AT&T Technical Journal
Vol. 64, No. 6, July–August 1985
Printed in U.S.A.

The 5ESS *Switching System:*

Operational Software

By J. P. DELATORE, R. J. FRANK, H. OEHRING, and L. C. STECHER*

(Manuscript received January 3, 1984)

The operational software of the *5ESS*™ switching system has been designed to meet specific objectives for capacity, functionality, and reliability. It has also been designed to accommodate changing technology, changing system applications, and an ever-increasing feature application set. Structured programming techniques, high-level languages, and modular design techniques have been used to obtain these objectives. Specifically, the operational software architecture is organized as a set of functional software components utilizing the advantages of a generalized operating system and database manager to achieve a high degree of hardware independence. A primary cause for hardware churning is the introduction of new peripheral units in support of new feature applications. The peripheral control software shields the majority of the operational software from these changes. Standardized and locally well-defined interfaces to the operations support systems provide information to the customer for the administration and maintenance of the switching system.

I. INTRODUCTION

The operational software of the *5ESS* switching system has been designed to meet specific capacity, functionality, and reliability objectives. It has also been designed to accommodate changing technology, divergent system applications, and an ever-increasing feature application set. To meet these objectives the design stresses structured programming techniques, use of high-level languages, and modular design techniques.[1]

In this paper we describe the operational software architecture within the overall *5ESS* system design. We show the organization of the software as functional components (see Fig. 1) and discuss the

* Authors are employees of AT&T Bell Laboratories.

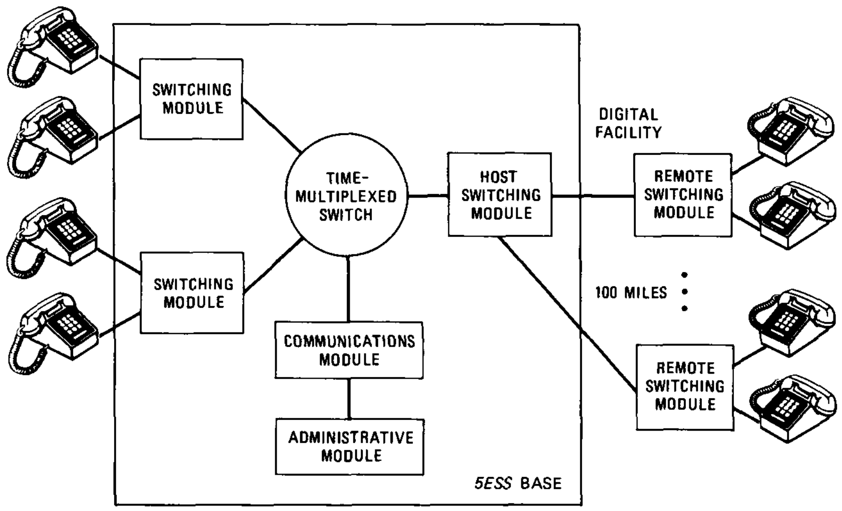


Fig. 1—Architecture of the *5ESS* switching system.

advantages of using a generalized operating system and database manager to achieve a high degree of hardware independence.

We also discuss control of the peripheral hardware and administrative services such as measurements and billing. The call-processing scenario ties these design components together. Finally, we demonstrate how the remote switching capability extends the design.

II. SOFTWARE ARCHITECTURE

The software system of the *5ESS* switching system (see Fig. 2) comprises subsystem modules that communicate with each other using concisely defined message protocols and logically based primitives. The subsystems are designed to be loosely coupled, and almost all subsystems are largely independent of the system hardware and the internal data structures.

An operating system isolates the application programs from the specific processors and allows portability of these programs across processors. The operating system is designed to present a single virtual machine environment to application programs operating in a distributed and decoupled multiprocessor environment.

The data base management subsystem provides the interface between the other subsystems and the physical data. Using a relational database design, it shields the application programs from the actual implementation of both dynamic and static data.

Similarly, the peripheral control subsystem serves the hardware-independent subsystems by managing and controlling the switching

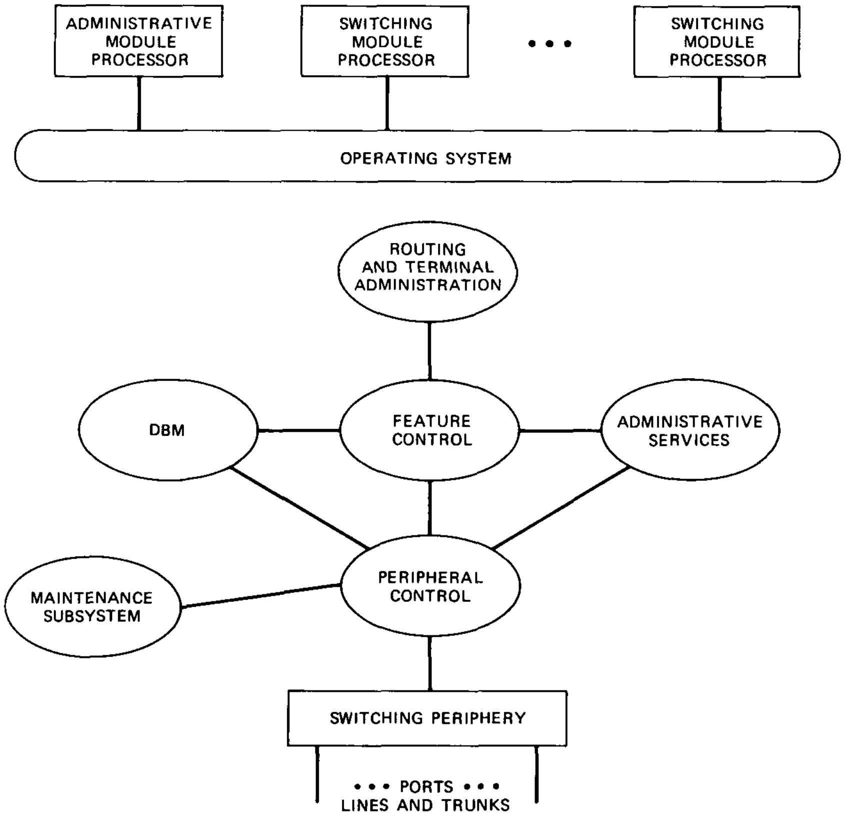


Fig. 2—Software architecture of the *5ESS* system.

periphery. The periphery includes the switching networks, service units, and line and trunk units. Lines and trunks terminate at the line and trunk units, respectively, and are referred to as *terminals* in this paper. The peripheral control subsystem performs all signaling, supervision, alerting, digit reception, and outpulsing functions. A change in a peripheral unit usually requires program changes only in this subsystem and is not propagated into the application software. All the programs to control the switching periphery are resident in the Switching Module (SM)* except for programs that control the Time-Multiplexed Switch (TMS), which are resident in the Administrative Module (AM).

The feature control subsystem is responsible for sequencing all logical call-processing actions associated with residential, business,

* Acronyms and abbreviations used in the text are defined at the back of the *Journal.*

toll, and system features. This subsystem receives terminal inputs in the form of supervision changes or dialed digits, interprets this input based on the terminal's current state, and, when needed, obtains more information from the Data Base Manager (DBM). It then requests that the peripheral control subsystem perform the required actions. Programs of the feature control subsystem that perform basic line and trunk call-processing functions are resident in the switching modules. Optional features or features invoked less frequently are implemented by program modules that can reside in either the administrative or switching module. Placement of these program modules is determined based on how frequently they are used and the resulting trade-offs of processor capacity and memory resources. This can be accomplished with minimal program modification. The software architecture and design philosophy permits the flexibility to best match feature usage with processor placement, thus assuring that the total system resource utilization can be balanced properly.

