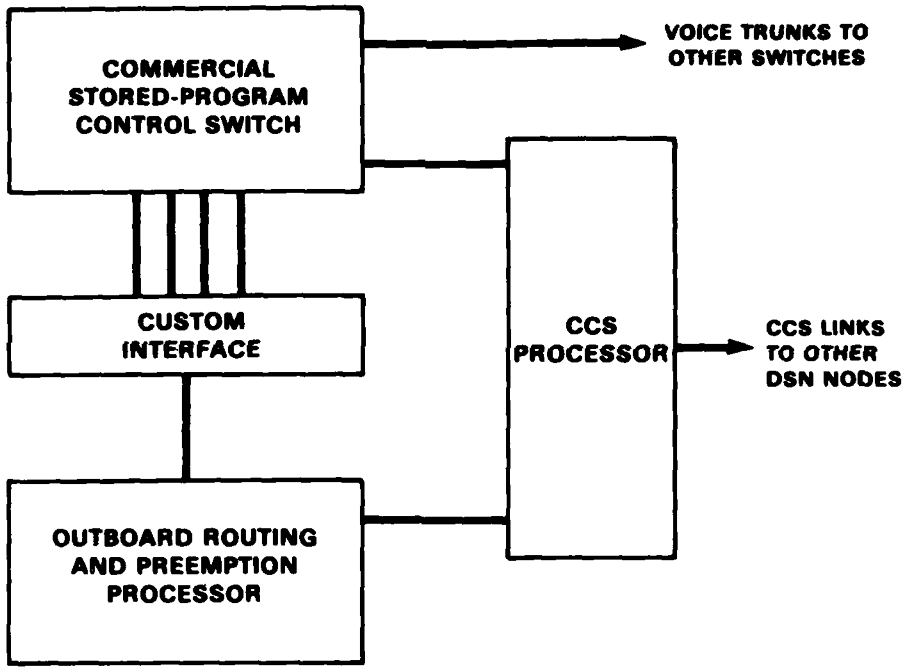

COMMERCIAL STORED-PROGRAM CONTROL SWITCH
VOICE TRUNKS TO OTHER SWITCHES
CUSTOM INTERFACE
CCS PROCESSOR
CCS LINKS TO OTHER DSN NODES
OUTBOARD ROUTING AND PREEMPTION PROCESSOR
138863-N

Figure 5. Block diagram of a DSN node including a CCS processor that performs link-level part of CCS protocol for both switch and an outboard routing and preemption processor.

Data Encryption Standard (DES), provided that each implementation satisfies Federal Standard 1027, "Telecommunications: General Security Requirements for Equipment Using the Data Encryption Standard." If the techniques developed in EISN are to be incorporated into future operational DoD communication systems, therefore, a means of implementing suitable EFTO must be shown to be feasible. In particular, EISN has the potential to serve as a test-bed for addressing a complex problem that appears not to have been fully resolved in the past, namely implementation of EFTO on a broadcast DAMA satellite channel.

A study has been carried out in FY84 aimed at:

(a) Analysis of the impact of EFTO on a DAMA scheme, specifically the PODA algorithm used in EISN;

(b) Definition of the critical issues in EISN EFTO design;

(c) Design of an appropriate system architecture; and

(d) Investigation of implementation alternatives.

The following paragraphs briefly summarize the results of the study. A more complete report is being prepared as a separate Project Memorandum, including more detailed discussions of architectural alternatives, satisfaction of government requirements, initialization and synchronization techniques, and implementation issues. Two significant conclusions were reached: (1) the crypto should be located between the controller/codec and the modem in the Earth Station Interface (ESI), and (2) the design of an EFTO implementation for EISN is feasible, but too complex to be satisfied with simple modifications of currently available DES equipment.

A straightforward approach for insuring that all EISN traffic is encrypted would be to require end-to-end encryption of all traffic using the satellite link; however, this would require separate crypto equipment for each conversation. Implementation of EFTO by bulk encryption of the satellite link is potentially simpler, cheaper, and more easily administered. However, several factors make the implementation of bulk encryption more complicated for EISN than for existing satellite systems. A key issue is that the wideband satellite network is a packet broadcast system based on a repetitive time-division multiple access (TDMA) frame, in which demand assignment is realized by changing slot assignments from frame to frame to accommodate control and data traffic offered by the members of the network. One of the member stations is the "leader," meaning that it establishes start-of-frame timing by transmitting a leader control packet at the beginning of each frame; any station can be leader, and a change can be made any time the current leader develops problems. Any station with outgoing traffic to send must first request the necessary time slots by means of a reservation request that is either piggy-backed on a previous transmission or inserted in a special reserved control subframe. All stations receive all requests, apply the Priority-Oriented Demand Assignment (PODA) algorithm to them, and arrive at the identical conclusions as to slot assignments for the new packets from each requesting station. It is clear, therefore, that any bulk encryption technique used in EISN must support full interconnection of all stations; conventional pairwise encryption management and synchronization is an unacceptable solution.

The critical problem in this broadcast DAMA bulk encryption for EISN is to establish and maintain crypto synchronization among all the stations in the net, including provisions for recovery from synchronization loss and for re-entry or late arrival of a station. Fortunately, the structures of EISN and the PODA algorithm permit important simplifications in the problem, as described below. The keys to these simplifications are the facts that:

(1) The length of each PODA frame is precisely 2^{16} bit times at 3.088 Mbps, and

(2) Each PSAT knows in advance the precise clock count at which any burst in a particular frame may arrive.

The second fact is subject to two perturbations: any station may be off in its transmit timing by up to ± 4 clock times, and any station may transmit a short burst (or none) if it happens to have insufficient data to fill out its reserved slot. The former can be accommodated by a variable delay line, as described below, and the latter is accommodated by clocking the receive crypto 2^{16} per frame, whether data are arriving or not.

An additional perturbation to this simple scheme is that a site which has just been turned on and is attempting to join the net has no prior knowledge of the starting point of the 16-bit time counter. It can, therefore, not recognize leader packets (the normal first step in timing acquisition). This can be accommodated by modifying the modem so that it transmits a unique leader-packet identifier prior to the encrypted contents of the leader, as described below.

2.5.2 Design of an EISN EFTO Test-Bed

Initial attempts were made to design a low-cost experimental EFTO implementation for EISN by exploiting the LC76, a commercial T-carrier data encryption unit available from LINKABIT, with minimal modifications. Upon closer study it became clear, however, that it would be necessary to design a new system in accordance with the approach outlined in this report.

In order to evaluate the possible locations for a crypto system in EISN, first consider the normal configuration shown in Figure 6 of the key elements of the wideband satellite network (WB SATNET) equipment at each EISN site. To the left of the PSAT are (in general) multiple host computers, sending and receiving data packets associated with multiple end-to-end connections between users. All this information is multiplexed into a single data stream by the PSAT, and therefore a crypto system to implement EFTO should be inserted somewhere to the right of the PSAT. The full-duplex Interface Controller/Codec Unit (ICCU) of the Earth Station Interface (ESI) provides error protection by encoding and decoding packets, with possible code rate variations from packet to packet in accordance with instructions contained in the packet headers. Therefore, the ICCU must be able to read the packet headers in unencrypted form and, hence,

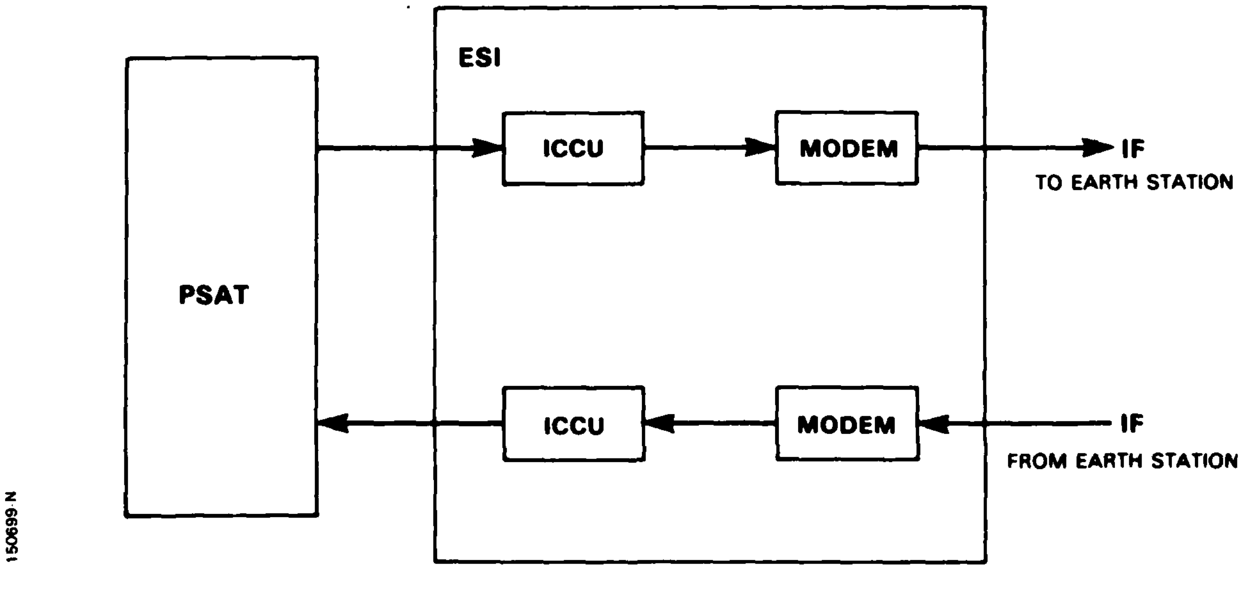


Figure 6. Current system configuration.

the crypto must be located to the right of the ICCU. Since the crypto operates on digital data, and since everything to the right of the ESI modem is analog, the appropriate choice for the location of the crypto is between the ICCU and the modem.

Figure 7 shows the insertion of the transmit and receive sides of a generalized (full-duplex) crypto into the data paths between the ICCU and the modem. On the transmit side, the crypto is

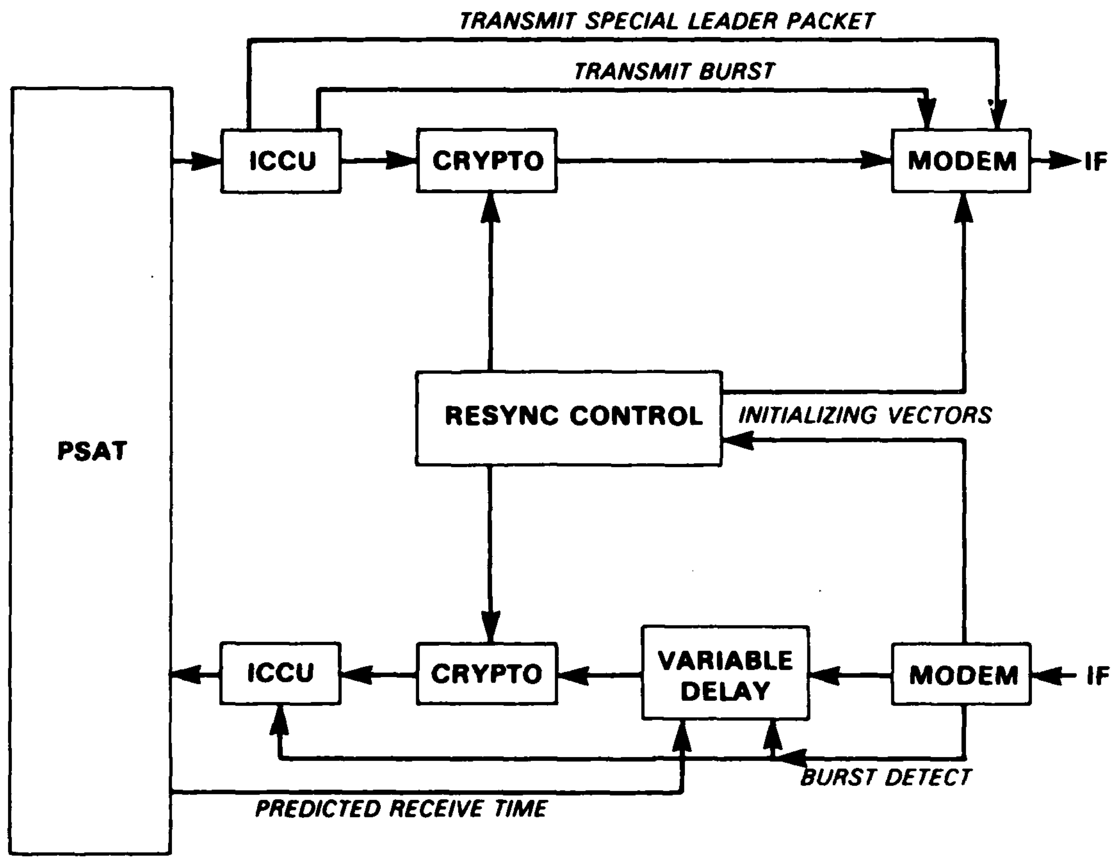


Figure 7. System configuration with EFTO.

clocked by the 3.088-Mbps transmit clock from the modem and, hence, produces 2^{16} output bits per frame, whether or not the ICCU is producing data bits. The "Transmit Burst" signal (which is already implemented in the ESI) is an envelope pulse which will enable the modem only when bursts of (encrypted) data are actually present. A consequence of this design is that the crypto must be operated in a non-feedback mode, so that a transmitting station can be generating the correct keystream bits without knowing what is being transmitted by the other stations in the net. Note the "Transmit Special Leader Packet" envelope signal which has been added to cause the modem to generate the unique leader packet identifier. Both envelopes bypass the transmit crypto, which is a potential cause for concern in obtaining certification for the system (although in this case it seems likely to be certifiable, as explained below). The function of resynchronizing the crypto is symbolically shown in Figure 7; mechanisms for actually accomplishing resynchronization are discussed below.

