AT&T Technical Journal
Vol. 64, No. 1, January 1985
Printed in U.S.A.

System 75:

Switch Services Software

By W. DENSMORE, R. J. JAKUBEK, M. J. MIRACLE, and J. H. SUN*

(Manuscript received July 11, 1984)

The switch services software of System 75 provides the basis for an extensible office communication system, supporting a wide variety of voice and data-switching services. This paper presents the software architecture of the System 75 switch services. The concepts of user, group, and process-per-call form its foundation. We introduce the architecture by stepping through a simple station-to-station phone call, and proceed to the derivation of a call model based on the topology of a call. This call model is realized as a layered set of cooperating processes that execute under the Oryx/Pecos Operating System on the System 75 switch processor. The software layers and processes are discussed and a call walk-through is used to illustrate the process interactions.

I. INTRODUCTION

The System 75 software supports a wide spectrum of terminals that range from a single line station to a sophisticated digital station for simultaneous voice and data communications. It also supports more than 150 features for office and business communication needs. Among them are the station features for handling multiple, simultaneous calls, features used for covering unanswered calls, routing features to select

* All authors are members of AT&T Information Systems Laboratories, an entity of AT&T Information Systems, Inc.

the least expensive network facility, capabilities that allow messages to be left for the called party automatically, and terminal dialing and modem pooling features for data communications services.

This paper describes the switch services software architecture of System 75, a framework that supports a wide variety of features and terminals. Section II states the design challenges and Section III illustrates a basic call scenario leading to the derivation of an essential call model. The concepts of user, group, and call are introduced, and are mapped to a basic software structure. In Section IV, this structure is generalized to a layered software architecture consisting of a set of cooperating processes. A call walk-through is used to illustrate the interactions among processes.

II. DESIGN CHALLENGES

The challenges in building switching software for an office communication system include:

- The vast number and variety of features to be supported
- The need to integrate different office services
- A wide variation in the capabilities of existing and future terminals
- Stringent real-time response criteria
- The asynchronous and concurrent nature of the external world.

Our design began with an analysis of the feature operations and resource management requirements of a switching system. Then, the essence of the feature and terminal operations were extracted into functional modules and the basic primitives of the system were defined. Next, we formulated a call model by analyzing the dynamic behavior of a call and the relationships between the functional modules. The functional modules were then layered into a set of cooperating processes, with the primitives of the system provided through message-based interfaces. Finally, information hiding and synchronization techniques were applied to simplify the software structure and to enforce stronger partitioning of functions.

A primary goal of this modular and disciplined architecture is to minimize the effort required to add new terminals and features. This architecture should also be easy to understand by software developers so that the architectural integrity is preserved over the product life. Moreover, the architecture should simplify the integration of more data-processing-like functions in the future.

Trade-offs exist between implementing the design goals and meeting the real-time requirements of a switching system. For example, a virtual terminal interface provides for uniform implementation of features across all terminal types at the expense of access time to the terminal. The primitives of the system are carefully defined to balance between generality and efficiency.

