VTAM Concepts

Mini-Course 1

VTAM Overview

MINI-COURSE 1. ACF/VTAM Overview

Introduction

Advanced Communications Function for the Virtual Telecommunication Access Method **(ACF/VTAM)** is an IBM program product that controls communication between resources in a data communications network. ACF/VTAM resides in a host processor and runs under control of a virtual operating system (OS/VS or VSE).

ACF/VTAM consists of programming modules that provide services that allow VTAM application programs in the host processor to communicate with each other and with terminals controlled by ACF/VTAM. An ACF/VTAM access method and the resources that it controls is called a domain. Two or more domains may be connected to form a multi-domain network. In a multi-domain network, VTAM application programs and terminals in one domain may communicate with VTAM application programs and terminals in another domain.

For convenience, we will henceforth refer to ACF/VTAM as VTAM.

Logical Structure of a VTAM Network

VTAM implements systems network architecture (SNA), therefore a VTAM network is structured according to SNA (see Figure 1-1).

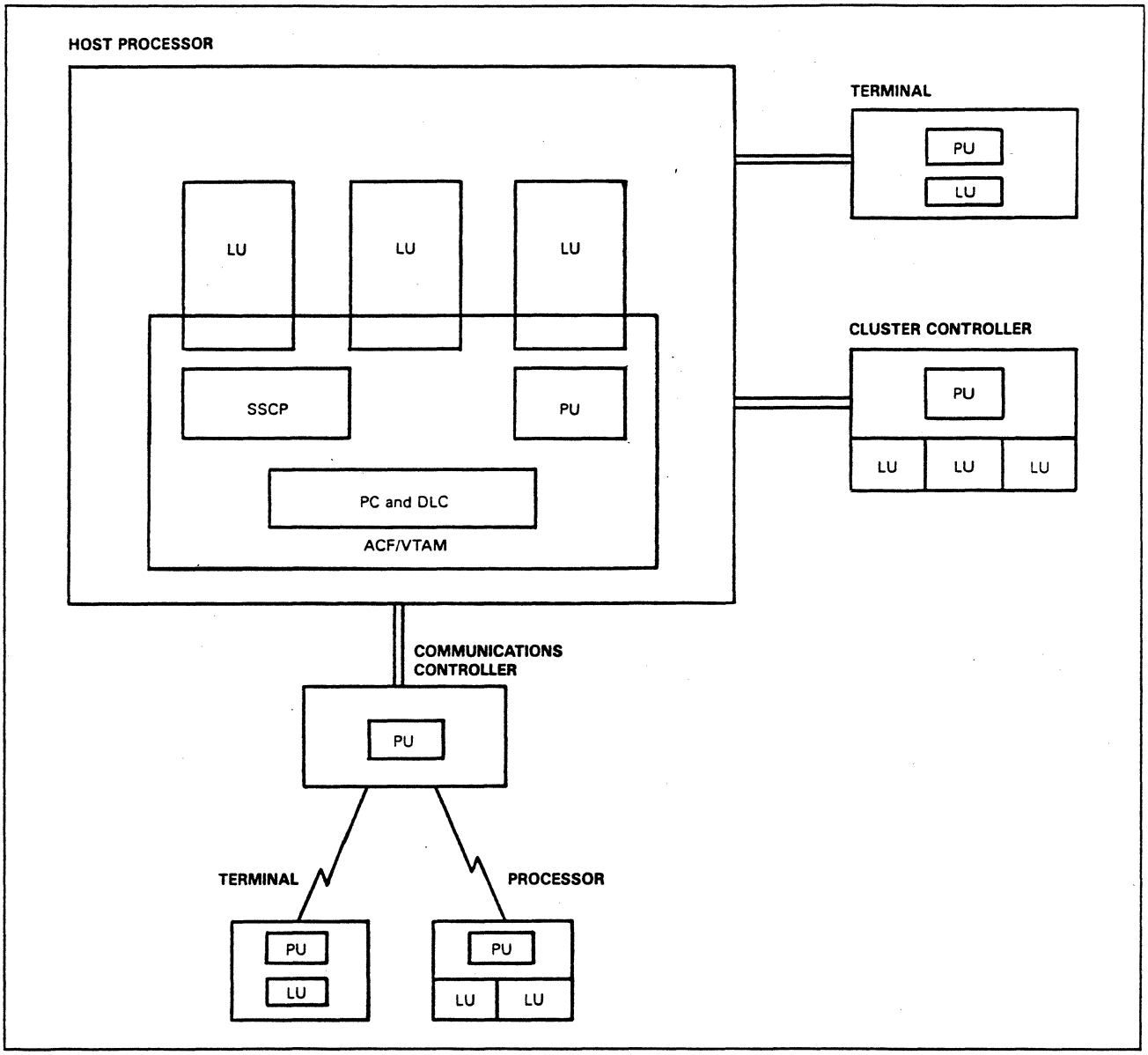

Figure 1-1. Logical Structure of a VTAM Network

Since VTAM implements SNA, it consists of the SNA components system services control point (SSCP), physical unit (PU), path control (PC) and data link control (DLC) components, and supplies part of each host logical unit (LU). The host logical units (LUs) may be supplied by IBM subsystems (for example, CICS, IMS, and TSO) or they may be user-written logical units.

VTAM sees the network as physical units and logical units rather than as terminals, controllers, and processors. Each terminal controller and each processor is seen as a single physical unit and one or more logical units. Each communications controller and its NCP is seen as a physical unit.

Major Components of a VTAM Domain

The type and number of major components that make up a VTAM domain depend on the domain configuration. Figure 1-2 shows a configuration that includes channel-attached terminals (local terminals) and link-attached terminals (remote

terminals). This figure illustrates major components of a VTAM domain and you may want to refer to the figure as we discuss each of the components.

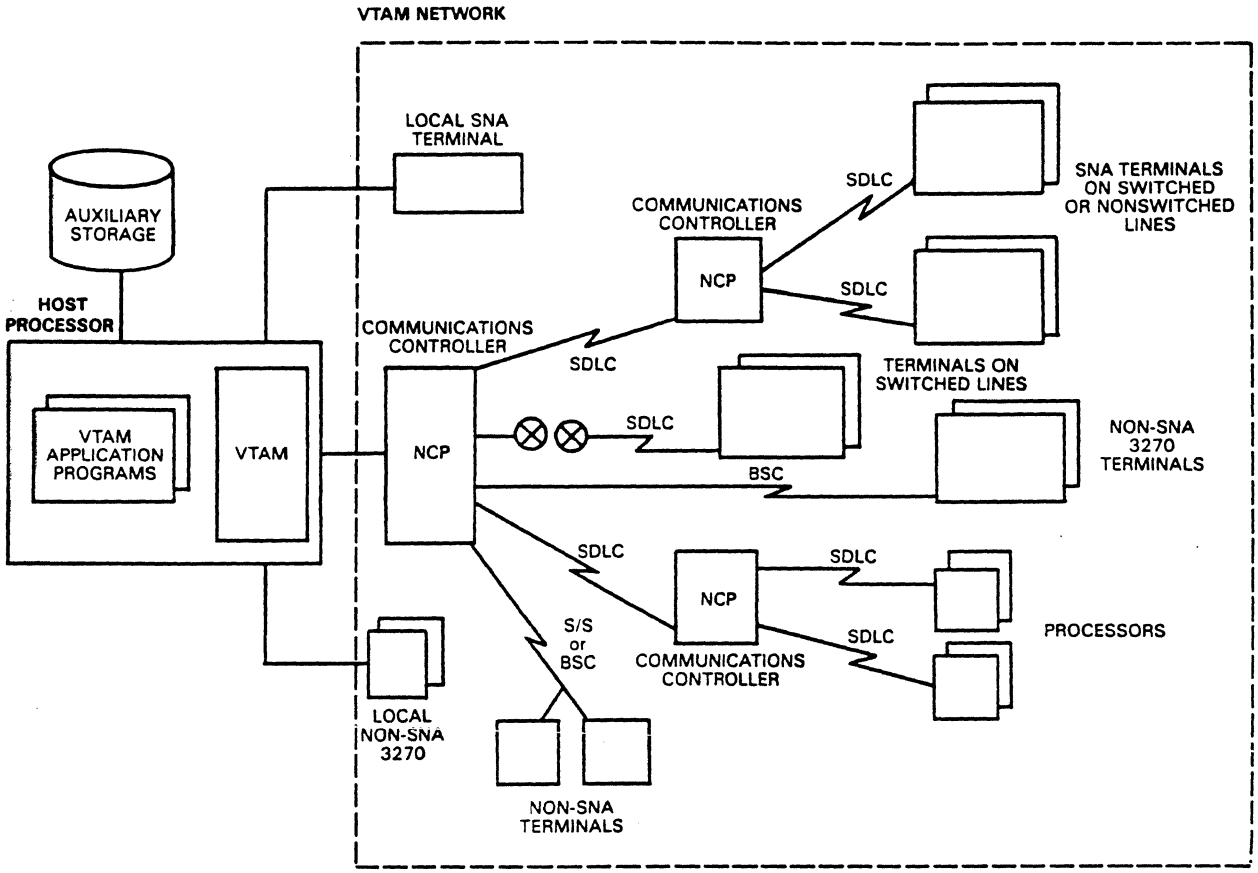

Figure 1-2. Major Components of a VTAM Domain

Major components of a VTAM domain include the following:

- VTAM application programs
- VTAM
- Network control programs (NCPs)
- Terminals and distributed processors
- Data links

VTAM Application Program

A VTAM application program is a program that uses VTAM macro instructions to communicate with resources in the VTAM network. A VTAM application program must identify itself to VTAM before it can communicate with other resources in the network. Once it has identified itself to VTAM, a VTAM application program uses VTAM macros to access VTAM services. VTAM application programs use VTAM macros to establish sessions with other logical

units, to receive information from and to send information to those logical units, and to control the sessions.

A VTAM application program performs the function of a logical unit. By identifying itself to VTAM, a VTAM application program establishes a session with VTAM, called an SSCP-LU session. For convenience, we'll refer to a VTAM application program as a VTAM program henceforth. Using the term VTAM program rather than logical unit is a way of differentiating it from logical units outside the host processor. Logical units outside the host processor reside in peripheral nodes and can be called peripheral logical units.

VTAM

VTAM is the central point of control for its domain. Once VTAM obtains control of the resources (physical units, logical units, and data links) to establish its domain, it provides services which allow logical units to communicate with each other. The services performed by VTAM include:

- Controlling the allocation of network resources (for example, links, communications controllers, and terminals)
- Establishing, controlling, maintaining, and terminating sessions between resources in the network
- Permitting use of resources without specific knowledge of their location
- Transferring data between network resources
- Permitting VTAM programs to share resources in the network
- Permitting a VTAM operator to monitor and alter the network
- Permitting the network configuration to be changed while the network is being used
- Initiating the detection and correction of problems in the operation of the network

Network Control Program (NCP)

A network control program (NCP) resides in a communications controller to which terminals and processors are attached. Each NCP, under VTAM direction, manages the part of the network that is attached to its controller. The NCP, along with its communications controller, provides such functions as the following:

- Transmitting data between network nodes
- Controlling lines
- Controlling buffering
- Deleting and inserting communication control characters
- Detecting permanent and temporary line errors
- Gathering line statistics

- Activating and deactivating lines
- Closing down portions of the network
- Handling recoverable line errors
- Providing error statistics to VTAM
- Testing communication links

Although these activities are performed in communications controllers, they are actually managed by VTAM. For example, when the NCP is to deactivate a link, the command to deactivate comes from VTAM.

Terminals and Distributed Processors

Each terminal or distributed processor is represented to VTAM as a physical unit (PU) and one or more logical units (LUs). To a VTAM program, a terminal or distributed processor is one or more logical units. A VTAM program is not aware of physical units (PUs).

Data Links

Data links include telecommunication links and host processor data channels. Telecommunication links may be managed by the following link protocols.

- Synchronous data link control (SDLC)
- Binary synchronous control (BSC)
- Start-stop (S/S)

SDLC links connect SNA products and certain models of the non-SNA 3270 family of terminals. The BSC line discipline is supported for BSC 3270s. And certain start-stop devices and BSC devices are supported via the network terminal option (NTO) that resides in the communications controller.

Data channels are managed by the data channel protocol which consists of channel command words (CCWs). Communications controllers and terminal controllers can be directly attached to the host processor via data channels. Channel-attached devices are called local devices.

VTAM Network Configurations

A VTAM network may consist of a single domain with local and/or remote networks or of a multi-domain network with local and/or remote networks.

Single Domain Network

Figure 1-3 illustrates a single domain network. A variety of SNA and

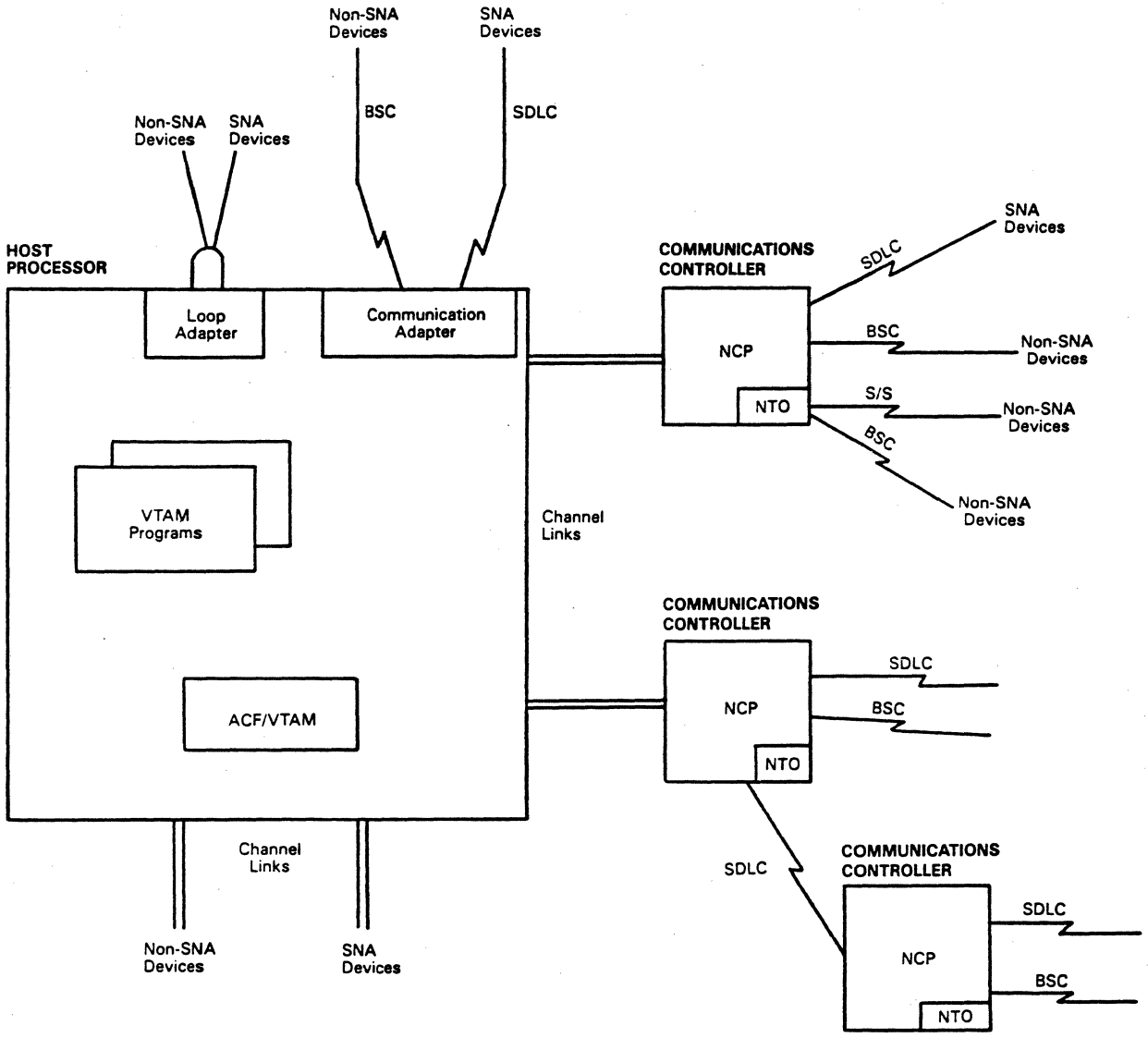

Figure 1-3. VTAM Single Domain Configuration

non-SNA devices are supported in the local and remote network.[1]

The local network is connected to the host processor by channel links (data channels) and by loop adapters. Terminals attached via the loop adapter feature of the host processor appear to VTAM as channel-attached terminals. Note, however, that not all processors support the loop adapter feature.[2]

Terminals remote to the host processor can be linked to VTAM by a communications controller or by a communication adapter (if supported by the host processor[2]). SNA terminals, communications controllers, and BSC 3270 control

1 Specific devices that are supported are listed in the ACF/VTAM General Information manual (Form Number: GC27-0608).

2 Host processors that support the loop adapter and communication adapter features are listed in the ACF/VTAM General Information manual (Form Number: GC27-0608).

units may be connected by the communication adapter. SNA terminals and BSC 3270 control units may be connected by a communications controller. One or more communications controllers may be link-attached to other communications controllers.

Certain S/S and BSC devices are supported in conjunction with the network terminal option (NTO) program product which runs in a communications controller.

Multi-Domain Network

One domain may be connected to another as shown in Figure 1-4.

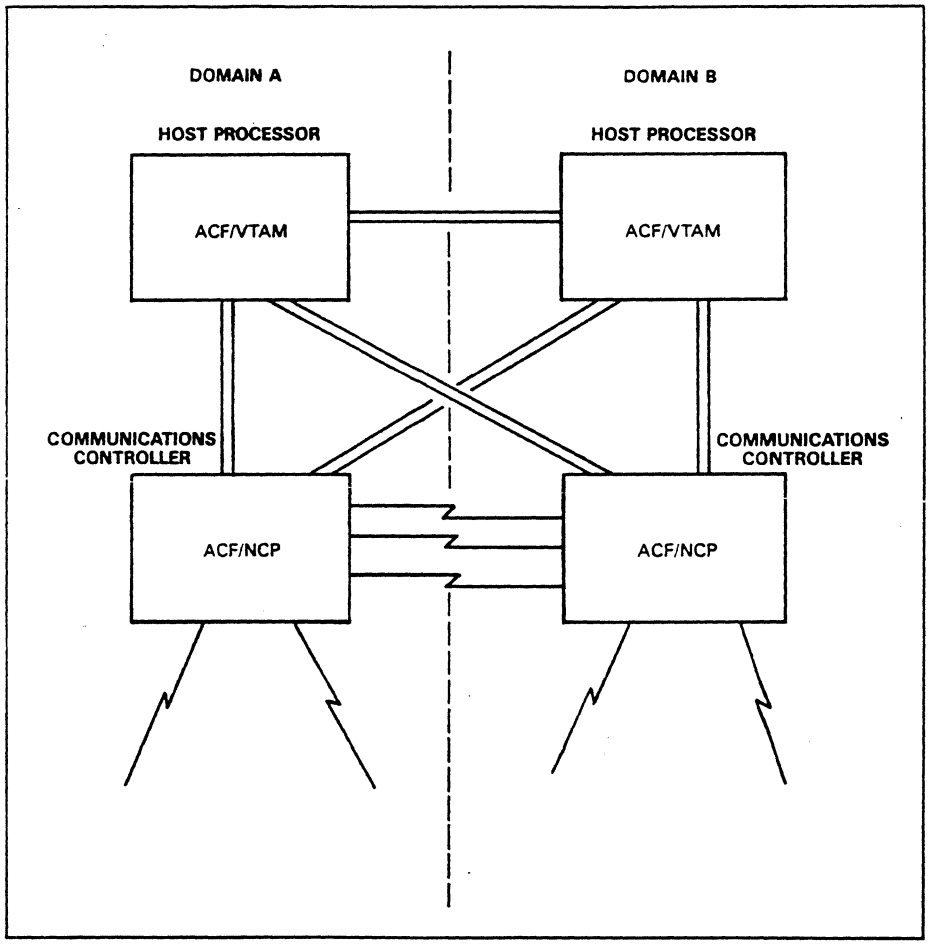

Figure 1-4. Multi-Domain VTAM Network

Each domain consists of a VTAM access method and the network controlled by that VTAM. This figure illustrates the various ways that the two domains may be connected. The host processors may be channel-attached, the communications controllers in each domain may be channel-attached to the host processor in the other domain, or the two communications controllers may be attached to each other by one or more SDLC links. Any or all of these connection methods may be used to connect a pair of domains. Any number of domains may be interconnected to form a network providing the capability for resources in each domain to communicate with resources in any of the other domains.

We'll assume a single-domain network for all of our discussions except in Mini-Course 16, which deals with multiple domain communication.

Major Functions of VTAM

VTAM monitors and controls the network resources that are defined to it. As the central point of control, VTAM performs several major functions. Four of these functions include:

- Starting and stopping the network
- Changing the configuration dynamically
- Allocating and sharing of network resources
- Handling input/output processing

Starting and Stopping the Network

VTAM allows the user to define the network resources to VTAM. The user defines network resources by writing definition statements and including them in a VTAM definition library. Certain definitions must be stored in the VTAM definition library before VTAM can be started. Once these definitions are placed in the VTAM library, the system operator can invoke a procedure to start VTAM execution. VTAM, once it is started, obtains these network resource definitions and uses them to monitor and control the network.

VTAM can also shut down the VTAM network. VTAM does this by sending commands to network resources to deactivate them. Typically, the VTAM network operator prompts VTAM to deactivate network resources.

Dynamically Modifying the Configuration

The VTAM network operator can submit commands to VTAM to modify the network configuration while the network is being used (see Figure 1-5).

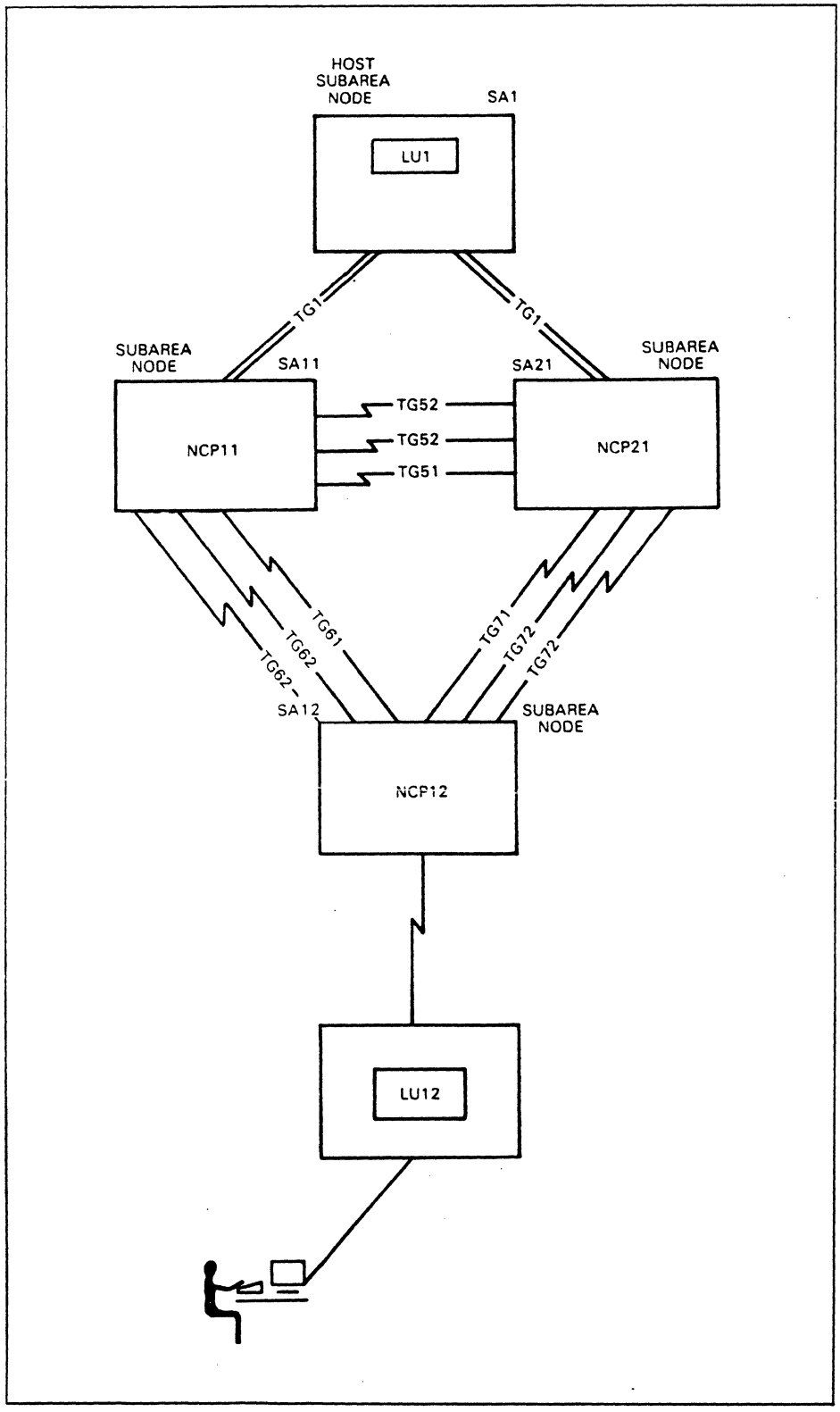

Figure 1-5. Modifying Network Configuration

Assume that NCP12 is located in a different time zone than the other part of the network and that the users of the NCP12 resources terminate activity at 5:00 p.m. each day.

The operator can issue deactivate commands to VTAM to deactivate NCP12's resources and to deactivate NCP12. Once NCP12 is inactive, VTAM knows nothing about NCP12 and its resources. It's as though that part of the network doesn't exist.

Any part of the VTAM network can be activated or deactivated at any time.

In addition to the network operator, an authorized VTAM program can send commands to VTAM to request activation or deactivation of network resources.

Allocating and Sharing Network Resources

VTAM allows the users of a network to share resources of that network. A resource can be shared concurrently or serially. For example, a VTAM program can communicate concurrently with several other logical units, but a logical unit in a terminal controller can communicate with only one VTAM program at a time.

Figure 1-6 illustrates the status of resources in a VTAM configuration at a particular time.

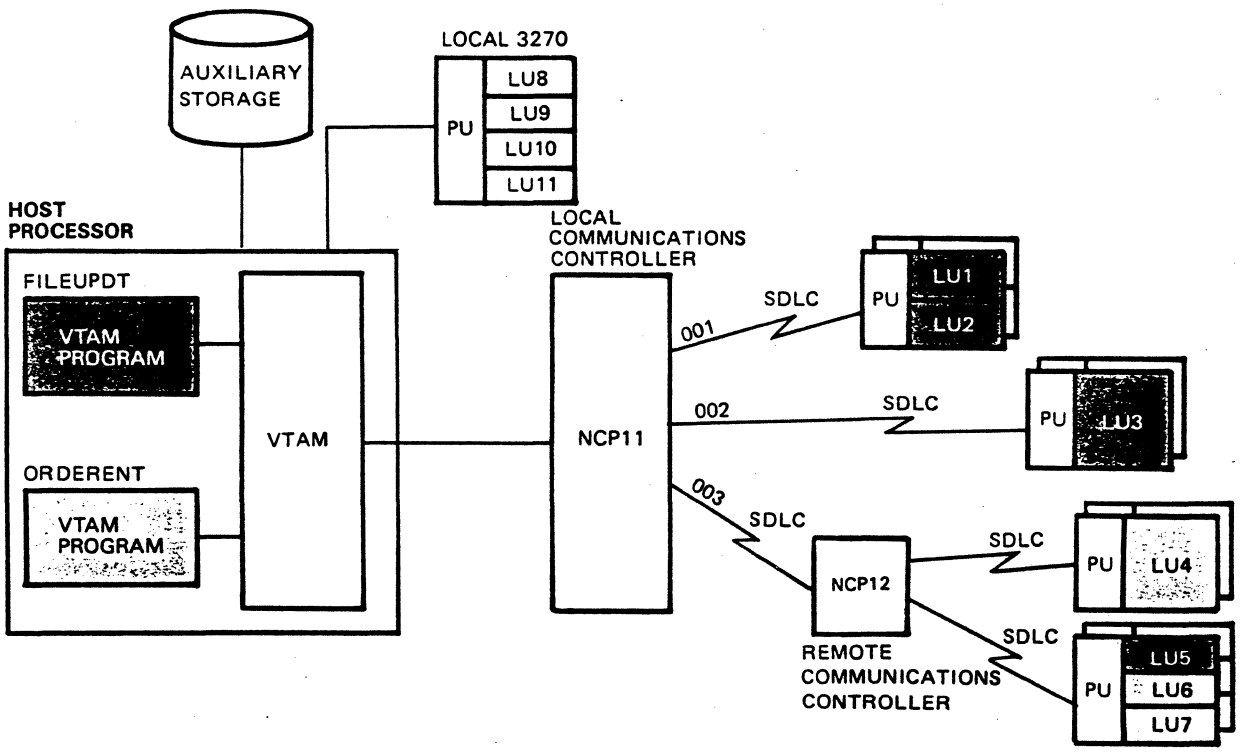

Figure 1-6. VTAM Network Resources

The VTAM program FILEUPDT is in session with four logical units: LU1, LU2, LU3, and LU5. The VTAM program ORDERENT is in session with three logical units: LU4, LU6, and LU7. FILEUPDT and ORDERENT are shared concurrently and logical units LU1 through LU7 are shared serially. If ORDERENT requested a session with LU1, VTAM would not allow it. LU1 is a serially shared resource and it already has an LU-LU session. VTAM will not allocate the resource (LU1) for the ORDERENT-LU1 session.

Once processing has completed on an LU-LU session, usually the session is terminated. For example, processing completes on the FILEUPT-LU1 session and on the FILEUPDT-LU3 session and the sessions are terminated. Now it is possible for LU1 and LU3 to establish a session with FILEUPDT, ORDERENT, or another VTAM program to process more data. We'll assume that ORDERENT requests a session with LU1 and VTAM allocates the resources for the session. Network status is shown in Figure 1-7.