In the receive side of Figure 7, a variable delay line and a crypto have been added. The variable delay line is essentially an 8-bit tapped shift register which (by selecting the right tap) can introduce up to ±4 bits of offset in the received bit stream, thereby bringing it into synchronism with the expected time of arrival of the burst. The "Burst Detect" signal derived from the incoming data by the modem is compared with the "Predicted Receive Time" signal from the PSAT to determine the correct delay value. Note that the receive-side crypto produces an output bit for each tick of the 3.088-Mbps system clock, but only those which follow assertion of the Burst Detect signal are interpreted by the ICCU. The ICCU knows the time of occurrence of the last bit in the burst, because the burst length is among the items in the burst header.

Two basic approaches to synchronization are possible in the EISN environment: (1) transmission of a new, independent initialization vector (IV) at the beginning of each packet, or (2) maintenance of continuous network-wide synchronization throughout the (extended) intervals between resynchronization events. The former is very straightforward from a system design point of view, but it is unattractive for practical reasons in that the increased system overhead required to transmit a new IV for each packet would seriously degrade overall system efficiency. This is particularly true in view of the relatively noisy satellite channel used for EISN, which implies a need for repetitive transmission or other techniques to insure error-free reception of every bit of the IV. *The second approach appears to be preferable* in spite of the expected complexity of implementing the Resync Control and Variable Delay modules shown in Figure 7, which are required as described below.

The two most critical issues in designing the crypto synchronization mechanisms for EISN are provision for 1027 compatibility, and provision for individual stations to acquire synchronization with an operating network. The 1027 issue is multi-faceted; while many of the details can be left to the designer of the actual equipment, certain key points must be accommodated in the basic architecture, as described below. The sync acquisition functions, which are intertwined with the 1027 constraints, include: network initialization from a cold start, re-establishment of communications with stations that have lost crypto sync, and addition of new arrivals to the network. These points are best addressed in terms of a brief description of normal network operation without crypto, as follows.

When an EISN station first goes on the air, it carries out a received frequency search procedure. If no signals are found within a suitable time-out interval, the station assumes that it is the only one in operation and establishes PODA frame timing with itself as the "leader," transmitting a uniquely identified leader packet at the beginning of each PODA frame. If, instead, the station finds transmissions already in progress, it will acquire bit timing on each received burst while searching for leader packets. Having found a leader packet and hence the start-of-frame time, the station must determine its own round-trip propagation delay time to the satellite, so that it will know the precise amount by which to advance its own transmissions in order to place them in the correct time slots on the downlink. Propagation delay is measured by transmitting a ranging packet in a special time slot called the ranging subframe, whereupon the station can begin requesting capacity as needed for outgoing traffic, thus becoming a fully functioning member of the net. Propagation delay is remeasured often enough during normal operation to guarantee that accumulated timing drift will never cause slot misalignments greater than ±4-bit times within each frame.

The requirement that leader packets be identifiable to new arrivals implies that, in an EISN EFTO system, leader packets must contain a unique unencrypted identifier. The "Transmit Special Leader Packet" signal from the ICCU to the modem has therefore been added in Figure 7 to cause the modem to add the required identifier to every leader packet transmitted. All the information transmitted on the uplink in Figure 7 is encrypted, except the 36-bit burst timing acquisition sequence on every burst and the unique identifier on leader packets, both of which are generated within the modem itself. The two transmit envelope signals in the figure both bypass the crypto, and are therefore expressly prohibited by Federal Standard 1027 unless it can be proven that the bandwidths of the bypass signals are sufficiently low that no significant amount of information could be leaked around the crypto. Since the envelope pulses occur at a very low rate (once per PODA frame, or about 47 Hz), and since no significant variation in their timing or other properties is possible without crashing PODA, it appears likely that 1027 approval could in fact be obtained for these bypasses.

Several modifications must be made to the receive side of the modem in order to achieve initial acquisition, ranging, and crypto synchronization. The first is addition of a decoder for the unique leader packet identifier, so that start-of-frame time can be reported by a signal to the ICCU. Since the receive crypto will be unsynchronized, the ICCU will be unable to interpret any bits in the received bursts. Round-trip delay can be measured without being able to interpret the contents of packets, however, and can therefore be done immediately by transmitting a ranging burst in the subframe reserved for the purpose.

Crypto resynchronization must take place often enough to permit new arrivals, or stations attempting to regain synchronization after losing it, to suffer no more than a reasonably small delay before joining the net. While the exact choice of a resynchronization interval is probably best left for the developers of an actual system, it seems that a suitable figure would be comparable to the time required to join the present-day unencrypted EISN network, which is typically a few tens-of-seconds to a few minutes. There are five important components in the resynchronization process: (1) the leader station must generate an IV; (2) the IV must be transmitted unencrypted, and in a clearly identified location, so that unsynchronized stations can receive it; (3) special measures must be taken to insure that the IV is received correctly, in spite of the noisy satellite channel; (4) in accordance with Federal Standard 1027, each transmit crypto must complete a set of internal tests upon resynchronization to insure the integrity of all its crypto functions; and (5) provisions must be made for all stations to cut over to the new IV simultaneously.

Component (1) implies that the IV must be generated within the transmit side of the modem in Figure 7 (modified as necessary), since all data from the ICCU or the PSAT get encrypted. Component (2) implies that the modem must be modified so that it attaches yet another special identifier to the particular leader packet which contains the IV. The latter action would be initiated by the Resync Control module.

Component (3) of the resynchronization process cannot be satisfied by the error-correcting coding built into the ICCU, since unsynchronized stations would be receiving a garbled bit stream. The leader must therefore have recourse to a redundancy scheme such as repetitive transmissions to increase the probability of correct IV reception, entailing substantial transmission overhead. Moreover, the modem must be modified to autonomously detect each incoming IV and pass it to the Resync Control module in Figure 7.

2.5.3 Test-Bed Configuration and Experiment Planning

The output of the study has been an outline of the design for a viable experimental crypto implementation in the EISN test-bed. While no plans currently exist for such implementation, the design is available for future consideration.

It is clear from the discussion above that sites with the EFTO modifications would be incompatible with unmodified sites. Since it would be desirable to conduct extensive tests of a crypto implementation before investing the time and effort to outfit the entire network with the modification, it appears that it would be preferable to begin with only two sites. There would then have to be a period of mutually exclusive operation of the network. Experiments would be planned and executed to test all aspects of the functioning of the crypto system.

2.6 PRELIMINARY INVESTIGATION OF APPLICATION OF EXPERT SYSTEMS TECHNIQUES TO SYSTEM CONTROL

The development and test of new routing algorithms has been a major focus in this program. Specific algorithms have been identified which offer substantial performance improvements under conditions of stress, as reported earlier. The assumed environment was a network of routing and control centers joined by communication trunks and CCS links, operating with substantial autonomy but with careful coordination among centers as to algorithms and parameters in use.

It is interesting to consider how best to achieve this coordination with respect to routing algorithms as well as the other aspects of higher-level system control, as the Defense Switched Network evolves toward the environment assumed above. It appears that the control of such a sophisticated system would constitute a set of highly complex, diagnostic and deductive tasks for which human operators would require skills developed over long periods of training and experience. The potential exists for applying expert systems technology, developed in the artificial intelligence research community, to this system control problem as a means to capture and retain the knowledge and skill of expert human system controllers, making it possible for new or less-skilled operators to progress through tasks and decision sequences almost as well (up to a certain point) as if the expert were actually present.

During FY85, Lincoln will be undertaking a study effort aimed at defining applications of expert systems techniques to system control. An initial look was taken during the latter part of FY84 at the nature of key system control problems, and at the possibility of expert systems applications. Contacts were made with Computer Sciences Corporation (CSC) and Honeywell on the basis of recent study efforts carried out by both organizations in related areas for DCEC. Both studies were concerned primarily with expeditious restoral of dedicated circuits for high-precedence military users in the face of circuit disruption due to equipment failures or hostile actions. CSC had focused on study and evaluation of algorithms for selection of new routes for failed circuits,[7] and Honeywell had implemented a hardware/software test-bed for studying man-machine interaction in the control of such a system.[8] After discussions with key people involved with the two studies, we concluded that the problems that would be faced by the human operators appeared to be well suited to an expert systems approach; that is, they are complex and demanding, yet they appear amenable to sufficiently complete and detailed definition to be incorporated in a computer system. Promising areas include diagnosis of system faults based on arrays of remote monitoring inputs, and the development of an interactive system model and data-base management system.

3. EISN ROUTING AND CONTROL FACILITY INSTRUMENTATION

3.1 INTRODUCTION

The purpose of the EISN system is to provide a system-level test-bed for the evaluation of advanced communications networking techniques, including survivable network routing algorithms using a mix of transmission media, for application in the DSN. Figure 8 illustrates the phased development of EISN, beginning with an interim routing/control experimental facility based on a Packet/Circuit Interface (PCI) and a Telephone Office Emulator (TOE), and culminating with deployment of an advanced facility including off-the-shelf digital circuit switches and flexible outboard Routing/Control Processors (RCPs). The potential future development of an expert system controller is also noted in the figure. An overview of the experimental architecture and application of EISN, together with the broader Wideband Satellite Network of which it is a part, was given in References 9 and 10.

Prior to FY84 the key accomplishments included development and deployment of the PCIs, TOEs, and Internet Packet Gateways (IPGs) at the EISN sites. Experiments performed with this network in FY83 and FY84 have been aimed at validation of the basic test-bed concepts, as noted in Section 2.3 of this report, including PCI-based satellite/terrestrial alternate routing, call preemption, and packet/circuit interoperability. In addition, a series of packet data experiments have been carried out between the IPGs at DCEC and Lincoln.

A major FY84 effort has been the development of the RCP hardware, software, and special interfaces, and integration of this equipment with a mix of off-the-shelf digital switches, as described in the following two sections. These efforts have resulted in a unique routing and system control test-bed capability which will be used for experimental validation of routing and preemption algorithm results obtained earlier by analysis and simulation. The Army has provided separate funds for replication of two RCPs and associated interfaces, to be installed at Army sites to support their participation in EISN test-bed activities.