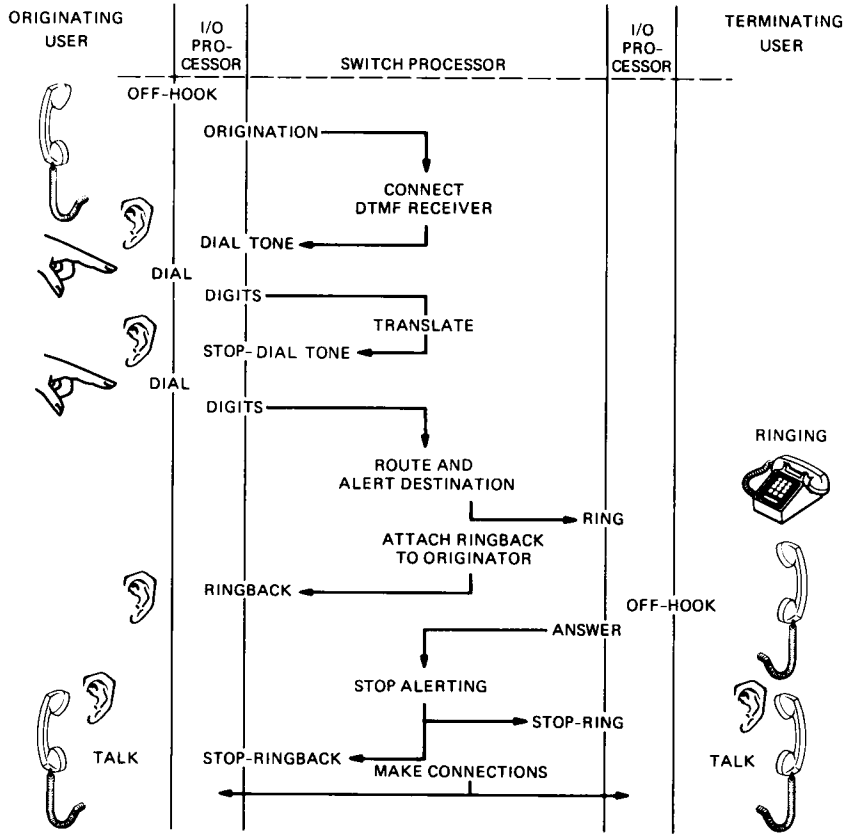

Fig. 1—Basic call example.

III. ARCHITECTURAL MODEL

To introduce the design of the switching software, we will step through a basic telephone call and develop a call model. Then the System 75 realization of this call model will be presented.

3.1 Basic call example

A user originates a call by going off-hook on his/her phone. This change is detected by an I/O peripheral and an off-hook signaling message is sent to the switch processor (see Fig. 1). On receiving this message, the resources for processing this call are allocated and messages are sent to the switch network for connecting a Dual-Tone Multi-Frequency (DTMF)* receiver and giving dial tone to the user.

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

The call processing software interprets each digit dialed by the user and routes the call to the terminating station when all the digits have been dialed. The call is signaled to the terminating user with ringing, and the call progress is indicated to the originating user with ringback tone. When the terminating user answers the call, the switching network is instructed to remove the ringing signal and the ringback tone, and establish a talking path between the originator and the terminator. Finally, when either the originator or the terminator goes on-hook, the call processing software tears down the circuit connection and deallocates all the resources associated with the call.

This example illustrates the major functions of System 75 in processing a basic call:

- Terminal Handling
  The voice terminals supported by System 75 range from a single-line analog station to a simultaneous voice/data station with display, multiple call appearances, feature buttons, and data module. A variety of trunks are used to interconnect with other switching systems or a central office switch.
- Resource Management
  In addition to terminals, there are other resources in the system that need to be managed. These include the DTMF receivers, the time slots for circuit connections, tone generators, and the internal software records for call processing, messaging, measurements, and call detail recording.
- Call Sequencing Control
  Of the more than 150 features suported, many involve complicated sequencing logic to bring a call from one state to another. For example, the call coverage feature specifies the selection of new call destinations (coverage users) if no answer occurs in a specified time interval at either the principal destination or the current coverage destination. A conference call is another example where, in response to a user's conferencing request, the internal records and the circuit connections of the two initially distinct calls are merged to establish a common talking connection for all the parties.
- Routing and Termination Selection
  Routing and termination refer to the selection of a terminating endpoint or set of endpoints for a call. There are a wide variety of algorithms for routing and termination selection, such as hunting, bridging, coverage, least-cost routing, and routing data calls through pooled modem resources.

3.2 Call model

By analyzing the dynamics of the software functions in the above

list of functions, we can derive a call model that contains three major components: the *call*, the *group*, and the *user*. Highest in the hierarchy is the call, which ties all the parties of a connection together. Next is the group, which appears as a party on the call and contains a set of users. The user is an entity that models a terminal or a set of terminals that belong to a system user. This hierarchy is illustrated in Fig. 2. Let's examine the call, group, and user concepts in more detail.

3.2.1 The call

The call is associated with a set of connected parties. It is defined by a record of these parties plus the sequencing control logic of the call. It resolves asynchronous actions of various parties and directs the system's responses to these actions for the various feature operations. The call abstraction separates the common sequencing logic and information of a call from the group feature operations and the terminal handling.