The routing and terminal administration subsystem provides central routing and screening functions in support of the other subsystems. For example, the feature control subsystem passes a set of dialed digits to this subsystem, which in turn determines the destination of the call. The call destination is characterized by the switching module and terminal to which the call will be completed. In addition to the central switching resource allocation and routing function, the subsystem provides supporting functions, such as terminal status information, for other subsystems. Because of the global nature of these functions, this subsystem resides primarily in the administrative module.

The administrative services subsystem provides message accounting or billing; traffic and plant measurement collection and outputting; network management functions; and interfaces to the external operations support systems, such as the Remote Memory Administration System (RMAS) and the Engineering and Administrative Data Acquisition System (EADAS). The programs in this subsystem collect data from the other subsystems on a per-event basis, and perform analysis and storage for delivery at prescribed intervals or on demand to the external support system via data links. In addition, queries and commands from these external systems are distributed by this subsystem to the appropriate operational subsystem for action. The administrative services program modules are principally resident in the administrative module with data collection programs placed in the switching modules to retrieve per-event descriptions.

The maintenance subsystems perform fault recovery and system integrity functions and are resident in the appropriate module. Various audit programs are automatically brought into play when inconsistencies or hardware failures are detected. They are also scheduled on demand when the integrity of dynamic data is suspect.

III. OPERATING SYSTEM

The Operating System for Distributed Switching (OSDS)[2] provides the processing environment required by operational, switch maintenance, and system integrity software resident in the different modules. OSDS performs five major functions:

1. Processor isolation
2. Concurrency control
3. Intra- and interprocessor message communication
4. Resource scheduling
5. Timing.

The operating system is designed to support the decoupled network view of the *5ESS* switching system design, i.e., it allows the view of the switching system as a collection of independent processors with well-defined interfaces via a concise message protocol. In essence, the design can be viewed as a collection of independent switches administered as one switching center.

A common machine-independent portion of OSDS includes the implementation of process management, interprocess communication, timing services, and scheduling. The machine-dependent portion, a small part of the total operating system, interfaces with the hardware processor and the host operating system supplied with the administrative module processor. It includes interrupt handling, process dispatching, basic memory management, interprocessor communication, and communication to the host operating system.

3.1 Processor isolation

The *5ESS* switching system architecture is designed to have a long lifetime. As hardware technology evolves, it is desirable to upgrade a processor without affecting the design of the software to any measurable extent. OSDS is designed to provide a virtual machine environment whose interface to the operational software remains constant through processor evolution. The fact that the operating system hides the unique hardware configuration from the operational software assures the expeditious porting of such software and preserves the investment in software development.

3.2 Concurrency control

The operating system concept of a process was adopted to facilitate understanding of the large number of concurrent activities in the switching environment. Concurrency is a significant design consideration owing to the large number of active terminals or phone calls at the same time. The software structuring concept of a process enables an operational software designer to program each activity (e.g., providing dial tone or ringing a line) as a sequence of separate tasks in a process. Multiple instances of this process are then executed concurrently by the operating system to provide the concurrency control necessary to simultaneously process a multitude of telephone calls. This approach reduces the complexity of the design, since the designer can focus most of his/her attention on the external behavior of the process and its interaction with the environment and not on the interactions of the processes with each other. The process concept also structures the software system as highly modular parts with limited and well-defined interfaces. Communication with other processes takes place via messages routed to the appropriate destination by the operating system.

3.2.1 Structure of processes

Within the scope of OSDS there are two process types: terminal process and system process.

3.2.1.1 Terminal process. All the activities on an active terminal are controlled by at least one terminal process. A typical call has two terminals and two associated terminal processes controlling the originating and terminating party. Each terminal process thus keeps a partial view of the call and responds only to well-defined inputs from its associated terminal or other processes.

This terminal-oriented process approach reduces programming complexity because it allows the designer to concentrate on the events specific to an individual terminal, the action necessary for processing such events, and interaction with other processes. The terminal-oriented process approach is more natural because the designer can assume the terminal user's point of view in designing a program to control the terminal. Feature enhancements can be implemented by a simple addition to feature control programs that are callable by terminal processes. This approach is also well matched to the distributed aspects of the call-processing job when a call originates in one module and terminates in another. Having one terminal process in each module permits a natural division of the processing between the two modules.

Terminal processes are created and terminated on demand. Typically, these processes are created on origination and terminated at disconnect time. They are activated by the arrival of an external stimulus or input signal from the terminal or another process. After processing a signal, the terminal process takes a "real-time" break by releasing control to the operating system and waits for further input. There are separate control programs for different terminal types. For example, there are residential, coin, and multiparty terminal control programs. These programs determine the control sequence required for their particular terminal type. The majority of the processes in the system are terminal processes, sharing the use of the various control programs.

3.2.1.2 System process. There are also some long-standing processes that function as servers within or outside of call processing. Examples are dispenser processes, which distribute inputs to terminal processes; the routing and terminal allocation process, which assists in routing the call; the data base manager process, from which a terminal process can request data characterizing the physical terminal; and the system audit process, which performs the routine software sanity checks. These are called *system processes* since they provide services on a systemwide basis. They are typically created at system initialization and remain active indefinitely. Each system process carries out a specific function and handles requests from more than one terminal process. In general, a system process is a self-contained program and does not share programs with other processes.

3.3 Message communication

Message communication is the major communication mechanism between processes. Basically, a sender prepares a message in a message buffer and then invokes the operating system using a message communication primitive to transmit the message. The operating system determines the destination of the message. If the message is addressed to a process within the same processor, the operating system copies the message to the receiving buffer and schedules the receiving process. In case the message destination is external to the processor, a communication control program is invoked to transmit the message through a physical link. In a similar fashion, incoming messages are received by the communication control program and transferred for proper routing to OSDS.

3.4 Resource scheduling

Resource scheduling involves the central allocation and assignment of memory resources and processor time. When the operating system schedules a process, it assures that sufficient memory resources are available for that process to run. The operating system is designed in a self-regulating manner intended to meet system performance at the rated call-carrying capacity and to satisfy stringent response times to the craft personnel. It also responds to external stimuli from an overload control program to take corrective actions, if necessary, and preserve system throughput.

The scheduling is done at different priority levels with each type of process assigned a specific priority. Overload control reacts to momentary bursts of traffic input or resource shortages and provides input to the scheduling algorithms. A sanity timer protects the system against any process taking control of the processor for longer than a maximum allowable time. A cyclic timer interrupt assures that high-priority jobs with stringent timing requirements are executed with the appropriate frequencies.

3.5 Timing

The operating system administers two classes of timing services. The first, essentially a time of day clock, is used for billing purposes and for time-stamping events. Second, each OSDS processor maintains a clock used in conjunction with an ordered queue of timing requests that allows each process to request control from OSDS at specified time intervals.

IV. DATA BASE MANAGEMENT SYSTEM

The Data Base Management System (DBMS)[3] promotes the overall *5ESS* system objectives of modularity, portability, and system evolution by providing a single access to the data which is totally at the logical level as viewed by the users of the data. The DBMS supports the distributed architecture of the system by providing distributed data and distributed management of the data. Most data required by a particular processor are contained within that processor. The DBMS makes the location of all data transparent so that other programs need not be concerned about on which processor the data actually reside. The DBMS is a specialized distributed system based on the relational data model and is subject to stringent performance and reliability requirements. The balance between real-time constraints and reliability requirements is largely achieved by a concurrency mechanism that provides the necessary real-time access while maintaining a consistent view of the data.

The Data Base Management System provides:

1. Data definition
2. Data access
3. Concurrency control
4. Memory partitioning and write protection
5. Data backup and recovery
6. Data administration.