3.2 RCP HARDWARE AND DIGITAL SWITCH INTEGRATION

It was reported in the FY83 Annual Report[2] that United Technologies LEXAR UTX-1200 switches had been selected for the Lincoln and DCEC EISN sites, and that both switches had been delivered and installed at Lincoln. In the course of developing the RCP concept and of making a selection of switches, a study was initiated of the general requirements for switch features that will permit implementation of DSN routing and preemption capabilities by means of outboard processors, as reported in Section 2.4. The UTX-1200 switches meet these criteria, with certain expectations that have been accommodated by the use of special trunk interfaces. Specifically, in order to (1) enable physical preemption of trunk calls in progress, and (2) report availability of individual trunks upon call completion, the Wescom CO Controller and E&M Controller interfaces have been procured, as reported last year. In order to send a second dial tone, and to

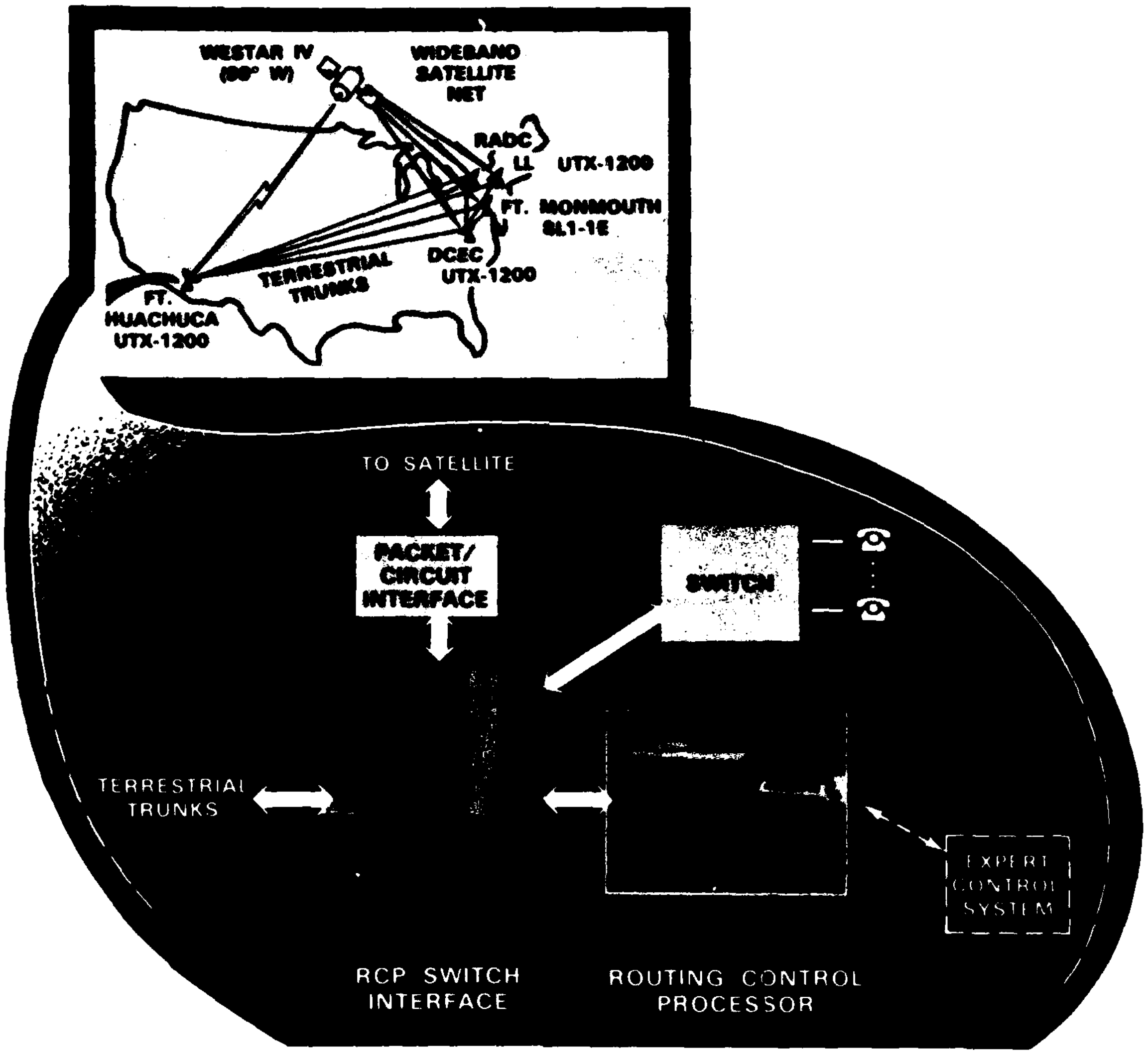


Figure 8. Advanced routing and system control test-bed.

receive the calling party's dialed digits, the Wescom DTMF Controller interface was procured, also as reported last year. The RCP control interface to the UTX-1200 requires a modified attendant console and a custom microprocessor-based interface. In addition, each RCP requires a shelf of telephone modems for CCS and for log-in ports, and a set of switches is required to permit convenient selection of the various remote and local interface combinations. In FY84, all these interface subsystems were integrated in a single cabinet for the Lincoln EISN site, and an identical interface cabinet has been integrated for the DCEC site. Each cabinet required a substantial amount of effort in the wiring, installation, and step-by-step functional checkout of the dozens of individual circuits and modules contained in it. The main advantage of this integrated interface cabinet approach is that it allows the detailed checkout to be performed once for each cabinet, at Lincoln, followed later by a very simple field installation at DCEC. This integrated approach also tended to help organize the tasks of setting up, modifying, and checking out the switch data bases to reflect the various types of trunk and subscriber circuit configurations required in the RCP experiments.

The added funds provided by the Army have been used to procure the components for two additional interface cabinets, and to procure a new SL-1 switch/RCP interface and a new UTX-1200 switch, as follows. The Army decided to implement an advanced EISN routing/control facility at Ft. Monmouth, based on a small Northern Telecom SL-1LE telephone switch already in hand whose parameters and capabilities were roughly equivalent to those of the UTX-1200. After close examination, we determined that the SL-1LE would function satisfactorily with the three Wescom interfaces described above, namely the Tone TX/RX, the E&M Controller, and the Central Office Controller. The Attendant Console Interface would have to be different from the Lincoln-built interface to the UTX-1200, however, to accommodate the proprietary design features of the SL-1LE. Arrangements were made with Northern Telecom to design and build the required Attendant Console Interface under subcontract to Lincoln, with delivery scheduled for early FY85.

The Army Communications Command decided to equip the EISN site at Ft. Huachuca with a UTX-1200 switch, and the purchase was completed by Lincoln. Installation of the switch took place at Ft. Huachuca in June 1984. The required number of copies of the Wescom special interfaces have been purchased, an additional UTX-1200 attendant console interface has been constructed, and cabinets have been purchased to hold all the interface equipment for the Army sites. This equipment is in the process of assembly and checkout to support FY85 delivery to the Army sites.

3.3 RCP SOFTWARE DEVELOPMENT

RCP software development has led to a complex, real-time software structure that supports both real-call demonstrations and phantom-call distributed simulation tests. Current software supports real calls originating from RCP telephones in networks with up to three RCPs. In addition, distributed simulation runs in networks with up to 3 nodes can be made when phantom calls are offered from phantom-call controller software processes within RCPs. All RCPs accumulate call statistics during such runs, which are combined and summarized at the end of a run.

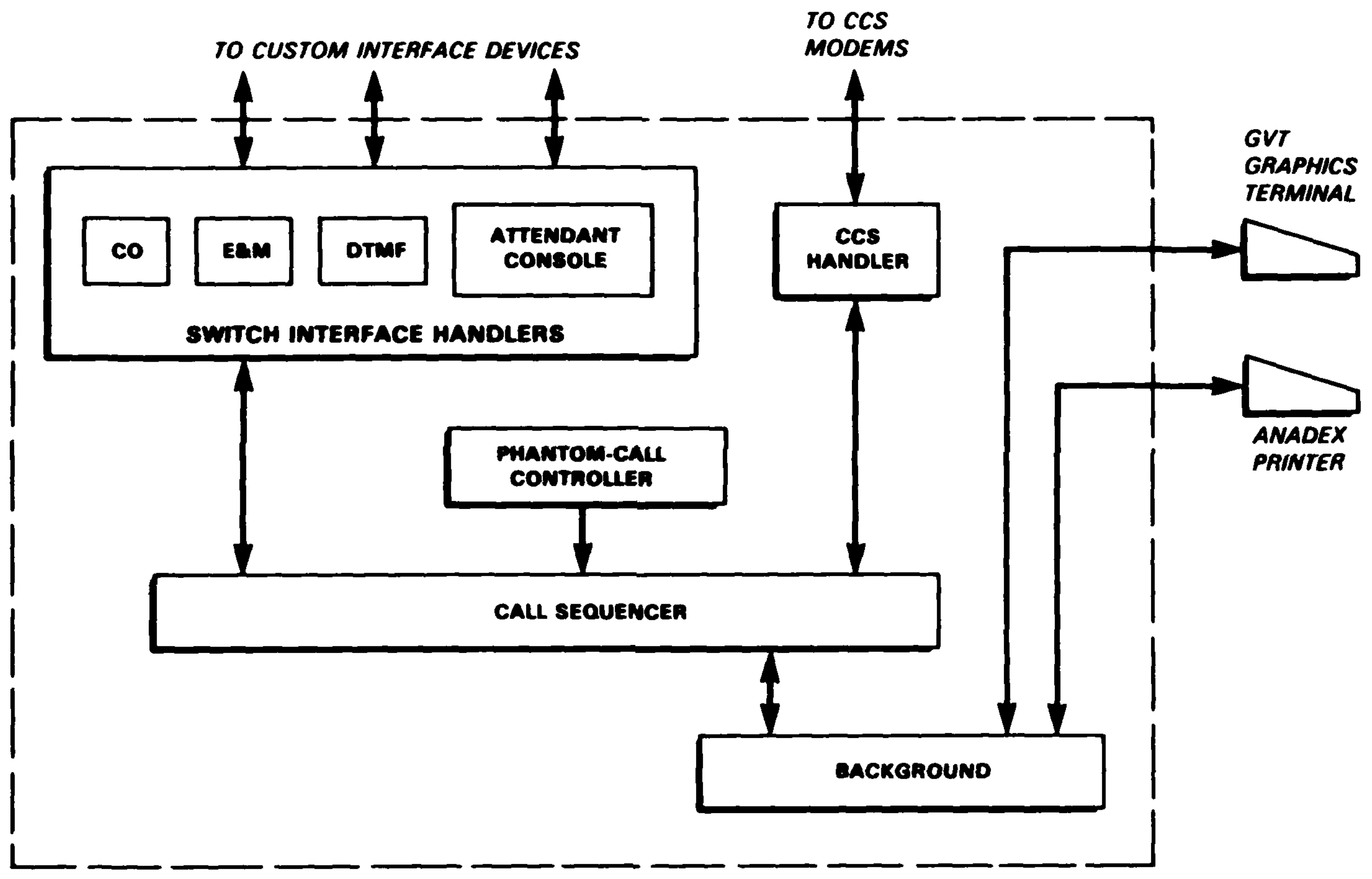


Figure 9. Current RCP software structure.

150744-N

Results obtained in this way can be compared with results obtained with the off-line call-by-call simulator to validate simulator results and the implementation of routing and preemption algorithms within the RCP. Distributed simulator runs can also be used to debug and test the RCP protocols and the logic required to support the new routing procedures.

A block diagram of current software structure of the RCP is presented in Figure 9. It includes multiple software processes that communicate through shared memory segments provided by a real-time version of the UNIX operating system called VENIX. Software processes include:

(a) A foundation process (not shown) to bring the system up and recover from software errors;

(b) A CCS handler to implement the link-level component of the CCS protocol using a TTY link to monitor link activity and restart transmission if the link goes down, and to keep statistics on CCS transmissions;

(c) A Central Office (CO) trunk interface handler used to monitor and control voice trunks going from the RCP switch to a Bell System central office switch;

(d) An E&M trunk interface handler used to monitor and control voice trunks that use E&M signaling and go from the RCP switch to the PCI or to another RCP switch;

(e) A Dual-Tone Multiple-Frequency (DTMF) interface handler used to send and receive DTMF tones on the attendant console headset;

(f) An attendant console interface handler used to monitor the lights and display on the attendant console, and to perform the same operations that an operator sitting at an attendant console can perform;

(g) A phantom-call controller to generate and terminate phantom calls, and keep statistics on them;

(h) A call sequencer to implement new high-level CCS protocols and the logic required to support new routing and preemption procedures, and to keep statistics on all calls;

(i) A background process to interpret and collect statistics and provide a user interface; and

(j) A timer process to provide time-outs to detect error conditions for the CCS protocol and to initiate periodic statistics logging when required.

The user-interface built into the background process is designed to simplify RCP control and monitoring. The RCP is controlled by selecting and altering the values of items on menus displayed on an intelligent computer terminal. This simplifies the task of setting up distributed simulation runs, of examining statistics, of selecting routing and preemption procedures, and of altering network controls. For example, statistics and information describing the status of real and phantom calls are displayed and updated in real time on a monitor screen. Simulation runs that use phantom calls can be be rapidly set up and initiated by changing items on a menu, and RCP behavior can be examined by watching the monitor screen as it is updated in real time.

Tests using RCPs in 2- and 3-node test-bed networks have been used to debug all important software components including logic and protocols, user interaction via menus, statistics collection, interface handlers, and the complex interaction between RCPs that takes place in networks. These tests led to a number of important modifications in RCP software and also demonstrated that it is possible to run multiple virtual RCPs within each PDP-11/44 without extensive changes to the existing operating system. In one test, two virtual RCPs were operating within one PDP-11/44 and a third was within a second PDP-11/44. Distributed simulation runs within this 3-node network yielded results that were similar to those obtained with the call-by-call simulator, and also uncovered a number of complex resource-allocation bugs in the current software. This technique of using many virtual RCPs within each PDP-11/44 will make it possible to perform distributed simulations in test networks with five PDP-11/44s and 10 to 20 virtual RCPs.