3.2.2 The group

The group models a collection of users who are associated to provide special call and feature operations for the customer. There are several group types in System 75; each one contains special algorithms for the operations of that group type. For example, the hunt group specifies how a user should be selected from a group to receive a call. The group abstraction hides the internal operation of group features from the call, and provides a uniform structure for handling such group-oriented features as hunting, bridging, multiple attendants, and trunk groups.

3.2.3 The user

The user models the end user in the system, who may have a single telephone instrument or possibly a collection of interacting terminals. The user abstraction is a terminal handler that provides a set of

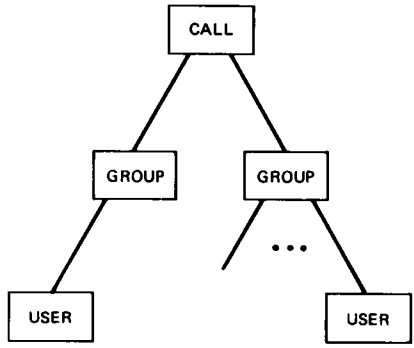

Fig. 2—Basic call model.

resources for which the different services compete: voice and data channels, status indicators, displays, ringers, etc. Trunks and data lines are also modeled as users. The intent of the user abstraction is to hide the terminal-specific operations from both the group and the call operations.

3.3 Realization of call model

The functional modules and dynamics of the call model are mapped into a software realization consisting of a layered set of cooperating processes. The processes execute under the Oryx/Pecos operating system[1] on the switch processor of the System 75 control complex.[2]

The call is realized by a call record and a single, transaction-oriented process that controls a call from origination to completion. A set of call processes exist to handle multiple calls in the system.

The group and user are realized by a group-manager and a user-manager process. Conceptually, there is a process per instance of each different group or user; in practice a single group-manager and a single user-manager process are multitasked to handle all instances of groups and users. The group-manager and user-manager processes provide a set of primitives that are independent of the group and user types of the system. These primitives serve as the building blocks for the upper-level software (e.g., the call process), where the feature sequencing is implemented.

This call model is a hybrid of both the process-per-call, or transactional structure,[3] and the process-per-terminal, or functional structure of switching systems.[4] The call process uses a process-per-call approach, whereas the group and user managers support the process-per-terminal approach. This hybrid design permits synchronizing the interactions of various terminals, as in the process-per-call approach, and permits isolating and distributing the terminal processing software, as in the process-per-terminal approach.

IV. SWITCH SERVICES SOFTWARE STRUCTURE

So far we have presented the call-process, and the group-manager and user-manager processes. There are additional processes that provide messaging and station services and the network and resource management functions of the switch. To organize the software, we define a service-control layer, a resource layer, and a driver layer of software (see Fig. 3). Within each layer, the various processes provide further information hiding and separation of functions.

4.1 Service control layer

The Service Control Layer contains a Service Dispatcher (SD) process and a process for each of the different services of System 75.