4.1 Data definition

Any database management system should provide a measure of data independence. In the *5ESS* switching system, data are defined off-line and changes can be introduced by redefining database relations and automatically recompiling user programs. The goal is to introduce database changes with a minimum number of client program changes. Because the schema remains stable during a given software release, it is not necessary to define the database on line. A data definition language accepts user definitions of the relations and their domains and generates user header files, data dictionaries, and documentation. Figure 3 shows the data definition process and the major software components in a switching module. The administrative module contains

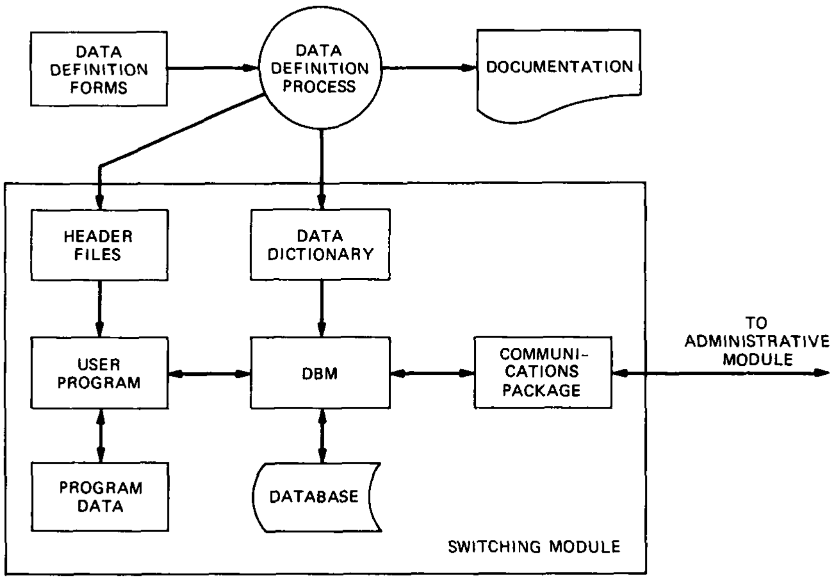


Fig. 3—Overall DBM architecture.

similar components. The user header files contain the C-structure layout (or template) of the relations and are used to compile the client programs. The data dictionaries describe all the relations, attributes, and domains in the database. They are used internally by the DBMS software.

4.2 Data access

The DBMS provides two levels of interface in accessing the database. The two levels of interface achieve different degrees of data independence for users with different performance and data usage requirements. The basic interface is the tuple-level access, which is employed by real-time critical users, e.g., call-processing programs. The user at this level can retrieve tuples of the base relations. The base relations are designed from an operational point of view and are used by all internal subsystems. The base relations represent the actual physical data stored in the database. A higher level of interface, called the view-level access, is designed to provide a higher degree of data independence for users with less stringent real-time requirements. The view-level access operates on view relations, which are virtual relations formed at execution time by combining information from one or more base relations. The view-level access provides an interface to administrative operating telephone company personnel in a *5ESS* switching system and also to Operations Support Systems through external data links.

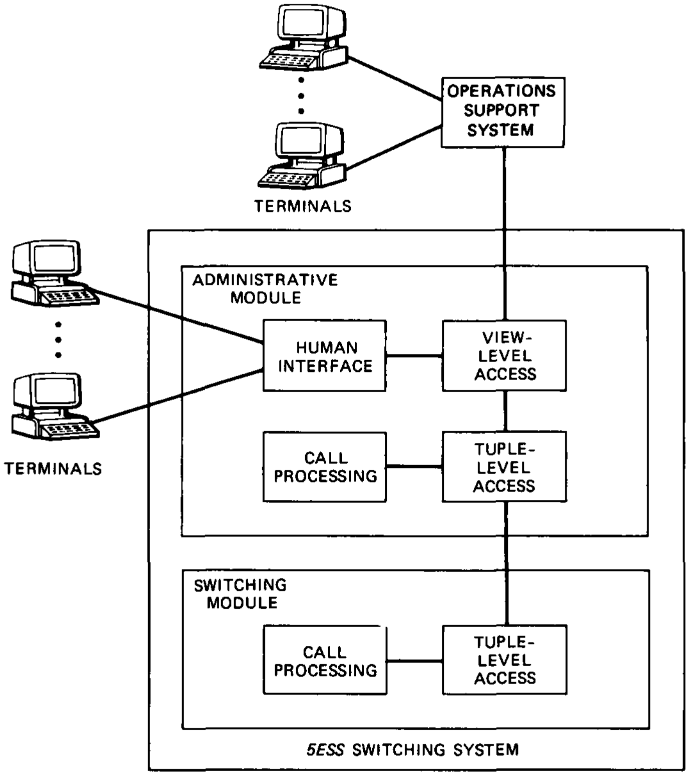


Fig. 4—DBM interfaces.

Since the view-level access is used to interface with the external users, it is only available in the administrative module. Figure 4 depicts the database management interfaces in the *5ESS* switch. All of the resident application programs access data through the DBMS on the processor on which they are executing via a set of interface primitives. These primitives are function calls, as opposed to messages, which are the communication mechanism between processes. For accessing the data on a remote processor, the DBMS automatically generates all the necessary interprocessor queries (in terms of messages).

The database is isolated from application programs. A function call causes some data to be copied from the database to a user buffer instead of providing a physical pointer to the database. This is done to ensure integrity of the data in the database. To fulfill the real-time requirement of call processing, the capabilities of the interface primitives are somewhat restricted. Only single tuples within base relations can be accessed.

4.2.1 Data access primitives

The DBM provides a basic set of primitives that serves all users. The set of primitives is "complete" so that the user can access any part of the database, but it is "basic" enough so that the DBMS does not have to perform any part of the application for the user. The tuple-level access provides control and protection functions in addition to data access. It manages transactions, controls concurrent access, and handles data distribution.

4.3 Concurrency control

Concurrency control in the DBMS is a mechanism that allows multiple users to access the data and protects them from getting inconsistent data due to concurrent updates. The policy on concurrency control has a great influence on the design of the concurrency mechanism. It determines the efficiency of shared data usage or, in other words, the degree of concurrent access.

Call processing is primarily a reader of the relatively permanent data and it contributes a large portion of the data access activity. Since call processing requires rapid real-time response, its access to the data has priority. Most changes to the database are not real-time critical and the frequency of such activities is rather low in comparison to call processing. Concurrency control is designed to provide efficient access to read-only users with potentially delayed access to update users.

The concept employed to control concurrent changes is based on data duplication. All updates are performed on a copy of the real data. During the update period, all readers of the database can still access the original version of the data. When the update is completed, the updated copy will become the real data for future users. The old copy will be discarded when all current read transactions on this relation are completed.

This mechanism allows only one writer to the database in the system to update the same relation. This, however, does not restrict the number of readers who have read-only access to the relation. Furthermore, it does not restrict the number of writers who are operating on the disjoint parts of the database. This situation fits quite well in the *5ESS* switching system's environment, where there are many readers and relatively few, infrequent writers. Simultaneous writers of the same data page are queued internally to the DBMS, so virtual concurrent update operation is still seen by the user in this case.

Since the concurrency control scheme is based on duplication of data, it is desirable to organize the storage structure in such a way that the amount of duplication is minimized. A tree-like storage structure is most suitable for this kind of application. The amount of data duplication required to perform an update in a relation consists of all the blocks along the path from the root node to a leaf node. All relations are in a tree-like structure, regardless of what kind of access method is employed.

When an update transaction terminates, the old data are replaced by the new data by changing the pointers to the control data structures. If the update process terminates abnormally (e.g., owing to software asserts or hardware reset) before the commitment of update, the current version of the data remains intact. The memory for new data that are never committed will be reclaimed immediately if possible, or will be subjected to garbage collection through data audits.