Software development during FY84 was preceded by an initial design phase during which the overall requirements of RCP software were specified, a software design philosophy was established, and design documents were written to provide initial specifications of:

(a) The real-time software structure of RCP software processes and techniques to communicate between processes;

(b) The Dialing Plan to be used in networks containing RCPs, PCIs, and TOEs;

(c) The CCS protocol to be used in RCPs and PCIs to support new routing and preemption procedures;

(d) The RCP user interface including descriptions of menus, statistics displays, and single-key system control commands; and

(e) Specifications of the interface protocol to be used between the RCP and custom interfaces built by Wescom and Northern Telecom including the E&M, DTMF, and the Attendant Console interfaces.

These documents are being used as design guides as RCP software is being written. In addition, the user-interface document was recently updated and rewritten to create an RCP users' guide that describes how to use and control the RCP from an RCP terminal. Another document also has been written recently that describes the structure and contents of an RCP configuration file. This file is read in when an RCP is started. It contains network topology information, Bell System numbers to dial up for voice trunk and CCS links, RCP dialing plan numbers for trunks and stations, the routing table for this RCP, and other information required to start up an RCP.

Software development proceeded from control of local real calls in a 1 RCP/switch node, to control of real and then phantom calls in 3-node networks. Major software additions were required to add the topological information, more complex CCS protocol and routing logic, statistics, and internal CCS and inter-process communication queueing required for multiple-node networks. Major additions also were required to route calls over satellite links. This entailed a new link-level CCS protocol to support a broadcast satellite connection and extensive additions to the logic used in the call sequencer to support satellite links. In addition, a number of features were added to simplify debugging in multi-node networks and distributed simulation runs. These included a trace feature that writes internal RCP events to a trace file to determine the logic flow within an RCP, statistics logging to write statistics information to a file at pre-specified intervals, and error- and warning-message handling that writes error and warning messages to a special system window on the control terminal.

Extensive software development and debugging were required to develop the switch interface handlers, call sequencer logic, link-level CCS protocol, and the user interface. Switch interface handlers orchestrate interactions beween the switch, the four custom interfaces, and the RCP. Their development required writing software for the 6802 microprocessor in the CO trunk interface controller because Wescom did not have the manpower for this project. CO software was written on an HP development system using the E&M trunk signaling software as a model. Call sequencer logic was initially described in the CCS protocol document. Then it was tested extensively and modified to handle unforeseen states and error conditions that were found during network tests. The link-level CCS protocol implemented in the CCS handler was also initially described in the CCS protocol document. Network tests led to additions to the error recovery capability and statistics collection included in the CCS handler. The user interface provided by the background process underwent a number of revisions to simplify control of distributed simulation runs in the RCP. These included the addition of statistics logging during simulation runs and the addition of statistics similar to those in the call-by-call simulator.

Software additions are planned for FY85 to support real-call demonstrations in a network with up to 5 nodes, and distributed simulation runs in a network with 10 to 20 nodes. These demonstrations will use modified forward routing and mixed-media routing with guided preemption. They will require additional logic and expanded CCS protocols to support these routing and preemption procedures.

3.4 PACKET/CIRCUIT INTERFACE (PCI) INTEGRATION

The major PCI integration activity in FY84 has been development of voice and CCS links between a PCI and a RCP-equipped switch. These links will provide satellite trunking in the advanced EISN test-bed. Other PCI work in FY84 emphasized support of experiments at the field locations, and implementation of modifications to improve convenience and dependability of operation. Also, opportunities were taken to correct system problems as they were discovered. The objective of transferring PCI technology to military personnel and contractors was pursued through preparation of a manual, as described below.

3.4.1 Voice and CCS Link Between PCI and RCP

The routing and connection of calls in the RCP network are controlled by CCS based on CCITT No. 7. The several voice trunks between any two sites are paralleled by a single dialed-up telephone connection on which CCS is sent between RCPs by modems to control all the voice trunks. For controlling voice paths over the WB SATNET, the CCS shares the same satellite path. It is the function of the PCI to prepare CCS messages from its RCP for transmission over the wideband satellite. In addition, the PCI must packetize the speech routed over the WB SATNET by the RCP and initiate changes in the site's stream reservation as the traffic from the RCP over the WB SATNET increases or decreases. The stream reservation is a "promise" by the WB SATNET to support a certain average transmission rate from a host on one of the PSATs.

During FY84, considerable changes and additions to the PCI UMC-Z80 code were made to enable the PCIs to handle CCS traffic. In particular, it was necessary to modify the basic timing structure of the program in order to accept data at 9600 baud from the RCP without losing speech samples from the digital channel bank. The capability for the PCI to support CCS traffic was established late in FY84. Additional extensions to the PCI software were in progress to support the associated speech traffic from the RCP.

3.4.2 PCI Support

As we gained experience in using the PCI at the field sites, several steps were taken to make that use easier and more dependable. An automatic bootup procedure was installed at field sites so that the PCI would automatically reload and start its programs after a power outage. A kit was sent to each site to correct a flaw in the TOE that caused it to malfunction after one of its phones rang but was not answered. A manual for EISN users, including a detailed section on the use of the PCI, was prepared by RCA under contract to the Army. Extensive briefings and detailed documents prepared by Lincoln people contributed importantly to that manual.

At the Lincoln site, the voice path between the PCI and the LEXAR UTX-1200 telephone switch, through the switch's peripheral trunk controllers, was tested successfully. (Calls were set up manually, not yet by CCS.) At Ft. Monmouth, direct connections between the PCI and the Northern Telecom SL-1 switch were used successfully. Both connections by 4-wire E&M trunks and by T1 carrier were used at Ft. Monmouth.

3.5 EISN EXPERIMENT CONTROL AND SUPPORT

Operation of a switch/RCP facility involves a number of setup and control functions. On an ongoing basis, because the system is still evolving, there is a requirement for distribution of new software from the development facility at Lincoln to each RCP. In order to run actual experiments, telephone calls must be dialed-up for CCS and voice trunks, RCP configuration files and processes must be set up and controlled, the PCI must be initialized and controlled, and performance data and statistics must be collected. These functions can be managed by hand, with multiple directly connected terminals, when only one or two RCPs are involved in an experiment;

the complexity of controlling multi-site experiments will quickly become excessive, however, with increases in the number of deployed RCPs.

In order to make it possible for one or a few individuals to initialize and control multi-site experiments, a degree of automation and centralization is being introduced. Modems and error-correction units have been purchased for software downloading and log-in control. Each site will have matching modems and error controllers in its RCP/switch interface cabinet. Software facilities have been designed and are being implemented for multiplexing of multiple terminal connections to remote sites through one or two computers at Lincoln. These automation support facilities will be integrated and utilized on an evolving basis during FY85, as RCPs are installed at remote sites and multi-site experiments are conducted.

4. DATA PROTOCOLS AND VOICE/DATA EXPERIMENTS

For the past two years, our goal in the area of data protocols has been to explore the performance of the DoD standard protocols, TCP and IP. IP (the Internet Protocol) supports the delivery of datagram packets in an Internet made up heterogeneous networks interconnected with gateways. IP datagrams are guaranteed neither to be delivered, nor to be delivered (if they are) in the order in which they were offered to the Internet. To provide the ordered, reliable delivery needed for applications such as file transfer or interactive terminal support, an additional protocol layer is needed. TCP (the Transmission Control Protocol) provides these services as well as end-to-end flow control to allow a receiver of a packet stream to control the rate of transmission by the sender.

During the first year of this activity we developed measurement tools, extended the capabilities of our IP/ST gateways, and carried out experiments that examined the behavior of TCP file transfers in competition with other data traffic. During this second year we extended the measurements to include situations in which the data traffic contended with packet-voice traffic for network resources, and we conducted a study of the issues to be faced in implementing a TCP that would aim to achieve good performance in a general network environment. In this report, we summarize the experiment results and present our conclusions from the study of TCP implementation issues relative to performance.

4.1 SUMMARY OF EXPERIMENT RESULTS

4.1.1 Data Experiments

During FY83, we conducted a series of experiments in which a TCP file transfer contended for a channel of fixed capacity with other IP data traffic having Poisson arrival statistics. The experimental configuration and results were discussed in detail in our previous Annual Report.[2] We repeat our conclusions from those experiments here as background for the discussion of the issues facing a TCP implementation.

The experiment results showed that an aggressive transmission policy that accepted packet losses of a few percent could achieve significantly higher file-transfer rates than a policy that adjusted its rate so that no losses occurred. Factors-of-two-or-more improvements in rates were obtained at the same time that higher channel utilization was achieved. On retransmission policy, we concluded that only the first segment on a queue of outstanding (unacknowledged) segments should be retransmitted when a time-out occurred.

The results also suggested that networks could be operated successfully with high average link utilization and that, under such conditions, the buffer requirements in the nodes would be modest.

4.1.2 Voice/Data Integration Experiments

Our explorations of voice/data integration made use of the configuration shown in Figure 10. We used Measurement Hosts (MHs) to generate and measure simulated voice and data traffic, and Packet Voice Terminals (PVTs) to place real voice calls against a background of simulated traffic. The MHs were PVTs with special timer cards providing precise timing for packet generation rates and delay measurements. One set of MHs generated ST voice packets with a multi-talker statistical talkspurt model. The other set generated either IP datagrams with Poisson statistics, or it simulated file transfers using the TCP/IP protocol model. Measurements included counts of packets sent, received, retransmitted, etc., as well as delay histograms. Instrumentation in the gateways provided additional information about traffic characteristics.

Most experiment runs made use of the direct connection between the gateways since the path through the PSAT and the satellite was not always available, and we observed no effect other than overall delay in going through the satellite. Parameters for the voice talkspurt model were

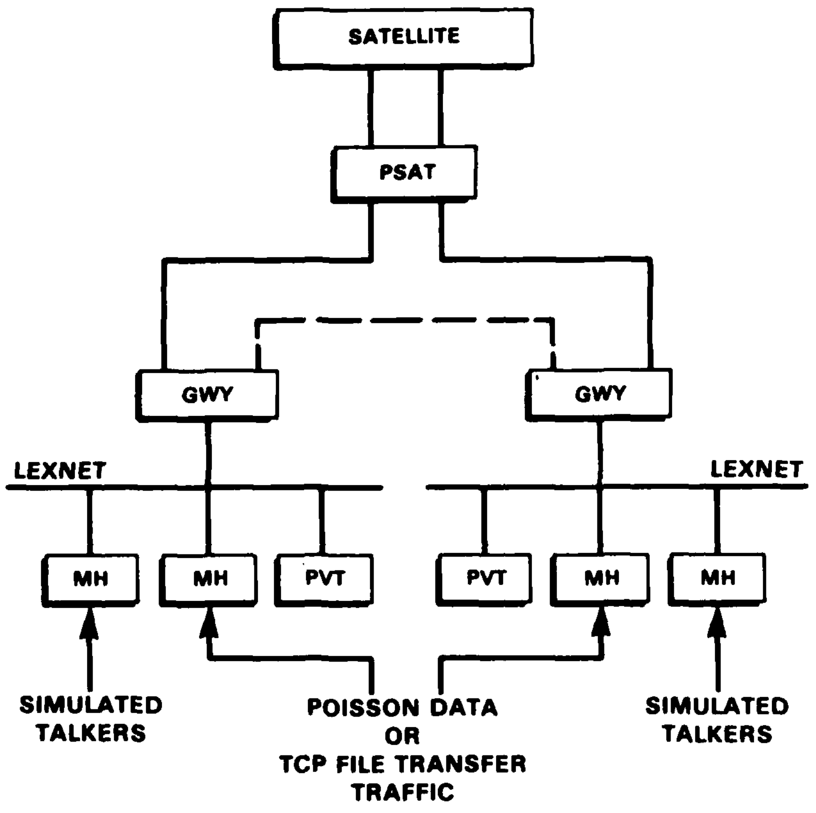
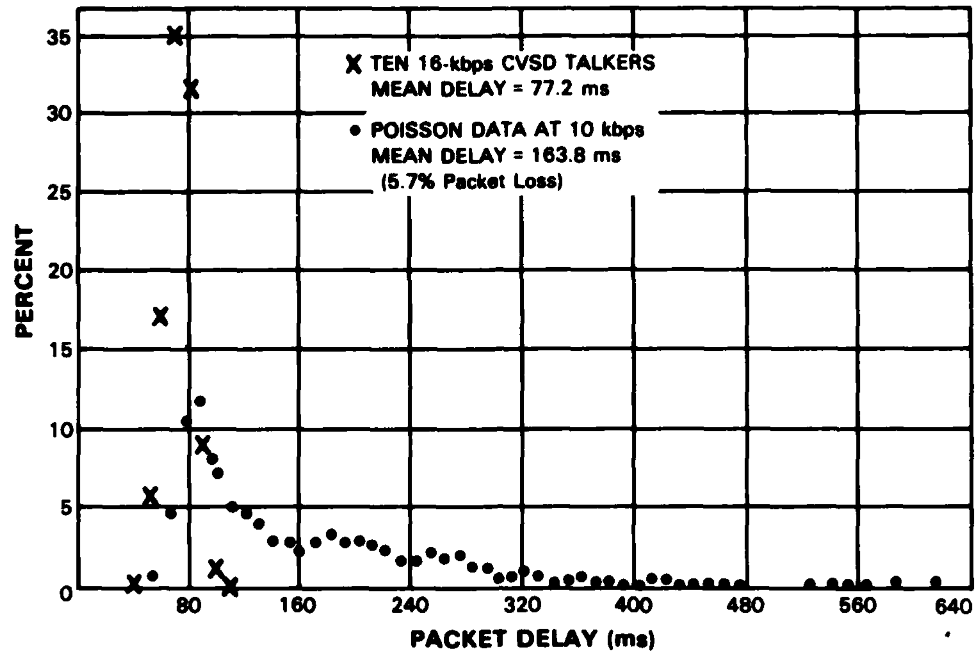