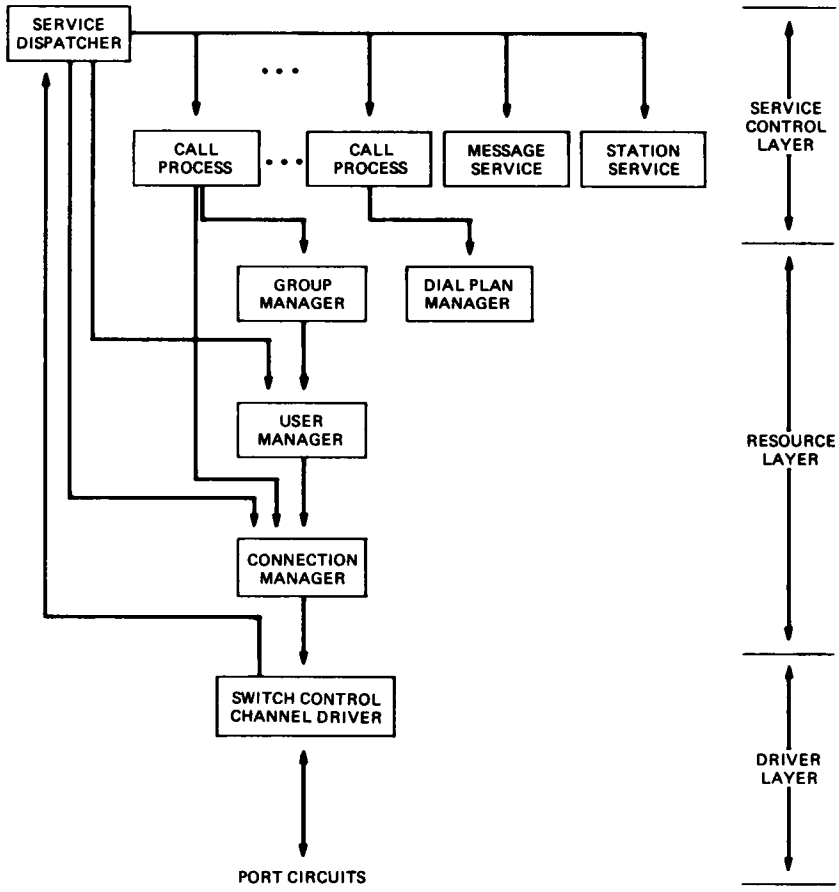

Fig. 3—Switch services software structure.

The service processes execute concurrently under control of the service dispatcher. User actions are translated into service commands by the resource layer and these commands are executed by a service process through primitives supplied by the resource-layer processes. Attention is given to the proper synchronization of user actions to guarantee that out-of-sync situations and deadly embraces cannot occur between concurrent transactions.

4.1.1 Call service

The Call Process (CP) provides the control and sequencing logic for call set-up and take-down and for a variety of feature operations in the system. Since a process-per-call is costly in terms of system resources, only a limited number of call processes exist to serve all calls. A call process is allocated to each call in a transient (e.g., dialing)

state and is deallocated from calls that have reached a stable (e.g., talking) state. The information is saved in a call record in the service dispatcher. The service dispatcher manages the pool of call processes, creating, allocating, suspending, and resuming them to/from calls.

There are many primitives supported by the resource-layer processes. The function of the call process is to invoke these primitives in the proper order and thereby produce the required response to the external user. It does this by analyzing the service-command message together with the current call state (e.g., idle or talking), and then invoking the appropriate sequencer code for that phase of the call. A call sequencer handles a special set of operations such as routing, answer, or drop (see Fig. 4). The operations are performed by invoking the primitives of the resource layer, which results in the driving of the hardware circuitry.

In addition to managing the call processes, the service-dispatcher process receives the signaling messages from the network and forwards them to the appropriate terminal handler process for interpretation into service commands. The appropriate service process, e.g., the call process, is then dispatched to execute the service command.

4.1.2 Message service

The control for the messaging services such as leave-word-calling and manual-message-waiting is provided by the Message-Service (MSG) process. The message service is a permanent server process, providing service for all users in the system. Message services can be invoked from a call by user terminal input or by other services. When message services are accessed via a call, the message-service process interacts with a call process. The message-service process accesses the

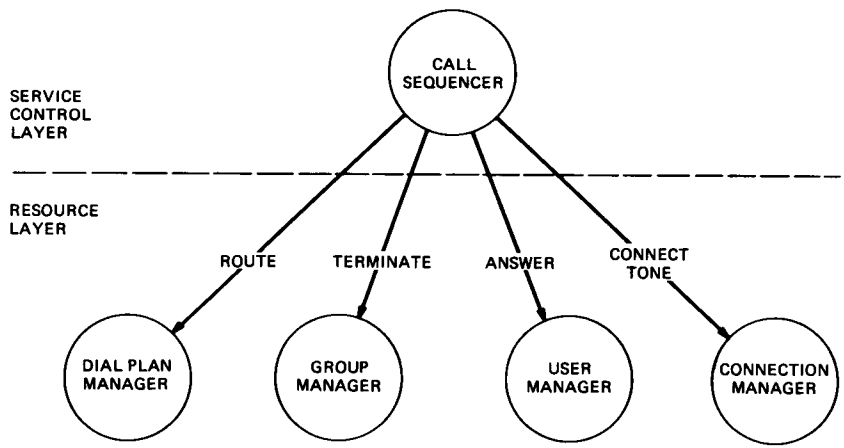

Fig. 4—Examples of resource layer primitives.

resource-layer processes for terminal input/output and translation data access.