4.4 Memory partitioning and write protection

Write protection is a mechanism to protect the database against "wild" writes by any process. If an area of the memory is write protected, a process writing into this area must first turn off the write-protection mechanism. Otherwise, an interrupt will be generated and appropriate action will be taken on this process. Most unintentional writes to the database can be caught by this mechanism. Providing write protection for the whole database would create unacceptable real-time penalties for the telephone switching operations. To achieve a balance between real-time response and reliability, the memory is partitioned into a write-protected area and a non-write-protected area.

Relatively permanent data are stored in write-protected memory. Examples of write-protected data are the data dictionaries that store the schema of the relational database, and information specific to the individual office: hardware configuration, telephone line and trunk data, digit dialing analysis data, and routing data. Another class of data that is more dynamic in nature is stored in the non-write-protected memory. This class of data is used to define system dynamics such as the state of a phone call, the status of hardware equipment, work queues, etc. Dynamic data do not rely on disk backup for data recovery. Instead, data are recovered through a system of audits that either recover the data from the office-dependent data or from program initialization.

4.5 Data backup and recovery

For access efficiency, most of the data are stored in main memory. Since a power failure can destroy the entire main memory content, secondary disk storage units are utilized to back up all main memory. The disk backup is also used to restore main memory data whose integrity is suspect. An elaborate set of audit programs are used to constantly check the data integrity. The DBMS continually keeps the disk data logically equivalent to the main memory to prevent loss of the latest data updates in the event of data recovery from disk.

4.6 Data administration

A data administration capability is provided by the *5ESS* switch to enable the operating telephone company and the telephone subscriber to dynamically change the feature capabilities available to them. The programs used to provide the data administration capability are called Recent Change and Verify (RC/V) programs. The primary design objective is to provide a user-friendly, interactive interface with maximum flexibility. A comprehensive set of error checks are made on all data entered, with particular emphasis on identifying the exact cause of the error and the specific steps required to correct the error. All error checks must pass successfully before the data in the *5ESS* switch will be updated, thereby ensuring a high degree of data integrity.

The RC/V program provides the ability to add, delete, update, or verify the database using

1. A menu select/mask interface
2. A text interface
3. An operations support system interface (see Fig. 4).

The menu select/mask interface is supported from multiple recent change terminals (CRTs), which can be utilized concurrently, while the other two interfaces are associated with unique terminals. All interfaces are provided over dedicated facilities. The user specifies one of two levels of menu select. The first level of RC/V menu select allows the user to specify which form (user view) is being used. These can be verify only, or a combination verify and recent change forms. The second level requires the user to select the type of operation to be performed (i.e., recent change—new, out, change, or verify) on that form. After the user specifies both the form and the operation, the form is displayed and the RC/V data interpretation and processing subroutines are invoked.

The menu-select program also determines if a particular user terminal is allowed to perform the requested RC/V operation. Following is a list of valid terminal types and a description of the allowed RC/V operations:

Description	*Operations Allowed*
Local-RC/V	All
Remote-maintenance	All
Repair service bureau	Verify only
Network administration	Verify all data Network, administrative data

Line administration	Verify all data
	Line group data

A user may obtain a permanent copy of all recent change activities by specifying an auxiliary printer. In this case the menu program will generate a print file that consists of a serialized record of the RC/V operations that occurred in a terminal session. Each time the database is recent changed or verified, a copy of the completed form is written, along with the time and view transaction identifier and an indication of the time the transaction was actually committed in the database.

As an additional option, a scratch print file contains a copy of the input forms used, along with any error messages obtained. The file is written whenever the user specifies. This allows the user to obtain a paper copy of his or her input and allows errors in input to be resolved off-line.

The RC/V program performs range and syntax checks on each attribute put into the system. In addition, data consistency checks are made by comparing all attributes of a particular form when the user indicates that input for a particular form is complete. Finally, before committing any database changes, a data integrity check is made so that the data input on a particular form does not conflict with any data already in the database.

V. PERIPHERAL CONTROL

In the *5ESS* switch software architecture, it is the function of the peripheral control subsystem to isolate application programs from the details of switching machine hardware. It provides the sequencing and control for operations on the switching network, service units, and line and trunk units that comprise the switching periphery. As a result, other subsystems can view the switching hardware as a collection of logical resources, unencumbered by the details of the particular hardware implementation.

The abstract resources that peripheral control provides are logical ports, paths, bridges, and connectors. A logical port is an abstraction of the customer terminal's attachment to the system. Bridges and connectors represent the various types of conferencing capabilities. Paths represent the interconnections of logical ports, bridges, and connectors.

The peripheral control subsystem is divided into four parts:

1. Port control—implements the concept of logical ports and provides a set of primitives to operate on them.

2. Switching resource allocator—controls allocation of various switching resources, such as time slots and service units.

3. Network control—coordinates path operations in the digital and metallic access networks.

4. Peripheral I/O—provides low-level sequencing and control of the switching periphery, including the handling of time-critical I/O.

Each of the four major components of the peripheral control subsystem is described in more detail in the following sections.

5.1 Port control

Port control software resides only in the switching module and provides a set of primitives to operate on logical ports. Through these primitives, the application software can deal with line and trunk terminations as conceptual entities, unconcerned with the detailed sequence of operations required to control the hardware. For example, a primitive is provided to "activate" a line termination port by establishing a path through the line concentrator, performing false cross and ground tests, and initializing the associated software data structures. Another primitive is available to apply dial tone (or other appropriate signal) to a logical port. This approach provides an abstract view of the switching periphery that leads to higher productivity in the development of new features. It also allows for the periodic introduction of improved, lower-cost hardware units without modification of the application software.

5.2 Switching resource allocator

The switching resource allocator provides a set of primitives to allocate and deallocate switching resources. These resources can be divided into two categories: (1) paths in the switching network, and (2) service units. Path resources include line concentrator paths, peripheral-side Time-Slot Interchanger (TSI) time slots, and metallic access paths.*

Service units include the High-Level Service Circuit (HLSC), the Local Digital Service Units (LDSUs), and conference circuits in the global digital service units. The HLSC is a circuit in the line unit that is used to verify concentrator paths, provide cadenced ringing, and perform the needed operations for coin phones. The LDSUs are used to generate and decode digital tones used in call processing. Conference circuits are used for bridging and three-way calling.

In addition to allocation and deallocation of switching resources, functions are provided to queue for, or preempt, resources. Queueing would be used, for example, when circuit diagnostics must be run on a hardware unit that is currently in service. Preemption allows a client to gain control of a resource regardless of its current status. Preemption is not a routine activity. It is used primarily in support of high-priority maintenance actions.

* Metallic access paths are used by the terminal maintenance subsystem for testing lines and trunks.

5.3 Network control

Network control coordinates the actions required to set up and release call paths. It presents a high-level abstraction for feature control to operate on the switching network. It is through the network control software, which resides both in the switching and administrative modules, that the abstractions of paths, bridges, and connectors are realized.

Central network control software coordinates path operations, such as path setup, path teardown, and network reconfiguration, that involve the Time-Multiplexed Switch (TMS) or multiple modules. Through interactions with local network control, it determines what new paths and conference circuits are needed, reserves the appropriate resources, and sends orders to the TMS to connect the paths.

Local network control software is responsible for making connections between the originating or terminating port and the module network control and timing link. It also plays an essential role in coordinating the sequence of path operations required for establishing and releasing an intermodule path, and in inserting or dropping bridges and connectors. The coordination required is a function of the characteristics of the network and is transparent to application software such as feature control.