Figure 10. Voice/data integration experiment configuration.

set for ten 16-kbps (CVSD rate) conversations with packets containing 40 ms of speech transmitted only during talkspurts. The voice traffic was given priority in access to the link which had a fixed capacity with the characteristics of a WB SATNET stream. Such a stream offers a periodic opportunity called a "slot" in which the gateway can send a group of packets of predetermined maximum total length. The data traffic rates were varied in a series of experiment runs to explore the extent to which the leftover capacity could be utilized. In some runs, the Poisson traffic model was used to measure data delay as a function of offered load. In others, the TCP file-transfer model was used to determine the rate at which a single file transfer could proceed through the time-varying residual channel. The capacity of the link connecting the gateways was set to a value roughly twice the average voice requirement.

In order to efficiently utilize the network capacity, we added a gateway-to-gateway (GTG) fragmentation and assembly mechanism to the gateways. The use of GTG fragments allowed efficient filling of the stream slot without regard to the sizes of the IP datagrams on the data queue. Without some kind of fragmentation, IP datagrams larger than the slot size could never be sent, and others might wait long periods for a slot with enough leftover capacity. The IP protocol defines a fragmentation mechanism that could be used in this situation, but IP fragments have a relatively high header overhead (20 bytes per fragment compared with 8 for GTG fragments) and, since they are reassembled only at the destination host, their use in a general context would increase the number of packets that downstream networks would have to handle. Further, the reassembly of IP fragments is a complex process since fragments may arrive out of order, and there is no straightforward way to decide when to stop waiting when a missing fragment is detected. In contrast, the reassembly of GTG fragments is relatively simple since the fragments arrive in order, and the failure of a fragment to arrive in its expected order is a clear signal that the reassembly can be aborted. Our implementation did not attempt to provide reliable delivery of reassembled IP datagrams by using any sort of acknowledgment/retransmission mechanism. This policy is consistent with the definition of IP datagram service.

Overall, the results showed that the gateway multiplexing mechanisms worked as intended. Voice traffic received the desired preferential treatment, with throughput and delay almost unaffected by the level of contending data traffic. With overall traffic levels below saturation of the gateway port to the LEXNET, we observed only a small increase in mean voice packet delay and dispersion as data traffic increased. Concurrently, we observed a marked increase in mean data delay and dispersion (as well as data packet loss) as channel capacity limits were approached. Figure 11 shows histograms of measured delays for a typical experiment run in which Poisson data traffic averaged about 80 percent of the mean voice-traffic load. For this run, overall offered traffic was about 90 percent of channel capacity. The plot shows a rather tight and symmetrical distribution for the voice-packet delays, and a skewed distribution with a long tail plus some lost packets for the data. For really high traffic loads that pushed the LEXNET port into saturation, we observed voice-packet losses as well. The ST protocol is designed to avoid such load conditions by refusing connections that would saturate the communication capacities. Unfortunately, IP lacks any mechanism for dealing with such a situation, and an overload of data traffic can thus spoil packet-voice communications.



150697-N

Figure 11. Delay histogram: voice/data multiplexing experiment.

The results from experiments using the Poisson data traffic showed that the GTG fragmentation and reassembly capability allowed about a 90-percent utilization of the link capacity, with data losses of a few percent and acceptable values for mean data delays and delay dispersions. The best results obtained with the TCP file transfer showed about 85-percent utilization. In this case utilization was lower because the packet losses caused retransmissions, and some transmission opportunities were missed waiting for time-outs. This performance is much better than we achieved in earlier experiments without GTG fragmentation, but a quantitative comparison is not meaningful because the performance without fragmentation depends very strongly on the particular choice of data-packet size in relation to voice-packet size and stream slot size.

Channel utilization could be improved by increasing the size of the output data queue in the gateway at the expense of a corresponding increase in average delay and delay variance. It is clear from both queueing theory and experience that the price in delay increases rapidly beyond the 85- to 90-percent region of average utilization that we have been exploring experimentally. Excess buffer capacity is likely to be detrimental to overall performance because it can result in large delay variances in situations such as ours where the effective channel capacity for the class of traffic varies rapidly. Protocols such as TCP that use a measurement of round-trip delay to set retransmission time-out values perform poorly in situations where high delay variances are encountered. They will tend to retransmit excessively if the estimated delay is lower than the actual, or waste opportunities if the estimate is too long. By limiting queue lengths (ideally on a connection basis, for fairness), network nodes can reduce delay variance and improve overall performance for such protocols.

It should be noted that the values for channel utilization obtained in these experiments are strongly dependent on the particular combination of voice-traffic load and link capacity chosen for the experiments. If the number of talkers had been larger and the link capacity correspondingly increased, we would expect to achieve better utilization of the residual capacity because there would be less variation in the voice load from moment to moment. Similarly, fewer talkers in a smaller channel would lead to greater variability and less success in utilizing the residual capacity.

4.2 PROTOCOL ISSUES

The main conclusion from our experiments was that the TCP protocol can achieve good performance under conditions of heavy network load if its critical parameters are set for a good match to the network characteristics. In our experiments, these settings were made using knowledge about the network that was not directly available to the TCP module running in our measurement hosts. The experimenter set the parameters manually and varied them to find optimal operating points. In this section, we address the issues involved in implementing a TCP module that could automatically adapt to perform well under a variety of network conditions. For example, hosts on the present Internet can experience differences in network characteristics that can limit useful file-transfer rates to values that range over three orders of magnitude. Achieving good performance over such a range of conditions is a significant challenge for a TCP implementation. Our goal here is to examine the mechanisms that must operate to adjust flows in the Internet, and to consider the problems that must be addressed by the implementers of the IP and TCP modules that will support the transfers.

The basic mechanism that regulates the rate of flow is the interaction between the TCP window parameter and the delays introduced by the network. TCP deals with a data stream that is treated as a numbered string of 8-bit units called "octets." The window (W) represents the number of unacknowledged octets that the sender is allowed to have in flight at any one time. At the start of a transmission, the sending TCP will create one or more segments containing at most W octets, and launch them into the network. It will then wait for acknowledgments (ACKs) to arrive in packets sent from the receiving TCP. (TCP can piggyback the ACKs in data packets if there is concurrent flow in the reverse direction, but for the typical file-transfer application there is no substantial reverse flow, and the ACKs will be carried in packets that have no other function.) The time taken for a segment to be acknowledged is called the round-trip time (RTT). As ACKs arrive at the sender, new window space becomes available and new segments can be created and launched. The interaction between the window and the RTT limits the transfer rate to a value that can be no greater than W divided by RTT. The actual rate of transfer can be much lower, either because the host computers cannot keep up with the CPU and/or IO loads involved in supporting the rate, or because packets are lost in the net or retransmitted unnecessarily. For the purpose of this discussion, we assume that host computer capabilities are not limiting factors and focus only on the effects of network delays, packet losses, and retransmissions.

If the Internet never lost packets and, consequently, retransmission was never required, the interaction between the window and the return of ACKs would result in a nearly optimal adaptation of the flow to the limit (W/RTT). If the window is large enough to keep the "pipe" between sender and receiver full, W/RTT will correspond to the limit set by the capacity of the network path and its delay. If, as is often the case, the window is not large enough, the flow will be limited by the window rather than the network capacity. If the window is too large, the sender may launch packets in a burst at a rate sufficient to momentarily overload gateway buffers, thereby causing packets to be dropped needlessly. In any event, the Internet model assumes that packets can be lost, and a retransmission mechanism must be provided to handle the possibility of such loss. This mechanism is not likely to work perfectly, and actual flow rates will be less-than-optimal values due to wastage of channel capacity with redundant retransmissions and to loss of transmission opportunities while waiting for time-outs.

The issues we wish to address are those faced by IP/TCP implementors in attempting to achieve good performance under a wide range of network conditions. The problems are complicated by the fact that an implementor does not have control of the behavior at both ends of the transfer. Generally, different people with different ideas create the programs that run at different sites. As a result, an individual implementation must be able to cope with a variety of behaviors on the part of implementations at other sites, and general design principles cannot assume complete cooperation between sender and receiver unless the extent of that cooperation is spelled out in the protocol definition.

The primary issues to be dealt with are:

(1) Choice of a window size by the receiver,

(2) Choice of segment sizes by the sender,

(3) Choice of an acknowledgment policy by the receiver,

(4) Choice of a retransmission mechanism by the sender, and

(5) Choice of a pacing mechanism by the sender to handle the case where the window is too large, causing losses on bursts.

As will be argued in the following discussion, good performance with respect to (4) requires a reliable estimate of RTT. Since getting such an estimate is a difficult if not unsolvable problem, we have an additional issue:

(6) Reliable estimation of RTT by the sender.

The following sections discuss these issues in detail and also address other matters such as the interpretation of Internet Control Message Protocol (ICMP) messages that are generated by gateways and sent to hosts in some cases when packet delivery problems are encountered. The effects of IP fragmentation are also considered when they interact with the other issues.

4.2.1 Window Size Issues

The window size is set by the receiver primarily on the basis of the buffer size that the receiver is able to commit to reordering arriving segments. One of the assumptions of the Internet model is that packets can both be lost and can arrive out of order. A receiving TCP is expected to provide enough buffering to handle reordering so that retransmission will be minimized. The receiver is expected to accept and hold any segment that arrives with a good checksum and that falls within the advertised window. Since buffer space is limited and a TCP module generally serves multiple concurrent connections, the space available for any one connection may well be small enough to cause the advertised window to seriously restrict the cross-net flow. This limitation is particularly severe for networks such as the WB SATNET which have both high channel capacities and long delays. To get megabit-per-second rates through a network with a 2-s RTT would require a window of 2 Mbits, which is more buffer space than a host is likely to be able to commit to this purpose.

The TCP is robust with respect to packet loss and it can operate as a simple "Go Back N" communications protocol that requires the entire window's worth of data to be retransmitted in the event of a lost packet or out-of-order packet arrival. If the probability of packet loss or disorder was expected to be small, the receiver could advertise a window larger than the available buffer and gamble that retransmission would not be excessive. Unfortunately, the receiving TCP has no information on which to base a decision to advertise an inflated window. It has no direct means of discovering that the net has more capacity than that set by an arbitrary window and actual RTT. Similarly, it has no way to discover that the window is too large and that packets are being lost because of too high a burst rate as the sender attempts to use the advertised window.

The protocol allows the receiver to reduce the window to shut down flow due to local congestion caused by limitations within the host system. Since we have decided to ignore host limitations in our study, we assume that the receiver will simply pick a window size consistent with its buffer capacity and keep that value fixed for the duration of a file transfer. W/RTT will thus pose an upper bound on the cross-net flow unless the sender can deduce that a higher rate is possible.

4.2.2 Segment Size Issues

The transfer of data through a packet network will proceed most efficiently and rapidly if the data are sent in packets of the maximum size that can be carried by the network. Such packets will make optimum use of buffers in the net and minimal use of the CPU resources in the network nodes. The problem for the sending TCP is to make a good guess at a segment size to use for its transmissions. It can know *a priori* the maximum packet size that can be sent on the net or nets to which its host is connected, but the Internet model says that it cannot assume such knowledge for paths through the general Internet. If the sending TCP picks too small a size, it will pay a price in packet overhead and flow rate. If it picks too large a size, the packets may be dropped by gateways that find them to be too big for some network, or they may be fragmented if the IP header flag bits permit fragmentation. When fragmentation occurs, the portions of the Internet downstream from the point of fragmentation experience an increased load in terms of the number of packets per second to be handled. If packet losses occur beyond that point, the probability of successful delivery goes down as the n^{th} power of the probability of losing an individual packet, where n is the number of fragments into which a packet is broken.