4.1.3 Station service

The Station-Service (SSV) process provides miscellaneous station services such as integrated directory service, time-of-day display service, and programming some translation data from the user's terminal. The station-service process is a permanent server, handling all users in the system, similar to the message-service process. The station-service process accesses the resource-layer processes for terminal I/O and data access. When station services are accessed via a call, the station-service process also interacts with a call process.

4.2 Resource layer

The resource layer provides general resource management for the services, service-specific functions, and the line/terminal signaling. The system resources managed include the switch network, DTMF receivers, tones, trunks, telephone terminals, data terminals, groups, and databases like the system dial plan and the name/number directory. Call routing, queueing, terminal administration and maintenance, and feature activation primitives are some of the service-specific functions provided.

The software at the resource layer is functionally organized around the user and group concepts, the switch network, and the routing and directory database. A process exists for each of these main functions. Primitives are provided to the services by these processes for resource access. These primitives use the synchronous message facilities of the operating system to support the processing of user actions.

4.2.1 Group management

The Group-Manager (GM) process has all the translation data for group membership and group properties, and it maintains the state of the group and its members. Service-specific functions are provided by the group manager to manipulate the groups for the different services and features of the system. The group executes the service command on the users or members in the group according to the type of group. For example, in response to a call service terminate command, the coverage group would sequence the termination to each idle member in the group (an alert-all algorithm), while the uniform-call-distribution hunt group would select the longest idle member to receive the call.

4.2.2 User management

The User-Manager (UM) process contains both the user and terminal management software and status information. It presents an

abstract user or virtual terminal interface to the upper layers of software, while handling the signaling with terminals at the Driver Layer. Terminal access contention between the switch services, maintenance, and system administration is arbitrated by the user manager.

The user manager provides service-specific primitives to manipulate the users and terminals in the system. For example, the terminate primitive would, for a user with a multibutton telephone, select an idle call button, update the button-status lamp to flashing, and start ringing the telephone. The lamp and ringer operations are performed by sending signaling messages to the port circuit that interfaces the telephone.

Signaling-message interpretation is done by the terminal-handler functions in the user manager. Low-semantic-level messages, e.g., off-hook or button-push, are interpreted into service-level commands like originate and answer. A service-control-layer process then executes these commands by invoking the corresponding primitive provided by a resource-layer process.

4.2.3 Data management

The Dial-Plan-Manager (DPM) process provides access to and interpretation of translation databases in the system, including the system dial plan, the name/number directory, user permissions, least-cost routing patterns, and speed-calling numbers. Customized access to the data is provided for the call service. For example, digit analysis and the choosing of the initial routing destination are performed by the dial plan manager in response to a request from a call process.

4.2.4 Network management

The Connection-Manager (CM) process is responsible for the management of the network resources and for network control signaling. It abstracts the physical characteristics of the switch network and provides network connection primitives. Primitives to access the network resources like the DTMF generators and receivers are provided. Contention between the switch services, system maintenance, and system administration for the network resources is arbitrated here.

The communications network in System 75 is distributed onto each port board.[1] Network control messages to do time-slot assignment for listening and talking, gain adjustment, and six-party conferencing are sent by the connection manager to the port boards.

4.3 Driver layer

The driver layer encompasses the operating system drivers and the firmware in the intelligent port circuits of the System 75 communications network. The drivers include the Switch Control Channel

Driver (SCD), asynchronous data channel drivers, a timer driver, and bus drivers. The SCD interfaces the switch processor to the network control channel of the Time Division Multiplexed (TDM) bus (see Fig. 5). The network control (or archangel) acts like a full-duplex message switch between the switch processor and the microprocessors in the port boards (commonly referred to as angels). Error correction is provided between the SCD and the angels and flow control is managed by the archangel.