5.4 Peripheral input/output

Peripheral input/output software is responsible for the sequencing and control of the switching periphery. Of all the peripheral control components, it operates at the lowest level, closest to the hardware. It performs both time-critical I/O jobs (10-ms interrupt level) and base-level I/O jobs whose time constraints are not as stringent. Peripheral I/O software typically performs these jobs at the request of port control or network control programs and provides isolation from the details of the peripheral hardware.

Interrupt-level (foreground) processing is responsible for handling those tasks that require an immediate response. Examples of jobs that fall into this category are

1. Service requests from the periphery
2. Line origination scanning
3. Dial pulse and multifrequency outpulsing.

Base-level (background) processing has less stringent timing constraints. Background processing is started on a 50-ms cycle or on the reception of a report from foreground. It provides such services as

1. Dispatching digit reports to port control
2. Sending origination reports to origination-handling software
3. Implementing the overall scheduling strategy for I/O jobs.

The foreground and background peripheral I/O functions provide a fundamental set of services for controlling the peripheral hardware and responding to external stimuli. These programs interact with the other components of peripheral control to provide the application software with an easy to use, abstract interface to the switching periphery.

VI. CALL-PROCESSING SOFTWARE

Call processing[4] is principally the responsibility of three subsystems: Feature Control (FC), Routing and Terminal Administration (RTA), and Peripheral Control (PC). The roles and interactions of these three subsystems can be better appreciated by considering the flow of a simple line-to-line call. For this purpose let us assume that customer A originates a call to customer B (see Fig. 5). When customer A originates a call, the off-hook is detected by the peripheral control foreground program. Peripheral control, in turn, calls the RTA programs, which create a feature control terminal process to control the call. Checks are made to determine the identity of the line, the type of service offered (individual, two-party, coin, etc.), the type of features offered, and the type of signaling used. A path through the concentrator is set up, a power cross test is run, a digit receiver is attached, and dial tone is returned to the customer. When the first digit is received, dial tone is removed. For the first digit and for each of those that follow, messages are sent to the terminal process, and the digits are subsequently analyzed. If there is no permanent signal, partial dial, or disconnect, the call continues as normal.

At this point a request to terminate to the selected party is sent to the administrative module using a route request message. Here the RTA software determines the terminal associated with the called number (customer B). A path is then allocated between the originating and terminating terminals, the required time slots are identified, and the TMS is ordered to provide the connection. A termination message is sent to the interface module of customer B, and RTA then creates a terminating terminal process after checking for special features on the terminating line. A path through the concentrator is established, a power cross test is run, the appropriate ringing signal is started, and audible ringing is connected through the terminating Time-Slot Interchange (TSI), the TMS, and the originating TSI, back to the originating party.

When customer B answers, an off-hook is detected by the PC foreground program, which informs the terminating terminal process after checking for special features on the terminating line. Ringing is removed by the hardware when customer B goes off-hook; subsequently, the audible ringing is removed and the terminating line is cut through. The parties are now talking.

Assume that the terminating party, customer B, hangs up or disconnects. In this instance, the peripheral control program sees the

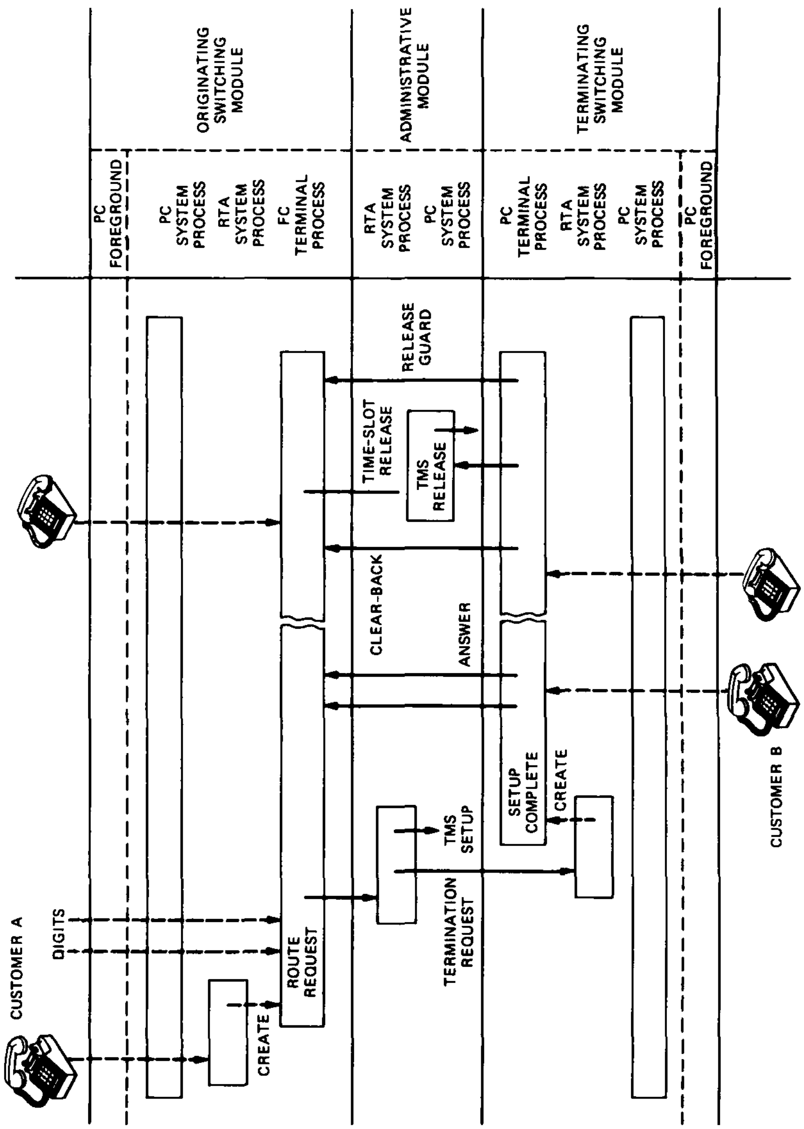


Fig. 5—Line-to-line plain old telephone service call.

on-hook and reports it to the terminating terminal process, which in turn sends a clear-back message to the originating terminal process. The originating process then begins disconnect timing. At this point, if either the originating party hangs up or the disconnect timing expires, the originating terminal process releases the call's time slot by sending a path release message to the administrative module. The terminating terminal process is also notified to release the path. It responds by idling the appropriate time slot, sending a message to the administrative module, sending a release guard message to the originating terminal process, idling customer B's port, and terminating itself. When the originating terminal process receives the release guard message, it idles customer A's port and terminates itself. Meanwhile, after the administrative processor receives the path release message from each process, it instructs the TMS to idle the path, thereby completing termination of the call.

There are many variations of this basic sequence, but the above description is the typical flow for a simple line-to-line call. In the case of three-terminal calls, such as call waiting or three-way conferencing, there is still only one terminal process associated with each terminal, and one of these terminal processes is responsible for providing the overall coordination required during the call. Even with multiterminal calls that are made up of two or more three-terminal calls, only one terminal process is required for each terminal and overall coordination responsibilities rest with one of the terminal processes. Only with interactive multiterminal calls is there a need for a separate call-coordinating process.

VII. ADMINISTRATIVE SERVICES

Administrative services[5] encompass collecting data, processing or formatting data, and outputting data to Operations Support Systems or local crafts people in the office. These wide-ranging data operations include billing, measurements, network management, and service evaluation. Much of this administrative services software is based on the processing of events—call events and system events. Call events are associated with an individual call. Examples are answer or disconnect time, called number or origination data. System events, on the other hand, are not associated with an individual subscriber, but instead consist of such things as traffic or maintenance usage data or processor utilization data. Events are stored on a per-switching-module basis, with each module processor having a call-event buffer and a system-event buffer.