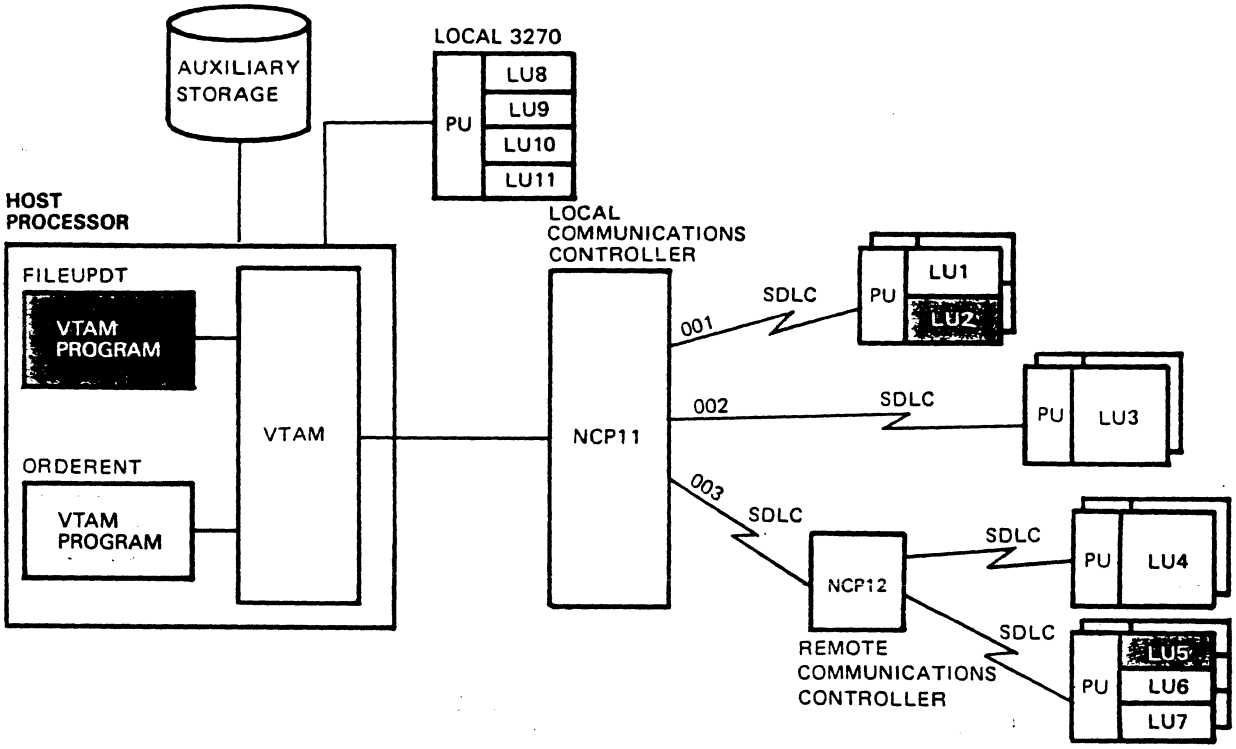

Figure 1-7. Allocating Network Resources for an LU-LU Session

Resources that make up the paths between logical units are also shared. Looking at Figure 1-7, the shared resources include VTAM, the channel link, the NCP, and the SDLC links. VTAM uses path elements on behalf of logical units only as long as needed to complete a specific data-transfer request.

Input/Output Processing

Once a session is established between a VTAM program and another logical unit, VTAM manages the transmission of data between the two. To illustrate this process, we'll use the session between FILEUPDT and LU3 as shown in Figure 1-8.

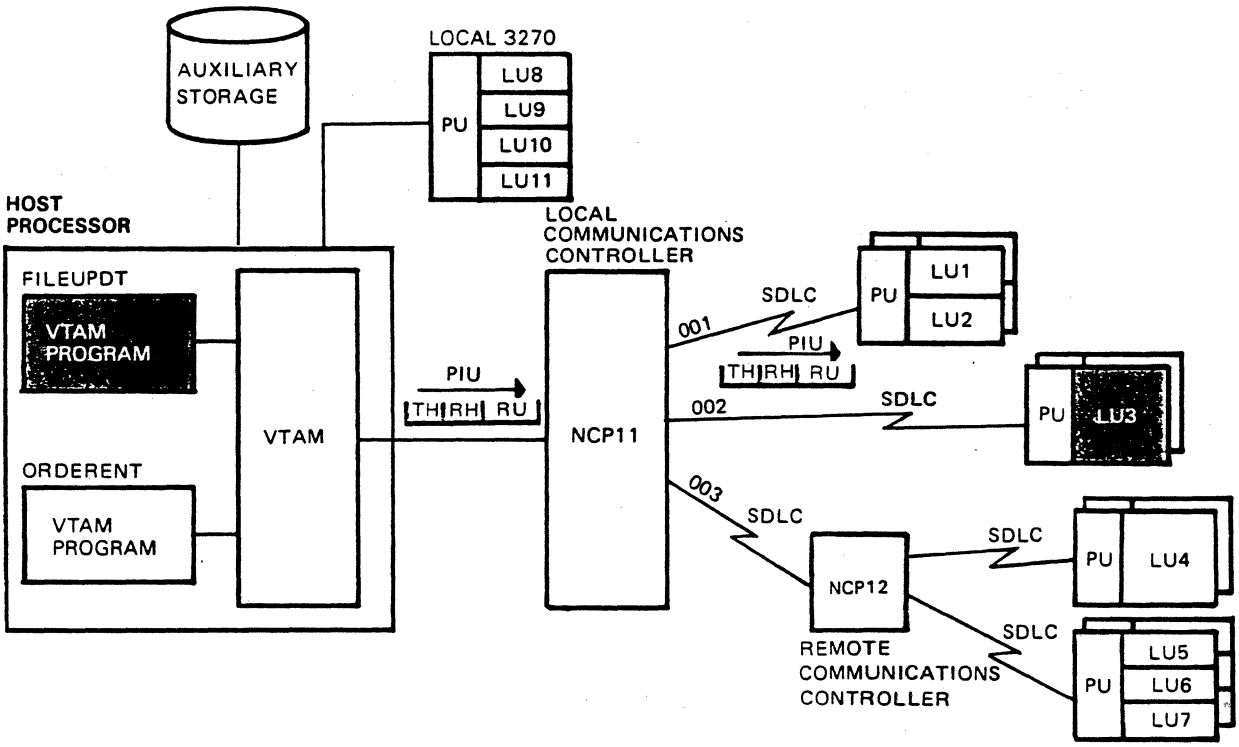

Figure 1-8. VTAM Program Sending to Logical Unit

FILEUPDT has data to send to LU3, so it issues a VTAM macro instruction that requests VTAM to transmit the data. FILEUPDT supplies information in the request that tells VTAM such things as:

- The location of FILEUPDT's data
- The address of LU3
- How to handle the request
- Whether a response is to be returned

VTAM uses this information to generate a path information unit (PIU). The PIU is shown in Figure 1-8 going from VTAM to NCP11. The request header (RH) contains information describing the request unit (RU) and describes how the request is to be handled. The transmission header (TH) contains the origin and destination addresses, the sequence number of the request, and other information. The request unit (RU) contains the application data that is being sent to LU3.

Upon receipt of the PIU, the NCP determines from the transmission header that the destination address is LU3. Therefore the PIU is sent over data link 002 to LU3. LU3 will return a response if the appropriate bits in the request header indicate that one is required.

Now we'll assume that LU3 has data to send to FILEUPDT and that the flow is as shown in Figure 1-9.

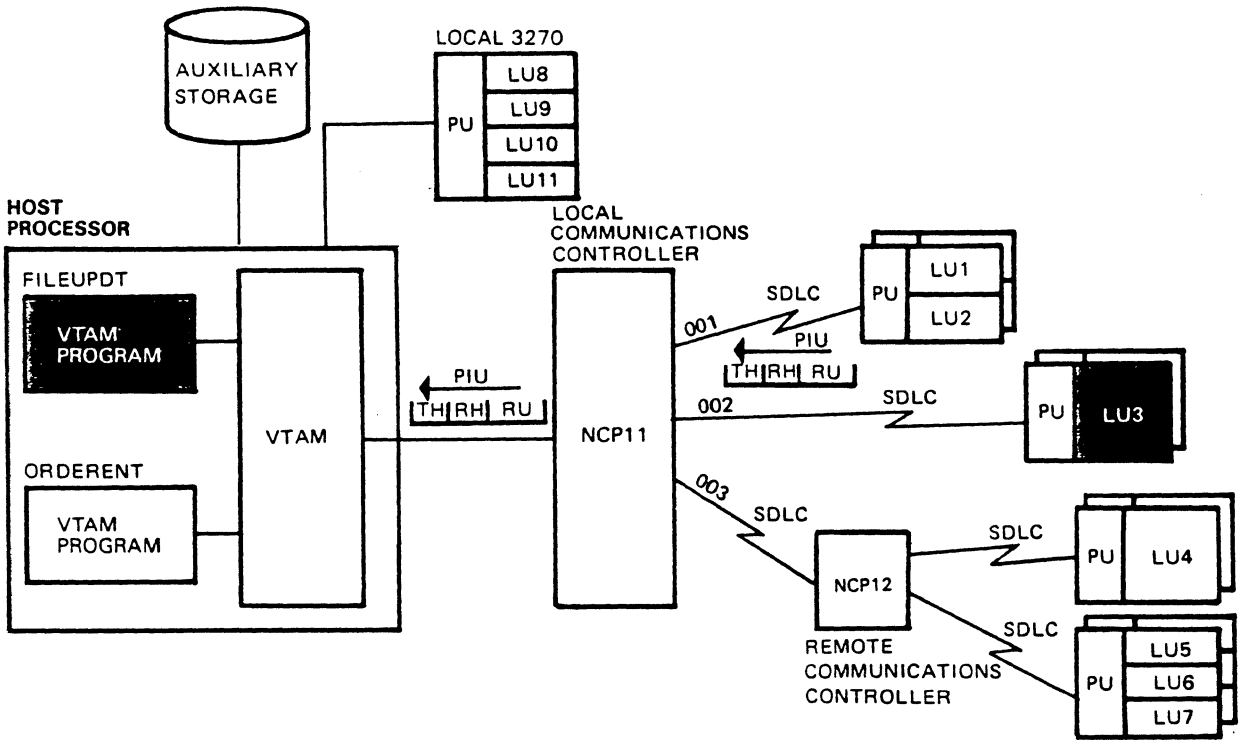

Figure 1-9. Logical Unit Sending to VTAM Program

A path information unit (PIU) is generated by LU3 and is sent across the link to its NCP. The NCP sends the PIU on to VTAM, and VTAM presents the data to FILEUPDT if FILEUPDT has issued a VTAM macro to receive transmissions from LU3. Otherwise, VTAM stores the data until a VTAM macro is issued to receive the data.

From this discussion you should note that VTAM does not initiate the input/output activity; but, upon request, it manages the transmission.

Routing Within a VTAM Domain

When a session is established between two network addressable units (SSCP, PUs, LUs), the session is assigned to and uses a network communications path called a **route**. Session traffic flows on that route for the duration of the session.

There are physical routes, called **explicit routes (ERs)**, that must be defined and the definitions are stored in the VTAM route table and in NCP route tables. There are logical routes, called **virtual routes (VRs)**, that are used in the process that assigns sessions to appropriate explicit routes.

Explicit routes

Explicit routes are defined between an origin subarea and a destination subarea. Figure 1-10 shows a VTAM network configuration that includes four subareas (subareas 1, 11, 12, and 21). One or more explicit routes must be defined between subareas 1 and 11, between subareas 1 and 12, and between subareas 1 and 21.

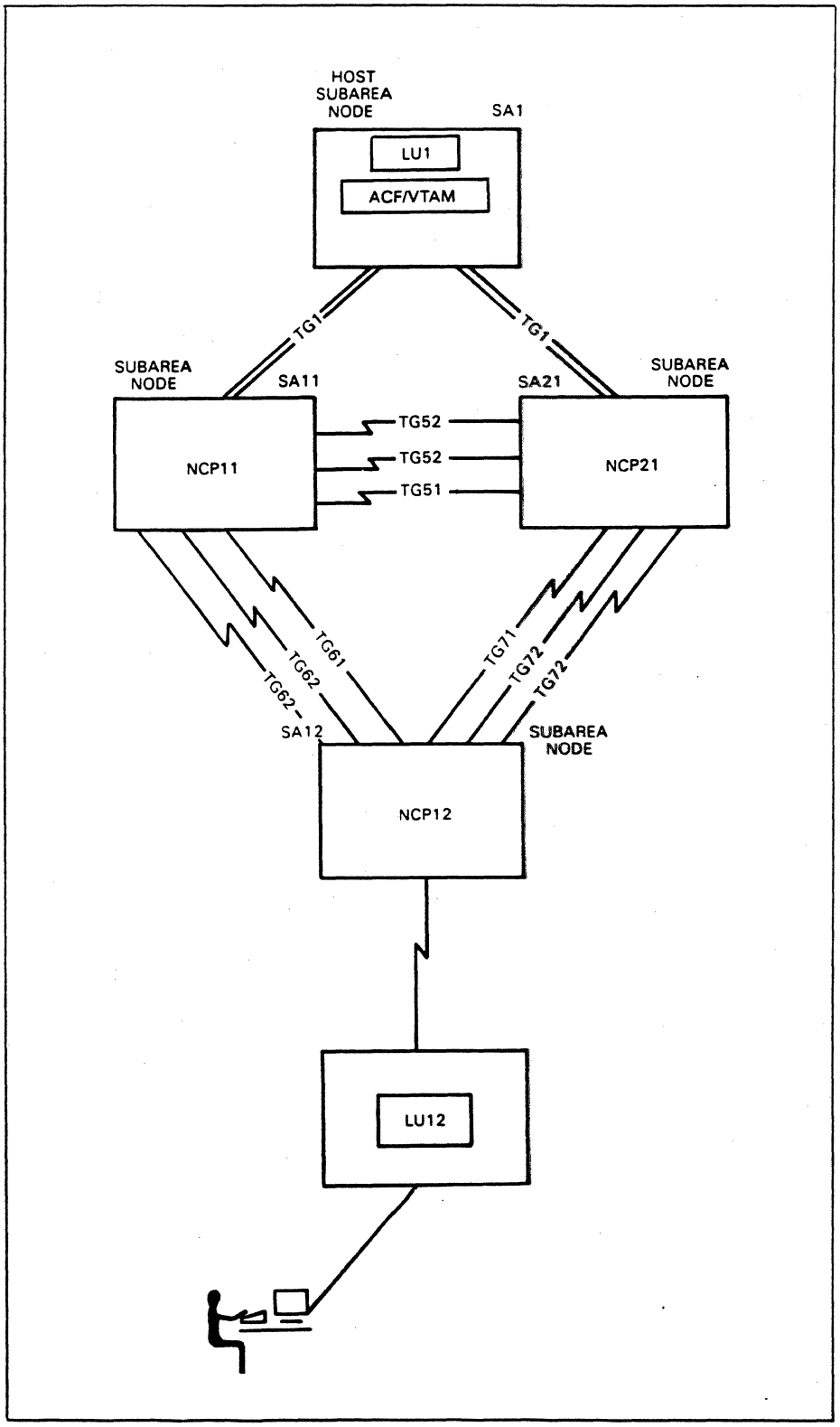

Figure 1-10. Routing

We'll discuss the routes between subareas 1 and 12. There are several possible paths between the host subarea node in subarea 1 and the subarea node in subarea 12 (NCP12). Here are three examples:

```
SA1.TG1.SA11.TG62.SA12
SA1.TG1.SA21.TG72.SA12
SA1.TG1.SA11.TG52.SA21.TG71.SA12
```

The first notation identifies the route from subarea 1 over TG1 to subarea 11, and over TG62 to subarea 12. The first route includes the subarea nodes in subareas 1, 11, and 21 and the transmission groups TG1 and TG62. The third route includes the subarea nodes in subareas 1, 11, 21, and 12 and the transmission groups TG1, TG52, and TG71.

We'll assume that two explicit routes are defined between subareas 1 and 12 as shown in Figure 1-11.

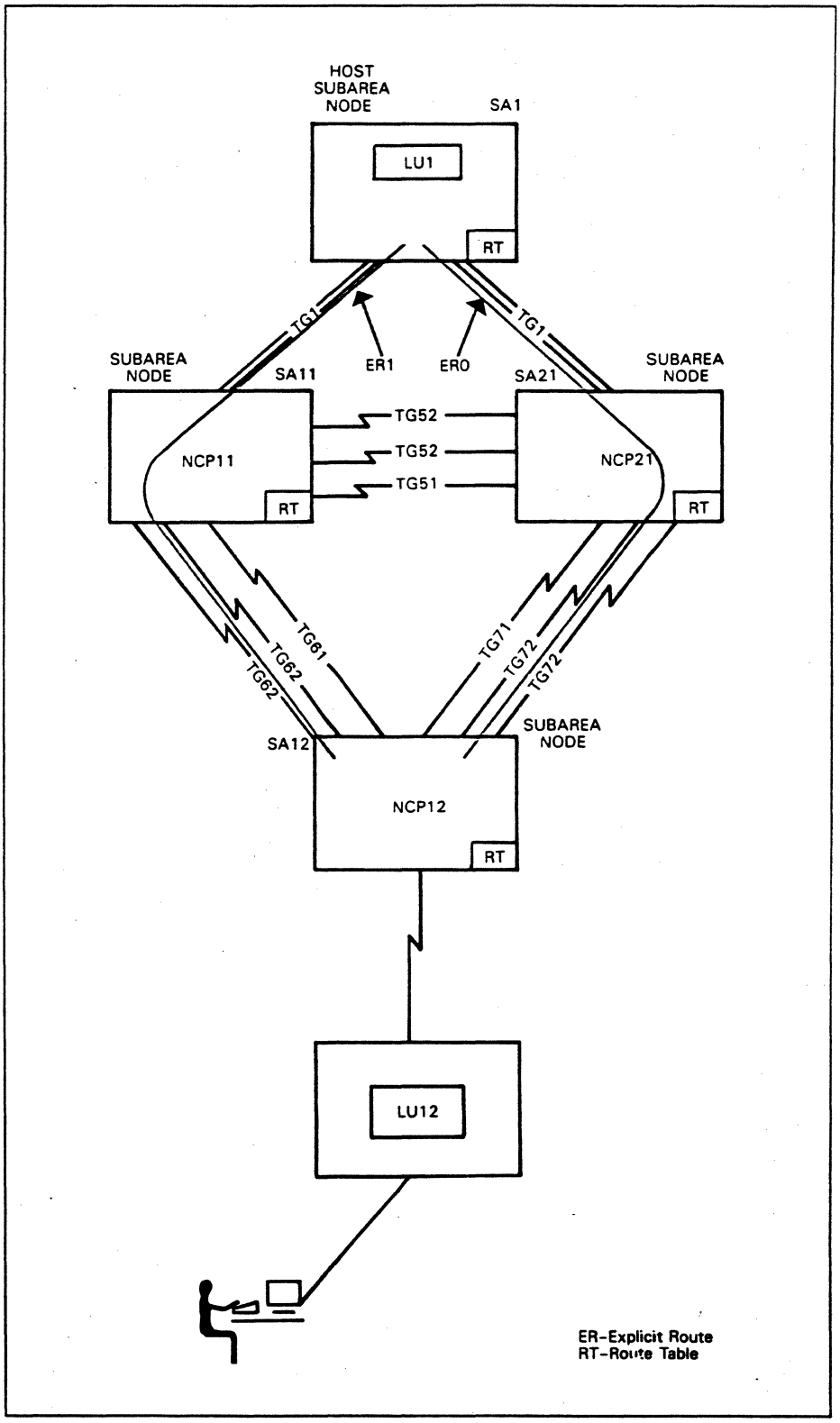

Figure 1-11. Explicit Routes

One explicit route is numbered zero and the other is numbered one. A maximum of eight explicit routes can be defined between the two subareas, ER0 through ER7. ER0 and ER1 are the same length and are the shortest possible routes. Both

routes also include a two-link transmission group. We'll designate ER0 as the primary route for all sessions and ER1 as an alternate route. Sessions are established on ER0 if the route is operational, otherwise sessions are established on ER1.

We'll also assume that two explicit routes are defined between the other two pairs of subareas as shown in Figure 1-12 and in Figure 1-13. Explicit routes between subareas 1 and 11 are shown in Figure 1-12 and explicit routes between subareas 1 and 21 are shown in Figure 1-13. ER0 is to be the primary route and ER1 is to be an alternate route. Other alternate routes can be defined if required.

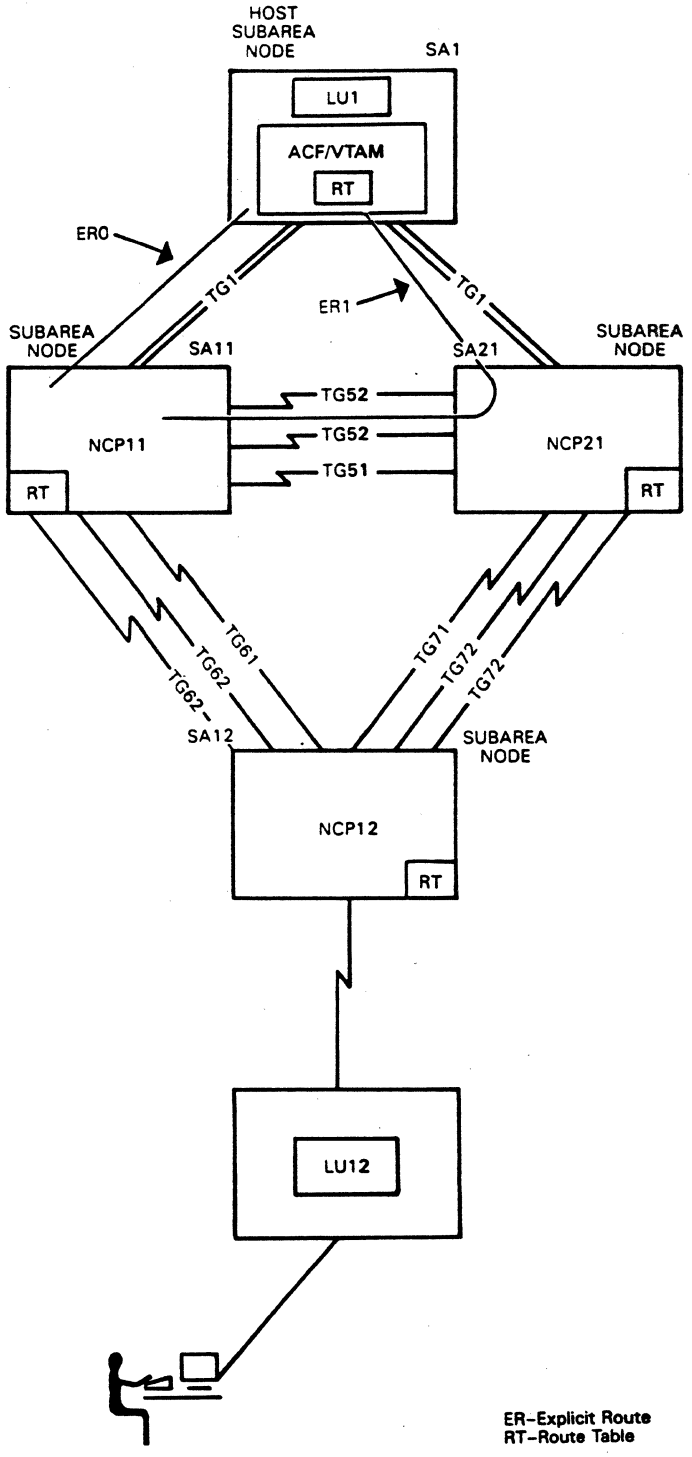

Figure 1-12. More Explicit Routes

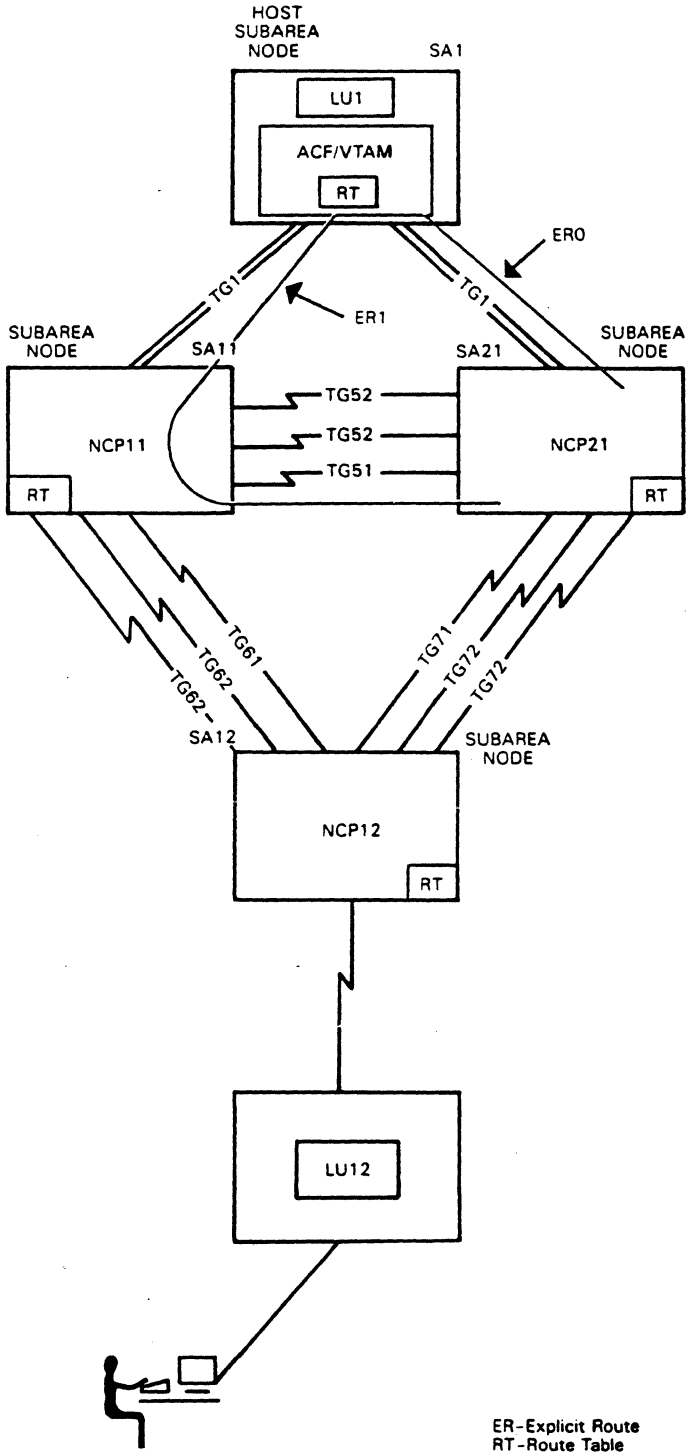

Figure 1-13. More Explicit Routes

Virtual Routes

Sessions are not assigned directly to an explicit route. Sessions are assigned to a virtual route and the virtual route is mapped to the appropriate explicit route. The mapping of the virtual route to the explicit route causes the session to use that explicit route.

The mapping of virtual routes to explicit routes is done in the host subarea node. Definitions are included in the VTAM route table to accomplish the VR to ER mapping.

A virtual route consists of a virtual route number and a transmission priority number. For example:

```
VR=(0,1)
```

The virtual route number is zero and the transmission priority number is one. A session assigned to this virtual route is assigned virtual route number zero. The traffic that flows on this session is assigned a transmission priority of one. There are three transmission priorities, 0, 1, and 2, with 2 being the highest. Session traffic with a higher transmission priority can pass session traffic with a lower transmission priority at certain points in the network.

We'll assume that the VTAM route table includes definitions that map virtual route number zero to explicit route zero and maps virtual route number one to explicit route one. Now we have two virtual routes:

```
VR=((0,1),(1,1))
```

The first virtual route is the primary route and the second virtual route is an alternate route. Other virtual routes could be added to this list. The first virtual route in the list is the most desirable for a session, the next virtual route is the second most desirable, and so on.

The various types of sessions in a network require specific routing services. For example, interactive sessions need a route that provides good response. Batch sessions need a route that can handle large volumes of traffic, and sessions that handle confidential data need a route that provides security features.

Class of Service

A list of virtual routes is established for each set of sessions that require unique routing services. Each virtual route list is named so the list can be selected when assigning sessions to routes. Each named list of virtual routes is called a **class of service**. All classes of service are stored in a class of service table that resides in the host subarea node and is accessible to VTAM. Figure 1-14 illustrates a class of service table that contains four classes of service.

- ISTVTCOS
- BATCH
- INTERACT
- SECURE

```
ISTVTCOS  ((0,2),(3,2),(4,2))
BATCH     ((1,0),(4,0))
INTERACT  ((0,1),(3,1),(4,1))
SECURE    (2,0)
```

Figure 1-14. Class Of Service Table

ISTVTCOS is the class of service for all VTAM sessions (SSCP-PU, SSCP-LU, and SSCP-SSCP). INTERACT could be used for CICS and IMS type sessions, RJE type sessions could use the BATCH class of service, and sessions that handle confidential data would be assigned the SECURE class of service.

Assigning a Session to a Route

Referring to Figure 1-12, assume that LU12 initiates a session with LU1 and the session initiation request specifies the INTERACT class of service. The first virtual route in INTERACT (see Figure 1-13) specifies VR0 and TP1. Assuming that VR0 is operational, the LU1-LU12 session is assigned to VR0 with a transmission priority of 1. Definitions in VTAM's route table will map virtual route number 0 to the appropriate explicit route. We've assumed that explicit route zero is the best route so VR0 is mapped to ER0. Therefore LU1-LU12 session traffic will use explicit route zero and the traffic has a transmission priority of one.

Each time that a session (SSCP-PU, SSCP-LU, LU-LU) is initiated, a class of service is specified, a virtual route is selected from the class of service, the virtual route number is mapped to an explicit route, and the session uses that explicit route.

Please turn to Mini-Course 1 in your Personal Reference Guide and do Exercise 1.1.

VTAM Concepts

Mini-Course 2

VTAM Program Concepts

MINI-COURSE 2. VTAM Program Concepts

Introduction

A VTAM program is a logical unit that resides in a host subarea node. It uses VTAM macro instructions to invoke the services of VTAM for communication with other logical units. VTAM programs are written in assembler language.

Figure 2-1 shows two VTAM programs and three application processing programs (end users).