A functional message set is used between the switch processor and the angels. Signaling and network control messages are sent from the user and connection manger processes to the port circuits. The signaling messages are common across all types of ports. This helps to

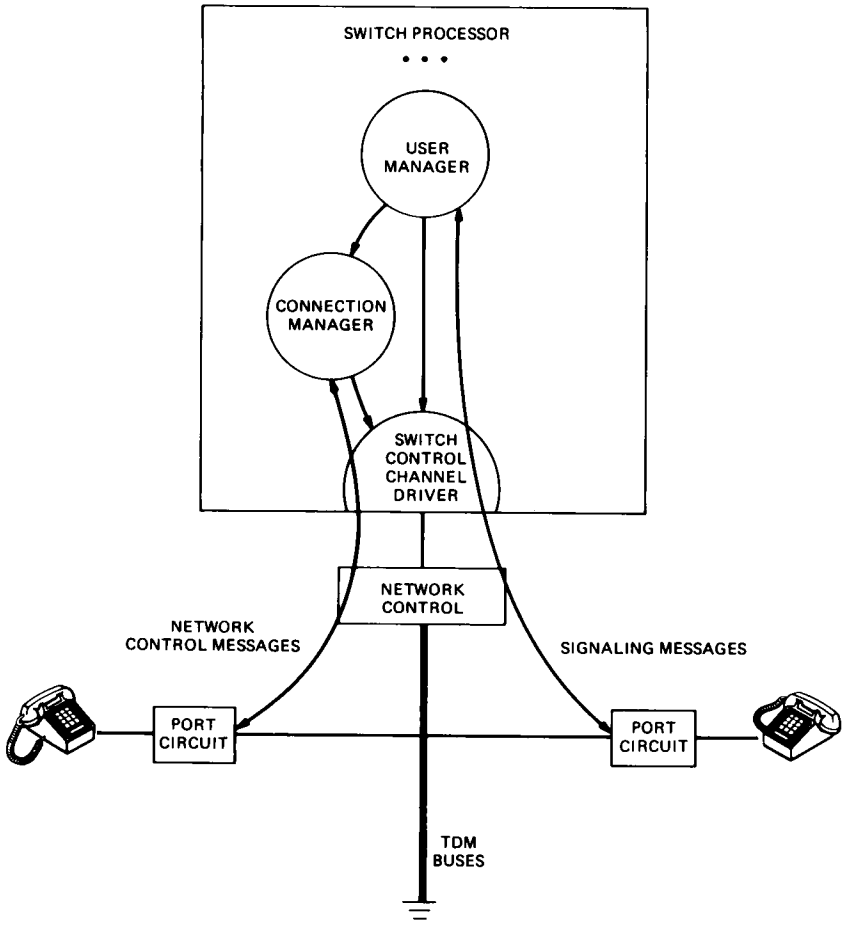

Fig. 5—Communications network control.

isolate the user manager from terminal-specific differences, providing the first level of terminal abstraction.

The angels off-load the switch processor by executing the low-level port-scanning, driving, and timing functions. This includes the ringer, tone, and lamp cadences, button scanning, and the timing for trunk signaling.

4.4 Call walk-through

A high-level walk-through of a call to aid in understanding the operation of the switch services software is presented here. The call is a station-to-station call between multifunction stations.

4.4.1 Origination

A station user goes off-hook with an idle call button selected, indicating the user wishes to "originate" a call. The off-hook signaling message is received by the service dispatcher, which forwards this signaling message to the user manager for interpretation. The user manager examines the terminal and user state and returns a call service originate command for the user back to the service dispatcher. The service dispatcher then allocates a call record, assigns an idle call process to this new call, and forwards the service command to the call process.

The call process enters the call setup mode, first requesting the connection manager to reserve network resources for the new call and then commanding the group manager and user manager to originate for the user. The user manager requests the connection manager to connect the originating user to the call and handles the lamp indications to the originating user. The origination message sequences are shown in Fig. 6.

Next, the call process requests the group and user managers to collect digits, or get a destination for the call. The user manager determines how to collect digits for this type of user and starts the collection, typically by requesting a DTMF receiver from the connection manager. Finally, the connection manager is requested to connect dial tone on the call. This finishes the setup part of the call and the user is now in a digit-collection state.