It is the task of the call-processing and maintenance programs to recognize and report events to the administrative services subsystem using primitives that increment counters and associate data on a per-

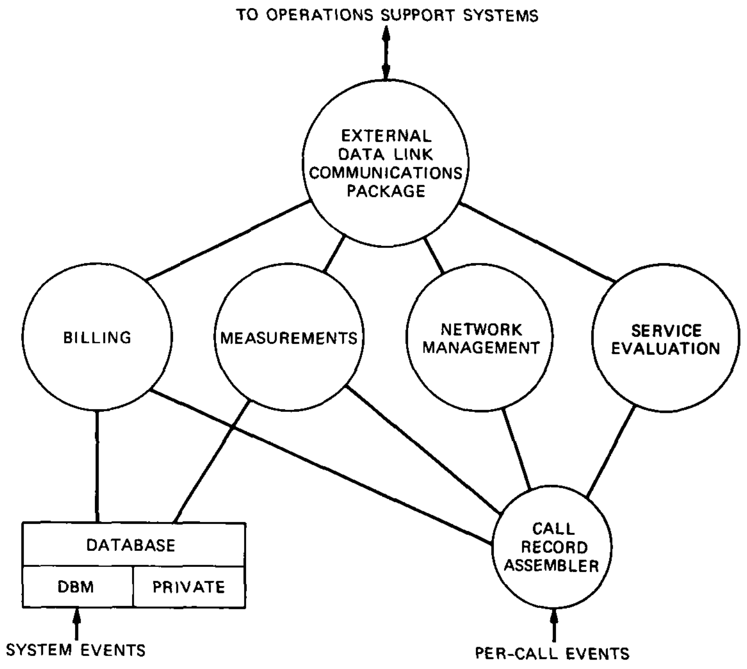
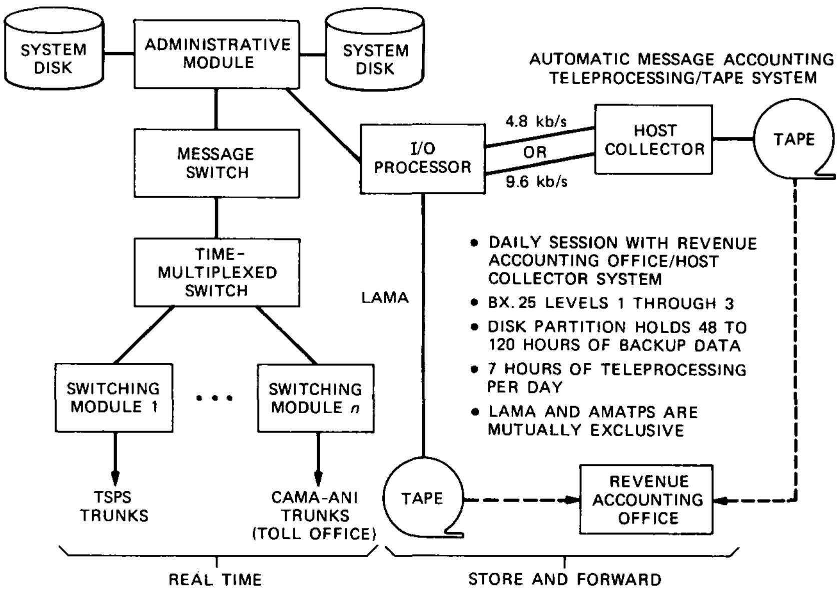


Fig. 6—Structure of administrative services software.

call basis. For example, the feature control subsystem reports the end of dialing, answer, and disconnect. The peripheral control subsystem recognizes that a circuit has been seized or a dial tone applied. The terminal maintenance subsystem recognizes line or trunk troubles, or per-call failures. In all instances, however, the event data collection and processing have been separated as much as possible from the call-processing and maintenance programs, and the data from a single event may be used for several administrative purposes. Only the event data needed are collected, and as new data are required, new administrative primitives are defined.

All the temporary data associated with an individual call are stored in a call record. The data are reported to the Call Record Assembler (CRA), which correlates the data, stores them in the proper call record, and makes the information available to the administrative services software. The memory for this record is dynamically allocated as needed.

The administrative services software is functionally organized as shown in Fig. 6. Per-call events are received by the CRA and system events are generally handled by the data base management subsystem. These data are then made available to the billing, measurements, network management, and service evaluation programs for appropriate



SYSTEM DISK
ADMINISTRATIVE MODULE
SYSTEM DISK
AUTOMATIC MESSAGE ACCOUNTING TELEPROCESSING/TAPE SYSTEM
MESSAGE SWITCH
I/O PROCESSOR
4.8 kb/s
OR
9.6 kb/s
HOST COLLECTOR
TAPE
TIME-MULTIPLEXED SWITCH
LAMA
• DAILY SESSION WITH REVENUE ACCOUNTING OFFICE/HOST COLLECTOR SYSTEM
• BX.25 LEVELS 1 THROUGH 3
• DISK PARTITION HOLDS 48 TO 120 HOURS OF BACKUP DATA
• 7 HOURS OF TELEPROCESSING PER DAY
• LAMA AND AMATPS ARE MUTUALLY EXCLUSIVE
SWITCHING MODULE 1
SWITCHING MODULE n
TSPS TRUNKS
CAMA-ANI TRUNKS (TOLL OFFICE)
TAPE
REVENUE ACCOUNTING OFFICE
REAL TIME
STORE AND FORWARD

Fig. 7—Billing options.

processing and formatting of output records. These programs, in turn, interface with the external data link communication package, which provides a path to the many external data links that interconnect the *5ESS* switch with the operations support systems.

7.1 Billing

The Automatic Message Accounting Teleprocessing/Tape System (AMATPS) provides stand-alone Local Automatic Message Accounting (LAMA) tape operation and/or teleprocessing of automatic message accounting data to a centralized Host Collector system (HOC). The *5ESS* switch supports many different billing configurations. Alternatively, the *5ESS* switch can support Centralized Automatic Message Accounting (CAMA) operation with Traffic Service Position System (TSPS) or with toll offices such as the *4ESS*™ switching system.

These billing configurations involve either real-time operation or a store and forward arrangement. Real-time operation implies that billing information is moved out of the office as quickly as possible. CAMA operation satisfies this definition. The possible real-time configurations are illustrated in Fig. 7, where Automatic Number Identification (ANI) information is output to TSPS or to a toll office for CAMA operation.

With the store and forward arrangement, billing data are stored in a single-entry format on a disk. These data can be forwarded on demand to a LAMA tape, as shown in Fig. 7. Alternatively, the data may be teleprocessed on a polled basis to a HOC. The interface to the HOC can be a 4.8-kb/s dial-up link or a 9.6-kb/s dedicated link.

7.2 Measurements

A second major portion of administrative services is the measurement package. These measurements fall into three different categories: plant, traffic, and service evaluation. Some of these measurements are peg counts, others are usage counts, still others are overflow counts. All of these counts are processed and subsequently provided to the craft either locally or remotely through operations support systems in a very human-oriented form. All craft reports are appropriately formatted with titles provided so that no templates are required to decipher the data.

The measurement package provides data to EADAS, the No. 2 Switching Control Center System (No. 2 SCCS),[6] and the Master Control Center (MCC). Twenty-four-hour plant measurement data are provided to the SCCS and the MCC; 30-minute traffic reports and Division of Revenue data are sent to EADAS, and a 15-minute traffic report is always sent to the MCC, while the 30-minute report is provided on demand. Optionally, this information is available on a local printer.

7.3 Network management

The *5ESS* switch provides a very powerful network management package, essentially as robust as that available with the *4ESS* toll switch. The package provides real-time surveillance data and implements an assortment of network management controls. All of this is intended to optimize the call-handling capacity of the switch and to maintain external network sanity during periods of traffic overload or failure.