Fragmentation is a particularly unfortunate situation in the IP/TCP environment because of the functional split between TCP and lower-level IPs. Fragmentation and reassembly are IP functions, but retransmission is a TCP function. As a consequence, a receiving IP module that is trying to reassemble a fragmented IP datagram cannot take any advantage of a retransmission of the same underlying data to fill in a missing piece because the IP header information does not identify the packets as having related information. Further, the receiving IP module has a difficult problem in deciding how long to retain an incomplete set of fragments. These tie up valuable buffer space and should be discarded as soon as possible, but not before waiting long enough for the Internet to have had an opportunity to deliver a reasonably delayed packet. The dispersion of fragment arrival times can be estimated, but the computation is tricky since the tail of the distribution is the important part. Worse, it is important not to average dispersions across TCP connections, since the fragmentation points may be very different for different Internet paths. Information for identifying the TCP connection is not available in the IP headers, and the IP module would have to use sending host addresses for identifying fragments as belonging to a connection or not.

It is clear that a sending TCP should avoid having its segments fragmented if it can be arranged. The easy way to achieve this result is to pick a segment size small enough to get through the most restrictive network in the Internet. Another approach would be to pick a larger size that had a good chance of avoiding fragmentation, and accept the cost of fragmentation for the small fraction of cases for which the size was too large. It is possible to probe the Internet to find out what size packets can escape fragmentation by setting the "DON'T FRAGMENT" bit in the IP header and watching for ICMP messages coming back saying that packets were dropped because they were too large. This latter approach is cumbersome, requiring the sending of a sequence of different-size packets. It is also not guaranteed to work because the ICMP messages either may not be generated or may not succeed in making their way back in all cases. Also, if the network path changes during the course of a file transfer, the maximum packet size limit may change as well. Our view is that an attempt to measure the maximum segment size is not likely to be worth the cost, and that a compromise maximum size is the best choice under the circumstances.

The actual segment size used for a packet cannot always be the maximum. The remaining window space will often limit the segment size that can be dispatched. In this situation, it is important to avoid sending very small segments unless the "PUSH" function is requested by a higher-level protocol to get the last part of a transaction through the network. If care is not taken to avoid small segments, a regenerative condition called "silly window syndrome" can occur that will severely degrade performance. This problem and its solution have been addressed by Clark[11] and will not be pursued in detail here.

4.2.3 Acknowledgment Policy Issues

TCP requires that a receiver return acknowledgments to a sender to indicate the successful arrival of data. The receiver may send an ACK piggybacked on a data packet moving in the reverse direction if it has traffic to send. If there is no such traffic, as is generally the case for file transfers, the ACK must be sent in a packet that performs no other service. It is not required that each arriving segment be ACKed. Since ACKs are cumulative, one ACK can cover a number of segments, and Clark[11] recommends using this capability as one defense against the "silly window syndrome." For this purpose, he suggests not ACKing segments until a reasonable fraction of the window (say, one-third) has been received. While we agree that such an ACK policy can help to prevent the problem of sending many small packets instead of a few large ones, for cases where the window size is large in relation to reasonable segment sizes this ACK policy denies the sender information about the behavior of network delay characteristics by reducing the rate at which the RTT can be sampled. We recommend sending an ACK whenever a segment of reasonable size is received, even though that segment does not represent a significant fraction of the window. Of course, segments with the PUSH flag set must be ACKed independent of segment size.

4.2.4 Retransmission Issues

TCP depends heavily on retransmission to provide robustness against packet loss. The TCP definition recommends that a sender estimate the mean RTT through the net and retransmit unacknowledged segments on a time-out set somewhat longer than the mean RTT. The usual mechanism is to maintain a queue of segments that are awaiting ACK or retransmission. As each segment is dispatched, an entry is placed on the queue with a timestamp indicating when the corresponding segment should be retransmitted if an ACK for it is not received before that time is reached. Retransmitted segments are not required to have a one-to-one relationship to previously transmitted segments. If the original segment was not of maximum size due to window limitations and ACKs that have arrived in the meantime have granted new window space, the retransmitted segment could be larger than the original. Similarly, for interactive applications new data may have arrived during the time-out, allowing a larger segment to be sent at retransmission time.

For the case in which the window is larger than the chosen segment size, there will be more than one segment on the retransmission queue when a lost packet triggers the retransmission process. During the time required for the ACK to the retransmitted segment to get back to the sender, the retransmission times for the other segments on the queue are likely to occur. As the experiments reported in the previous Annual Report[2] demonstrated, it is wasteful to retransmit those segments if the probability of losing packets is low. It is more efficient to retransmit only the first segment and wait for it to be ACKed. However, if packets are lost in bursts, retransmitting the entire queue as each element times out will give a higher overall file-transfer rate at the expense of some wasted (duplicate) retransmissions.

Time-outs are not the only mechanisms for triggering retransmissions. The arrival of an ICMP "SOURCE QUENCH" message can be a better queue for retransmission than a time-out. A QUENCH message often means that a gateway has dropped a packet that it was unable to handle due to lack of resources. From the gateway's point of view, the purpose of the QUENCH is to request the host to reduce the rate at which it is sending packets so as to alleviate a congestion problem. From the host's point of view, the QUENCH also means it is likely that the QUENCHed message will need to be retransmitted, and in most cases the QUENCH carries enough information to allow the TCP to schedule a segment for retransmission. Such a retransmission would be likely to take place at an earlier time than would be the case if the TCP waited for a time-out. The result of more timely retransmissions is a higher file-transfer rate for the same rate of packet loss, as our experiments have shown.

Unfortunately, there are some problems with using QUENCH messages to queue retransmission in real-world implementations. First, the QUENCH is an ICMP message that properly belongs in the IP layer, and some implementations may not pass the QUENCH up to the TCP layer. (This is not required by the protocol specifications.) Second, if the dropped packet was a fragment that was not the first fragment of a segment, the QUENCH will not provide enough information for the TCP layer to identify the associated segment for retransmission. Third, there is no guarantee that a QUENCH will be generated when a packet is dropped or that, if it is generated, it will make it back to the sender. Fourth, gateways may send QUENCH messages to slow down transmission rates without actually dropping packets, in which case retransmission of a QUENCHed packet would be wasted. It is therefore necessary to retain the time-out mechanism, and the best that can be done is to augment it with QUENCHes when they are available and when experience shows that they correlate well with packets really being dropped.

Achieving good performance with IP/TCP requires a retransmission mechanism that performs very well. If it is too aggressive, channel capacity is wasted with unnecessary retransmissions, and the probability of congestion in the network is increased by the extra traffic. If it is too conservative, transmission opportunities are missed waiting for time-outs to occur, and channel capacity may be not be fully utilized. In both cases, the rate at which files are transferred or the delays perceived by interactive users will be worse than could be achieved with just the right mechanism. Unfortunately, good performance depends on a good estimate of the network RTT for setting time-outs and, as we argue in Section 4.2.6, there is no simple algorithm for obtaining a reliable estimate under all conditions. Since the effects of an aggressive mechanism are likely to be more damaging to overall network performance, current thinking favors a conservative approach. If the time-out is set to be several times the estimated RTT and increased with successive retransmissions, errors in estimating RTT will have less effect on performance than the setting of 1.3 to 2.0 times the estimated RTT suggested in the TCP Specification Document (RFC 793). Of course, the time-out must be bounded on both sides to reasonable values independent of the RTT estimate. We would recommend a high starting value for the time-out when there is no meaningful estimate of RTT, because none or only a few ACKs have been received in the recent past.

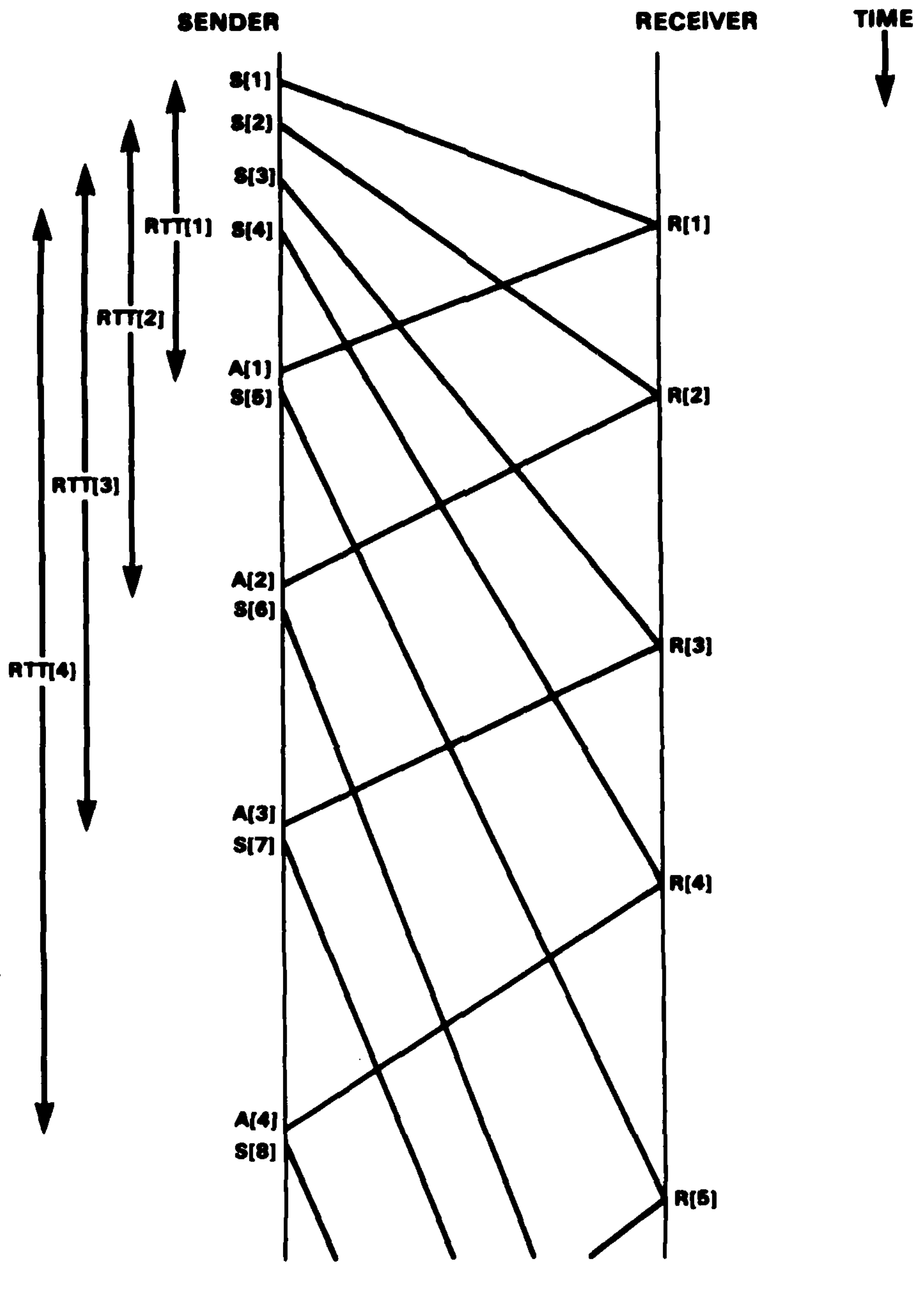
4.2.5 Pacing Issues

In a first-order sense the flow of ACKs, together with the window, is sufficient to control the rate at which segments are transmitted. Figure 12 shows a simplified scenario for the start of a file transfer in which the window size is sufficiently large to allow four segments to be generated and sent in a burst at the beginning. The time at which segment "i" is dispatched is indicated by the symbol "S[i]." Similarly, the times of receipt of the segment and of the arrival of the ACK back at the sender are indicated by the symbols "R[i]" and "A[i]," respectively. The timing shown corresponds to a network path that has delays limited in part by the available channel capacity. The scenario assumes that no packets were lost in the burst, and one can readily see that transmission of segments following the initial burst are spaced in an appropriate fashion. For this case, no special pacing mechanism would be needed.