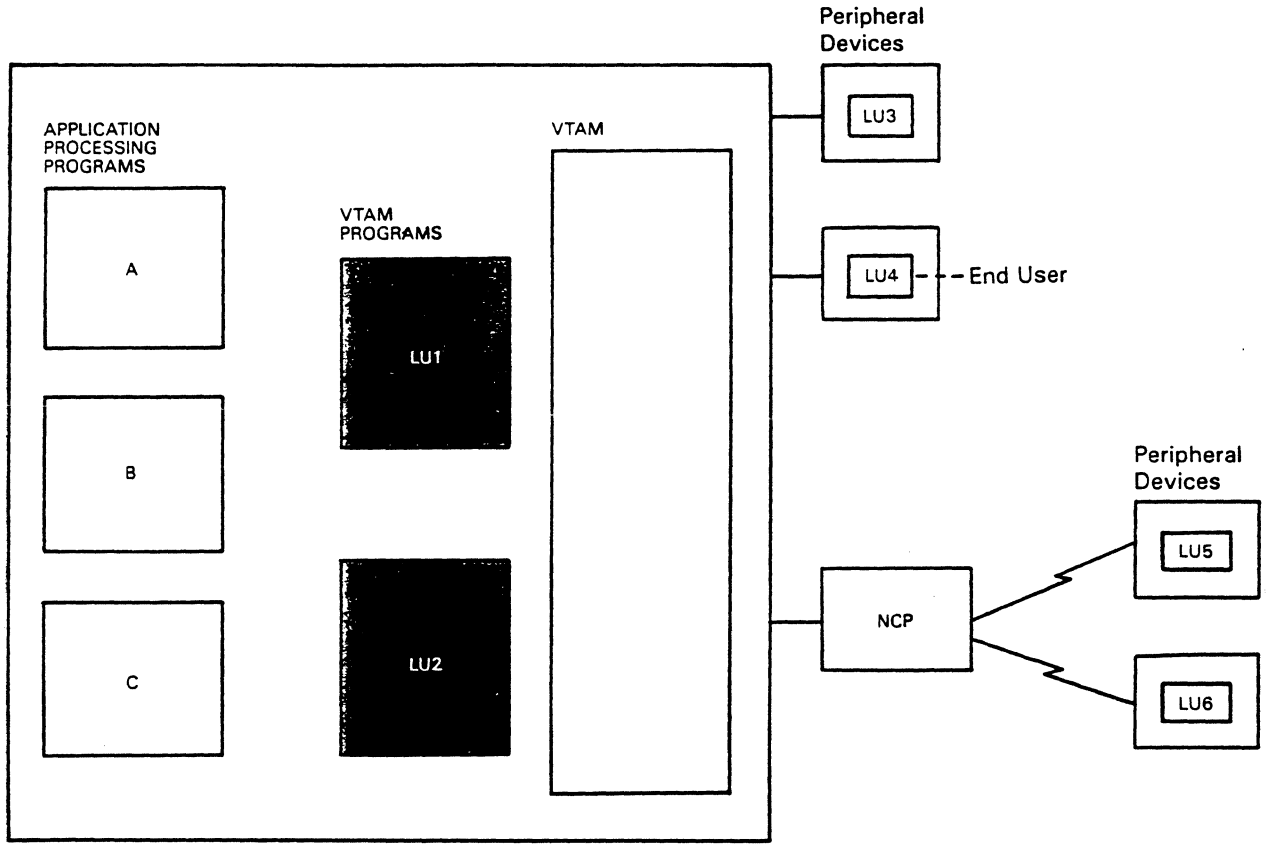

Figure 2-1. VTAM Programs

The VTAM programs can establish sessions with peripheral logical units and with other VTAM programs. For example, the two VTAM programs can establish a session with each other. The VTAM programs can interface to the three application processing programs shown in the figure. For example, application processing program A has information to send to the end user associated with logical unit LU4. Application processing program A gives the information to logical unit LU1; LU1 establishes a session with LU4 and sends the information to LU4, and LU4 gives the information to its end user.

LU4's end user can send information to application processing program A on the LU1-LU4 session. The end user gives the information to LU4, LU4 transmits it to LU1, and LU1 gives the information to application processing program A.

Functions of VTAM Programs

Each VTAM program performs several functions and these functions are related to end users, other logical units, and to VTAM. Figure 2-2 shows one VTAM program and lists the functions of a VTAM program.

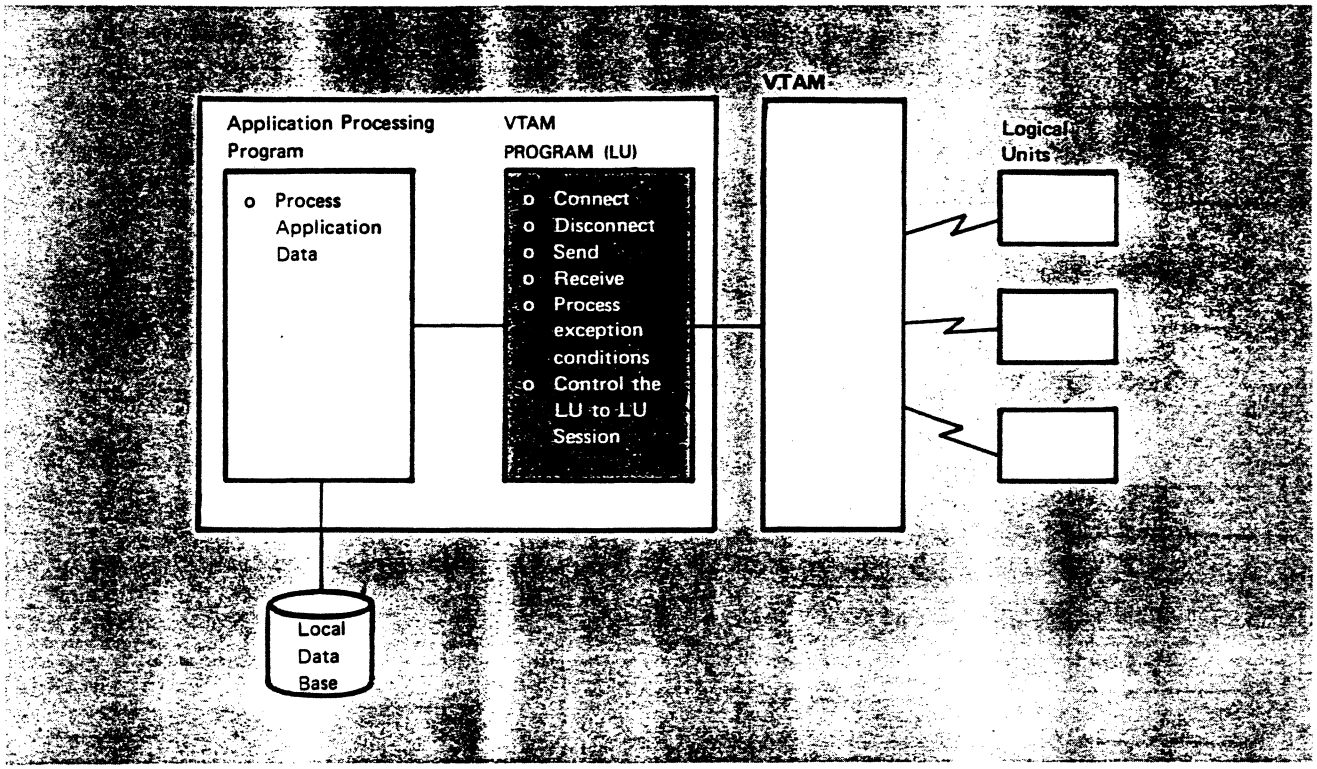

Figure 2-2. VTAM Program

Each VTAM program performs the following functions:

- Identifies itself to VTAM
- Disconnects itself from VTAM
- Establishes and terminates sessions with other logical units
- Sends application related data to and receives application related data from other logical units (includes accepting data from application processing programs for transmission over sessions and receiving data on sessions and giving the data to the appropriate application processing program)
- Processes exception conditions that occur in a session
- Participates with its session partner to control the session (includes quiescing data transmission and shutting the session down)

The purpose of the VTAM program is to transmit data between application processing programs and logical units.

Application processing programs process application-related data received from VTAM programs and give data to the VTAM programs to be transmitted to an end user.

VTAM Processing Instructions

The VTAM program contains VTAM macro instructions and some processing instructions. Processing instructions within a VTAM program enable the program to make decisions concerning its communication with logical units. For example, assume that the VTAM program sends data to a logical unit and a physical error occurs. The VTAM program must test for this condition and decide what to do. The processing instructions can instruct the program to ignore the error, to resend the data, or to terminate the session. The processing instructions in the VTAM program do not process the application data.

VTAM Macro Instructions

VTAM macro instructions provide an interface between the VTAM program and VTAM. Each VTAM program must use the VTAM macros to invoke VTAM services and these services are required for VTAM programs to communicate with other logical units.

VTAM macro instructions are grouped into the following categories:

- ACB-based macro instructions
- Declarative macro instructions
- RPL-based macro instructions
- Manipulative macro instructions

ACB-Based Macro Instructions

ACB-based macro instructions are **not** VTAM macros. However, they're required to identify a VTAM program to VTAM and to disconnect a VTAM program from VTAM. ACB-based macro instructions include:

- OPEN
- CLOSE

Once a VTAM program is started, it must notify VTAM that it is to be recognized as an active element in the network. The VTAM program does this by issuing an **OPEN** macro instruction and upon completion of the OPEN operation, an SSCP-LU session is established. The VTAM program is recognized as an active element in the network and can issue VTAM macros to access VTAM services.

A VTAM program removes itself as an active element in the network by issuing a **CLOSE** macro instruction. The CLOSE macro instruction tells VTAM to terminate the SSCP-LU session and to mark the VTAM program as no longer present in the network. Once the CLOSE operation completes, the VTAM program can no longer use VTAM macro instructions to access VTAM services.

Declarative macro instructions are used to build and to initialize VTAM program interface control blocks. The control blocks are the VTAM program's interface to VTAM. A VTAM program communicates information to VTAM and VTAM communicates information to the VTAM program via the control blocks. Declarative macro instructions include:

- ACB
- RPL
- NIB
- EXLST

Access Method Control Block (ACB) Macro Instruction: The *ACB* macro builds and initializes an Access Method Control Block (ACB). The ACB control block identifies the VTAM program to VTAM and to the SNA network as a logical unit.

Request Parameter List (RPL) Macro Instruction: The *request parameter list (RPL)* macro builds and initializes an RPL control block. Every request that a VTAM program makes for session establishment or communication must refer to an RPL. The RPL control block describes the VTAM program request to VTAM.

Node Initialization Block (NIB) Macro Instruction: The *node initialization block (NIB)* macro builds and initializes a NIB control block. The NIB control block is used during the initiation and establishment of LU-LU sessions to identify and define those sessions.

EXLST Macro Instruction: The *EXLST* (exit list) macro builds and initializes an EXLST control block with addresses of the VTAM program exit routines. The EXLST macro instruction may specify addresses for the following VTAM exit routines:

- LERAD
- SYNAD
- DFASY
- RESP
- SCIP
- TPEND
- RELREQ
- LOGON
- LOSTERM
- NSEXIT

An exit routine is scheduled by VTAM when a specific event occurs, for example, when an error occurs or when a logon is received for the VTAM program.

RPL-Based Macro Instructions

RPL-based macro instructions include those VTAM macro instructions that refer to a request parameter list (RPL) control block. Each RPL-based macro instruction specifies that VTAM perform a particular operation and the referenced RPL provides VTAM with the information it needs to perform that operation.

There are four categories of RPL-based macro instructions:

1. Session establishment macro instructions
2. Communication macro instructions
3. Program operator macro instructions
4. Macro instructions that assist in session establishment or communication

Session Establishment Macro Instructions: Session establishment macros are used by a VTAM program to initiate sessions with other logical units and to terminate existing LU-LU sessions.

When a VTAM program acts as a primary logical unit, it uses the following macro instructions to establish and terminate LU-LU sessions:

- OPNDST
- SIMLOGON
- CLSDST

OPNDST requests VTAM to establish a session with the designated logical unit in which the VTAM program will act as the primary logical unit.

SIMLOGON requests VTAM to initiate a session on behalf of the designated logical unit in which the VTAM program is to act as the primary logical unit.

CLSDST requests VTAM to terminate the session between the VTAM program and the designated logical unit when the VTAM program is acting as the primary logical unit.

When a VTAM program acts as a secondary logical unit, the following macro instructions are used to establish and terminate LU-LU sessions:

- REQSESS
- OPNSEC
- TERMSESS
- SESSIONC

REQSESS requests VTAM to initiate a session with a designated logical unit in which this VTAM program will act as a secondary logical unit. The REQSESS

macro instruction is used by a VTAM program to initiate a session with another VTAM program.

OPNSEC is issued by a VTAM program that is to act as a secondary logical unit. When the VTAM program receives a BIND request from another program, it issues the OPNSEC macro instruction to accept the BIND parameters.

TERMSESS is issued by a VTAM program that is acting as a secondary logical unit. TERMSESS requests VTAM to terminate the session between this VTAM program and the designated logical unit.

SESSIONC is used to send negative responses to a BIND request and to send certain command requests (for example, STSN, RQR, and SDT).

Communication Macro Instructions: Communication macro instructions are used by a VTAM program to send information to and to receive information from another logical unit. Communication macro instructions include:

- SEND
- RECEIVE
- RESETSR
- SESSIONC

SEND requests VTAM to transmit a request or a response on a session to a specific logical unit. Data in a request is transferred from a data area in the VTAM program.

RECEIVE requests VTAM to transfer request and response information to the VTAM program. These requests and responses are transmitted by the VTAM program's session partners.

The **RESETSR** (reset send-receive) macro can be used to change the mode of a logical unit that is in session with the VTAM program. The modes are used to restrict input from logical units.

The **SESSIONC** (session control) macro is used in the recovery from sequence errors that occur in VTAM program to logical unit sessions. It can stop data flow and clear the data path, perform the recovery, and start data flow following recovery.

Program Operator Macro Instructions

Program operator macro instructions can be used by an authorized VTAM program to control or display the status of the network, to receive messages from VTAM, and to reply to VTAM messages that require a reply.

Program operator macro instructions include:

- SENDCMD
- RCVCMD

A VTAM program that is authorized to use these two macros can essentially perform the same function as the VTAM network operator. *SENDCMD* is used to send operator commands to VTAM. *RCVCMD* is used to receive VTAM messages.

Macro Instructions That Assist in Session Establishment or Communications: These macro instructions include *CHECK, EXECRPL, INQUIRE, INTRPRET*, and *SETLOGON*. Later we'll discuss the use of some of these.

Manipulative Macro Instructions

A VTAM program places values in control blocks (ACB, RPL, NIB, and EXLST) that are used by VTAM when the VTAM program requests VTAM to perform actions on its behalf. These control block values describe how VTAM is to handle the operation. Manipulative macro instructions can be used to place values in the control blocks.

Manipulative macro instructions include:

- GENCB
- SHOWCB
- TESTCB
- MODCB

Manipulative macro instructions build, modify, access or test control blocks during program execution. These are **not** VTAM macros.

GENCB can be used to build ACBs, EXLSTs, NIBs, and RPLs during program execution and can initialize designated fields within the control blocks with specified values.

SHOWCB obtains the value or values from one or more fields of a control block (ACB, EXLST, NIB, or RPL) and places them in an area in the VTAM program where they can be examined.

TESTCB compares the contents of a control block field against a user-specified value. The comparison sets the condition code in the PSW.

MODCB changes the contents of one or more specified fields of a control block (RPL, ACB, EXLST, or NIB) by inserting specified values in the fields.

The programmer has the option of either using these manipulative macro instructions or performing the same functions with assembler language instructions.

VTAM and DB/DC Systems

Discussions to this point have been in relation to a user-written VTAM program. This means that the user codes the VTAM program using assembler language, and codes the application processing program using assembler language or a higher level language. Now let's turn our attention to DB/DC systems where the user doesn't code the VTAM program. The VTAM program is included as part of the DB/DC system.

Many VTAM programs may communicate concurrently with logical units through VTAM. One of the VTAM programs may be CICS/VS. Like a user-written VTAM program, CICS/VS must use VTAM macro instructions to communicate with other logical units in a VTAM network.

Figure 2-3 shows the relationship between CICS/VS, a VTAM program, and VTAM.

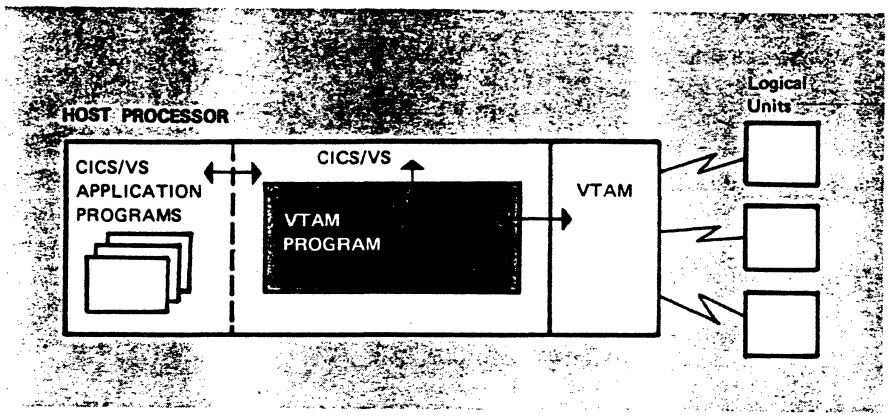

Figure 2-3. VTAM Relationship to CICS/VS

Figure 2-3 shows only those logical units that belong to VTAM. CICS/VS may communicate with other resources through another access method (BTAM, for example).

The CICS/VS application programs shown in Figure 2-3 are user-written programs. They use CICS/VS macro instructions to request CICS/VS services. However, CICS/VS application programs may communicate with VTAM logical units only through a VTAM program.

The VTAM program, a module within CICS, receives data from a logical unit. CICS/VS gives the data to the appropriate application processing program. Also, CICS transfers data from an application processing program to the VTAM program so the data can be transmitted to the appropriate logical unit.

The CICS generation process generates a VTAM program as part of CICS.

Figure 2-4 shows the relationship between IMS, a VTAM program, and VTAM.

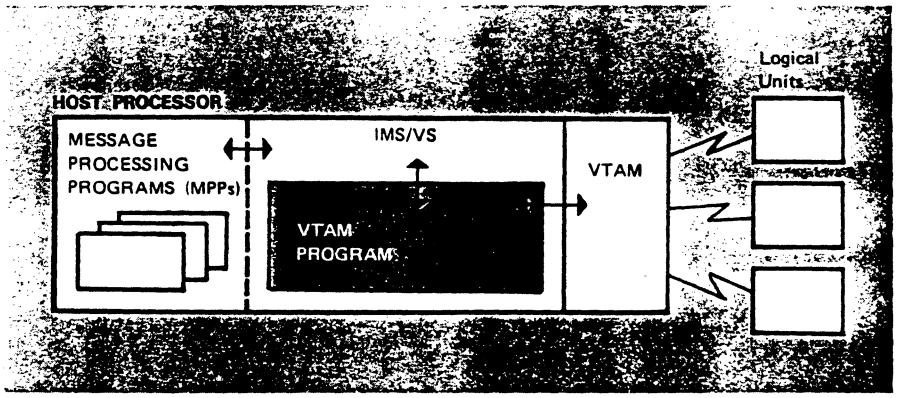

Figure 2-4. VTAM Relationship to IMS/VS

Figure 2-4 shows only those logical units that belong to VTAM. IMS may communicate with other resources through another access method.

The message processing programs (MPPs) shown in Figure 2-4 are user-written programs. The MPPs communicate with IMS in order to send data to a logical unit or to receive data from a logical unit. IMS, in turn, communicates with logical units through the VTAM program.

The IMS generation process generates a VTAM program as part of IMS/VS.

Please turn to Mini-Course 2 in your Personal Reference Guide and do Exercise 2.1.

Mini-Course 3

VTAM Definition Concepts

Mini-Course 3. VTAM Definition Concepts

Introduction

Typically, each user that installs a VTAM network has different requirements. In fact, the day-to-day activities of many data processing installations may require more than one VTAM network configuration. VTAM provides for this possibility. A user may generate several sets of network resource definitions and each set defines a subset of the network resources. Each time VTAM is started, the user has the option of directing VTAM to select specific sets of definitions to activate specific parts of a VTAM network. The user can modify the network configuration while the network is being used. Figure 3-1 illustrates one possible configuration.

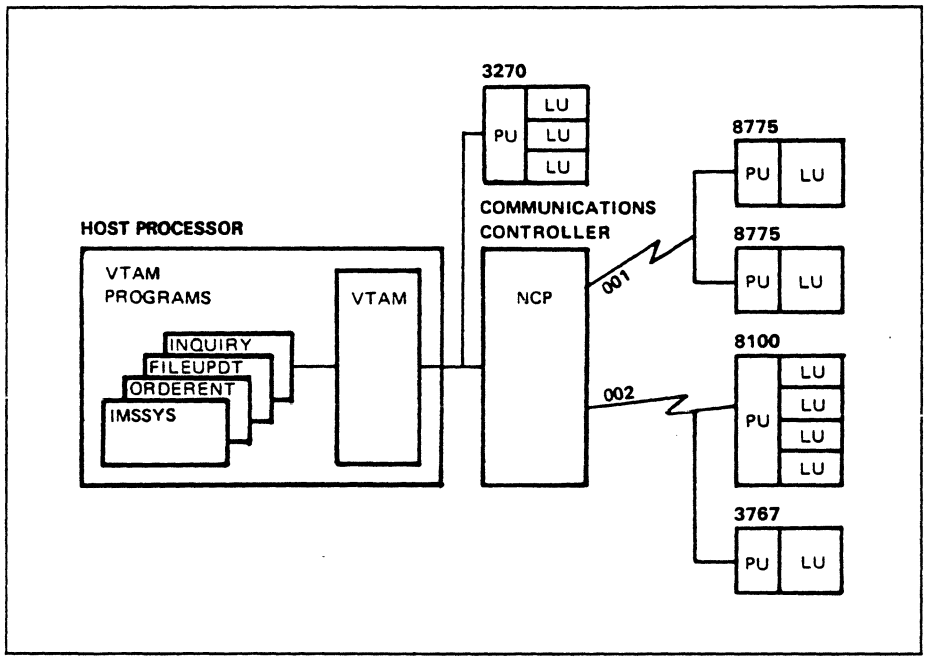

Figure 3-1. VTAM Network

The configuration in the figure includes four VTAM programs (LUs), one local SNA 3270 information display system, and one NCP subarea that includes two 8775 display terminals, one 3767 communication terminal, and one 8100 information system. All of these resources must be defined to VTAM.

Defining a Specific Network Configuration

A minimum of three sets of network definitions are required to define the network configuration shown in Figure 3-1 to VTAM.

- A set of definitions for the NCP subarea
- A set of definitions for the local SNA 3270 information display system
- At least one set of definitions for the VTAM programs (LUs)

The set of definitions for the NCP subarea is used for two purposes. First, the definitions are used to generate an NCP load module that is loaded into the

communications controller to control its network resources. The generated NCP will contain the addresses of all of its resources (data links, physical units, and logical units) along with other required information. Second, the definitions are used to provide VTAM with the information that it needs to manage the NCP subarea. VTAM obtains network addresses for the NCP resources from the resource resolution table (RRT) that is created during NCP generation.

The channel-attached 3270 must be defined to VTAM. In our example, one physical unit (PU) and three logical units (LUs) must be defined.

One to four sets of definitions are required to define the VTAM programs to VTAM. Each set of definitions for VTAM programs can include definitions for one or multiple VTAM programs. All definitions must be stored in the VTAM definition library.

At VTAM startup, VTAM can be directed to obtain all the network definitions, some of the network definitions, or none of the network definitions. VTAM uses the definitions to build resource definition tables (RDTs) to maintain the status of network resources. Once VTAM obtains a set of network definitions from the definition library and network addresses from the NCP resource resolution table and puts them in a resource definition table, VTAM has all the required information to monitor and control the associated resources.

Once VTAM is executing, the VTAM network operator can issue commands to VTAM to obtain more definitions from the definition library to build additional resource definition tables. VTAM can also be directed to remove sets of definitions from resource definition tables. Once a set of definitions has been removed from a resource definition table, VTAM knows nothing about the associated resources. As far as VTAM is concerned, those resources don't exist. So the VTAM network configuration can be changed while the network is being used.

VTAM Nodes

The sets of network definitions that we've been discussing are VTAM nodes. There are two types of VTAM nodes, *major* nodes and *minor* nodes.

VTAM major nodes include:

- VTAM application program
- Local non-SNA
- Local SNA
- Switched
- NCP
- Channel-attachment (CA)[1]
- Communications adapter (CA)[1]
- Cross domain resource manager (CDRM)
- Cross domain resource (CDRSC)

A VTAM network can contain a maximum of 255 major nodes. Each of the elements within a major node is called a minor node. Any major node may include one or more minor nodes. The number of major nodes and the number of minor nodes defined in each major node are determined by the user.

Figure 3-2 contains examples of major and minor nodes in a single domain. Study the figure, then read the discussion that follows.

---

1 Definitions for channel-to-channel attachments and for the communications adapter are included in the channel attachment major node. One set of definitions describes the channel-to-channel connection for two host processors while another set of definitions describes the links and resources attached to a communications adapter.

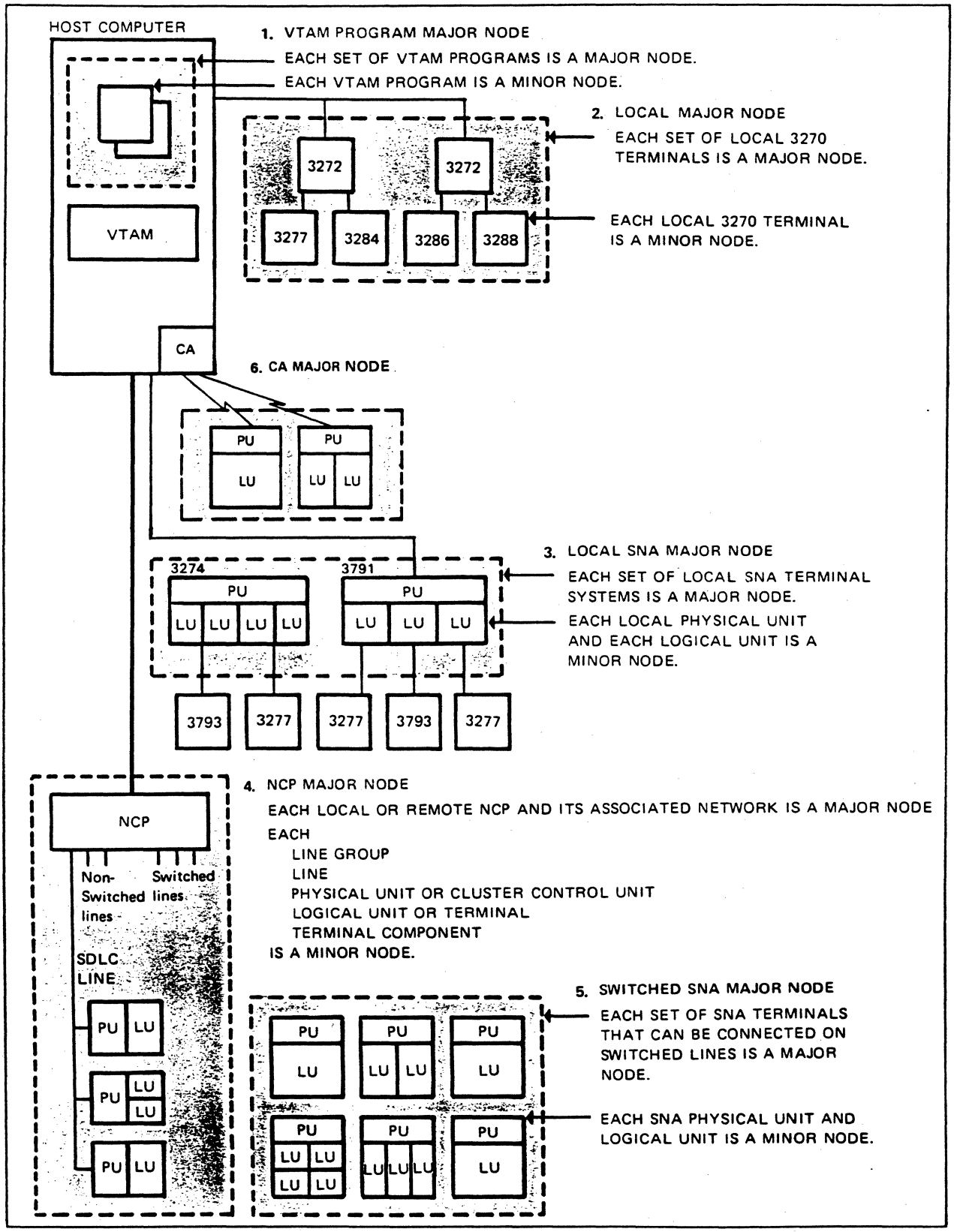

Figure 3-2. VTAM Nodes

VTAM Program Major Node

The VTAM program major node includes two minor nodes. Each minor node is a definition of a VTAM program (LU).

Typically, a VTAM program major node includes definitions for VTAM programs (minor nodes) that are to be active at the same time. Several sets of VTAM application major nodes may be defined and each major node will contain definitions for one or more VTAM programs.

VTAM activates a VTAM program major node by obtaining the definitions from the VTAM definition library and placing them in a resource definition table. Having done that, VTAM has all the facts that it needs about each of the VTAM programs defined in that major node.

Once a VTAM program major node has been activated, each of the VTAM program names can be used by a program to identify itself to VTAM as a VTAM program, that is, to form a session with VTAM (SSCP-LU). Once a VTAM program is in session with VTAM, it can establish sessions with other logical units.

Each VTAM program is defined with an APPL definition statement. The VTAM program major node example in Figure 3-2 would require two APPL statements.

Local Non-SNA Major Node

The local non-SNA major node includes four minor nodes; that is, it includes definitions for a 3277, 3284, 3286, and a 3288.

Each local non-SNA major node is defined with an LBUILD statement followed by one or more LOCAL statements. Each local 3270 terminal (minor node) is defined with a LOCAL statement. The 3272 controllers are not defined.

Local SNA Major Node

Item 3 in Figure 3-2 is an example of a local SNA major node. One or more sets of terminals can be defined as a major node. For example, a 3274 information display system and a 3790 communications system can be defined as one major node or as two major nodes.

A VBUILD definition statement is used to describe the local SNA major node. PU statements describe the characteristics of each physical unit, while LU statements describe the characteristics of each logical unit.