The *5ESS* switch interfaces used to support network management are shown in Fig. 8. They include interfaces with EADAS/Network Management (NM), the No. 2 SCCS, and a local terminal. The communication with EADAS/NM in the Network Management Center (NMC) takes place through the No. 1A EADAS, and the *5ESS* switch interfaces with EADAS through a dedicated 2400-b/s X.25 synchronous link. EADAS/NM is provided with data that includes five-minute surveillance data and thirty-second discretes. The crafts people, in turn, can invoke a variety of trunk group controls, including skip, cancel-to, cancel-from, and reroute. The reroute control allows out-of-chain routing during failure or overflow. Also available are code

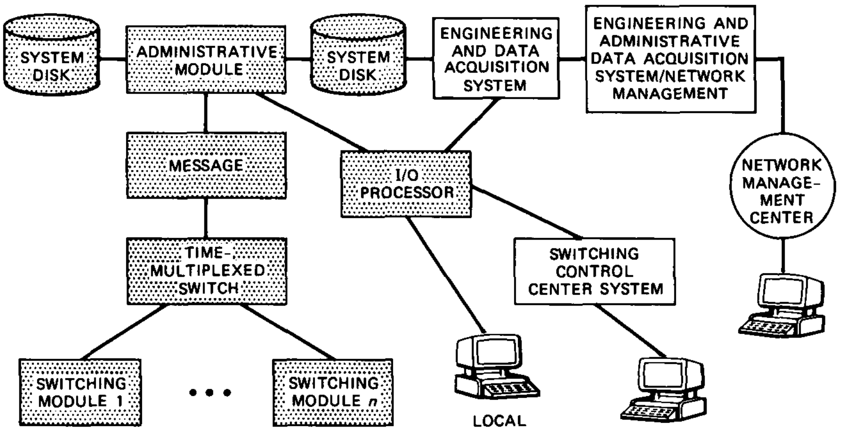

SYSTEM DISK
ADMINISTRATIVE MODULE
SYSTEM DISK
ENGINEERING AND DATA ACQUISITION SYSTEM
ENGINEERING AND ADMINISTRATIVE DATA ACQUISITION SYSTEM/NETWORK MANAGEMENT
MESSAGE
I/O PROCESSOR
NETWORK MANAGE-MENT CENTER
TIME-MULTIPLEXED SWITCH
SWITCHING CONTROL CENTER SYSTEM
SWITCHING MODULE 1
• • •
SWITCHING MODULE n
LOCAL

Fig. 8—Network management interfaces.

controls and call gapping. All network management controls that have been invoked can be retired with a single command from any of three different locations: the network management center, the switching control center system, or from a local terminal in the office.

7.4 Service evaluation

Service evaluation is a feature used to determine the quality of service experienced by the telephone customer. The process is now fully automated with the advent of the No. 2 Service Evaluation System (SES). The *5ESS* switch interface with the No. 2 SES consists of a dial-up 2400-b/s X.25 synchronous data link and from one to four monitoring links. The arrangement supports dial-line service evaluation, as well as Mechanized Evaluation of Call Completion Anomalies (MECCA). The latter involves monitoring outgoing calls that have long holding times without answer supervision. The process helps to detect fraud and to find trunks that are faulty from a billing standpoint. MECCA can be used for all line-to-trunk calls and up to four calls can be observed simultaneously.

VIII. REMOTE SWITCHING

The Remote Switching Module (RSM) is the first of a series of digital remote switching products realized within the basic architectures of *5ESS* switching system hardware and software. It provides to customers at the remote site the same services as those served by the host, and it meets the same demanding reliability requirements. The RSM, which can be located up to 100 miles from its host, permits economic replacement of Community Dial Offices (CDOs) with up to

4000 lines and the establishment of new wire centers with a smaller size than previously economical.

8.1 Remoting facilities

The RSM is a standard switching module that uses essentially the same hardware and software as an existing Switching Module (SM). It is connected to an SM at the host office using 4 to 20 digital facilities (see Fig. 1). Two or four of the digital facilities carry permanently reserved control time slots over which the RSM processors communicate with the rest of the system.

8.2 Operational software design for remote switching

The operational software for the RSM supports two modes of operation. The normal linked mode occurs when there is proper functioning of the communication channels between the RSM and the Administrative Module (AM) through an SM at the host. During linked operation, the RSM provides all of the calling features of the host office. The stand-alone mode is entered when communications over the digital facilities is lost owing, for example, to a cable cut. In the stand-alone mode, it provides normal handling of intramodule calls, including most custom calling services, and special handling of emergency and operator calls that would otherwise be routed to the host.

The operational software design for the RSM takes into account three factors introduced by remote switching operation:

1. Intermodule calls involving the RSM require an extra stage of time-division switching beyond that used for local switching modules. This stage is provided by the time-slot interchange unit of the SM in the host.
2. Intramodule calls must continue to be served during stand-alone operation. Thus the call-processing design must handle intramodule calls independently of the AM.
3. Special treatment of emergency calls is required during the stand-alone mode. In addition, intermodule calls must be given an appropriate failure treatment.

8.2.1 Call connections for remote switching

A major portion of the RSM-specific operational software controls the path allocation and path setup mechanisms necessary to incorporate the additional stage of switching into the call connection software. This software is completely within the peripheral control subsystem and, consequently, the application-level software, such as Feature Control (FC), is unaware of the changes required to interface with the expanded switching network. In all, five additional path types are supported (see Fig. 9).

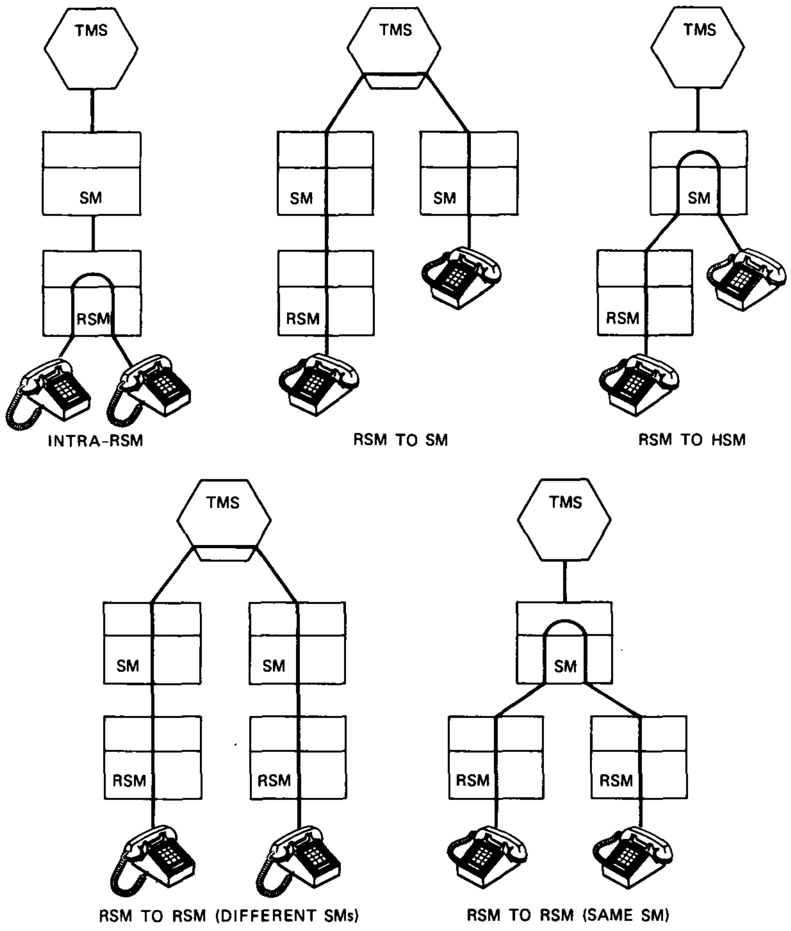


Fig. 9—Remote switching module path configurations.