Figure 13 shows a similar diagram for a scenario in which one of the segments (S[2]) is lost due to a momentary overload during the burst. After a time-out, S[2] is retransmitted (S[2]*), eliciting a cumulative ACK (A[5]) since the other packets were received successfully. At this point, a full window-size burst can again be attempted by the sender. One can easily see that this unproductive bursty transmission mode could persist if the transmission rate during the burst is sufficient to have a high probability of overloading the Internet. If no pacing mechanism is in effect, such bursts will be sent at rates limited only by the CPU resources in the host and the capacity of the local network port. These rates are likely to exceed the rates at which downstream gateways can absorb and buffer the packets, and some packet loss is to be expected.

Pacing can be a serious problem for hosts connected to fast cable nets that then connect via gateways to slow long-haul nets. It is less of a problem for a host connected to ARPANET or to one of its clones since the IMPs in those nets provide backpressure that tends to prevent a host from sending packets too rapidly. But even in the relatively benign ARPANET environment, a gateway to a slow net can easily be overloaded by an excessive burst rate. Clearly, a pacing mechanism is needed to spread out burst transmissions and thereby achieve more efficient use of Internet resources.

The solution to the burst overload problem is obviously to put some delay between the packets in the burst. The difficulty is to determine a good value for the delay. A fixed value long enough to handle the worst-case situation would exact a large penalty in achievable transfer rates. A variable delay adjusted according to feedback from the net would be ideal, but what feedback is available? The TCP gets information about the network characteristics from arriving ACKs and can use that information to estimate a pacing interval. It also gets information about lost packets from retransmission time-outs and from QUENCH messages. The QUENCH message is intended to inform the host that its transmission rate is too high, but since gateways do not track the rates for individual TCP connections, an arriving QUENCH may merely mean that some gateway was momentarily overloaded and randomly decided to send a QUENCH for some packet.



150696-N

Figure 12. Timing diagram for a file transfer starting with a successful burst of four segments.

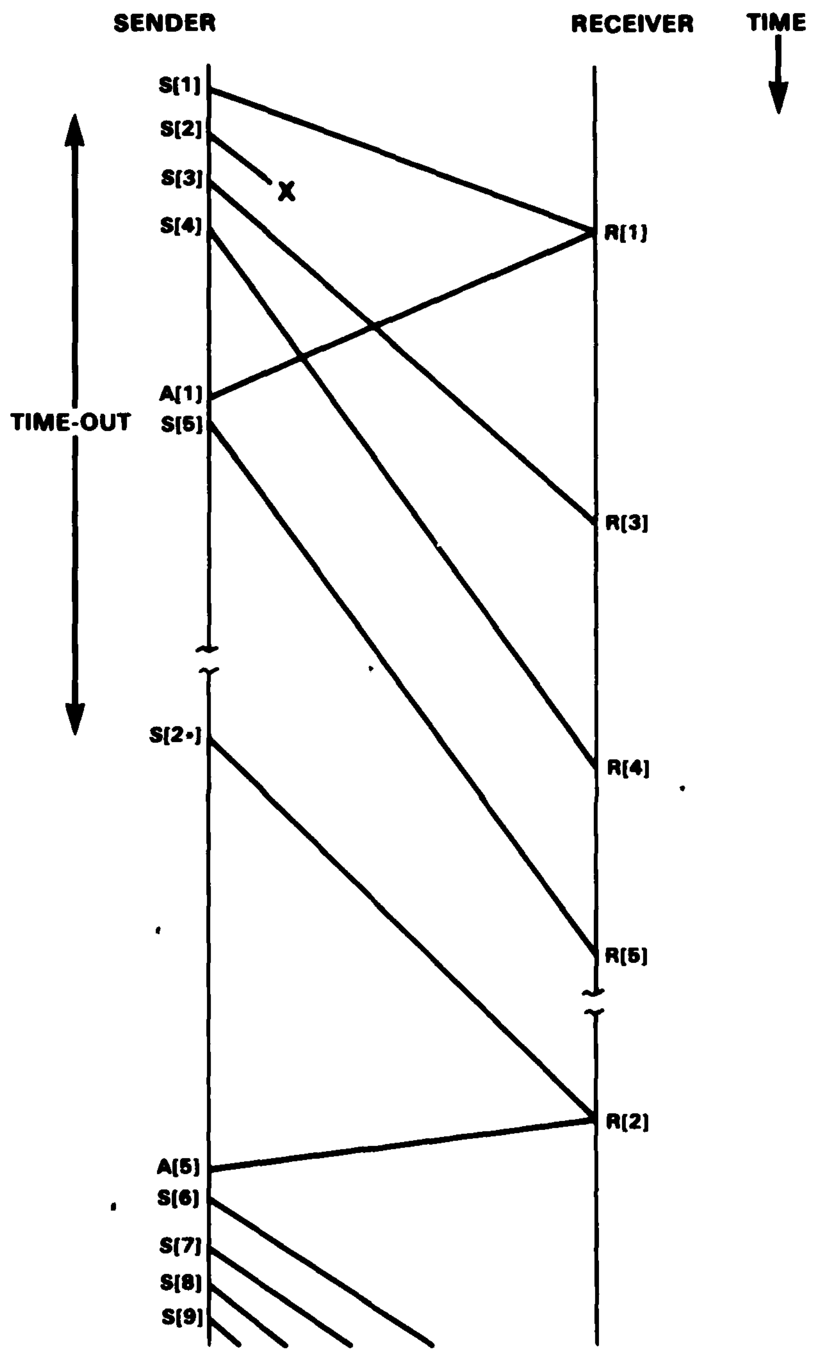


Figure 13. Timing diagram for a file transfer starting with a burst of four segments in which one (S[2]) is lost in network.

Unfortunately, packets can be lost due to excessive transmission rates without any QUENCHes coming back to indicate a problem. Such a situation will occur if the point of loss is the input to a gateway from a net that does not exert backpressure. The gateway is then unaware of the loss and is not in position to send a QUENCH. Feedback to the TCP in this case comes only from the need to retransmit on time-outs.

There is obviously information in the pattern of QUENCHes and lost packets in relation to the transmit patterns. For example, if losses correlate with position in a burst, and the losses are greater for later segments in the burst, it may be reasonable to infer that the burst rate is too high. Similarly, if ACKs consistently arrive with separations greater than those between the corresponding transmitted segments, it may be reasonable to infer that the transmission interval could be increased with little loss in transfer rate. Both the complexity of algorithms to use such information and the burdensome chore of keeping a history of packet departure times seem inappropriate for a relatively low-level protocol such as TCP. Therefore, we conclude that less sophisticated approaches must suffice.

For file transfers, a conceptually simple mechanism would be to estimate the mean time between arriving ACKs for a window's worth of data, and to use that estimate for pacing transmissions. This approach approximates the natural pacing that would occur with the arrival of ACKs in the absence of packet losses and retransmissions. Our experiments implicitly assumed such a pacing mechanism by transmitting at a fixed inter-packet spacing set by the experimenter. In the real world, this approach suffers from similar difficulties to those of estimating the RTT (see Section 4.2.6) and from the fact that at the start of transmission, when the need for pacing is most critical, the estimate is not yet available. Also, the approach is not useful for interactive applications where the need to send a window's worth of data on a continuing basis is not a likely situation.

Nagle[12] suggests a pacing mechanism for the interactive case that allows only one outstanding message at a time on the TCP connection. Data that are given to the TCP during periods when it is awaiting an ACK are buffered and sent in a segment as large as possible when the ACK arrives. This mechanism should work well for single-character keyboard inputs, but could cause extra delays when the computer end of the connection has larger quantities of data to send, as in the case of full-screen displays, in response to single key commands. To handle the latter case as well as file transfers, some further mechanism is needed.

We suggest a mechanism that would allow two packets to be outstanding in situations where both the chosen segment size and the window were large. This mechanism allows the quantity of data in the network "pipe" to build up to a value limited by the window and the RTT. Unfortunately, it does not guarantee that losses will not occur on the size-two burst. To handle such losses, we suggest augmenting the mechanism with a minimum delay between sending packets to the same gateway. This delay should be increased with retransmissions and be reduced with receipt of ACKs. The rate of increase should be more rapid than the rate of reduction, perhaps a doubling on each retransmission with many ACKs required for a reduction by half. Upper and lower bounds should be chosen to avoid ridiculous extremes. We suggest experimentation to adjust the parameters for this mechanism, and that the experiments should be carried out in the particular network environment in which a host is operating, since the choice of values will depend on local conditions and requirements.

Unfortunately, our suggested mechanism requires action at both the IP and TCP levels and is therefore more complex than one would like. The IP layer needs to introduce the delay because only it knows that queued packets are being sent to the same gateway and that the packets are therefore subject to the delay. The delay adjustment depends on the need for retransmission which is indicated both by time-outs and the receipt of QUENCHes. IP gets the QUENCH messages and can deduce to which gateway they correspond, so it can handle the delay increase for that case. But only TCP knows about the time-outs, so it must call for a delay increase by some special communication with the IP layer for that case. Also, only TCP can identify the ACKs (without cheating), and it must then notify the IP layer to decrease the delay. Since TCP may also receive the QUENCHes, it could in that case also call for a delay increase; but, in our view, the handling of QUENCHes for this purpose should be done at the IP level since the actions of protocols other than TCP can best be factored in at that level.

4.2.6 Reliable Estimation of Round-Trip Time (RTT)

Setting retransmission time-outs in an optimal fashion depends on obtaining a good estimate of the RTT through the network path. The RTT is defined as the elapsed time between sending a data octet with a particular sequence number and receiving the first ACK that covers that sequence number. Since ACKs are cumulative, it is necessary to search the retransmission queue to find the sending time of the segment with the highest sequence number covered by the ACK number. The transmission time for that segment is used as the sending time in the RTT computation.

Since RTT varies dramatically from segment to segment, in many cases, it is necessary to compute a smoothed RTT for use in setting the retransmission time-out. The recommended computation[13] for computing the smoothed RTT (SRTT) after the receipt of the ith ACK that resulted in RTT[i] is:

$$\text{SRTT}[i] = \text{ALPHA} * \text{SRTT}[i-1] + (1 - \text{ALPHA}) * \text{RTT}[i]$$

where ALPHA is the smoothing factor with suggested values in the range 0.8 to 0.9.

There are two difficulties with this estimation procedure. The first is a startup problem. A natural way to start would be to set RTT[0] = RTT[1] when the first ACK arrives. For scenarios such as that of Figure 12, the algorithm will yield an estimate much too low for some time. The proper SRTT for retransmission purposes in this case is closer to RTT[4] than to RTT[1]. SRTT[i] will eventually reach the right value, but in the meantime its use could cause wasteful retransmissions and consequent lost packets in a worse scenario than that of Figure 13. The retransmissions can be avoided by using an arbitrary upper bound (say 1 min.) for the retransmission time-out until a number of ACKs (say, ten) have been received that have contributed to the SRTT estimate. (Duplicate ACKs do not contribute new information on RTT estimates.)

The second problem results from retransmission and is more difficult to handle. Consider a scenario in which an SRTT has been established and is stable, and then a step increase in network delay occurs of sufficient magnitude to allow a segment to be retransmitted on a time-out before the ACK to the original transmission arrives. Figure 14 shows the timing for such a segment (S[i]) and its ACK (A[i]). The dashed lines show the expected timing predicted by SRTT.

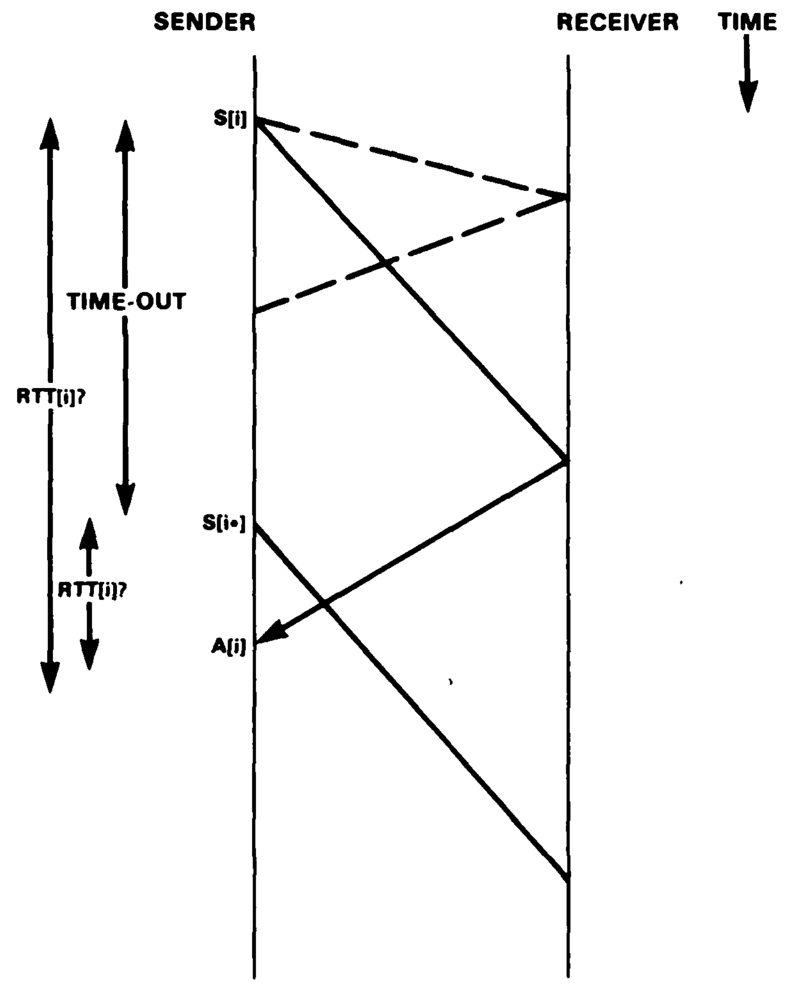


Figure 14. Timing diagram showing a network delay increase large enough to cause retransmission of segment S[i] on a time-out. Definition of Round-Trip Time (RTT[i]) is at issue.