NCP Major Node

Item 4 in Figure 3-2 is an example of an NCP major node. One or more NCP major nodes can be defined for each communications controller. Only one NCP major node can be active at one time for a given communications controller. NCP minor nodes include the following:

- Lines, each defined by a LINE statement
- Physical units, each defined by a PU statement
- Logical units, each defined by an LU statement

Switched SNA Major Node

Item 5 in Figure 3-2 is an example of a switched SNA major node. One or more major nodes can be defined for SNA terminals on switched lines. Each major node represents one or more physical units and their associated logical units.

A VBUILD statement begins the definition of a switched SNA major node. A PU statement describes each physical unit, an LU statement describes each logical unit, and one or more PATH statements are required for each PU statement.

Communication Adapter Major Node

Item 6 in Figure 3-2 is an example of a communication adapter major node.

ACF/VTAM supports SDLC and BSC lines attached to a communication adapter in a VSE system. A communication adapter (CA) major node can define one line and attached resources or it can define multiple lines and attached resources. A VBUILD statement is used to define the major node and LINE, PU, and LU statements are used to define the minor nodes.

Major Nodes Unique To Multiple Domain Networks

CDRM, CDRSC, and CA major nodes are used in multiple domain networks. A channel-to-channel attachment (CA) major node is defined for a channel that connects two domains, that is, the channel connects two host processors that contain VTAM.

CDRM major nodes are defined so that each domain can communicate with every other.

CDRSC major nodes are defined so that logical units in one domain can communicate with logical units in other domains.

Naming Major and Minor Nodes

Each major and minor node must be assigned a symbolic name. The following rules apply to assigning names:

- Duplicate major node names are not permitted.
- Duplicate minor node names are not permitted in the same major node.
- A major node and a minor node may not have the same name.
- Major nodes containing duplicate minor node names cannot be active simultaneously. If a major node that contains a duplicate minor node name is activated, the second minor node name is ignored.

Node Structure

VTAM definition statements are used to identify all major and minor nodes and to place each node within a hierarchical structure of controllable elements. Each major node structure has the general form shown in Figure 3-3.

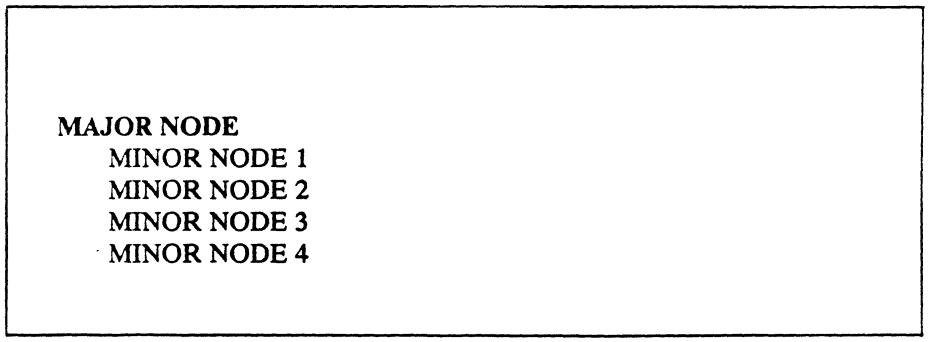

Figure 3-3. VTAM Node Structure

Each major node structure can be controlled as a whole, or portions of it can be controlled through the minor nodes within each major node; that is, an individual minor node can be activated as long as all nodes between the minor node and VTAM are active. A user can activate minor nodes by activating the associated major node. Similarly, deactivating a major node causes all subordinate nodes (minor nodes) to be deactivated.

Installing and Implementing a VTAM Network

An operational ACF/VTAM network is accomplished by first installing ACF/VTAM and then providing the definitions for ACF/VTAM to monitor and control the network. Installing ACF/VTAM includes the following activities:

- Defining VTAM, local devices, and VTAM system libraries to the operating system
- Loading VTAM modules in a VTAM library

Implementing the VTAM network includes the following:

- Defining nodes to VTAM
  - NCP
  - Local non-SNA
  - Local SNA
  - VTAM program
  - Channel-to-channel attachment
  - Communications adapter
  - Switched
  - Cross domain resource manager (CDRM)
  - Cross domain resource (CDRSC)
- Coding and including accounting and authorization exit routines

- Defining connection and disconnection procedures for logical units (USS tables and logon mode tables)
- Defining a class of service table
- Defining start options
- Defining configuration lists
- Defining paths

We will discuss the installation and implementation process in relation to the VTAM configuration shown in Figure 3-4. Study the figure, then continue reading.

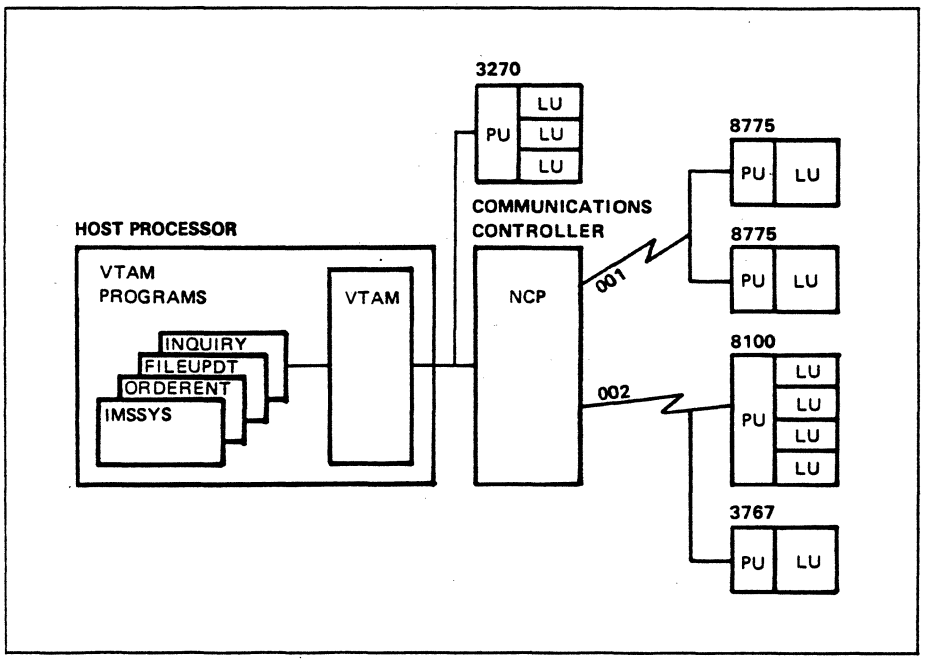

Figure 3-4. VTAM Network to be Defined

Installing VTAM

Installing ACF/VTAM includes defining VTAM, local devices, and VTAM libraries to the operating system and loading ACF/VTAM modules from the product tapes into the appropriate VTAM library. This is a system generation process. In Figure 3-4, the local 3270 controller and the communications controller must be defined to the operating system.

VSE. You no longer need to define ACF/VTAM to VSE because the VSE system generation statements already include ACF/VTAM support. Core image libraries contain VTAM load modules, tables, and system exit routines and source statement libraries contain network definitions. The maintain system history program (MSHP) is used to load VTAM modules from the product tapes to the core image library and is illustrated in Figure 3-5.

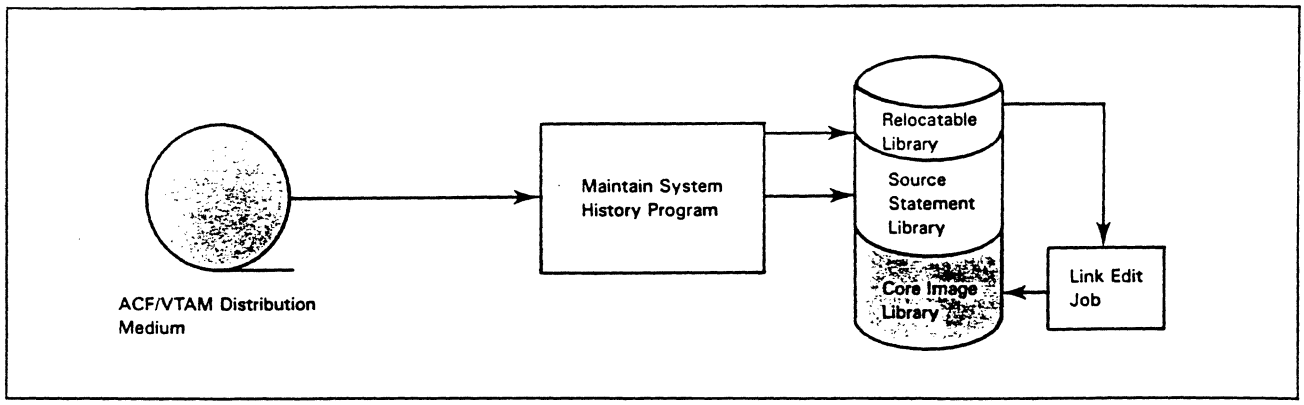

Figure 3-5. Installing ACF/VTAM Modules (VSE)

OS/VS. The VTAM libraries used by OS/VS1 and OS/MVS systems include VTAMLST, VTAMLIB, and VTAMOBJ. VTAMLST contains VTAM network definitions and VTAMLIB contains VTAM modules, tables, and system exit routines. VTAM stores its records of active major nodes in VTAMOBJ the first time that the major node is activated. The user does not access or store information in VTAMOBJ.

The system modification program (SMP) is used to load VTAM modules from the product tapes into VTAMLIB and is illustrated in Figure 3-6.

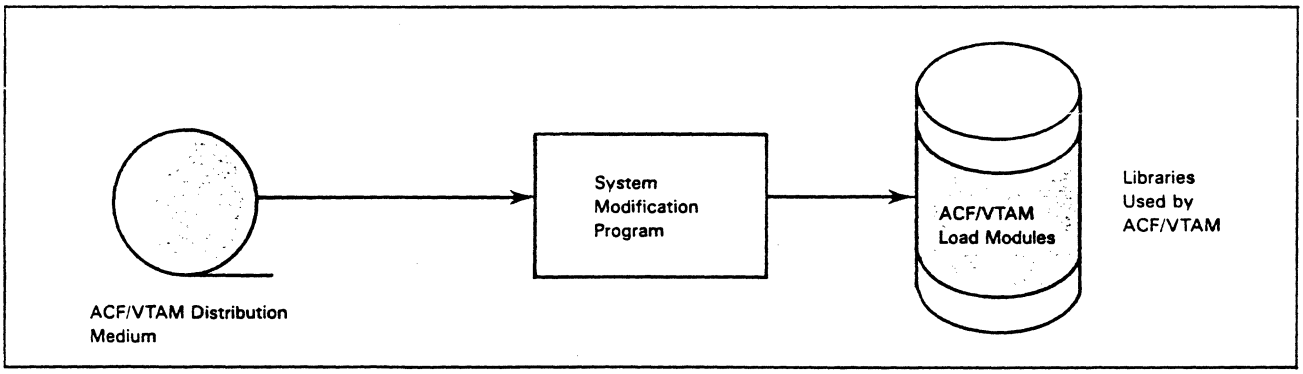

Figure 3-6. Installing ACF/VTAM Modules (OS/VS)

Please view the videotape for Mini-Course 3. Then turn to Mini-Course 3 in your Personal Reference Guide and do Exercise 3.1.

VTAM Concepts

Mini-Course 4

Controlling a VTAM Domain

MINI-COURSE 4. Controlling a VTAM Domain

Introduction

Controlling a VTAM domain includes starting and stopping VTAM as well as controlling the domain while VTAM is active. VTAM can be controlled by network definitions and by network operator commands.

Controlling VTAM with VTAM Definitions

VTAM is controlled at startup by start options and by configuration lists. VTAM start options, which tailor the VTAM domain, do the following:

- Assign a subarea number to the VTAM subarea
- Specify the number and size of each VTAM buffer
- Specify the maximum number of sessions that can be initiated or terminated at the same time

The configuration list specifies the major nodes that VTAM is to activate at startup.

Definitions of network resources can specify such things as the following:

- Whether a minor node will be activated when its associated major node is activated
- Whether a logical unit will be automatically logged on to an active VTAM program when the LU is activated

Controlling VTAM With Network Operator Commands

Network operator commands are used to start/stop, monitor, and modify the network. Network operator commands may be initiated by a network operator, or many of the commands may be initiated by a VTAM program called a program operator. The program operator can monitor and control a VTAM domain but it cannot start or stop VTAM.

Resource Definition Tables (RDTs)

Before getting into the discussion on starting VTAM, we'll explore the concept of how VTAM monitors the status of its network resources.

VTAM builds tables to keep track of its current configuration. It builds a table for a major node when there is a request to activate the major node. For example, assume that VTAM is to activate a local SNA major node (3270 system). VTAM builds a table and obtains the definition statements for that 3270 to make the necessary entries in the table. Later, the network operator may issue a command to activate an NCP major node. This causes VTAM to build a table for that node. Definition statements for that NCP and its network are obtained to make the necessary entries in the table.

VTAM builds table entries for each major node that is activated. VTAM maintains the status (active or inactive, for example) of each resource defined in each active

major node. VTAM deletes the table for a major node when that node is deactivated.

Each of the definition tables built by VTAM is called a **resource definition table (RDT)**. RDT entries include such information as:

- The symbolic name of each addressable resource (logical units or physical unit, for example)
- The network address of each resource
- The status of each resource (that is, active, inactive, or in session with another resource)

You should understand that RDTs identify VTAM's world. If it isn't in an RDT, then VTAM doesn't know about it.

Starting VTAM

Starting VTAM not only means that VTAM execution is initiated, but also that VTAM may obtain network definitions and build resource definition tables (RDTs) for the definitions. At start up, VTAM can obtain the network definitions and build the RDTs to support the whole network, part of the network, or none of the network. Some things that may happen during the start-up process are the following:

- Some or all nodes may be activated.
- Some or all logical units may be logged on to active VTAM programs.
- Selected VTAM facilities (such as TRACE) may be activated.

The network configuration shown in Figure 4-1 is the basis of our discussion in this mini-course. It shows the VTAM network that was defined in Mini-Course 3. We're going to discuss how VTAM is started and how it is prompted to obtain the definitions to support the nodes shown in the figure.

During discussions on activation and deactivation you should refer to the figure to keep track of the status of the network as it is modified.

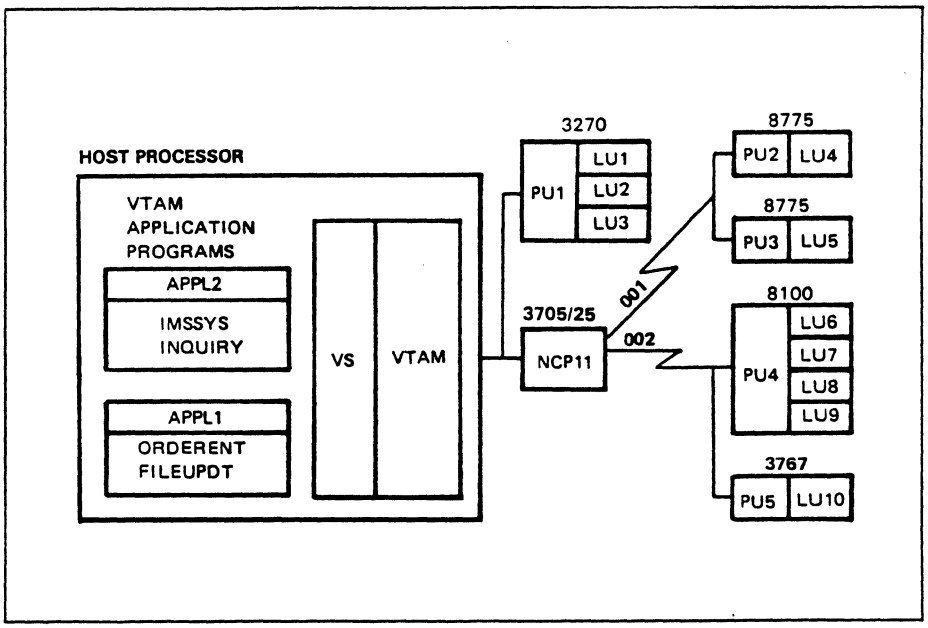

Figure 4-1. Configuration Defined to VTAM

The START Command

The format of the START command shown below is for an OS/VS system.

```
START procname [,,,(options)]
```

VTAM can be started by JCL or by a network operator command (EXEC command for VSE and START command for OS/VS).

The procedure name is required. It is the name of a cataloged procedure that contains the JCL needed to start and run VTAM. This procedure is written and named by the user. If any start options are included with the command, they override any duplicate options in the specified start option definitions.

The EXEC Command

The format of the EXEC command (for VSE) is as follows:

```
EXEC   PROC=name
```

In this command, "name" is the name assigned to a user-written cataloged procedure that contains the JCL needed to start and run VTAM. Start options are not allowed in the EXEC command, but VTAM can prompt the operator to enter options.

Configuring VTAM

Before beginning the discussion of starting VTAM, we need to define the terms **active status** and **inactive status**. A physical unit or a logical unit is active when it is in session with VTAM; that is, there is an SSCP-PU session and an SSCP-LU session. The systems services control point (SSCP) is a component of VTAM.

Figure 4-2 shows our network definitions and the START command issued by the network operator. Study the figure, then read the discussion that follows.

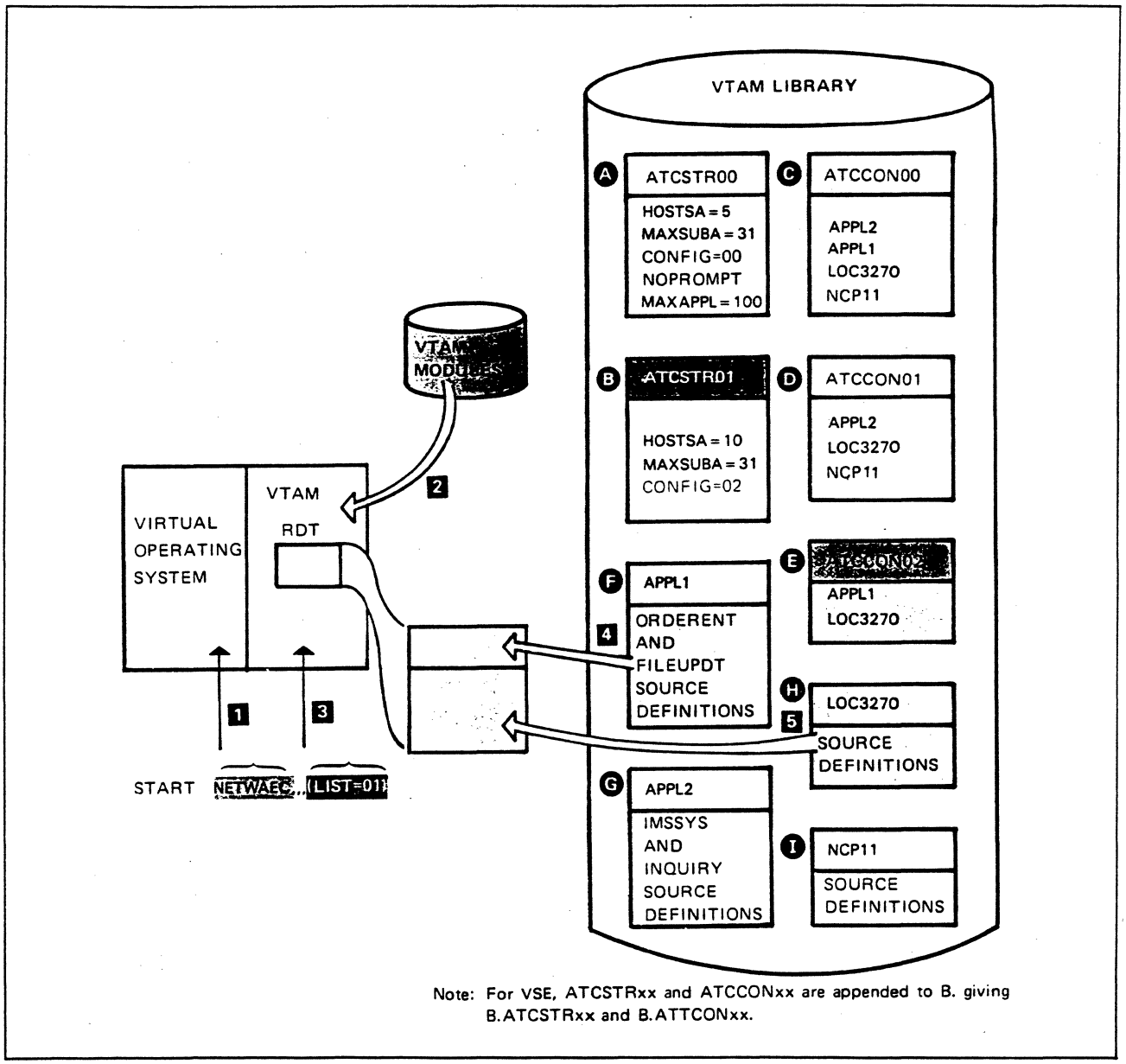

Figure 4-2. Starting VTAM

The network definitions in the VTAM definition library are labeled A through I. Items F, G, H, and I illustrate the definitions for the VTAM configuration shown in Figure 4-1. Items A and B show start options lists and items C, D, and E show configuration lists. Also shown is the VTAM load module library that contains VTAM modules.

We'll assume that the START command shown at the lower left of Figure 4-2 is submitted by the network operator. It specifies the start procedure NETWAEC (item 1) and a start options list LIST=01 (item 3).

Now let's talk about the function of the NETWAEC start procedure. It causes the VTAM modules (item 2) to be loaded into virtual storage and then starts VTAM executing.

VTAM selects the start list specified in the START command, start list 01 (ATCSTR01), and obtains the start options from that list to tailor itself.

ATCSTR01 contains three start options: (1) HOSTSA=10, (2) MAXSUBA=31, and (3) CONFIG=02. HOSTSA=10 specifies that VTAM and the major nodes that are part of the VTAM subarea are to be assigned subarea 10. MAXSUBA=31 specifies that the VTAM network will support a maximum of 31 subareas. CONFIG=02 identifies the configuration list (ATCCON02) that contains a list of major nodes to be activated by VTAM during this startup.

Once VTAM is initialized it will activate the major nodes. We'll discuss the activation process of major nodes after we complete our discussion of the other start options.

So far we've used only four start options. These start options came from two sources: one from the START command (LIST=01) and the three from the referenced start list (ATCSTR01). When starting VTAM in an OS/VS system, there are four sources of start options:

1. START command

2. Start options list specified in the START command

3. Start options list ATCSTR00

4. Defaults

The order of this list also indicates the start option selection hierarchy. Start options in the START command take precedence over all other option sources, the options list specified in the START command takes precedence over options from ATCSTR00 and defaults, and the options from ATCSTR00 take precedence over defaults.

Since VTAM has used start options from the START command and from ATCSTR01, it will obtain options from ATCSTR00 next. VTAM searches ATCSTR00 for those start options that weren't specified in the START command and in ATCSTR01.

The NOPROMPT option specifies that the network operator is not to be prompted for options. The PROMPT/NOPROMPT option is ignored when the START command specifies a start options list other than ATCSTR00, which it does in our example. HOSTSA, MAXSUBA, and CONFIG are not used because they're specified in ATCSTR01. MAXAPPL=100 will be used because it isn't specified in ATCSTR01. This option specifies that a maximum of 100 VTAM programs (logical units) will be supported by VTAM.

The VTAM default values are used for the remaining start options.

Now that VTAM is started and tailored according to the start options, VTAM uses the CONFIG parameter (in ATCSTR01) to determine which major nodes (if any) are to be activated. CONFIG=02 means that configuration list ATCCON02 (item E) contains the list of major nodes that VTAM is to activate at this time.

ATCCON02 specifies two major nodes (APPL1 and LOC3270), therefore VTAM will activate those major nodes. The two major node definitions are shown at items

F and H. VTAM obtains the definitions from the VTAM library, builds a resource definition table (RDT) for each major node and inserts the definitions in the tables. The two major nodes are considered to be active once the RDTs are completed. Figure 4-3 shows VTAM's view of the network at this time.

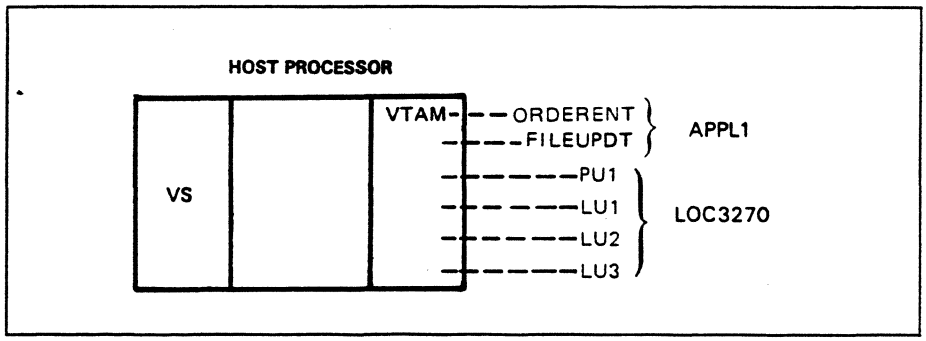

Figure 4-3. VTAM's View of the Network After Startup

The APPL1 major node definitions for the VTAM programs ORDERENT and FILEUPDT reside in one RDT. ORDERENT and FILEUPDT are minor nodes. The LOC3270 major node defines the minor nodes PU1, LU1, LU2, and LU3.

A major node is said to be active when the major node definitions are included in the RDT. PUs and LUs (minor nodes) are said to be active when they're in session with the SSCP. Therefore, PU1, LU1, LU2, and LU3 are not active at this time. However, all of the minor nodes are known to VTAM as their definitions reside in an active major node.

Minor nodes can be activated during VTAM startup if their definitions specify them as **initially active**, and if all nodes, links, and link stations in the path to VTAM are active. Since PU1 is connected to the host channel, there are no nodes between it and VTAM. Assuming that PU1 and LU1 are defined as **initially active**, they will be activated during VTAM startup in our example.

Figure 4-4 illustrates the activation of PU1 and LU1. Study the figure, then read the discussion that follows.

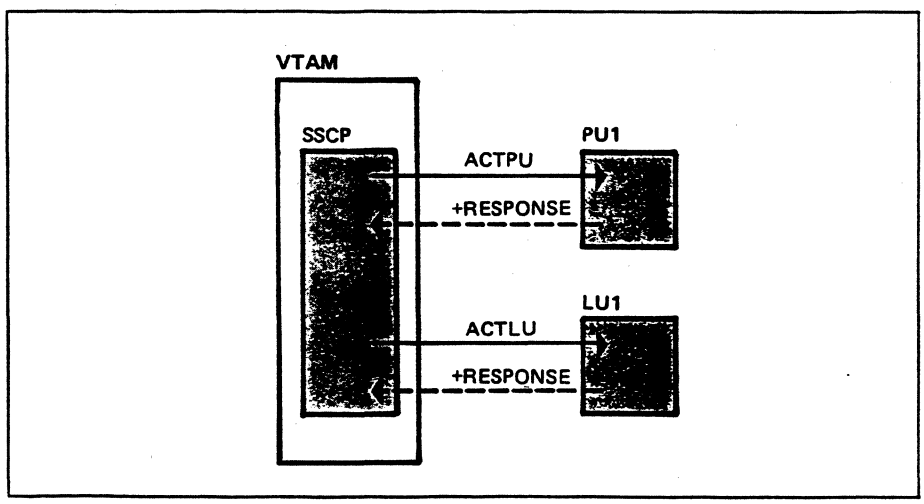

Figure 4-4. Activating PU1 and LU1

Once VTAM activates the LOC3270 major node, it scans the minor node definitions to determine if they're to be activated. The definition for PU1 contains ISTATUS=ACTIVE which causes VTAM to send the ACTPU request to activate PU1. LU1 is also defined as ISTATUS=ACTIVE, therefore VTAM sends the ACTLU request to activate LU1.

The startup operation is complete. VTAM has started executing, has initialized itself according to the specified start options, and has been configured to support the major nodes APPL1 and LOC3270 as specified in the configuration list ATCCON02. And now there is an SSCP-PU1 session and an SSCP-LU1 session.

Now test your understanding of the startup process with the following questions.

Questions

1. Which major and minor nodes are in the VTAM RDT(s) at this time?

2. Which major and minor nodes are active at this time?

3. Which nodes may establish sessions with active VTAM programs (LUs) at this time?

Answers

1.
```
APPL1 (major node)
        ORDERENT (minor node)
        FILEUPDT (minor node)

LOC3270 (major node)
  PU1 (minor node)
  LU1 (minor node)
  LU2 (minor node)
  LU3 (minor node)
```

2.
```
APPL1 (major node)

LOC3270 (major node)
  PU1 (minor node)
  LU1 (minor node)
```

3. LU1 (This is the only logical unit active at this time.)

Now we'll take a look at what happens when the START command contains no start options as shown in Figure 4-5. The START command located in the lower-left part of the figure is coded as follows:

```
START  NETWAEC
```

This START command includes only the procedure name and no options. You might wonder which start options list and configuration list, if any, will be used during startup. Start options list 00 (ATCSTR00) is used. In Figure 4-5 you can see that ATCSTR00 defines HOSTSA=5, MAXSUBA=31, CONFIG=00, NOPROMPT, and MAXAPPL=100.