4.4.2 First digit

When the digit is received by the service dispatcher, it is forwarded to the connection manager, which has data about the call that the DTMF receiver is connected to. The connection manager returns a message that indicates a digit and the call number the digit is for. The service dispatcher forwards this message to the call process allocated to this call. The call process removes dial tone from the call and then

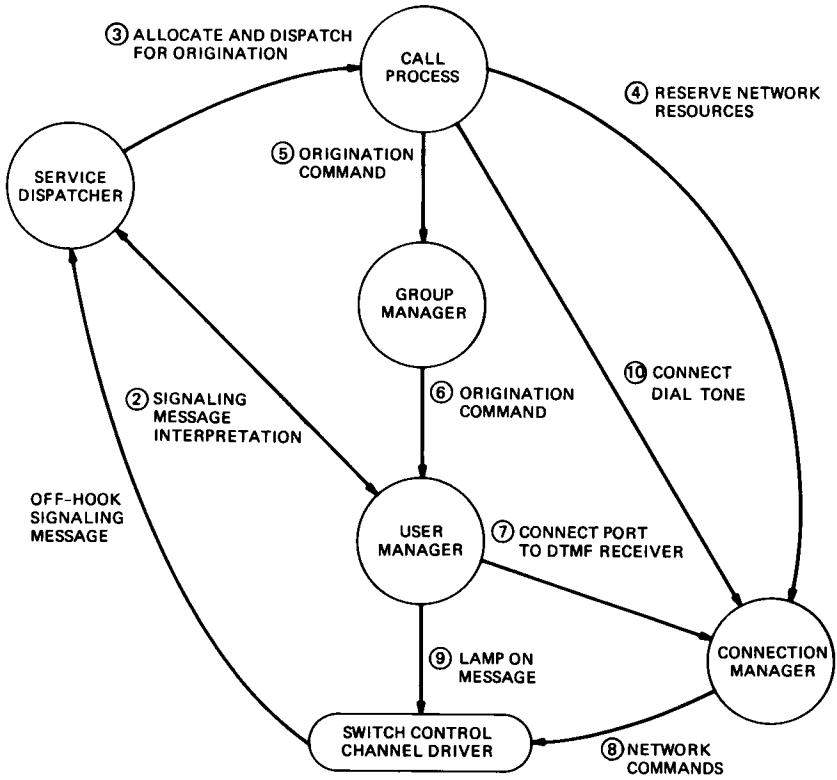

Fig. 6—Call origination sequence.

requests the dial plan manager to route the call based on the collected digit. The dial-plan manager returns data on how many more digits need to be collected. The call process then waits for the required number of digits or for an interdigit time-out or a hang-up.

4.4.3 Last digit

The last digit of a call is the signal that triggers the selecting and alerting of a destination. When the required number of digits have been received by the call process, it requests the dial plan manager to route the call based on the dialed digits. Permission checking is also done on the routing attempt. If enough digits have been dialed and the dialed destination is valid, the dial plan manager responds with the initial destination user. The call process then stops digit collection and the user manager releases any digit-collection resources from the call, such as the DTMF receiver.

Next, the call process attempts to terminate (i.e., bring in) the call to a destination user by requesting the group manager to terminate

the call. If the destination is simply a user, no further routing need be done, so the group manager requests the user manager to terminate the call. The user manager alerts the local user to the incoming call and replies to the call process that the user accepted the call. The call process then requests the connection manager to apply ringback tone to the originating user and "feeds back" the state of the termination and the identity of the destination to the group manager, which passes the information to the user manager—this results in the originating station receiving a display update if the user has a display.

The call is now in the terminating state, awaiting either an answer by a destination, an abandon by the originator, or a time-out to trigger the next stage of routing. Because the call is in a stable state, the call process is deallocated from the call.