The question of concern is what value of the sending time to use in computing RTT[i]. If we set RTT[i] = (A[i] - S[i]), SRTT will correctly respond to the increased network delay. However, if we set RTT[i] = (A[i] - S[i*]), SRTT will erroneously indicate a decrease in network delay. Unfortunately, the sending TCP cannot distinguish between the scenario represented here from one in which the original transmission or its ACK was lost, in which case (A[i] - S[i*]) would be the correct value to use.

We do not know of any algorithm that will compute the correct value in all such cases without making changes to the protocol to allow retransmissions to be distinguished and to convey that information back to the sender in the ACK. Such a protocol change is not likely to take place, and we would not argue that it should. Instead, we recommend simply using the original sending time in the computation and accepting the resulting loss of performance for the cases where segments are really being lost. This choice is conservative and leads to the correct behavior for the presumably more frequent case of increased network delay.

It should be noted that, at best, estimation of RTT is a noisy procedure. There is often considerable packet-to-packet variation in delay through networks, and short-term average delays can vary widely over periods of time that may be short in relation to the intervals between useful ACKs. The effect of packet-to-packet variation can be reduced by increasing the smoothing factor, but only at the expense of slowing the response to changes in the short-term average delay. The result is that there is likely to be significant error in the estimate and, consequently, it must be used conservatively.

4.3 CONCLUSIONS

In this report we have discussed a number of issues in TCP implementation that relate to performance. Difficulties have been identified and possible solutions presented, but it is clear that a practical implementation must involve many compromises, and that it is unlikely that good performance can be achieved over a wide range of conditions. It is also clear that IP/TCP is a complex and expensive protocol suite.

The scope of our study did not extend to the exploration of alternative protocols or changes to the presently defined IP and TCP, but our investigations have led us to the conclusion that the difficulties with the protocols and the underlying Internet datagram model are sufficiently severe that alternative approaches should be seriously explored.

5. EISN SYSTEM COORDINATION AND EXPERIMENT PLANNING

In February 1984, Lincoln Laboratory submitted the "Defense Switched Network Technology and Experiments Project Master Plan — FY84" to DCEC. The purpose of the document was to detail the specific activities that would be carried out under the FY84 program, as well as the key elements of the continuing activity planned for FY85. Three major experimental areas were discussed, namely: (1) evaluation of new survivable mixed-media routing algorithms and efficient multi-level precedence and preemption (MLPP) techniques as candidates for implementation in the DSN; (2) implementation, installation, and experimental use of an advanced telecommunications test-bed at the EISN sites across the country, including off-the-shelf computer-controlled telephone switches integrated with programmable outboard Routing/Control Processors (RCPs); and (3) development and evaluation of flexible, adaptive data protocol and voice/data integration techniques. In addition, the Plan described Lincoln's continuing activities as overall planner and coordinator for the EISN test-bed and for experiments carried out in the network.

With Lincoln coordination, the Wideband SATNET Task Force has continued to address problems of network performance and reliability. Gradual improvement has been achieved as problems have been identified and corrected. The most serious system problem that has been attacked in FY84 is the "five-stream bug" which was identified in late March 1984. The effect of this bug was to cause all PSATs in the network to crash whenever user traffic levels rose above the limit of five streams. Additional task-force-related activities during this period included: (1) moving the WB SATNET channel to a different satellite (WESTAR IV) with improved channel characteristics, in late July; (2) intensified user-level tests, with the network in quasi-operational status, in August and September; and (3) enhanced operational coordination between Western Union and the Bolt, Beranek, and Newman (BBN) Network Operations Center (NOC).

The five-stream bug caused no problems when only four or fewer sites had the capability to create streams on the net, and therefore an interim policy was established of disabling stream capability (by means of PSAT software patches) at all but four sites at any given time. This policy was administered by BBN, and arrangements were made for particular sites to obtain network access by contacting BBN prior to scheduled demonstrations or experiments. An example of such an exercise was a demonstration of simultaneous PCM calls to Ft. Monmouth and Ft. Huachuca from the Defense Communications Engineering Center (DCEC) via the PCI equipment, carried out during an Experimental Integrated Switched Network (EISN) Steering Group meeting at DCEC on 11 April 1984.

A concerted attack on the five-stream bug was carried out over several Task Force on-site work sessions. The specific cause of the bug was identified by BBN in late September; it involved data being inadvertently written on top of an area of PSAT program memory when an attempt was made to set up more than four streams. The correction required BBN to carry out some restructuring of the PSAT software. By the end of FY84, the revised PSAT code had been written and was ready for test.

With the goal of achieving improved performance for the wideband satellite channel, an agreement was reached between Western Union and the sponsors to transfer the net to a different satellite capable of providing increased signal-to-noise ratio. The transfer was accomplished for all seven network sites during a brief period in late July and early August, with Lincoln providing power and frequency calibration support for Western Union. The new parameters are: uplink frequency 6013 MHz, downlink frequency 3788 MHz, on Transponder 2-Cross on WESTAR IV, at 99° W longitude. The corresponding old parameters were 5959 MHz up, 3734 MHz down, Transponder 1 on WESTAR III, 91° W. In August, Lincoln supported Western Union in calibrating power and frequency for the eighth and ninth sites on the network, namely LINKABIT and Carnegie-Mellon.

On 9 August, the Wideband Network Task Force met at BBN (Cambridge, Massachusetts) to present a progress and status report to DCA and DARPA. The primary focus of the meeting was to address issues concerning achievement of stable, routine operational status for the network. Although several problems were known to exist (such as the five-stream bug), it was apparent that there was no fundamental impediment to quasi-operational use of the network on at least a part-time basis. It was decided, therefore, that the network would be made available for user traffic each Thursday and Friday until further notice, with debugging and maintenance confined to the other three days of the week.

On 11 September, a meeting was held at Western Union's WESTAR Control Center at Glenwood, New Jersey to clarify and enhance operational coordination between Western Union and the BBN NOC. The attendees included representatives of Western Union, BBN, and Lincoln Laboratory. At the meeting it was agreed that BBN NOC would report all Western Union troubles to Glenwood. In order to expedite correction of site problems, it was also agreed that WB SATNET site personnel could contact local Western Union repair centers directly if desired, for convenience, provided that BBN NOC was also contacted.

REFERENCES

1. Defense Switched Network Technology and Experiments Program Annual Report, Lincoln Laboratory, M.I.T. (30 September 1982), DTIC AD-A128932.
2. *Ibid.* (30 September 1983), DTIC AD-A140609.
3. R.P. Lippmann, "Steady State Performance of Survivable Routing Procedures for Circuit-Switched Mixed-Media Networks," Technical Report 633, Lincoln Laboratory, M.I.T. (29 December 1982), DTIC AD-A126213.
4. R.P. Lippmann, "Survivable Routing Procedures for Circuit-Switched Satellite-Terrestrial Networks," in Proc. 1983 Intl. Conf. on Communications (ICC '83), Boston, Massachusetts, 19-22 June 1983, pp. A1.3.1-A1.3.6.
5. R.P. Lippmann, "Routing Algorithm Development and Evaluation for Circuit-Switched Mixed-Media Networks," presented at ORSA/TIMS (Operations Research Society of America/The Institute of Management Sciences) 1984 Annual Meeting, Dallas, Texas, 26-28 November 1984.
6. R.P. Lippmann, H.M. Heggestad, and R. Berger, "Switch Features that Permit an Outboard Processor to Control Routing and Preemption," Technical Report 681, Lincoln Laboratory, M.I.T. (6 July 1984), DTIC AD-A146193.
7. Distributed Circuit Control for the DCS, Final Report on Phase 2 Protocol Development and Simulation, Computer Sciences Corporation (February 1984).
8. Digital Network Control Concept Validation System, Final Report, Honeywell Corporation (November 1982).
9. H.M. Heggestad, "Voice and Data Communications on an Experimental Wideband Internetwork System," J. Telecommun. Networks 3, 131 (1984).
10. C.J. Weinstein, "The Experimental Integrated Switched Network -- A System-Level Network Test Facility," MILCOM '83 Conf. Proc., Vol. 2, Arlington, VA, 31 October — 2 November 1983, pp. 449-456.
11. D.D. Clark, "Window and Acknowledgement Strategy in TCP," Network Working Group Request-for-Comments RFC 813 (July 1982).
12. J. Nagle, "Congestion Control in IP/TCP Internetworks," Network Working Group Request-for-Comments RFC 896 (January 1984).
13. Transmission Control Protocol, DARPA Internet Program Protocol Specification, RFC 793, Information Sciences Institute, University of Southern California (September 1981).

GLOSSARY

ACK	Acknowledgment
CCS	Common-Channel Signaling
CPU	Central Processing Unit
CSC	Computer Sciences Corporation
DAMA	Demand-Assignment Multiple Access
DCA	Defense Communications Agency
DCEC	Defense Communications Engineering Center
DES	Data Encryption Standard
DSN	Defense Switched Network
DTMF	Dual-Tone Multiple-Frequency
EFTO	Encryption for Transmission Only
EISN	Experimental Integrated Switched Network
ESI	Earth Station Interface
FTP	File-Transfer Protocol
GTG	Gateway-to-Gateway
ICCU	Interface Controller/Codec Unit
ICMP	Internet Control Message Protocol
IMP	Interface Message Processor
IP	Internet Protocol
IPG	Internet Packet Gateway
IV	Initialization Vector
MH	Measurement Host
MLP	Multi-Level Precedence
MLPP	Multi-Level Precedence and Preemption
NOC	Network Operations Center
PCI	Packet/Circuit Interface
PODA	Priority-Oriented Demand Assignment
PSAT	Pluribus Satellite Interface Message Processor
PVT	Packet Voice Terminal
RADC	Rome Air Development Center
RCP	Routing/Control Processor
RSC	Routing and System Control
RTT	Round-Trip Time

SATNET	Satellite Network
SRTT	Smoothed RTT
ST	Stream Protocol
TCP	Transmission Control Protocol
TDMA	Time-Division Multiple Access
TOE	Telephone Office Emulator
WB SATNET	Wideband Satellite Network

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-85-172 / **2. GOVT ACCESSION NO.** AD-A158039	**3. RECIPIENT'S CATALOG NUMBER**
4. TITLE *(and Subtitle)* Defense Switched Network Technology and Experiments Program	**5. TYPE OF REPORT & PERIOD COVERED** Annual Report 1 October 1983 — 30 September 1984
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Clifford J. Weinstein	**8. CONTRACT OR GRANT NUMBER(s)** F19628-85-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073	**10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS** Program Element No.33126K
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency 8th Street & So. Courthouse Road Arlington, VA 22204	**12. REPORT DATE** 30 September 1984
	13. NUMBER OF PAGES 66
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB, MA 01731	**15. SECURITY CLASS.** *(of this Report)* Unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

None

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

routing algorithms, Experimental Integrated Switched Network, Defense Switched Network, mixed-media routing, preemption, circuit-switched network, packet switching

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

This report documents work performed during FY 1984 on the DCA-sponsored Defense Switched Network Technology and Experiments Program. The areas of work reported are: (1) development and evaluation of routing and system control techniques for application in the Defense Switched Network (DSN), (2) instrumentation and integration of the Experimental Integrated Switched Network (EISN) test facility, (3) development and test of data communication techniques using DoD-standard data protocols in an integrated voice/data network, and (4) EISN system coordination and experiment planning.

[illegible]