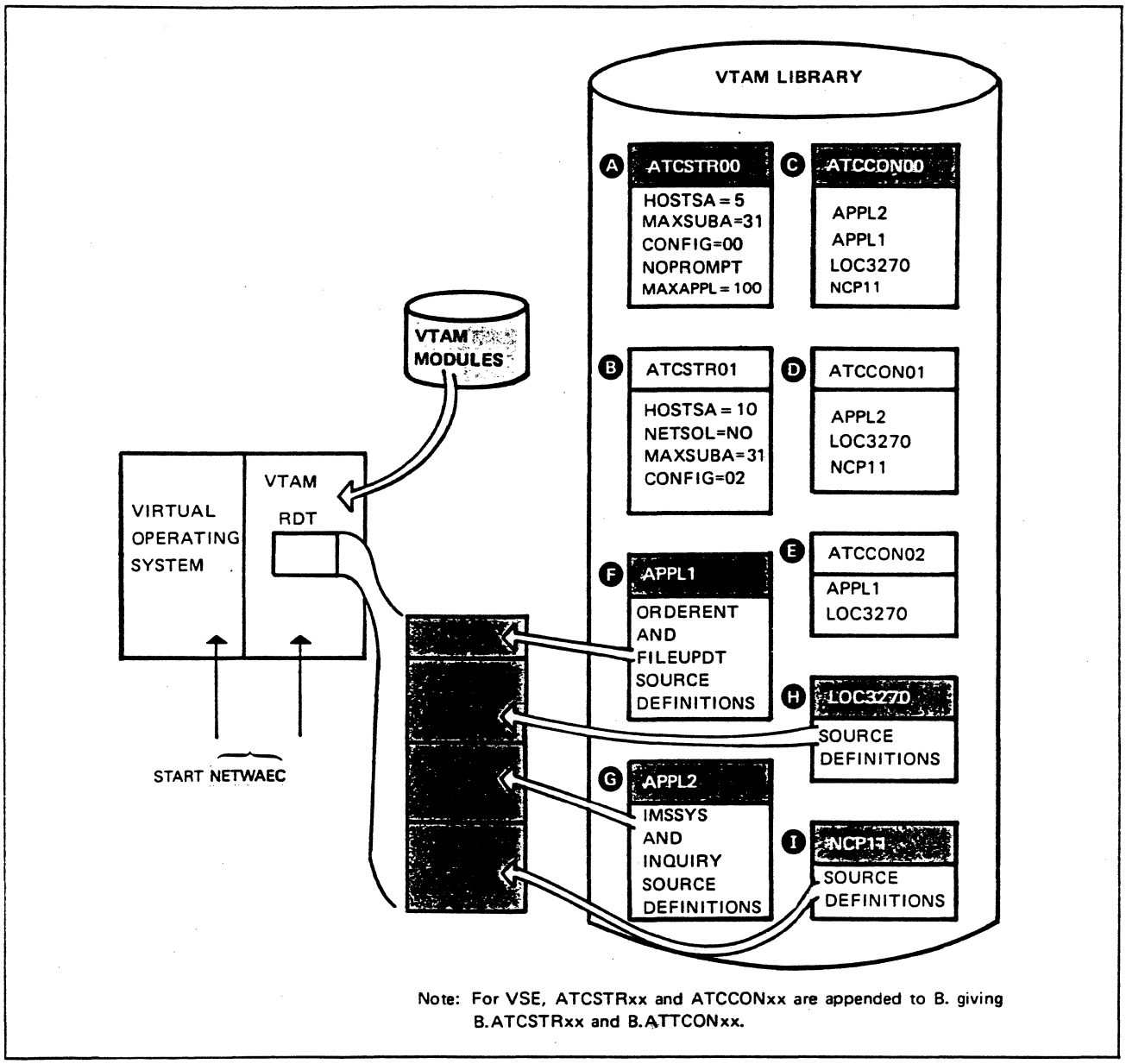

Figure 4-5. Selecting a Start Options List

VTAM and the resources in its subarea will be assigned subarea number 5, maximum number of subareas supported is 31, configuration list ATCCON00 defines the major nodes to be activated, NOPROMPT specifies that the network operator is not to be prompted for start options, and a maximum of 100 VTAM programs can be supported. ATCCON00 specifies that major nodes APPL1, APPL2, LOC3270, and NCP11 are to be activated. It just so happens that this includes the entire network.

RDTs are built and initialized for the major nodes APPL1, APPL2, LOC3270, and for NCP11.

This completes the discussion on VTAM startup. Now assume that VTAM is configured according to the START command shown in Figure 4-2. The major nodes APPL1 and LOC3270 are active as well as the minor nodes PU1 and LU1.

Modifying the Network Configuration

Now we'll talk about modifying the network configuration after VTAM startup completes. The network operator command VARY can be used to initiate activation and deactivation of major and minor nodes. Figure 4-6 shows a VARY command that requests VTAM to activate the major node NCP11.

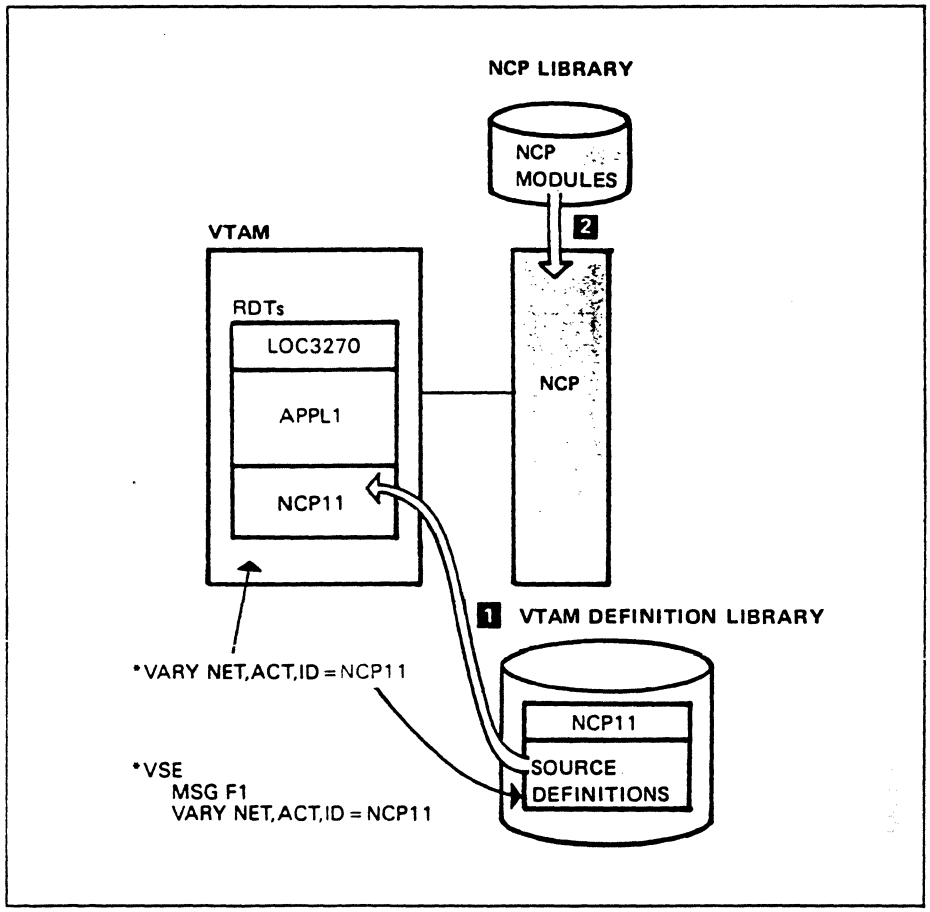

Figure 4-6. VARY Active NCP11

NET identifies a procedure which routes the VARY command to VTAM. ACT says that the specified node is to be activated, and ID identifies the node to be activated.

VTAM receives the VARY command and determines that it is to activate the NCP11 major node which includes the NCP, one 3767, two 8775s, and an 8100 (see Figure 4-1).

VTAM builds a resource definition table (RDT) and makes entries required to support the NCP11 major node. This information is obtained from NCP11 source definitions shown at item 1 in Figure 4-6. The definition library is the same one that we've been using all along. The figure illustrates only the one set of definitions, although we know that the VTAM definition library contains all the network definitions.

Now there are three RDTs containing three sets of definitions:

1. LOC3270 major node
2. APPL1 major node
3. NCP11 major node

This activation process also causes the NCP executable modules to be loaded into the appropriate communications controller (item 2). Figure 4-7 shows the SNA request flow which activates NCP11 as well as the two attached data links. Study the figure, then read the discussion that follows.

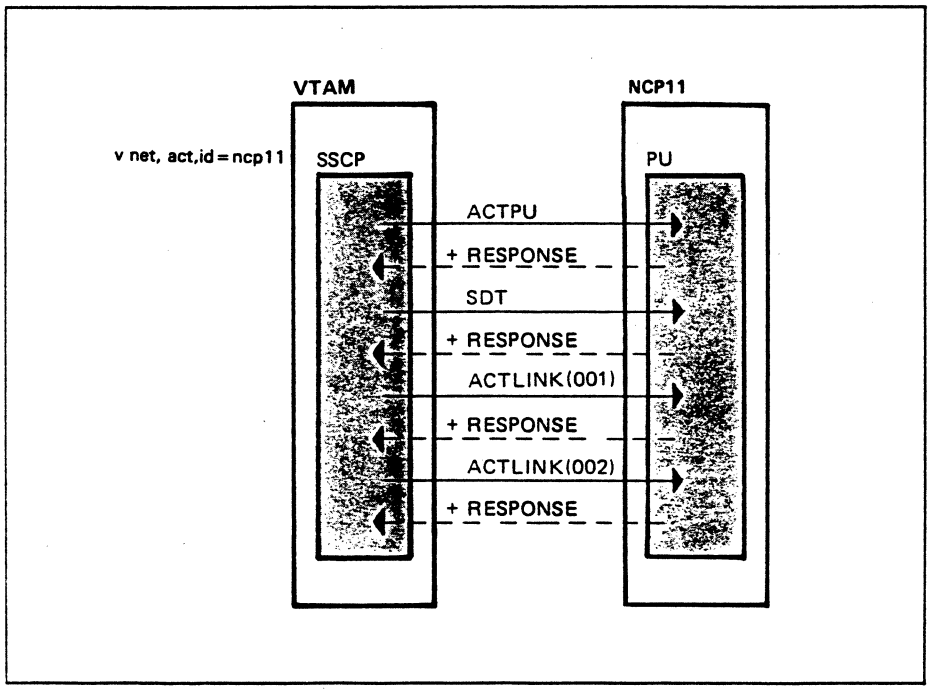

Figure 4-7. Activating NCP11 and Attached Links

The NCP is activated when the SSCP sends an activate physical unit (ACTPU) request to NCP's physical unit. This establishes the SSCP-PU session with NCP. The start data traffic (SDT) request is sent to place the session in data-traffic-active state so function management data (FMD) requests can flow on the session. Now VTAM can communicate with and through the NCP. Next, the SSCP sends activate link (ACTLINK) requests to NCP's PU requesting that it activate links 001 and 002.

Now turn back to Figure 4-1. Notice links 001 and 002 and the position of the physical units and logical units attached to those links.

Which of the physical units and logical units defined to NCP11 will be activated, if any? The physical units and logical units will be activated only if they were defined as ISTATUS=ACTIVE. We can go one step further and say that initially active logical units will be activated only if their associated physical units are activated. As an example, let's assume that PU4 and LU8 were defined as ISTATUS=ACTIVE. Since the NCP and data link 002 are active, it's possible to activate PU4 and LU8, in that order. Figure 4-8 shows the operator and SNA request sequence that accomplish this.

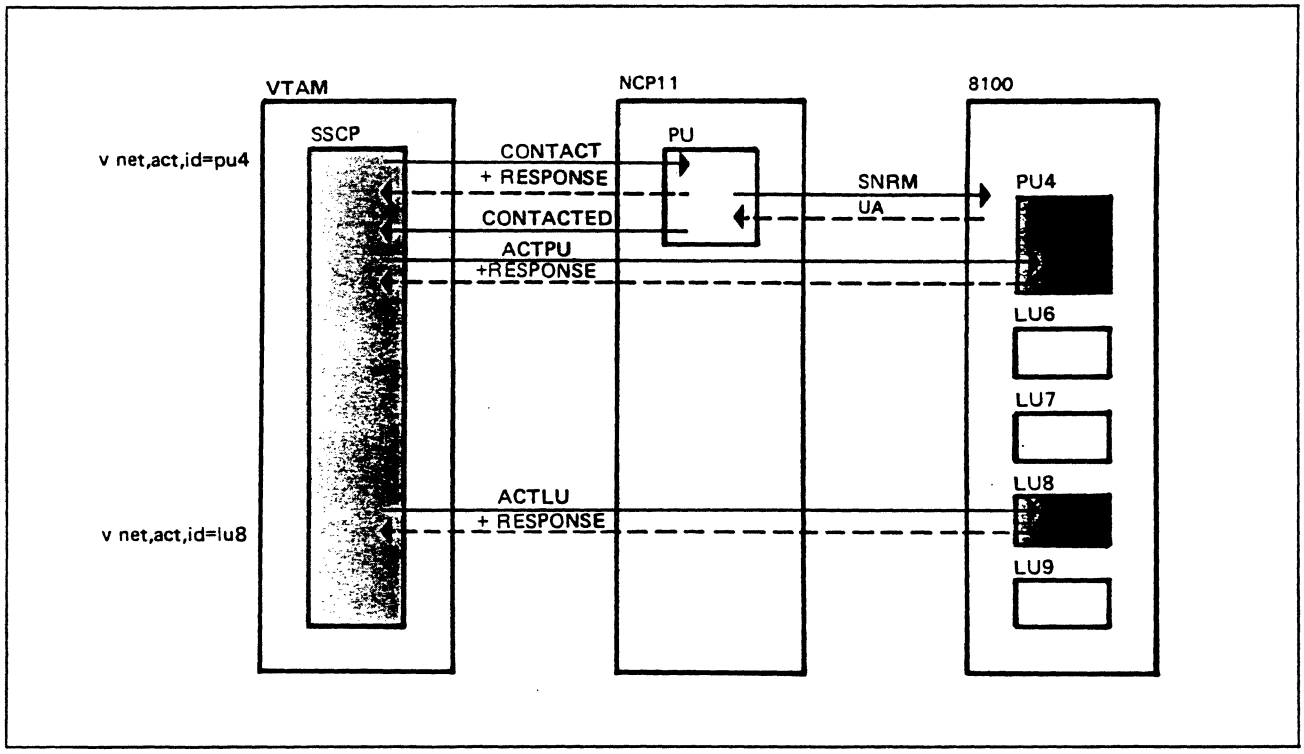

Figure 4-8. Activating PU4 and LU8

The operator command

```
v net,act,id=pu4
```

causes the SSCP to send a CONTACT request to the NCP's physical unit that requests it to contact the 8100 controller. The contact is performed at the data link control (DLC) level using SDLC commands set normal response mode (SNRM) and unnumbered acknowledgement (UA). This SDLC exchange says that the 8100 controller is "powered-on" and is ready to establish communication with the SSCP. Next, the SSCP sends the ACTPU request to PU4 to establish an SSCP-PU4 session. With that session established, the operator command

```
v net,act,id=lu8
```

causes the ACTLU request to be sent to activate LU8. This establishes the SSCP-LU8 session. Now both PU4 and LU8 are active. That is, the SSCP is in session with both PU4 and LU8.

The VTAM operator commands generally control the SSCP-PU and SSCP-LU sessions. The LU-LU sessions involve the logical units, and ordinarily the VTAM network operator is not involved with establishing and terminating LU-LU sessions.

At this point, what resources does VTAM know about? Earlier we said that VTAM knew about only those items in its RDTs. There are three RDTs at this time, one for the major node LOC3270, one for the major node APPL1, and one for the major node NCP11. Figure 4-9 illustrates VTAM's view of the system at this point in time.

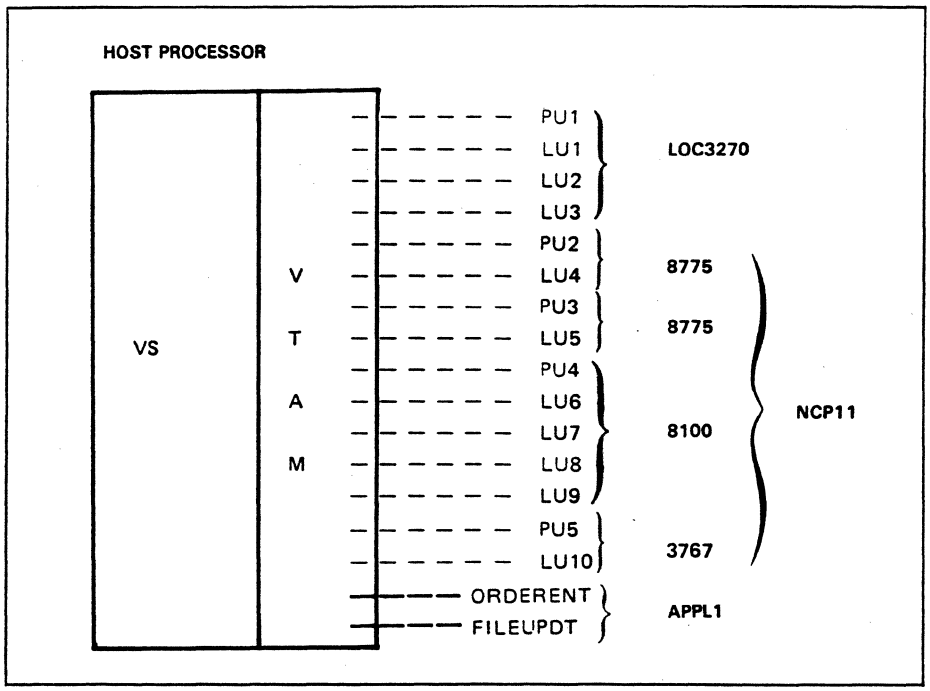

Figure 4-9. VTAM's View of the Network

The figure shows that VTAM knows about everything except the major node, APPL2. PU1, LU1, PU4, and LU8 are highlighted to indicate that they are active.

ORDERENT and FILEUPDT are not executing at this time. The figure indicates only that their definitions are now in a VTAM RDT.

Please turn to Mini-Course 4 in your Personal Reference Guide and do Exercise 4.1.

VTAM Concepts

Mini-Course 5

Identifying a VTAM Program to VTAM

MINI-COURSE 5. Identifying a VTAM Program to VTAM

Introduction

A VTAM program that is executing under control of an operating system (VSE or OS/VS) can issue an OPEN macro to identify itself to VTAM at any time. Specifications in the OPEN macro depend on the contents of VTAM's resource definition tables (RDTs). Therefore, we'll briefly review RDTs before getting into the concept of how a VTAM program identifies itself to VTAM.

Mini-Course 4 pointed out that VTAM's RDTs contain all the information required by VTAM to monitor and control the network. The symbolic name of each VTAM program and each logical unit is included in an RDT. The actual format of the RDT isn't important to grasp the concept of identifying a VTAM program to VTAM. Therefore, we can view the general representation of the RDT as shown in Figure 5-1 .

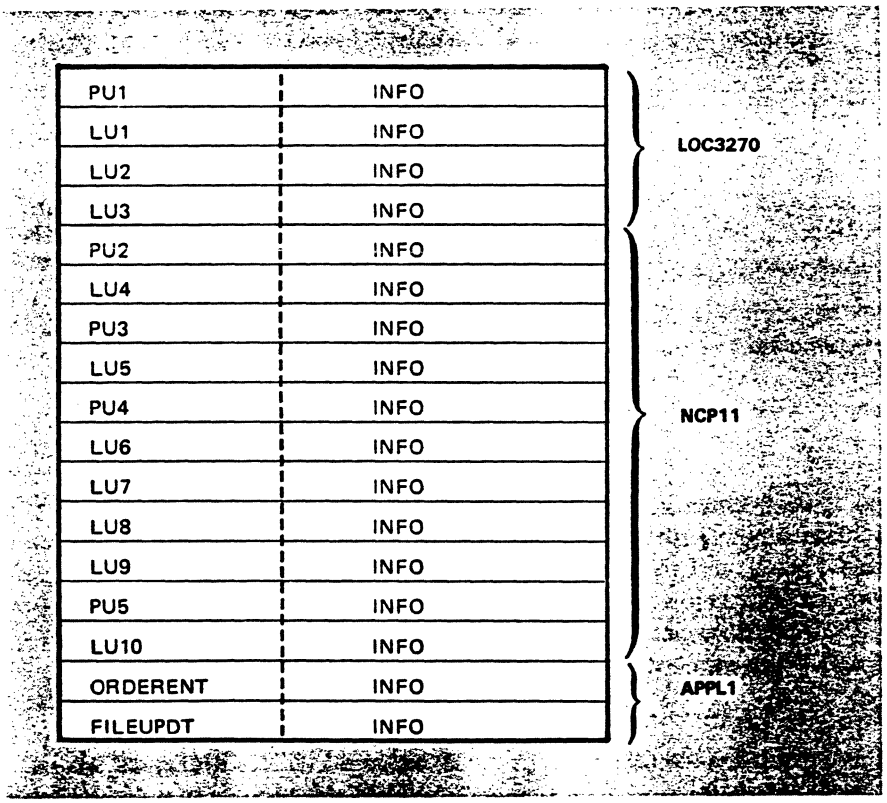

Figure 5-1. Resource Definition Table (RDT)

Figure 5-1 shows the symbolic names of all the physical units and logical units that VTAM knows about at this time. The figure also indicates that other information about each major node is contained in the RDT. With this picture in mind, let's consider how a VTAM program identifies itself to VTAM.

Activating a VTAM Program

A VTAM program issues an OPEN macro to identify itself to VTAM, which activates the VTAM program. The OPEN process must use a symbolic name that is contained in an RDT. Figure 5-1 includes two VTAM program entries (ORDERENT and FILEUPDT) in the RDT. An OPEN operation would have to identify either ORDERENT or FILEUPDT to successfully complete the open process. During this process, the VTAM program provides VTAM with other information during the OPEN process such as whether it will accept logons. The VTAM program provides the information in a control block, called the access method control block (ACB).

The ACB

The ACB identifies the VTAM program to VTAM and specifies certain processing requirements. The ACB identifies an APPLID (VTAM program identification) entry in the RDT and an appropriate password if the VTAM program was defined with a password. The ACB also specifies whether the VTAM progrm will accept logons and may identify some user-coded exit routines.

The OPEN process must specify an ACB, which is to say the ACB is opened. To VTAM, the ACB is the VTAM program. So it's quite proper to say that an ACB identifies itself to VTAM.

A VTAM program must specify an ACB in every VTAM macro instruction that it uses. For example, the VTAM program must specify an ACB when it sends data to, or receives data from another logical unit. The ACB provides information that tells VTAM some of the VTAM program's requirements.

Typically, a VTAM program will contain only one ACB, even though it can contain more than one. The use of multiple ACBs is a way of breaking a VTAM program into "sub-applications."

Opening an ACB

Now we'll look at an example that illustrates how a VTAM program identifies itself to VTAM. Figure 5-2 illustrates the relationship between the OPEN macro instruction, the ACB, and the RDT contents.

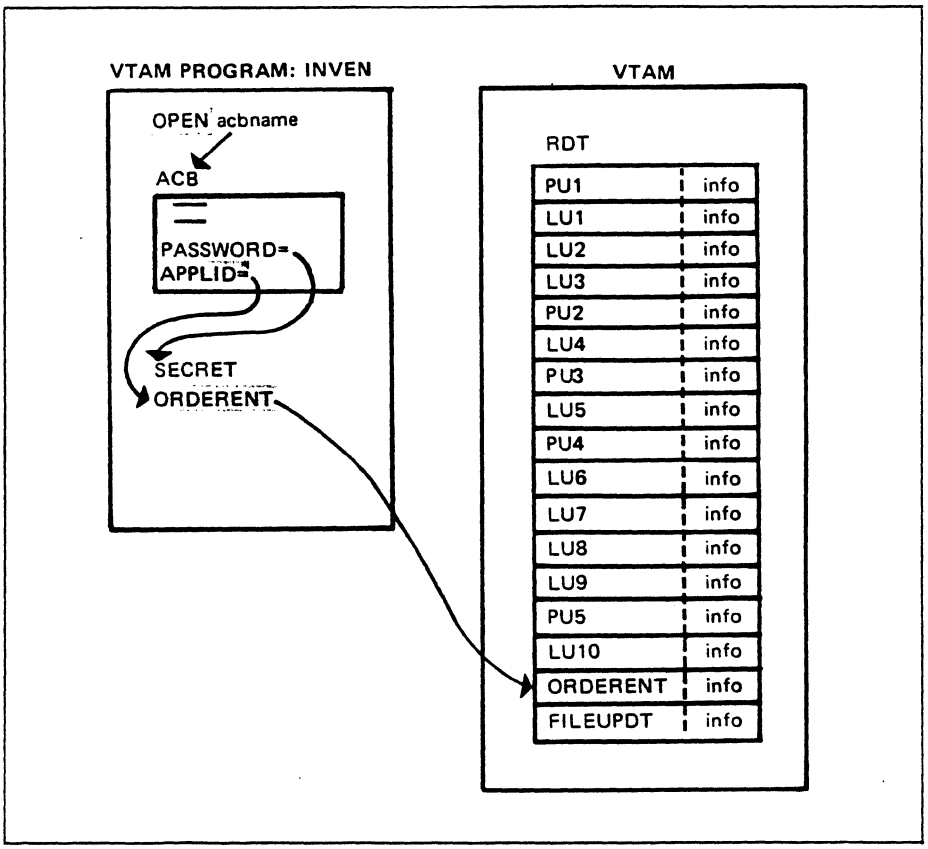

Figure 5-2. Opening an ACB

The OPEN macro instruction specifies the name of an ACB. This example shows only two fields of the ACB. The APPLID, field, identifies the symbolic name (ORDERENT) of the VTAM program as it is contained in the RDT. The other field identifies the password SECRET, assuming that VTAM program ORDERENT was defined with the password SECRET.

The VTAM program is stored in a program library under the name INVEN as shown in the top of the figure. The system operator starts the program executing under control of the operating system. Once INVEN is executing, it can issue the OPEN macro to identify itself to VTAM. The OPEN macro specifies the name of an ACB that identifies a VTAM program name that is contained in the RDT. The OPEN gives control to VTAM, and VTAM uses the program name identified by the APPLID field in the ACB to search its RDT.

Since the name ORDERENT is in the RDT, VTAM builds the necessary control block structure to identify the VTAM program to VTAM. VTAM uses these control blocks to manage any sessions in which the VTAM program participates.

Notice that the program name and the name specified in the ACB are different. The program name INVEN is known to the operating system while the name specified in the ACB, ORDERENT, is the one known to VTAM. Both names could be the same; that is, the program name known to the operating system could be ORDERENT.

Now we'll discuss the OPEN process for two other VTAM programs (IMSPROG and PROG2). Assume that both programs are executing. Study Figure 5-3 , then read the discussion that follows.

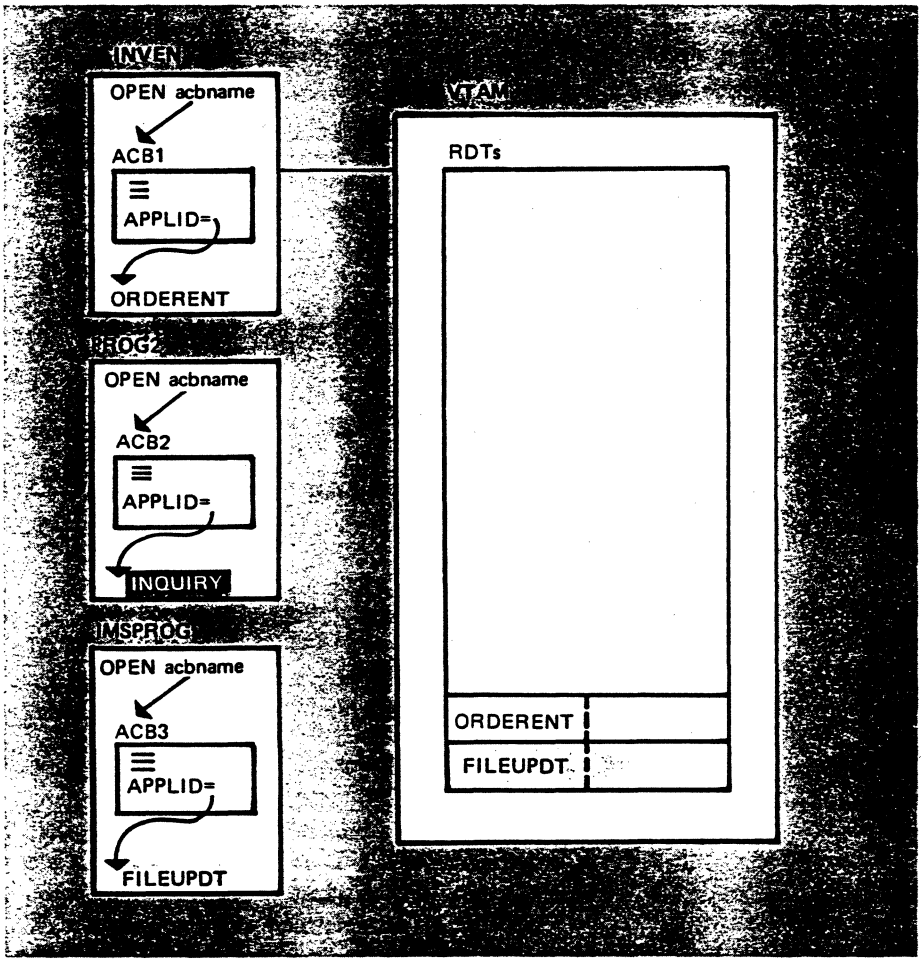

Figure 5-3. Identifying VTAM Programs to VTAM

PROG2 issues an OPEN macro to identify itself to VTAM via the name INQUIRY. The RDT shown in Figure 5-3 doesn't contain the name INQUIRY, because its associated major node, APPL2, hasn't been activated yet. Therefore the OPEN process completes unsuccessfully, and PROG2 is not identified to VTAM.

Although for our example we do not want PROG2 to be identified to VTAM yet, it would be possible if the operator command VARY was submitted to activate the APPL2 major node. This would cause the program names INQUIRY and IMSSYS to be placed in an RDT. Then PROG2 could identify itself to VTAM by issuing the OPEN that specifies an ACB identifying the VTAM program name INQUIRY. For now, however, let's consider the APPL2 node to be inactive.

Now look at IMSPROG in Figure 5-3. The OPEN macro instruction references the ACB (ACB3) which identifies the program name FILEUPDT. VTAM searches the RDT and finds that name. The OPEN process completes successfully and IMSPROG is identified to VTAM as FILEUPDT.

EXLST Exit Routines

VTAM provides for the designation and use of a group of special- purpose exit routines: LERAD, SYNAD, DFASY, RESP, SCIP, TPEND, RELREQ, LOGON, LOSTERM, and NSEXIT. The purpose of these routines is understood by both VTAM programs and VTAM. An EXLST exit routine is coded by the user as part of a VTAM program.

VTAM causes control to be transferred to a particular exit routine when a specific event occurs, such as a physical I/O error, a response from an LU, or when contact with an LU is lost. For example, a VTAM program issues a SEND macro instruction which directs VTAM to transmit a request to a logical unit. A logical error occurs, causing VTAM to give control to the LERAD exit routine. The coding provided by the user in the LERAD exit routine processes the error and returns control to VTAM.

The same sequence of operation occurs for each exit routine: VTAM gives control to the appropriate exit routine when a special event occurs, the user-coded exit routine processes the error, and control is returned to VTAM.