4.4.4 Answer

The call is answered when a destination party goes off-hook. The off-hook is handled like the origination off-hook, except that the user manager determines that the user is now answering a ringing call. The service dispatcher allocates an idle call process and passes the answer command to it. The call process retrieves the call record from the service dispatcher and enters the call answer sequencer. First, any routing timers are canceled, ringback is removed from the connection, and the call process informs the originating user, via the group manager, that the call has been answered. Next, the call process commands the group manager to answer the call for the user. The group manager handles all users being alerted for the incoming call and commands the requesting user to answer the call. The user manager handles the answer command from the group manager by stopping the alerting indications at the user's station and commanding the connection manager to connect the answering port to the call.

Since the calling user was connected in the origination phase, the calling and called parties now have a two-way connection. The call is in a stable state so the call process is deallocated from the call.

4.4.5 Drop

The call is dropped when a user hangs up. The on-hook signal is processed in the same way as the off-hook, with the user manager interpreting the on-hook (or a button push) into a drop command. The call process receives the drop and commands the group manager to drop the user off the call. The gruop manager forwards this to the user manager, which then handles the lamp indications to the user and also commands the connection manager to remove the port from the connection.

Since there is only one party left on the call, the call process

sequences the teardown of the call by sending drop commands to the group manager. The drop commands are forwarded to the user manager, which again changes the user's indications and commands the connection manager to remove the port from the connection. Since all parties have been dropped, the call process informs the connection manager that the call no longer exists. The connection manager idles its connection and resource records for that call. The call process then cleans the call record and tells the service dispatcher to free it and the call record from the call.

V. ACKNOWLEDGMENTS

The System 75 switch services software architecture reflects the efforts of many people within the AT&T Information Systems Laboratories. Many ideas were the fruits of exploratory work that preceded the development phase.

REFERENCES

1. G. R. Sager, J. A. Melber, and K. T. Fong, "System 75: The Oryx/Pecos Operating System," AT&T. Tech. J., this issue.
2. L. A. Baxter et al., "System 75: Communications and Control Architecture," AT&T Tech. J., this issue.
3. Wing H. Huen, "What Is Different About Operating Systems for Telephone Systems," IEEE Reprint CH1515-6/79/0000-0179.
4. L. E. McMahon, "An Experimental Software Organization for a Laboratory Data Switch," Proc. ICC '81 (June 1981).

AUTHORS

Wayne Densmore, B.S. (Electrical and Computer Engineering), 1975, Clarkson College of Technology, M.S. (Electrical and Computer Engineering), 1976, Clarkson College of Technology; AT&T Bell Laboratories, 1976–1983; AT&T Information Systems Laboratories, 1983—. Mr. Densmore has been involved with software development for the *Horizon*® communication system prior to working on System 75. He is currently involved with enhancements to the System 75 architecture. Member, IEEE.

Ray J. Jakubek, B.S. (Electrical Engineering), 1967, Manhattan College; M.S., 1968 and Ph.D. (Electrical Engineering), 1973, New York University; Bell Laboratories, 1968–1982; AT&T Information Systems Laboratories, 1983—. Mr Jakubek has been involved in the design of a wide range of customer premises support systems and products. He contributed to the software development of the CNCC, NCOSS and RMATS II support systems. Subsequently he has supervised the software development of many of the call processing features of System 75. Since December 1982, he has been Head of the Software Applications department. Member, Eta Kappa Nu, Tau Beta Pi.

Michael J. Miracle, B.S. (Electrical and Computer Engineering), 1979, University of Wisconsin; M.S. (Electrical Engineering), 1980, Stanford University; AT&T Bell Laboratories, 1979–1982; AT&T Information Systems

Laboratories, 1983—. Mr Miracle worked on System 75 switch software development in the areas of call processing and data switching and is currently working on System 75 enhancements. Member, IEEE, Eta Kappa Nu, Phi Kappa Phi, and ACM.

John H. Sun, B.S. (Electrical Engineering), 1972, National Chiao-Tung University, Taiwan; M.S. (Computer Science), 1977, University of Connecticut; Taiwan Telecom Research Laboratories, 1974–1975; ACCO-Bristol Company, 1978; Bell-Northern Research, 1979; AT&T Bell Laboratories, 1980–1982; AT&T Information Systems Laboratories, 1983—. Mr. Sun has contributed to System 75 switch services software architecture, design and implementation both as a developer and in his current supervisory capacity.