For a call between local switching modules, path setup is accomplished through interaction with network control in the AM. In a case involving the RSM, the procedure is very similar (identical from the feature control point of view). The roles of the originating and terminating terminal processes and the path setup protocol remain essentially the same. There are, however, some differences, which are hidden within peripheral control. The connection through the TMS must be made to the RSM's Host SM (HSM). This is different from the local SM case, where the TMS connection is always made to the originating or terminating module itself. Also, the setup of the originating or terminating portion of the path involving an RSM is more complicated since two modules, the host SM and the RSM, are involved. The primitive, which is used to set up the originating or terminating end of the path, determines if it is executing in an RSM and, if so, coordinates with software in the host SM to establish the required connections. This coordination function is placed within the existing call connection procedure so that this activity is transparent to the call-processing actions on the other end of the call.

8.2.2 Distributed routing for remote switching

The administrative module provides the call-routing function for ordinary calls, and routing data therefore reside in the AM. Stand-alone operation in the RSM requires a more distributed approach. In this case, the local routing algorithm in an RSM proceeds through data accesses and manipulations relating to resident destination addresses. If the destination of the call is an RSM terminal, defined in the extracted routing data stored in the RSM, all routing is handled by the RSM. If the data stored in the RSM are insufficient to determine the termination address, control is forwarded from the RSM to the AM, which completes the routing process.

The call connection and distributed routing techniques provide a full complement of services to lines served by the RSM. They also provide the basis for stand-alone call processing, a mode that may be assumed (a telephone company option) when communications with the host fail.

8.2.3 Stand-alone mode

In the stand-alone mode, the RSM continues to handle intra-RSM calls normally and provides for special treatment of other calls. Normal intermodule calls, originating at the RSM and terminating on another module (including outgoing calls), are automatically routed to either an announcement or a reorder tone (telephone company option) at the RSM. Emergency calls, originating on the RSM and normally handled as intermodule calls, may be routed to designated lines on the RSM during stand-alone operation. Up to ten directory numbers may be specified in the emergency category. Global hunt groups (hunt groups with appearances on the RSM and one or more other switching modules) are reconfigured to permit hunting over the local subset of the global group during stand-alone operation.

IX. SUMMARY

This paper has described the operational software structure used to provide the major call-processing and administrative function in the *5ESS* switching system. It demonstrates the functional decomposition of the software through use of a general operating system and DBM, as well as a high degree of hardware independence achieved by use of the operating system and the peripheral control subsystem.

X. ACKNOWLEDGMENTS

The authors wish to acknowledge the efforts of those designers, too numerous to mention individually, who contributed to the successful operational software system design.

REFERENCES

1. D. L. Carney et al., "The *5ESS* Switching System: Architectural Overview," AT&T Tech. J., this issue.
2. W. H. Huen et al., "*5ESS* Switching System Software: Operating System Structure," AT&T Tech. J., to be published.
3. M. R. Locher, L. R. Pfau, and D. W. Tietz, "*5ESS* Switching System Software: Database Management System," AT&T Tech. J., to be published.
4. D. A. Anderson et al., "*5ESS* Switching System Software: Call-Processing Software Structure," AT&T Tech. J., to be published.
5. J. M. Erickson et al., "*5ESS* Switching System Software: Billing and Administrative Services," AT&T Tech. J., to be published.
6. J. J. Bodner, J. R. Diano, and K. A. VanderMeulen, "Traffic Service Position System No. 1B: Switching Control Center System Interface," B.S.T.J., *62*, No. 3, Part 3 (March 1983), pp. 941–57.

AUTHORS

John P. Delatore, B.A. (Mathematics), 1963, College of Steubenville, Steubenville, Ohio; M.A. (Mathematics), 1965, Bowling Green University, Bowling Green, Ohio; AT&T Bell Laboratories, 1965—. Mr. Delatore has worked on Traffic Service Position System (TSPS) program design and TSPS test evaluation. He worked at AT&T from 1973 to 1975 providing computer-aided service cost methodologies. In 1975 he became Supervisor of the TSPS Growth and Field Support Group, and in 1977 he became Supervisor of the TSPS Planning Group. In 1979 Mr. Delatore was appointed Head of the Credit Card Data Base Feature Programming Department. Beginning in 1982 he worked on the database, factory system test software, and operational software for the *5ESS* switch. In 1984 Mr. Delatore assumed his current position as Head of the Local Switching Systems Engineering Department.

Rudolph J. Frank, B.S. (Electrical Engineering), 1966, Seattle University; M.S. and Ph.D. (Electrical Engineering), 1968 and 1971, respectively, Oregon State University; M.S. (Business Management), 1981, Stanford University; Pacific Northwest Bell, 1964–1966; AT&T Bell Laboratories, 1971—. At Pacific Northwest Bell, Mr. Frank was Supervisor in the electronics data processing area of the Comptroller's division. At AT&T Bell Laboratories he worked in the TSPS laboratory. In 1975 he was designated Bell Laboratories Visiting Professor to Southern University (Baton Rouge, La.). Mr. Frank became Supervisor of the *4ESS* Network Management Control Group in 1976 and has worked on several large software development projects relative to the domestic and international telecommunication markets. In 1980 he was selected as a Sloan Fellow by the Graduate School of Business at Stanford University. In 1981 Mr. Frank was appointed Head of the Toll Digital Systems Development Department, where he worked on toll features for both *4ESS* and *5ESS* switching systems. In 1985 he assumed the position of Director of the *5ESS* System Software Laboratory at AT&T Bell Laboratories in Naperville, Illinois. Member, IEEE, Eta Kappa Nu.

***Hans Oehring**, B.S., 1958, Lafayette College, and M.S., 1960, Ohio University,* both in Applied Mathematics; AT&T Bell Laboratories, 1960—. Mr. Oehring joined AT&T Bell Laboratories in 1960 as a Member of Technical Staff in 1*ESS* system development. He was named Supervisor of a 1*ESS* system development group involved in the development of business customer services in 1967. In 1977 he was promoted to Head of the 1/1A *ESS* Software Design Department, responsible for field support, planning, architecture, database and maintenance software. He transferred to the *5ESS* system project in 1980 as Head of the *5ESS* Operational Program Department, responsible for the design of the operating system, data base manager, peripheral control software, administrative services, and the overall software architecture. He became Head of the *5ESS* Application Maintenance and Growth Department in 1983. His department is responsible for the software design and development of the basic Integrated Services Digital Network operational and maintenance capabilities. The department also develops the fault-recovery software for many of the peripheral hardware units and provides the growth scenarios for these units. Member, Sigma Pi Sigma.

L. C. Stecher, B.S., 1967, Loyola University; M.S., 1968, Northwestern University; Ph.D., 1972, Northwestern University, all in Applied Mathematics and Computer Science; AT&T Bell Laboratories, 1970—. Mr. Stecher was involved in the design and development of the 4*ESS* maintenance and call-processing subsystems. In 1975 he became Supervisor of the 4*ESS* trunk maintenance development effort and later supervised an exploratory effort to define new network services for the evolving Stored Program Controlled Network. In 1980 he became Head of a department responsible for development of software for the Traffic Services Position System. In 1982 he became Head of a department responsible for the development of the *5ESS* Remote Switching Module. In 1983 Mr. Stecher took over responsibility for the *5ESS* Project Management, Architecture, and Planning Department. This department is responsible for determining the feature content of future *5ESS* switching system generics, specifying the software and hardware achitectural modifications necessary to develop these features, and providing the ongoing project management for the generic development.