If an exit routine is not provided for a specific error, VTAM gives control to the next sequential instruction. For example, a SEND macro instruction is issued, and control is not to be returned to the VTAM program until the SEND operation completes. A physical error occurs, but there is no SYNAD exit routine. Therefore VTAM gives control to the instruction following the SEND macro instruction.

Some, all, or none of these exit routines can be included in a VTAM program. The exit routines that are included in a VTAM program are specified in an exit list control block (EXLST) in the program. The identity of an EXLST exit routine is made known to VTAM either when the program is opened, or for certain types of exit routines, when the program establishes a session with a logical unit.

We'll now discuss how the EXLST exit routines are identified to VTAM at the time the OPEN macro is issued. The ACB macro instruction identifies to VTAM the name of an EXLST that contains the address of each exit routine within the program. Figure 5-4 illustrates how EXLST exit routines are identified to VTAM.

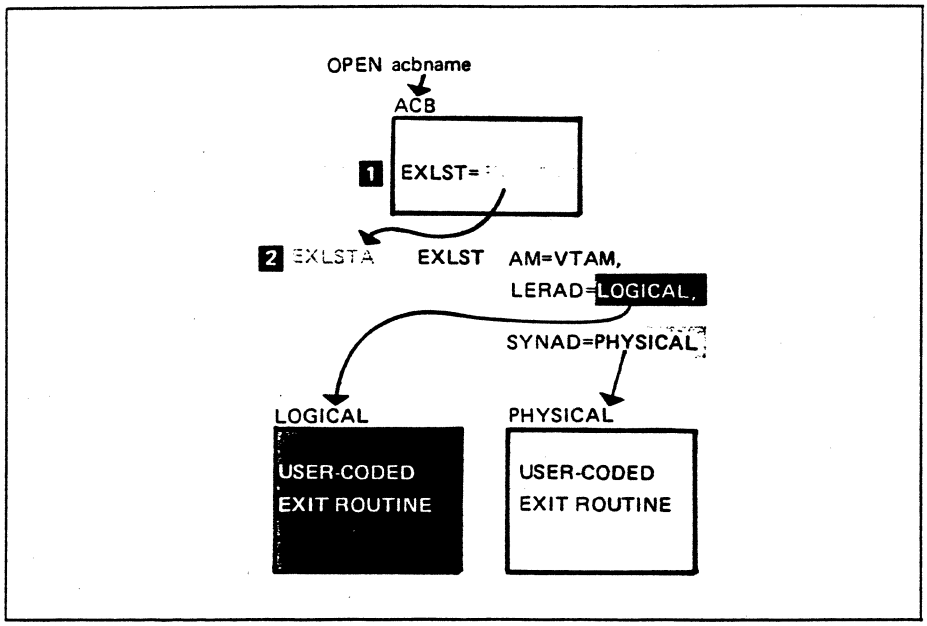

Figure 5-4. Identifying EXLST Exit Routines to VTAM

The ACB EXLST parameter (item 1) identifies an exit list located at EXLSTA. The EXLST macro instruction (item 2) builds and initializes the exit list control block. The exit list control block contains the addresses of the included exit routines. In this example, the control block identifies the LERAD and SYNAD exit routines to VTAM.

Please turn to Mini-Course 5 in your Personal Reference Guide and do Exercise 5.1.

VTAM Concepts

Mini-Course 6

LU-LU Session Concepts

MINI-COURSE 6. LU-LU Session Concepts

Introduction

VTAM program logical units and peripheral logical units must be in session with VTAM before sessions can be established between two VTAM programs or between VTAM programs and peripheral logical units. Once these SSCP-LU sessions are established session initiation requests can be submitted to VTAM to establish LU-LU sessions.

Peripheral logical units support one LU-LU session at a time while VTAM programs may be designed to support several LU-LU sessions at a time.

There are several sources for initiating sessions between logical units.

- Primary logical unit
- Secondary logical unit
- VTAM network operator
- Network definitions
- A third party logical unit

If an LU-LU session is initiated by a primary logical unit, this process is called *acquiring a logical unit.* All other session initiation requests are forms of *logons.* Whether acquiring an LU or issuing a logon, the session initiation request is submitted to VTAM (the SSCP). VTAM then notifies the appropriate VTAM program of the session initiation request. The VTAM program issues an OPNDST to BIND the session.

So there are three stages in establishing an LU-LU session:

1. A session initiation request is submitted to VTAM.
2. VTAM notifies the appropriate VTAM program.
3. The VTAM program issues an OPNDST to BIND the session.

This mini-course discusses the third stage, binding the session. Mini-Courses 7 and 8 discuss the other two stages of establishing LU-LU sessions.

BINDing the LU-LU Session

If a session is to be established between two VTAM programs, one of the programs will be the primary logical unit and the other program the secondary logical unit. In a session between a VTAM program and a peripheral logical unit, the VTAM program is the primary logical unit.

Regardless of how a session is initiated, the VTAM program that is acting as a primary logical unit issues an OPNDST macro instruction to bind the session.

To say it another way, the VTAM program that issues the OPNDST macro is the primary logical unit and the logical unit that receives the BIND request is the secondary logical unit.

The OPNDST macro must provide information to VTAM that specifies how VTAM is to handle the OPNDST operation. This information includes the symbolic name of the primary logical unit (VTAM program) and the secondary logical unit. The information for handling the OPNDST operation is provided in three VTAM control blocks:

1. Access method control block (ACB)

2. Request parameter list (RPL)

3. Node initialization block (NIB)

The OPNDST and associated control blocks are shown in Figure 6-1. The OPNDST specifies the appropriate RPL and that RPL, in turn, specifies an appropriate ACB and NIB.

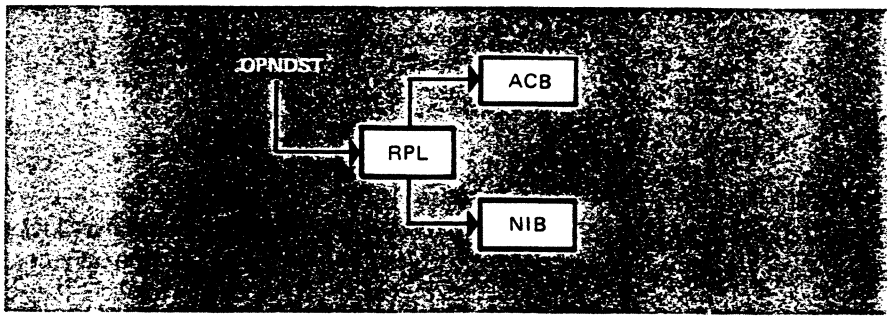

Figure 6-1. OPNDST and Associated Control Blocks

The RPL contents describe the OPNDST operation requirements to VTAM. The ACB identifies the VTAM program to VTAM. The NIB identifies the secondary logical unit to which a BIND request is to be sent.

SNA Request Flow for the OPNDST Operation

Figure 6-2 shows the SNA requests which flow as the result of issuing the OPNDST macro. Study the figure, then read the discussion that follows.

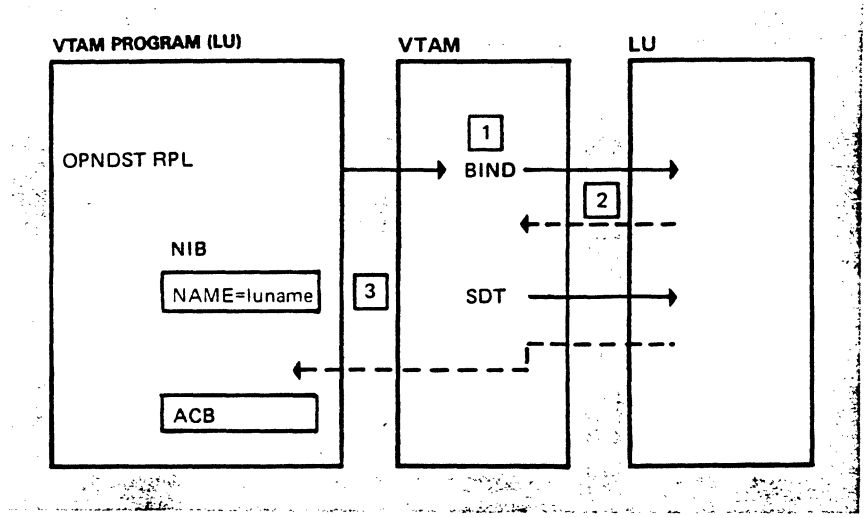

Figure 6-2. OPNDST and the SNA Requests

The OPNDST macro instruction references an RPL, the RPL references an ACB and a NIB. The NAME=operand in the NIB specifies the symbolic name of the logical unit to be connected to the VTAM Program. The OPNDST operation causes the BIND request to flow to the secondary logical unit (LU). The LU-LU session is established when the secondary LU returns the positive response.

The BIND request establishes an LU-LU session in the data traffic reset state. In this state, function management data (FMD) requests and data flow control (DFC) requests are **not** allowed to flow. This means that application data cannot be transmitted between the two session partners. The data traffic reset state is used for recovery situations such as testing and setting request sequence numbers.

FMD and DFC requests can flow when the session is in the data traffic active state. The *start data traffic (SDT)* request is used to change an LU-LU session from the data traffic reset state to the data traffic active state. The SDT request can be sent by the primary logical unit or by VTAM. In Figure 6-2, VTAM sends the SDT request. The VTAM program (the primary logical unit) obtains the responsibility to send the SDT request by including the operand SDT=APPL in the NIB referenced by the OPNDST macro. We will see a need for the VTAM program to send the SDT request when we discuss negotiable BIND requests in Mini-Course 10.

Status of a Logical Unit

The status of a logical unit can affect whether an OPNDST operation can connect the logical unit to the VTAM program. The logical unit may be active or inactive. As you may recall, a logical unit is active when it is in session with VTAM. Otherwise it is inactive. Also, a logical unit that is active is either available or unavailable. A logical unit is unavailable if it is at its session limit. An active logical unit may be connected to a VTAM program if it is available.

Communication Identifier (CID)

You just saw that an OPNDST specifies the symbolic name (not the network address) of the secondary logical unit that is to be connected. VTAM maintains a table of network resource symbolic names and their associated network addresses. Therefore, when a VTAM program issues an OPNDST, VTAM uses the symbolic name provided in the referenced NIB to search its table for the network address of the logical unit. VTAM places the network address in the BIND request to send the BIND to the destination logical unit.

Once the session is established between the VTAM program and the secondary logical unit, both logical units must specify network addresses for all transmissions on the session rather than specifying symbolic names. Each request and response sent by a session partner must include the destination network address and the origin network address. For example, the VTAM program sends a request to the secondary logical unit and that request must include the network address of the VTAM program as well as the network address of the secondary logical unit.
In VTAM, this combination of the origin and destination addresses is called the communications identifier (CID). The CID (origin network address and destination network address) is placed in the transmission header (TH) of each request and response sent by the VTAM program.

The two network addresses are made available to both session partners during the OPNDST (BIND) operation. The secondary logical unit obtains the two network addresses from the transmission header of the BIND request. VTAM makes the CID available to the VTAM program by placing the CID in the associated RPL and NIB at the completion of a successful OPNDST. VTAM places the CID in the ARG field of the RPL and in the CID field of the NIB. In Mini-Course 7 you will see why the CID is placed in two different control blocks.

Since VTAM places the CID in the RPL referenced by the OPNDST macro, that same RPL can be used by VTAM macro instructions that communicate with the logical unit. For example, if the OPNDST references RPL1 at the completion of the OPNDST operation, VTAM places the CID in RPL1. A SEND macro instruction can reference RPL1 to transmit data to the secondary logical unit since RPL1 already contains the CID for the session.

If RPL1 is used by the VTAM program to communicate with more than one LU, RPL1's ARG field must be set to the appropriate CID before a VTAM macro operation communicates with another logical unit. To do this, a VTAM program typically maintains a table of each logical unit's symbolic name and its associated CID.
At the completion of each OPNDST operation, the VTAM program places the VTAM-provided CID in the table with the appropriate symbolic name.

Specifying Session (BIND) Parameters

The BIND request that is sent to a secondary logical unit contains session parameters that specify the protocols (rules) that the two session partners must abide by. There are two sources for session parameters: logmode tables and the VTAM program.

Logmode Tables

Network definitions assign a logmode table to each logical unit that performs the role of a secondary logical unit. One logmode table could be associated with all of the logical units or there could be a different table for each logical unit. A logmode

table contains one or more named sets of session parameters. The logmode table that is associated with a logical unit should contain one or more named sets of session parameters appropriate for use by that logical unit. The name of a set of session parameters is called a logmode name.

A logon issued by a secondary logical unit includes a logmode name that specifies a set of session parameters in the associated logmode table. That set of session parameters is to be included in the BIND request that establishes the LU-LU session. However, the primary logical unit (VTAM program) can override the logmode name in the logon by specifying another set of session parameters to be included in the BIND request.

Session Parameters From a VTAM Program

Session parameters can also be supplied by the VTAM program. These session parameters are specified by the NIB that is used in an open operation.

Selecting Session Parameters for the BIND Request

Figure 6-3 illustrates the process for selecting session parameters. Study the figure, then read the discussion that follows.

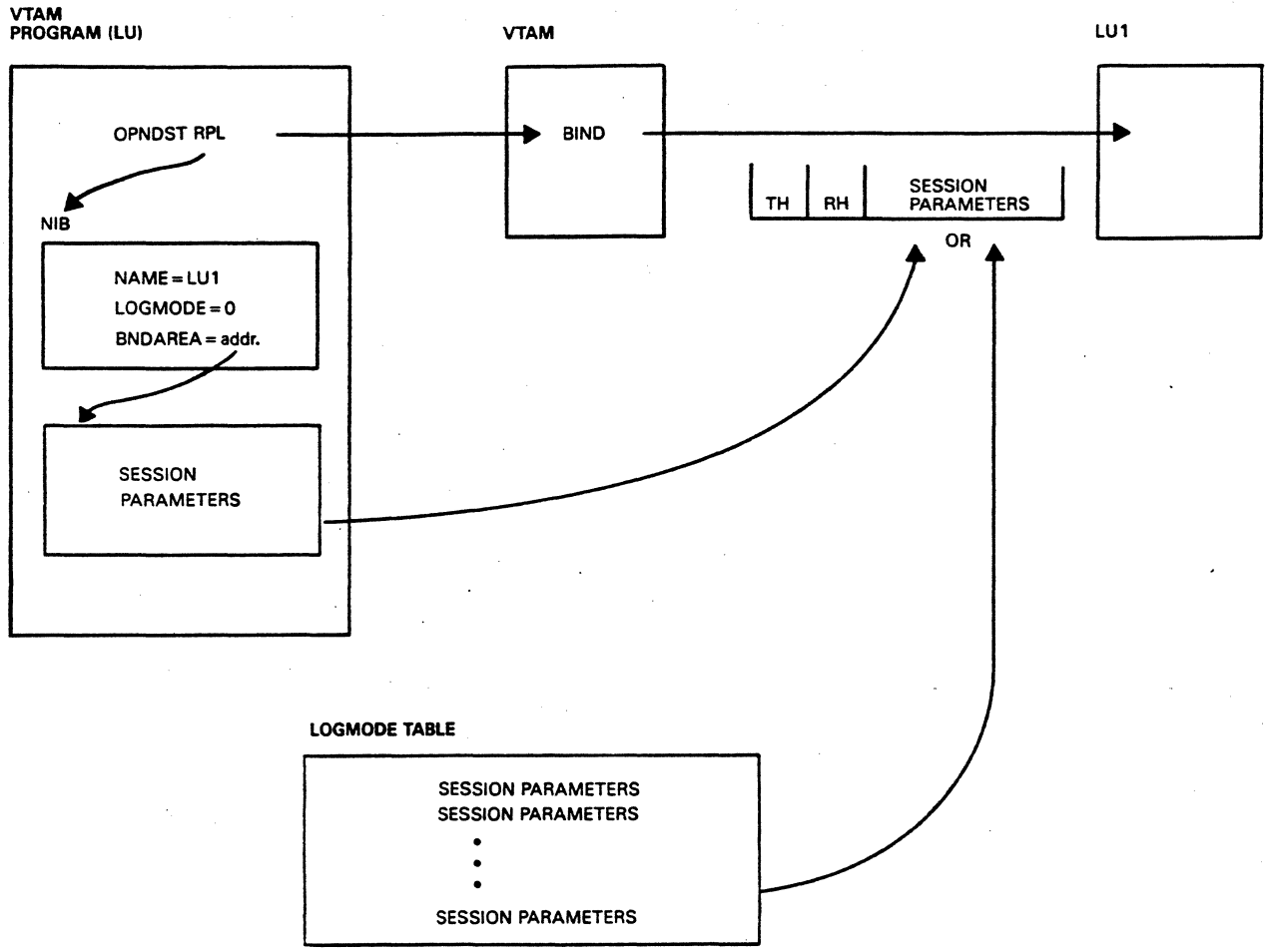

Figure 6-3. Selecting Session Parameters

Assume that the VTAM program initiates a session by issuing an OPNDST request. Two fields in the NIB referenced by the OPNDST macro specify the session parameters to be included in the BIND request:

1. LOGMODE
2. BNDAREA

Both fields are initialized by the NIB keyword operands, LOGMODE= and BNDAREA=. The setting of these fields determine the session parameters selected.

Case 1:
```
LOGMODE=0
BNDAREA=0
```

When both fields are set to 0, the first set of session parameters in the logmode table associated with the secondary logical unit is included in the BIND request.

Case 2:
```
LOGMODE=logmode name
BNDAREA=0
```

The set of session parameters named in the LOGMODE operand is obtained from the logmode table associated with the secondary logical unit.

Case 3:
```
LOGMODE=0
BNDAREA=address
```

The set of session parameters at the address specified by the BNDAREA operand is included in the BIND request.

At this point, you should understand that a VTAM program issues an OPNDST macro instruction to connect a secondary logical unit to that program. The OPNDST must reference an RPL, and the RPL must specify the appropriate ACB and NIB. The VTAM program is responsible for placing the required information in the control blocks before the OPNDST is issued. Also, VTAM places the CID (network addresses of the VTAM program and the secondary logical unit) in the RPL and in the NIB at the completion of the OPNDST operation. The CID must be used in any subsequent communication with the logical unit.

Please turn to Mini-Course 6 in your Personal Reference Guide and do Exercise 6.1.

VTAM Concepts

Mini-Course 7

Acquiring Peripheral Logical Units

MINI-COURSE 7. Acquiring Peripheral Logical Units

Introduction

A VTAM program may be authorized to acquire logical units. Authorization is granted by including the appropriate parameter in the APPL definition statement used to define the program. Acquiring means that the VTAM program takes the initial action to establish a connection with a logical unit. Without any prompting from the secondary logical unit, the program issues an OPNDST macro instruction to BIND the session.

Acquiring is suitable for VTAM programs that require access to a specific resource or resources in order to function. A printer is an example of a resource that might be required for a program to function.

A VTAM program can acquire a single logical unit or multiple logical units with an OPNDST macro instruction. The acquisition of a single logical unit is discussed first. The discussion is based on the system that was generated earlier, shown in Figure 7-1 . Study the figure to become familiar with the current status of the network.

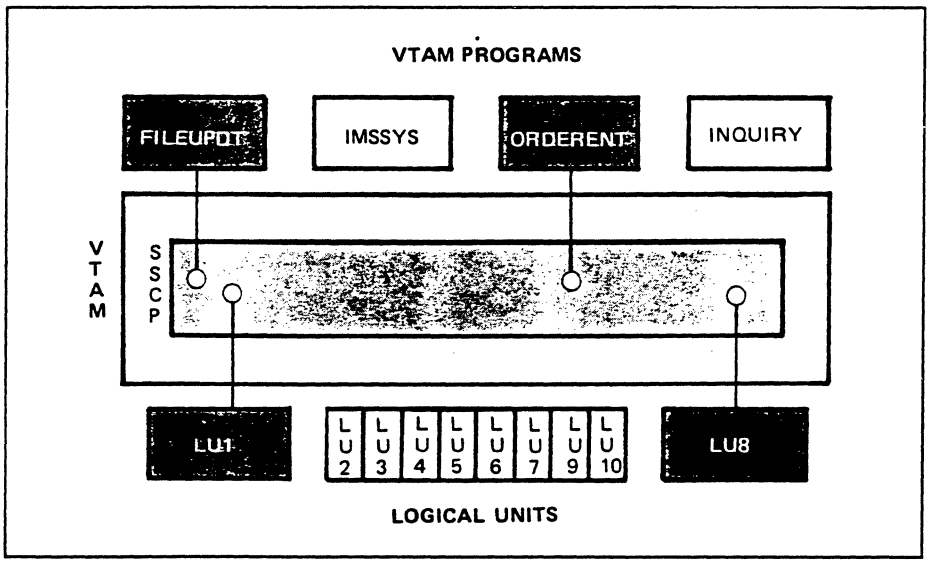

Figure 7-1. VTAM Configuration

The figure shows that the two VTAM programs, ORDERENT and FILEUPDT, are active because both have been identified to VTAM. Two peripheral logical units, LU1 and LU8, are active, that is, both are in session with VTAM. Also, LU1 and LU8 are available because neither LU is in session or queued for a session with a VTAM program.

Acquiring a Single Logical Unit

Assume that ORDERENT is authorized to acquire logical units:

```
ORDERENT  APPL  AUTH=ACQ
```

For this example, ORDERENT acquires logical unit LU1 by issuing the OPNDST macro, as illustrated in Figure 7-2 . Study the figure and then read the discussion that follows.

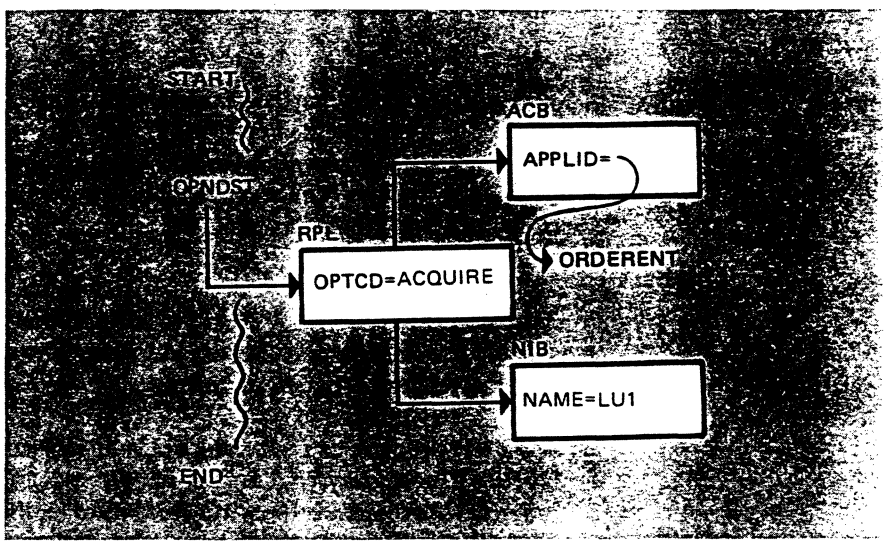

Figure 7-2. Acquiring a Single Logical Unit

The OPNDST macro instruction references an RPL, and the RPL references an appropriate ACB and NIB. The RPL operand OPTCD specifies the value ACQUIRE which informs VTAM that this OPNDST operation is to acquire a logical unit. VTAM utilizes the definition of ORDERENT to verify that the program is authorized to do an acquire. This OPNDST is referred to as an OPNDST ACQUIRE.

The ACB identifies the VTAM program ORDERENT to VTAM. The NAME operand in the NIB specifies that LU1 is the logical unit with which ORDERENT is requesting a session.

VTAM processes the OPNDST request and the ORDERENT-LU1 session is established. VTAM places the CID for this session in the associated RPL and NIB. VTAM then notifies ORDERENT that the operation is complete and now ORDERENT may communicate with LU1.

Acquiring Multiple Logical Units

A single OPNDST macro may be coded to acquire multiple logical units. This is accomplished by an OPNDST that references multiple NIBs, called a NIB list. Each NIB identifies and describes a single logical unit. Figure 7-3 illustrates the use of a NIB list.

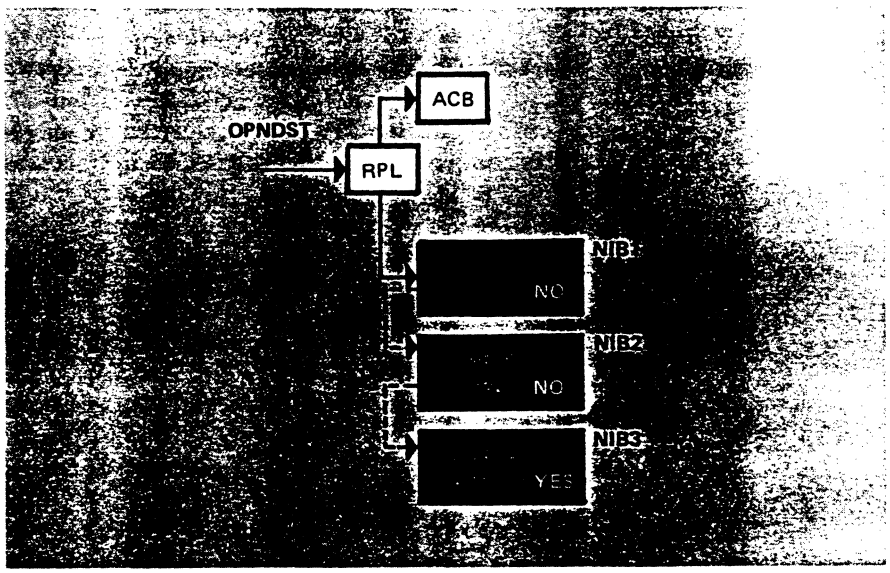

Figure 7-3. NIB List

The NIBs of a NIB list must be coded one after the other. The last NIB in the list must have the LISTEND operand set to YES, and each NIB before the last one must have the LISTEND operand set to NO, as shown in Figure 7-3.

The VTAM program has two options as to how many sessions will be established when an OPNDST references a NIB list: **any** and **all**. The OPTCD operand in the RPL is coded to specify the desired option:

```
OPTCD=CONANY or OPTCD=CONALL
```

Specifying OPTCD=CONANY allows one session to be established. VTAM scans the NIB list and a session is established with the first available logical unit; the rest of the list is not scanned. Referring to Figure 7-3, you see that VTAM will try to establish a session between the VTAM program (ORDERENT) and LU1. If LU1 is not available, LU2 is tried next and if LU2 is not available, VTAM tries LU3. The OPNDST will complete without a session being established if none of the logical units specified in the NIB list are available.

Specifying OPTCD=CONALL means that sessions are to be established with all available logical units specified in the NIB list. The OPNDST in Figure 7-3 could establish sessions with a maximum of three logical units (LU1, LU2, and LU3) if the RPL includes OPTCD=CONALL. How can ORDERENT determine which sessions were established? VTAM sets the NIB connection status field to YES (CON=YES) when a session is established with the specified LU. The VTAM program must test each NIB to determine if a session was established with the specified LU.

VTAM places the CID of each logical unit in the appropriate NIB when the VTAM program establishes a session with that LU. The VTAM program must save each CID for subsequent communication with each logical unit. The ARG field of the OPNDST RPL is not valid at the completion of the operation when a NIB list is used; therefore, the VTAM program will have to extract the CIDs from the NIBs and store them for future use.

At this point, you may see a reason why VTAM places the CID in both the RPL and and in the NIB. The CID is in the RPL after a session is established with a single logical unit. The program can communicate with the logical unit, using that RPL, without having to load the CID for that session into the RPL. On the other hand, the RPL can hold only one CID. Therefore, provisions must be made for saving CIDs when sessions are established with multiple logical units through the use of NIB lists. To accomplish this, VTAM places CIDs in appropriate NIBs.

OPNDST ACQUIRE Examples

Now you will study two OPNDST ACQUIRE examples and determine the LU-LU sessions that are established in each case. First, you may want to refresh your memory as to the initial status of the VTAM network by studying Figure 7-1. Next, study the first OPNDST ACQUIRE example, shown in Figure 7-4. Determine which LU-LU sessions will be established and then read the discussion that follows.

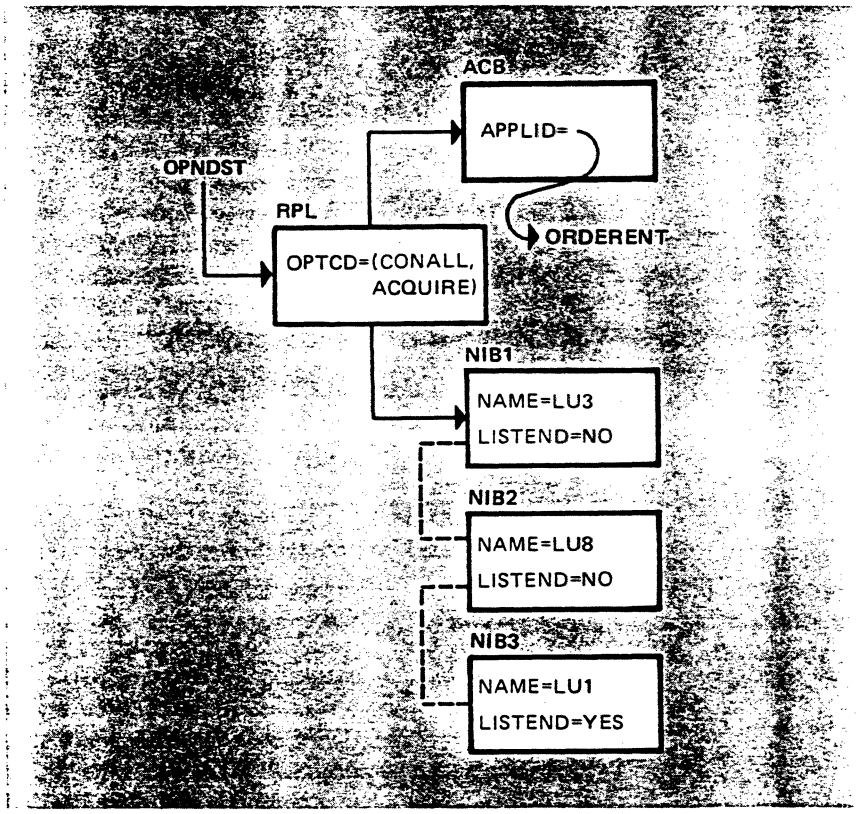

Figure 7-4. OPNDST ACQUIRE Example One

The OPNDST in Figure 7-4 specifies that sessions are to be established with all available logical units (OPTCD=CONALL) specified in the NIB list. Sessions are established with LU1 and LU8. A session is not established with LU3 because it isn't available, as you can see from Figure 7-1 . VTAM sets the connect field (CON) in NIB2 and in NIB3 to YES, to indicate that LU1 and LU8 are in session with ORDERENT. Also, VTAM places appropriate CIDs in NIB2 and in NIB3.

A second OPNDST ACQUIRE example is shown in Figure 7-5. For this example, assume that the status of the VTAM network is as shown in Figure 7-1. That is, ORDERENT, FILEUPDT, LU1, and LU8 are in session with VTAM but no

LU-LU sessions exist. Study the OPNDST in Figure 7-5 and determine which sessions will be established, then read the discussion that follows.

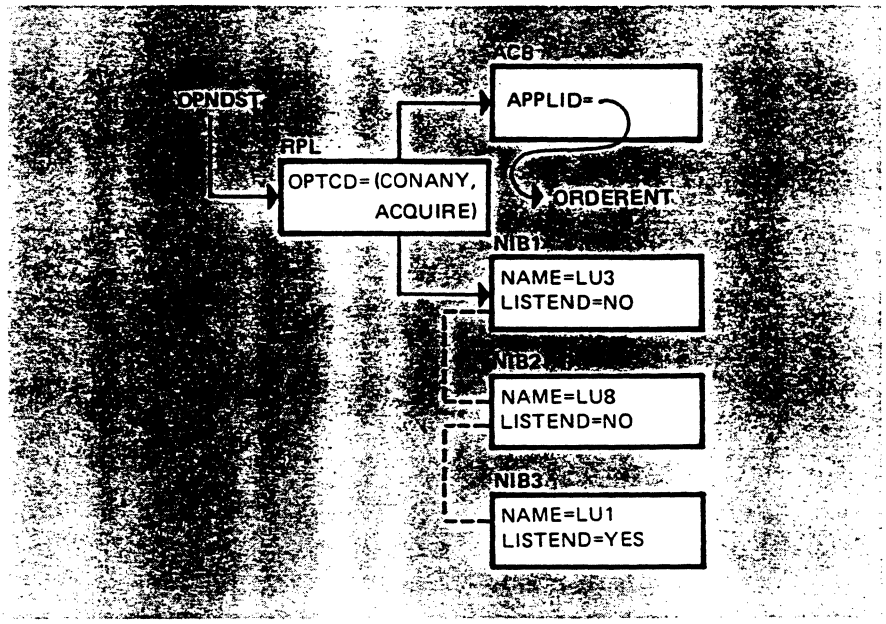

Figure 7-5. OPNDST ACQUIRE Example Two

OPTCD=CONANY says that only one session is to be established by this OPNDST. VTAM scans the NIB list and finds that the first available logical unit is LU8. The ORDERENT-LU8 session is established, VTAM sets NIB2's connect field to YES (CON=YES), and VTAM places the CID in NIB2's CID field and in the RPL's ARG field. VTAM then notifies ORDERENT that the OPNDST operation is complete.

What are the significant points about acquiring logical units? A VTAM program must be authorized to issue an OPNDST ACQUIRE macro instruction. Authorization is provided by coding the appropriate value in the program's definition statement. A VTAM program may acquire one or multiple logical units with an OPNDST macro instruction. The OPNDST references a single NIB in order to acquire one logical unit. The OPNDST references a NIB list in order to acquire multiple logical units. All NIBs used in an acquire operation must specify the name of a logical unit.

Sessions are established with available logical units. VTAM places the CID of a logical unit in the associated RPL and NIB when a session is established with that LU. The RPL's ARG field does not contain a valid CID when sessions are established with multiple logical units by a single OPNDST operation.

Please turn to Mini-Course 7 in your Personal Reference Guide and do Exercise 7.1.

VTAM Concepts

Mini-Course 8

Accepting Logons from Peripheral Logical Units

MINI-COURSE 8. Accepting Logons From Peripheral Logical Units

Introduction

A logon is submitted to VTAM by a secondary logical unit or on behalf of a secondary logical unit. A logon is a request for a session to be established between the submitting logical unit and a specific VTAM program. VTAM, upon receipt of the logon, notifies the specified VTAM program of the logon request. It is up to the VTAM program to issue an OPNDST macro to BIND the session.

An OPNDST with OPTCD=ACCEPT specified in the referenced RPL requests VTAM to process a logon, that is, send a BIND to the logical unit that makes the request. So we have an OPNDST ACCEPT to accept a logon and BIND the session, and we have an OPNDST ACQUIRE to BIND a session with a logical unit that did not submit a logon.

We will examine three aspects of logon requests:

- Generating the logon
- Logon contents
- How a logon is processed by a VTAM program

Generation of the Logon by a Logical Unit

With respect to format, there are two types of logons: *field-formatted* or *character-coded.* A logical unit generates a logon in one of these two formats and submits it to VTAM. Usually, programmable logical units generate field-formatted logons and non-programmable logical units generate character-coded logons.

Field-Formatted Logon

A field-formatted logon is an initiate-self (INIT-SELF) request. An INIT-SELF request is an SNA command that has a specific format. Logical units that submit field-formatted logons generate the INIT-SELF request for transmission to VTAM in a path information unit (PIU). Figure 8-1 illustrates a field-formatted logon in a PIU.

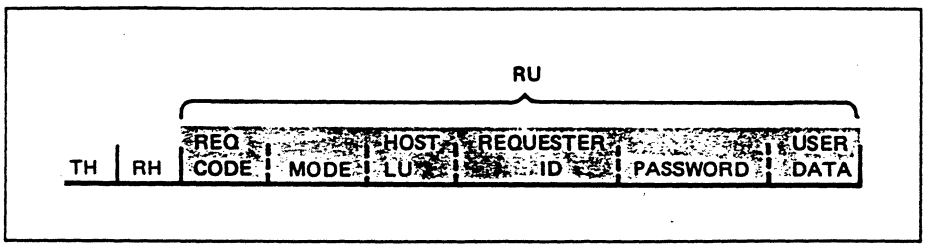

Figure 8-1. Field-Formatted Logon

The request unit (RU) portion of the PIU contains the logon. Note the following:

- The request code field contains the SNA command INITIATE SELF.
- The mode field contains a logmode name that identifies a set of session parameters for this session. The VTAM program and the logical unit, when connected, must abide by the protocols (rules) specified by the session parameters.
- The host LU field contains the symbolic name of the VTAM program with which the logical unit is requesting a session.
- The requester ID field contains the symbolic name of the logical unit submitting the logon.
- The password field can contain a password that can be validated by the VTAM program.
- The last field may contain user data.

VTAM can process a field-formatted logon in its existing form. Figure 8-2 shows a field-formatted logon being sent to VTAM. Study the figure and then continue.

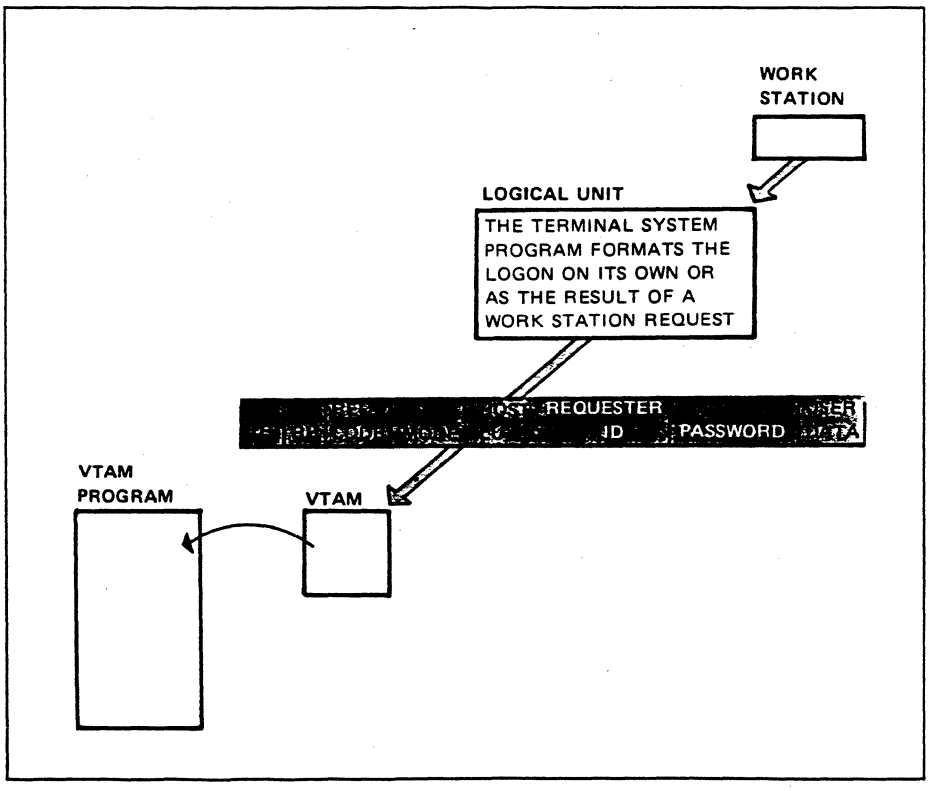

Figure 8-2. Sending a Field-Formatted Logon to VTAM

The logical unit creates and submits the logon either on its own or as the result of a prompt from a work station. VTAM, upon receipt of the logon, notifies the target VTAM program about the logon. Certain fields of the logon (requester ID, password and user data, and the session parameters specified by the mode field) can be accessed by the VTAM program. The program can use the logon message,

password, and requesting logical unit name to verify authorization for the session with the logical unit. The VTAM program may access the session parameters specified in the logon for the sake of determining if they're acceptable and to modify them if they are not acceptable. If the program modifies the session parameters, then the OPNDST must be set up to use the modified parameters rather than the session parameters referenced in the logon.

Character-Coded Logon

A typical logical unit that submits a character-coded logon is a terminal logical unit that is associated with an operator. The character-coded logon is user-defined and meaningful to the operator that enters the logon.

A character-coded logon can be as simple as the symbolic name of the VTAM program with which the submitting logical unit wishes a session with. For example, suppose the operator at a terminal logical unit wishes to establish a session with the CICS logical unit and that CICS's symbolic name is defined as **CICS**. The operator enters:

```
CICS
```

The terminal logical unit builds a PIU containing CICS and sends it to VTAM. The character-coded logon must be converted to a field-formatted logon before VTAM can process it. VTAM has two routines to process each type of logon:

1. The formatted systems services (FSS) routine processes field-formatted logons.

2. The unformatted systems services (USS) routine processes character-coded logons. It converts the character-coded logon to the field-formatted logon format and submits it to the FSS routine for further processing. A table, called an unformatted systems services (USS) table, must be available to the routine in order for it to make the conversion. USS tables must be defined and included in a VTAM library.

You may want logical units to provide the appropriate password in the logon:

```
PROGA X2634
```

PROGA is the symbolic name of the VTAM program and X2634 is the password assigned to PROGA.

It may be desirable to set up standard logon formats for your establishment. Appropriate USS tables must be defined so that VTAM can convert the character-coded logons to *field-formatted logons.*

Handling a Logon Submitted by a Logical Unit

We have examined the generation and submission of the various types of logons by a logical unit. Now let's see how a VTAM program handles a logon.

A VTAM program can accept a logon in the mainline code, or a LOGON exit routine can be included to accept logons. Typically, LOGON exit routines are used to accept logons. Regardless of whether a LOGON exit routine is included, you should keep in mind that a session is established between a VTAM program and a logical unit by an OPNDST macro instruction. We will now describe an OPNDST that is used to establish a session as the result of a logon.

An OPNDST must be coded specifically to accept a logon and BIND the session. The OPNDST must reference an RPL that includes OPTCD=ACCEPT. Furthermore, an OPNDST ACCEPT can be coded to accept a logon from a specific logical unit, or it can be coded to accept a logon from any logical unit.

We include OPTCD=(ACCEPT,ANY) in the RPL in order to accept logons from any logical unit. An OPNDST ACCEPT ANY will be completed by any logon that is directed to this VTAM program. VTAM places the name of the logical unit in the referenced NIB at the completion of the OPNDST operation. Also, VTAM places the CID of the logical unit in the RPL and and in the NIB.

We include OPTCD=(ACCEPT,SPEC) in the RPL referenced by the OPNDST to accept a logon from a specific logical unit. Also, we must include in the referenced NIB the name of the logical unit to be connected. An OPNDST ACCEPT SPEC will accept a logon only from the logical unit named in the referenced NIB. Figure 8-3 illustrates an OPNDST that accepts a logon from a specific logical unit.

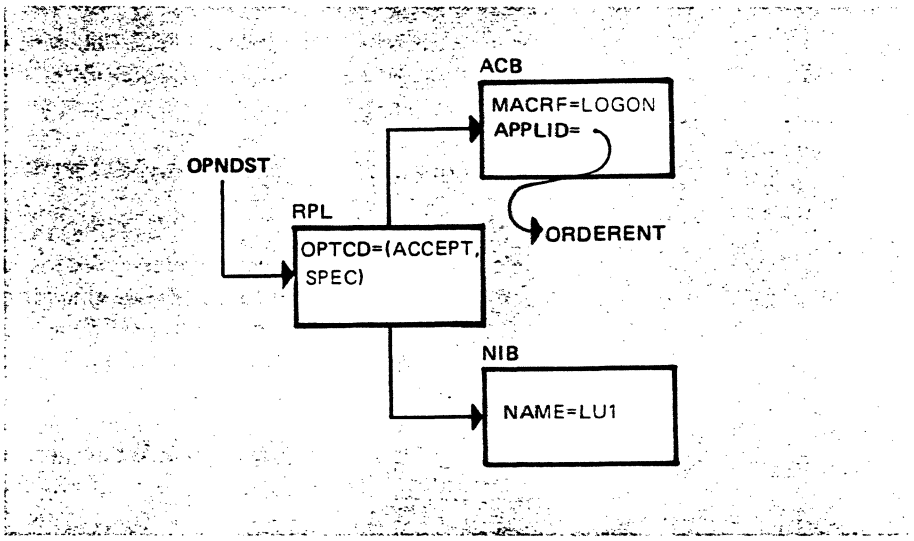

Figure 8-3. Accepting a Logon From a Specific Logical Unit

The ACB specifies that ORDERENT will accept logons. The RPL specifies that the OPNDST will accept a logon from a specific logical unit and the NAME operand in the NIB provides the symbolic name of the logical unit. The OPNDST will be completed when a logon is received from LU1.

Figure 8-4 illustrates an OPNDST that will accept logons from any logical unit.

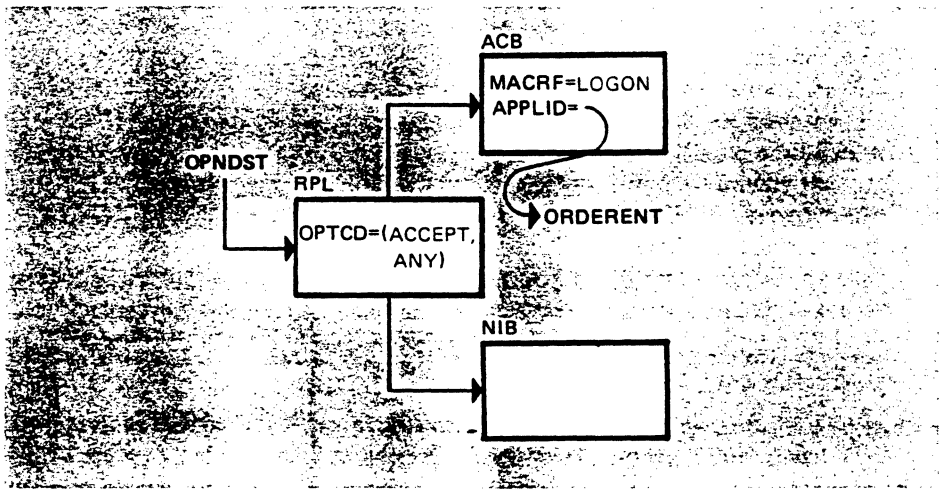

Figure 8-4. Accepting a Logon From Any Logical Unit

The referenced RPL includes OPTCD=(ACCEPT,ANY), which means that the OPNDST can be completed by a logon from any logical unit. This type of OPNDST is referred to as an OPNDST ACCEPT ANY. The NIB does not include a logical unit name. Even if it did, VTAM would not use the name, because this is an ACCEPT ANY.

At the completion of the OPNDST, VTAM places the CID of the session in the referenced RPL and in the referenced NIB. Also, VTAM places the symbolic name of the connected logical unit in the NIB. If the NAME operand of the NIB already contained a logical unit name, VTAM overlays it with the name of the logical unit with which the session was established.

Logon Exit Routine

This describes how a LOGON exit routine is used to handle logons. A LOGON exit routine is an asynchronous routine; that is, VTAM schedules the routine to get control when VTAM receives a logon for the program. Once the logon exit routine receives control, it processes the logon and then returns control to VTAM. At that time, VTAM may give control to the mainline code or to another exit routine.

A session is not established at the point in time when the routine gets control. The routine has the opportunity to obtain and examine the logical unit name, password, user data, and specified session parameters identified in the logon. The routine determines whether the logical unit has the authority to connect to this program. Assuming that the logical unit is authorized for a session with the program, the routine can either establish the LU-LU session or it can provide the necessary information to the mainline code to establish the session. Figure 8-5 illustrates one way that the LOGON exit routine can be used to process a logon. Study the figure and then read the discussion that follows.

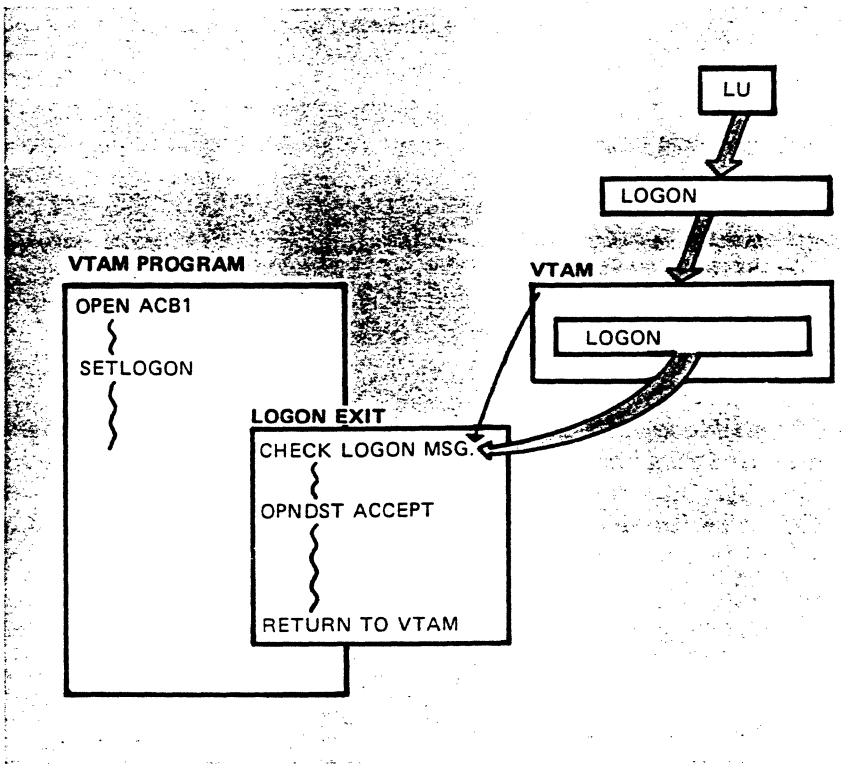

Figure 8-5. LOGON Exit Routing Usage

VTAM cannot schedule the LOGON exit routine until it has been activated. The mainline code of the VTAM program issues the VTAM macro SETLOGON to activate the LOGON exit routine. This allows the VTAM program to prepare itself before activating the LOGON exit routine to accept logons.

The figure shows a logical unit submitting a logon. The logon is routed to VTAM. VTAM examines the logon and determines which VTAM program that the logon request is for and schedules that program's LOGON exit routine. Once the exit routine gets control, it obtains and examines the logon message to determine if the logical unit is authorized for a session with this VTAM program. The logon message could consist of a password to establish the logical unit's authority.

Some VTAM programs, such as CICS and IMS, include in a table the symbolic names of all logical units that are authorized for a session. The program obtains the symbolic name of the requesting logical unit from the logon, compares it with names in its table, and processes the logon if the name is found in the table.

Assuming that the logical unit is authorized for a session, the exit routine shown in Figure 8-4 issues an OPNDST ACCEPT to BIND the session. When all processing is complete, the exit routine returns control to VTAM. VTAM can then return control to mainline code or to an exit routine.

Assume that the logical unit is not authorized for a session with the specified VTAM program. Rather than issuing an OPNDST macro, the LOGON exit routine issues a CLSDST macro to terminate the logon request.

The OPNDST does not have to be issued in the exit routine. The routine could verify authorization of the logical unit and let mainline code issue the OPNDST to establish the session. In that case, the exit routine must provide the logical unit

name to mainline code before the routine returns to VTAM. When mainline gets control, it can issue an OPNDST ACCEPT to establish the session.

You should realize that a LOGON exit routine has the opportunity to determine the authorization of a requesting logical unit before a session is established with the logical unit. If a LOGON exit routine is not included, connections must be established from mainline code. However, mainline code does not have the opportunity to determine the authorization of a logical unit to form a session with the VTAM program. Mainline code issues an OPNDST and eventually the OPNDST is completed, meaning that a session is established. The program can then determine if the logical unit should have been connected. If it should not have been connected, a CLSDST is issued to disconnect the logical unit. A LOGON exit routine can be more efficient when there is a lot of logon and logoff activity.

Other Logon Types

The previous discussion of logons was oriented toward logons issued by secondary logical units. Most of that discussion applies to the following logon types:

- Logon submitted by the network operator
- Automatic logon
- SIMLOGON by a VTAM program
- A VTAM program passing a logical unit to another VTAM program

The following discussion will point out how these logon types differ from those issued by secondary logical units.

Network Operator

The network operator can enter a VARY command that requests VTAM to issue a logon on behalf of a specified logical unit. The VARY command also specifies the name of the VTAM program to which the logon is to be sent. Once the logon is submitted, the remainder of the logon operation is the same as previously discussed.

Automatic Logon

A logical unit may be defined to connect to a specific VTAM program. This is accomplished by including "LOGAPPL=VTAM program name" in the logical unit definition. This indicates to VTAM that the logical unit is to be connected to the specified VTAM program. VTAM will automatically submit a logon on behalf of that logical unit any time that it is available. The rest of the operation is the same as before.

SIMLOGON

VTAM provides the facility for a VTAM program to initiate a logon. The program can do this by issuing a SIMLOGON macro instruction, which requests VTAM to issue a logon on behalf of the specified logical unit. The effect of the simulated logon is to schedule the program's LOGON exit when the specified logical unit is available. A subsequent OPNDST ACCEPT establishes the session.

SIMLOGON is handy for a VTAM program that is designed for logons; that is, the program doesn't acquire logical units. On occasion, the program may need to

initiate a session with a logical unit. So rather than including acquire coding in the program, a SIMLOGON is used. This can save additional programming.

CLSDST PASS

A VTAM program uses the CLSDST macro instruction to disconnect a logical unit. Also, the CLSDST can be coded to cause the logical unit to be passed to another VTAM program. This is an option and is implemented by setting the PASS option code in the CLSDST's RPL.

```
CLSDST RPL=RPL1,OPTCD=PASS
```

A program must be authorized in its definition statement to use this option.

For a CLSDST PASS operation, VTAM first disconnects the logical unit and then generates a logon for the logical unit. The VTAM program that issues the CLSDST PASS must indicate in the CLSDST's RPL the name of the VTAM program to which the logon is to be sent.

Queueing Logons to a VTAM Program

There are occasions when VTAM queues logons to a VTAM program to be processed later. If a VTAM program is executing, is in session with VTAM, and is defined to accept logons, VTAM will direct logons to the program. If the VTAM program has not issued an OPNDST ACCEPT and has not activated a LOGON exit routine, VTAM queues any logons for that program.

Assume that the VTAM program does not have a LOGON exit routine. Then OPNDST ACCEPTs must be issued in mainline code to process logons. A VTAM program can issue OPNDST ACCEPTs to be completed when a logon arrives for the program. As logons arrive for the VTAM program, an OPNDST will process the logon. If logons are queued to the VTAM program while an OPNDST ACCEPT is not issued, the next OPNDST ACCEPT that is issued will accept the first logon that was queued.

Assume that the VTAM program has a LOGON exit routine. VTAM queues logons to the VTAM program if the exit routine has not been activated. Once the exit routine is activated, the first logon in the queue drives the exit. VTAM also queues logons if the exit routine does not process the logons at the rate that they arrive.

Please turn to Mini-Course 8 in your Personal Reference Guide and do Exercise 8.1.

VTAM Concepts

Mini-Course 9

Terminating Sessions Between VTAM Programs and Peripheral Logical Units

MINI-COURSE 9. Terminating Sessions Between VTAM Programs and Peripheral Logical Units

Introduction

An LU-LU session may be terminated because communication between the two logical units has completed or because of some error condition. The termination may be an orderly shutdown of the session or it may be an immediate disconnection.

Orderly Session Termination: Orderly shutdown of a session is the typical way to terminate an LU-to-LU session. An orderly shutdown of a session can be initiated by the secondary logical unit or by the primary logical unit (VTAM program). A VTAM program that acquires a logical unit may make the decision that communication is complete. In that case, the program will initiate disconnection. On the other hand, a logical unit that issued a logon for connection may be more qualified to decide when communication is complete. In this case, the secondary logical unit will initiate disconnection.

An orderly shutdown consists of a dialogue between the VTAM program and the secondary logical unit to ensure that both resources are ready to terminate the session. The dialogue ensures that there are no outstanding requests to be processed.

Regardless of which logical unit initiates session termination, the VTAM program actually terminates the session by issuing a CLSDST macro instruction. The CLSDST macro causes VTAM to send an UNBIND request to the VTAM program's session partner and the session is terminated.

Immediate Session Termination: For immediate session termination, there is no dialogue between the two session partners. One of the session partners sends a request to VTAM for immediate session termination. The session may be terminated immediately by VTAM or by the primary logical unit (VTAM program). We'll discuss both cases.

The way each session partner may terminate a session will be described first for the case in which the primary logical unit initiates session termination. Then we will look at the case in which the secondary logical unit initiates the termination.

Session Termination by the VTAM Program

A VTAM program can request assistance from VTAM to terminate a session with another logical unit or the VTAM program can negotiate with the session partner for session termination. Figure 9-1 shows two ways that the VTAM program can initiate the process to terminate an LU-LU session.

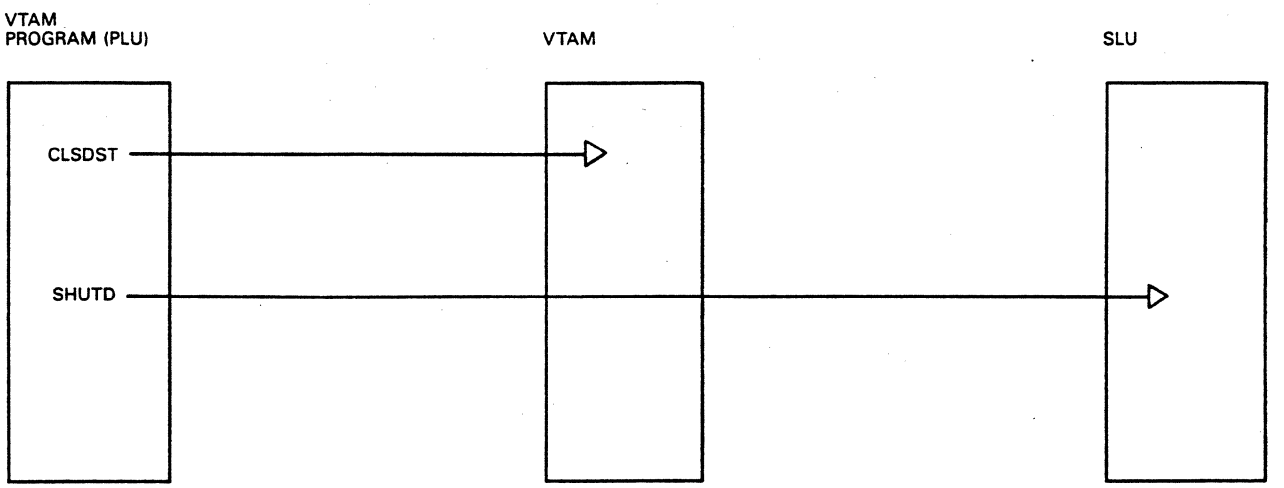

Figure 9-1. VTAM Program Terminates LU-LU Sessions

The VTAM program can issue a CLSDST macro to request VTAM to terminate a session with a specific logical unit and this is called an immediate termination. The session is terminated even though the secondary logical unit may not be ready to terminate the session. A VTAM program might use immediate termination when it has acquired a logical unit, such as a printer. In this case the VTAM program knows when it has sent a complete report to be printed. The VTAM program can terminate the session immediately without destroying any data.

Except for the above situation, the VTAM program normally uses an orderly session termination process. That is, the VTAM program can negotiate with a session partner for session termination. It does this by issuing a SEND macro to transmit a shutdown (SHUTD) request to its session partner. SHUTD is an SNA command.

Submitting Session Termination Request to VTAM

Figure 9-2 illustrates the command flow for immediate session termination.

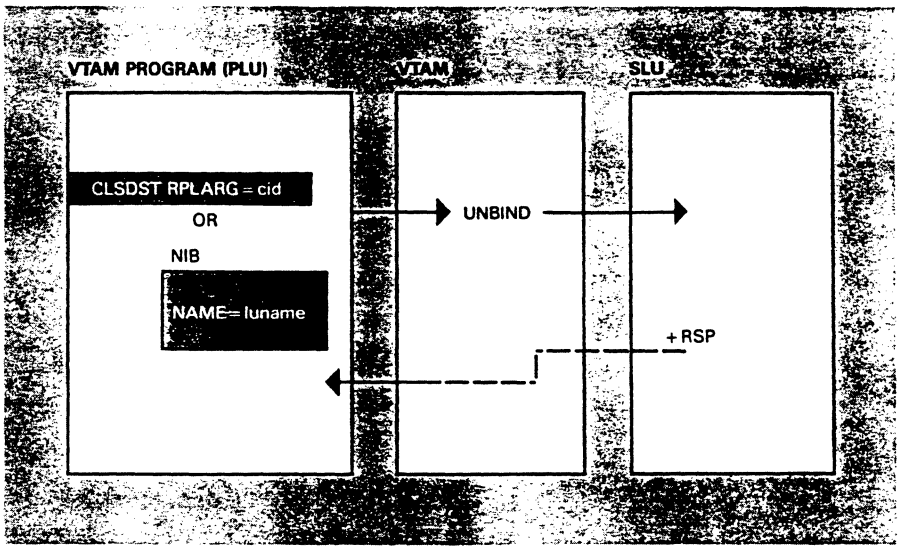

Figure 9-2. Submitting Session Termination Request to VTAM

The VTAM program issues a CLSDST macro requesting VTAM to terminate a session. The CLSDST macro references an RPL and the RPL specifies the appropriate ACB. There are two ways to identify the secondary logical unit to be disconnected. The CLSDST macro can identify the logical unit to VTAM either by placing the appropriate CID in the ARG field of the RPL, or by specifying a NIB that contains the symbolic name of the logical unit.

The CLSDST prompts VTAM to send the UNBIND request to the secondary logical unit, thereby terminating the session. The secondary logical unit returns a positive response to VTAM for the UNBIND, and VTAM notifies the VTAM program that the CLSDST operation is complete.

Submitting Session Termination Request to the Secondary Logical Unit

A VTAM program uses orderly session termination when a network or system operator submits a command to terminate execution of the VTAM program or to terminate execution of the operating system under which the VTAM program is running. The VTAM program wants to terminate all of its sessions at logical points, that is, at the completion of a transaction. Figure 9-3 illustrates the request flow when the VTAM program initiates orderly session termination with a specific peripheral logical unit.

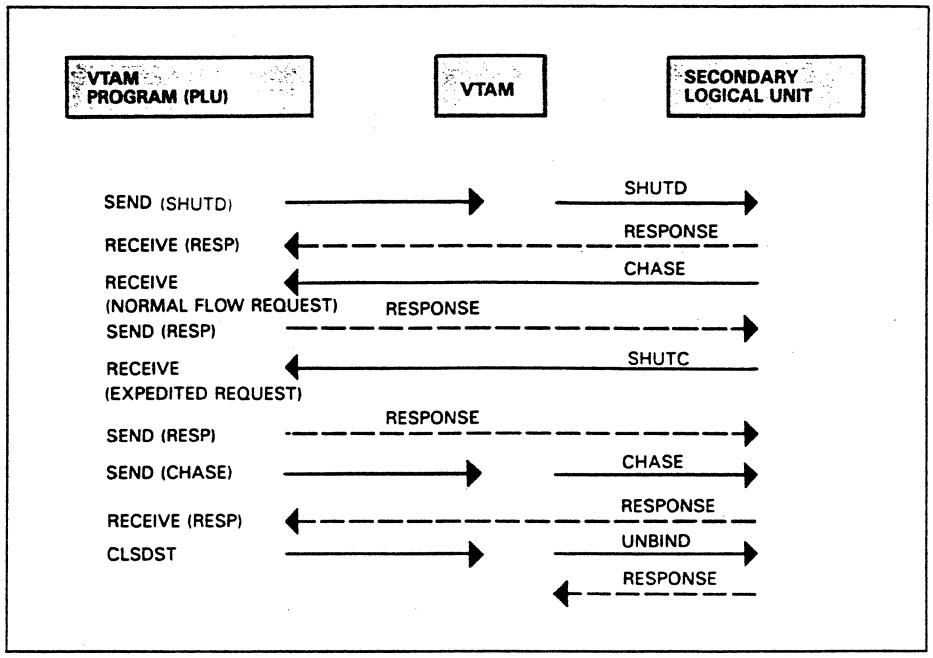

Figure 9-3. Submitting Session Termination Request to the SLU

The VTAM program issues a SEND macro that specifies that the SNA shutdown (SHUTD) command is to be sent to the logical unit that is specified in the the referenced RPL. The SHUTD request travels on the expedited-flow and notifies the receiving logical unit that the session is to be terminated at the completion of the current transaction. At the completion of the current transaction, the secondary logical unit must send a shutdown complete (SHUTC) request to the VTAM program signifying readiness for session termination. The sending of the SHUTC request is a promise by the secondary logical unit that it will not initiate any more transactions or data traffic.

Before sending the SHUTC request, the secondary logical unit may send a CHASE request to the VTAM program to verify that it has no outstanding requests on the session. The CHASE request travels on the normal-flow and the VTAM program accepts the request with a RECEIVE macro that accepts normal-flow requests. The VTAM program returns a response for the CHASE request only after responses to all other requests have been returned. The secondary logical unit knows that there are no outstanding requests when it receives the response for the CHASE request. Typically, non-programmable logical units don't have the capability to use the CHASE request.

The SHUTC request travels on the expedited-flow and can be accepted in the VTAM program either by a RECEIVE macro that is coded to accept an expedited request or by a DFASY exit routine. The VTAM program in Figure 9-3 uses a RECEIVE macro.

The VTAM program, once it receives the SHUTC request, has the option of using the CHASE request to determine if it has any outstanding requests. Once the response is received for the CHASE request, the VTAM program issues a CLSDST macro to send the UNBIND request to the secondary logical unit, terminating the session.

Session Termination by the Secondary Logical Unit

Like the primary logical unit (VTAM program), the secondary logical unit can initiate the process for terminating the LU-LU session. Figure 9-4 illustrates the ways that the secondary logical unit asks for session termination.

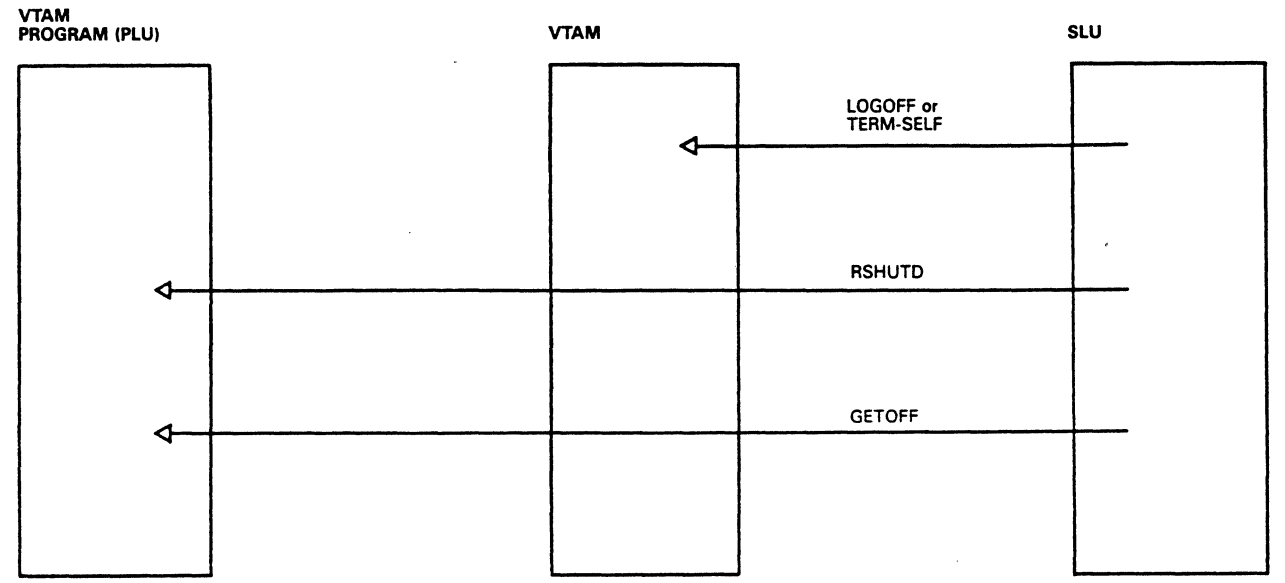

Figure 9-4. Peripheral Logical Unit Initiates Session Termination

The secondary logical unit can request VTAM's assistance to terminate the LU-LU session, or it can negotiate with the primary logical unit (VTAM program) for orderly session termination. The secondary logical unit transmits a user-defined logoff or a terminate-self (TERM-SELF) request to VTAM to get VTAM's assistance in performing immediate session termination.

The secondary logical unit can start an orderly session closedown by sending a request containing either a request shutdown (RSHUTD) or a user-defined character string to the VTAM program.

A secondary logical unit that initiates all transactions might use immediate session termination. For example, a terminal operator may enter a part number to find out how many items are in stock. Once the reply is received from the VTAM program, the operator submits a logoff for immediate termination of the session. There are no more transactions to be processed at this time.

Submitting Session Termination Request to VTAM

Figure 9-5 illustrates request flow when the secondary logical unit submits a session termination request to VTAM.

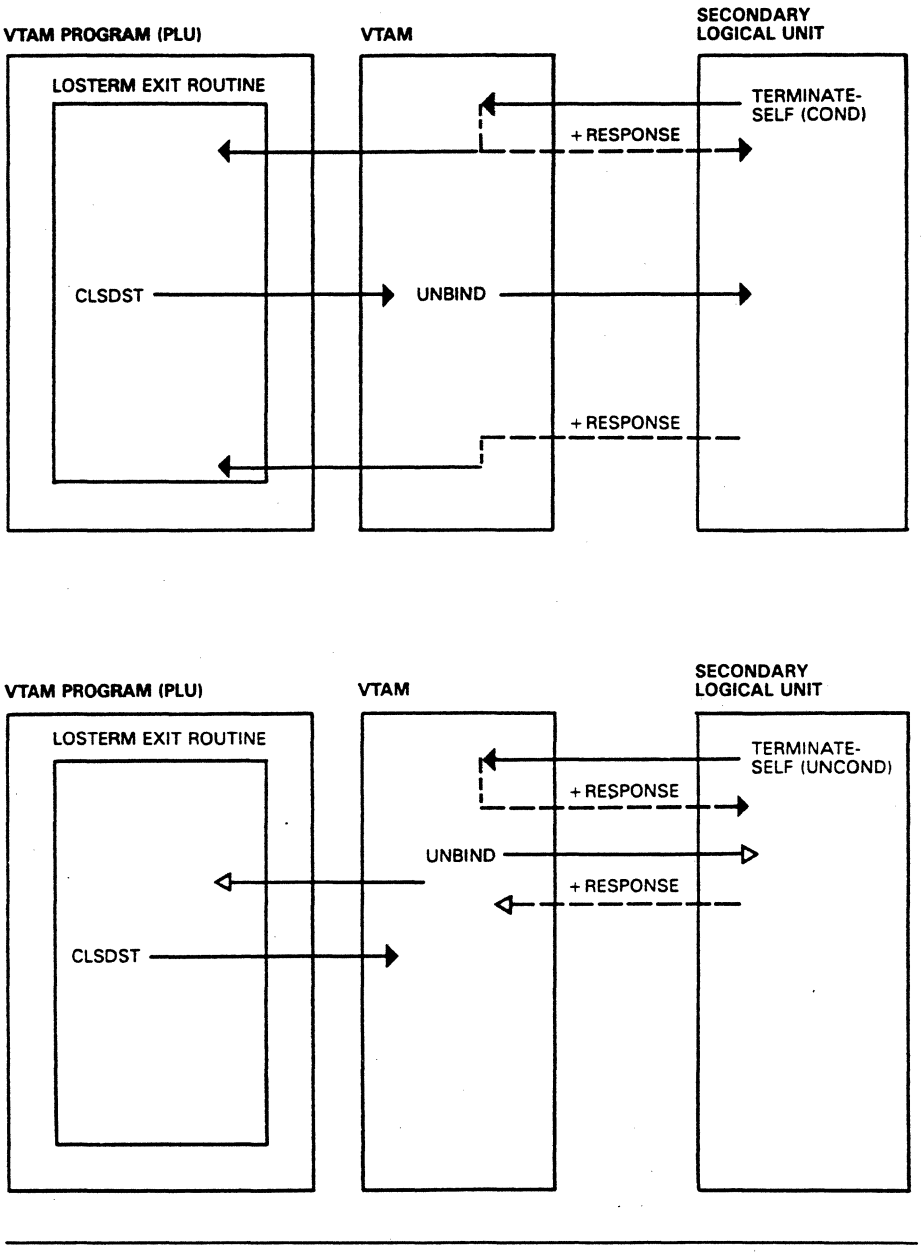

Figure 9-5. Submitting Session Termination Request to VTAM

The secondary logical unit sends the terminate-self (TERM-SELF) request to VTAM to get VTAM's assistance in terminating the session. A programmable logical unit may have the capability to generate and transmit the TERM-SELF request. Typically, a non-programmable logical unit does not have this capability. Normally, a terminal operator submits some user-defined character string and VTAM processes it against a user-supplied USS table to build the TERM-SELF request.

The TERM-SELF request can specify either conditional or unconditional termination of the session.

Conditional Session Termination: The top part of Figure 9-5 shows a conditional TERM-SELF request being submitted to VTAM. The TERM-SELF request causes VTAM to schedule the VTAM program's LOSTERM exit routine. The VTAM program must include the LOSTERM exit routine to be notified of the TERM-SELF disconnection request. The VTAM program should issue a CLSDST macro to send the UNBIND request to the secondary logical unit, thereby terminating the session.

Unconditional Session Termination: The bottom part of Figure 9-5 shows the request flow when the secondary logical unit submits a TERM-SELF request that specifies unconditional termination of the session. VTAM immediately sends an UNBIND request to the secondary logical unit, thus terminating the session. The VTAM program does not yet know that the session is terminated. If the VTAM program tried to send data to the secondary logical unit, the send would fail. VTAM would post an error code in the RPL referenced by the SEND macro that indicates that the session has been terminated.

Once VTAM has sent the UNBIND request to the secondary logical unit, it notifies the VTAM program by scheduling its LOSTERM exit routing. VTAM provides a specific code to the exit routine which says that the session has already been terminated. The VTAM program still must issue a CLSDST macro to clean up the session information that exists between the VTAM program and VTAM.

Submitting Session Termination Request to the VTAM Program

The secondary logical unit can use the SNA protocol RSHUTD or a user-defined character string to start session termination.

Figure 9-6 illustrates the request flow when the secondary logical unit uses RSHUTD to terminate the session. Study the figure and then read the discussion that follows.

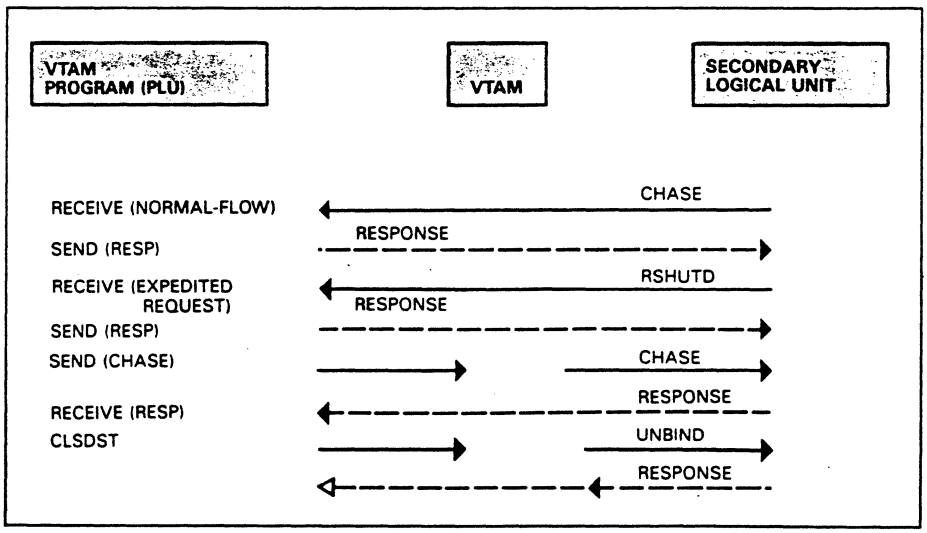

Figure 9-6. Secondary Logical Unit Submits RSHUTD Request

A secondary logical unit that has the capability may use the CHASE request to verify that there are no outstanding requests before starting session termination. Once the response is received from the VTAM program for the CHASE request, the logical unit sends the request shutdown (RSHUTD) request to the VTAM program. The RSHUTD request indicates to the program that the secondary logical unit is at the "end of job" condition, and would like the session to be terminated.

The VTAM program must make provisions to receive the expedited RSHUTD request. The request can be received either by a RECEIVE macro instruction that is coded to accept expedited requests or by the DFASY exit routine. The DFASY exit routine is entered (if provided) when an expedited request is received.

The VTAM program may or may not be ready to terminate the session. If not, communication with the logical unit continues. Assuming that the program is ready to terminate the session, it may want to use the CHASE command to verify that it has no outstanding requests.

Once the secondary logical unit returns a response for the CHASE request, the session can be terminated with assurance that there are no requests in the session that haven't been processed. The VTAM program issues a CLSDST macro to send the UNBIND request to the secondary logical unit, terminating the session.

Submitting User-Defined Session Termination Request

Assume that the VTAM program recognizes the character string GETOFF as a request to terminate the session. Figure 9-7 shows the request flow when the secondary logical unit submits GETOFF to the VTAM program.

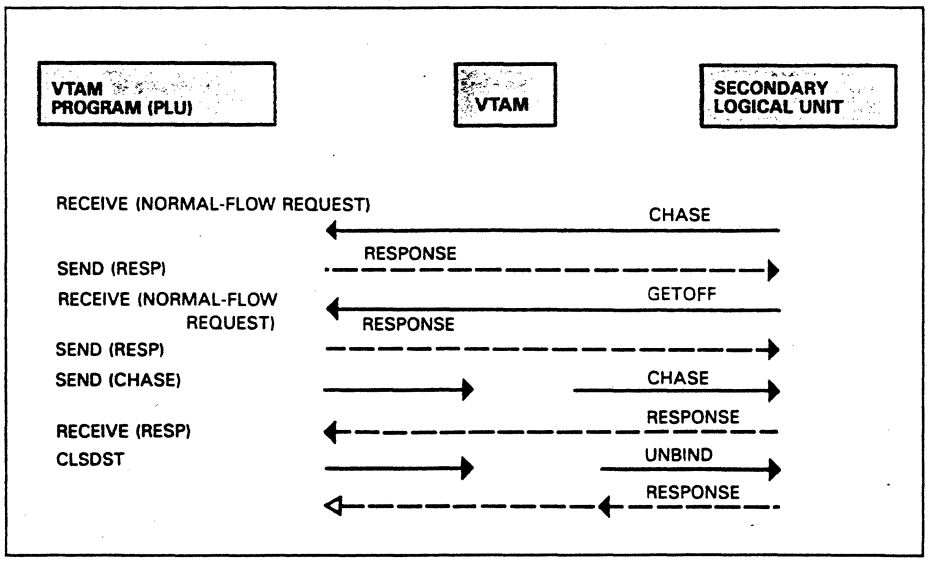

Figure 9-7. Secondary Logical Unit Submits User-Defined Logoff

This request flow is identical to the one shown in Figure 9-6. The only difference is that the secondary logical unit starts session termination by sending a request that contains the user-defined logoff GETOFF, rather than the RSHUTD request.

Please turn to Mini-Course 9 in your Personal Reference Guide and do Exercise 9.1.

VTAM Concepts

Mini-Course 10

Establishing Sessions Between VTAM Programs

MINI-COURSE 10. Establishing Sessions Between VTAM Programs

Introduction

In order for a VTAM program to connect to and communicate with another VTAM program, one of the programs must fulfill the role of a secondary logical unit and the other program must fulfill the role of a primary logical unit. When performing the function of a secondary logical unit (SLU) the program must be able to submit a logon, accept a BIND request, respond to the BIND request, and perform other SLU functions.

Figure 10-1 shows three VTAM programs (APPLA, APPLB, and APPLC) and two peripheral logical units (LU1 and LU2). APPLA has three LU-LU sessions and is performing the role of a primary logical unit (PLU) for all three sessions:

```
APPLA-APPLB
APPLA-APPLC
APPLA-LU1
```

APPLB is performing the role of a secondary logical unit (SLU) for both of its sessions:

```
APPLB-APPLA
APPLB-APPLC
```

APPLC performs the role of an SLU for the session with APPLA:

```
APPLC-APPLA
```

APPLC performs the role of a PLU for its sessions with APPLB and LU2:

```
APPLC-APPLB
APPLC-LU2
```

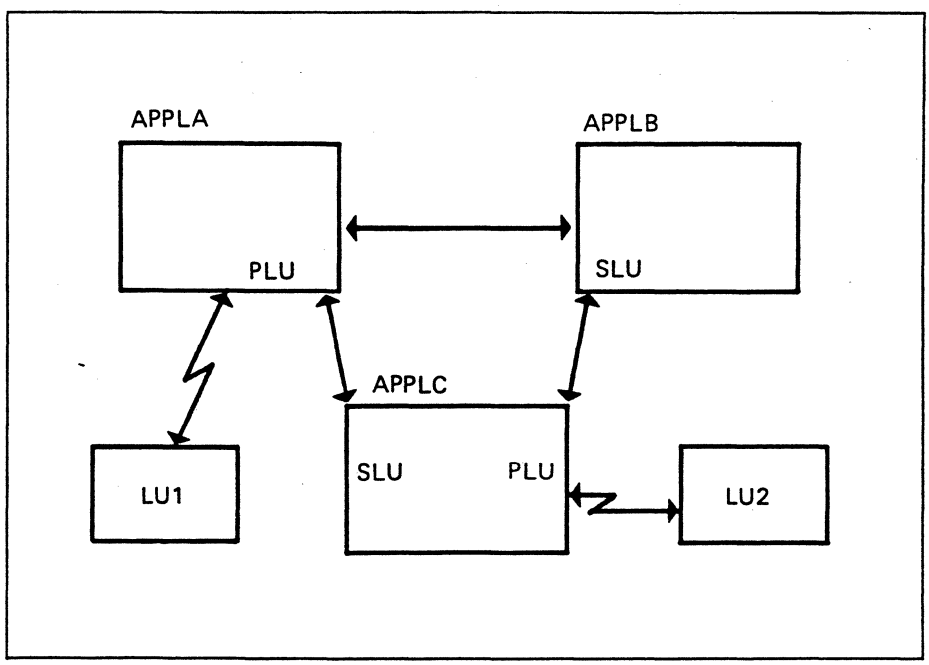

Figure 10-1. Primary and Secondary VTAM Programs

A VTAM program can perform the role of a PLU and an SLU at the same time. As an SLU it can connect to one or more PLUs. As a PLU it can connect to one or more SLUs.

The program that issues the OPNDST macro instruction automatically becomes the PLU. The program that submits a logon becomes the SLU.

Acquiring a Secondary VTAM Program (SLU)

Figure 10-2 illustrates a VTAM program acquiring a VTAM program. Study the figure, then read the discussion that follows.

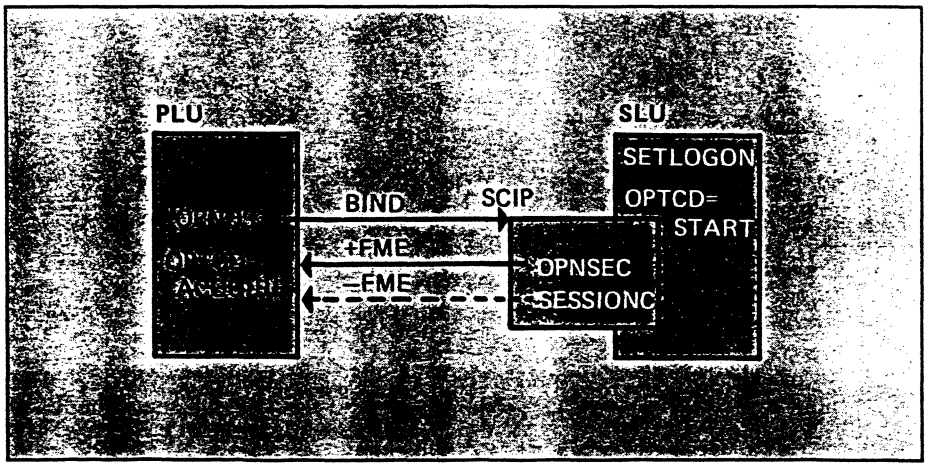

Figure 10-2. Acquiring a VTAM Program

The primary logical unit (VTAM program) issues an OPNDST to send a BIND request to the secondary logical unit (VTAM program). The secondary logical unit must include the SCIP exit routine and the routine must be enabled by the

SETLOGON macro instruction to receive the BIND request. The exit routine examines the session parameters received in the BIND request. If the parameters are acceptable, the exit routine issues an OPNSEC macro to send a positive response (+FME) to the PLU and the session is established. If the session parameters are unacceptable, the exit routine issues a SESSIONC macro to send a negative response (-FME) to the PLU to reject the session.

Submitting a Logon From a VTAM Program

A VTAM program can be designed to submit logons. Figure 10-3 illustrates a VTAM program submitting a logon to another VTAM program.

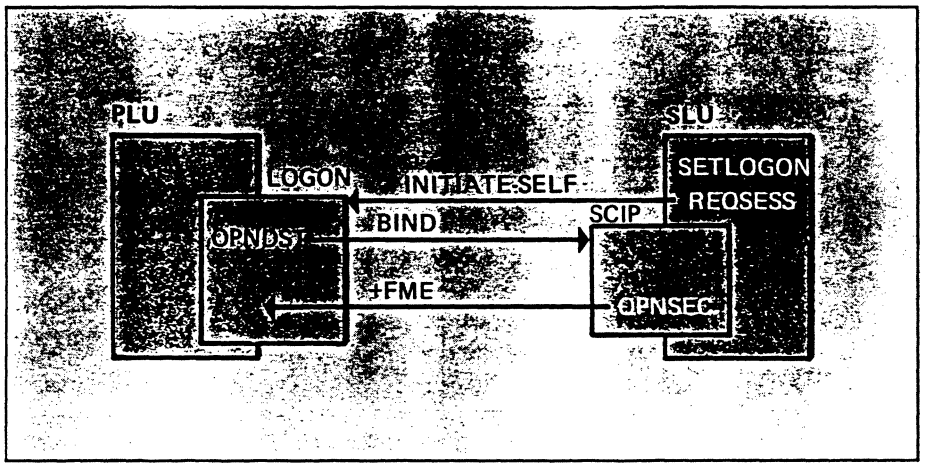

Figure 10-3. VTAM Program Submits Logon

The secondary VTAM program uses the REQSESS macro instruction to send an initiate-self (INIT-SELF) logon to the PLU. The REQSESS macro instruction causes the INIT-SELF request (logon) to be sent to the specified VTAM program. VTAM gives control to the primary logical unit's (PLU) logon exit routine to process the logon. The PLU determines that the requesting SLU is authorized for a session with the PLU and issues an OPNDST to send a BIND request to the SLU. The BIND request drives the SLU's SCIP exit routine. The SLU accepts the BIND request by issuing an OPNSEC macro instruction to send a positive response and the session is established.

Negotiable BIND

A primary logical unit (VTAM program) has the capability to send a negotiable or a non-negotiable BIND to a secondary logical unit. If a non-negotiable BIND request is sent, the secondary logical unit only has the option of accepting or rejecting the session by returning a positive or negative response. If the primary logical unit and the secondary logical unit support negotiable BIND requests, then they can negotiate which BIND parameters to use for the LU-LU session.

The negotiable BIND allows the secondary logical unit to modify session parameters received from the primary logical unit and return them in the positive response for the BIND. If the modified session parameters are acceptable to the primary logical unit, it allows the session to continue; otherwise, it terminates the session.

The primary logical unit causes a BIND request to be identified as negotiable by setting an operand field in the NIB referenced by the OPNDST.

NIB

```
PROC=NEGBIND
```

The secondary logical unit recognizes the BIND request as negotiable by examining a field in the BIND request.

Figure 10-4 illustrates a negotiable BIND operation when the secondary logical unit issues a logon. Study the figure, then read the discussion that follows.

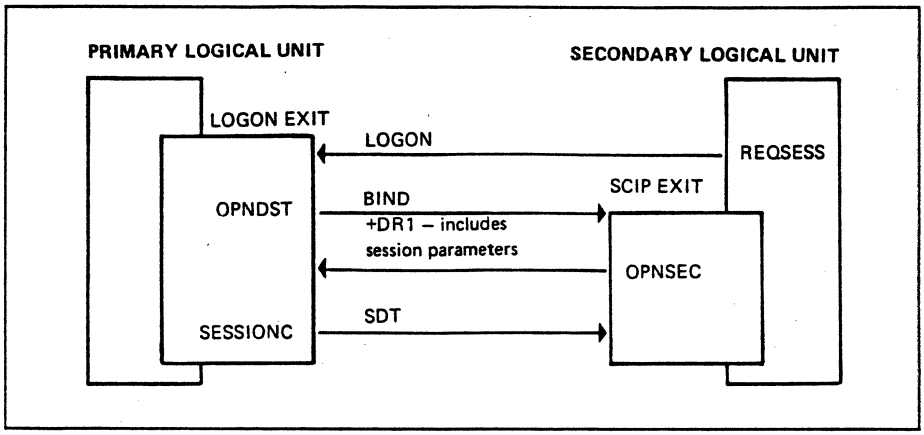

Figure 10-4. Negotiating Session Parameters

The secondary logical unit (VTAM program) issues a REQSESS macro to send a logon to the primary logical unit. The logon drives the primary logical unit's LOGON exit. The primary logical unit obtains the session parameters specified in the logon, modifies them, and stores them in its storage space. Next, the primary logical unit issues an OPNDST to send a negotiable BIND to the secondary logical unit. Two operands must be coded in the NIB referenced by the OPNDST to specify the BIND as negotiable and to identify the location of the modified session parameters to be included in the BIND:

```
PROC=NEGBIND

BNDAREA=address of modified session parameters
```

PROC=NEGBIND causes VTAM to mark the BIND request as negotiable. VTAM locates the modified session parameters using the address from the BNDAREA field and places the parameters in the BIND request.

The secondary logical unit has the option of accepting the BIND parameters, rejecting the BIND parameters, or negotiating for different BIND parameters. We'll assume that the secondary logical unit negotiates for different BIND parameters.

The secondary logical unit (Figure 10-4) obtains the session parameters and modifies them. The secondary logical unit must specify that a negotiable BIND response be sent by specifying PROC=NEGBIND in the NIB referenced by the

OPNSEC. The BNDAREA operand in the NIB must specify the address of the modified session parameters so VTAM can include them in the response.

The primary logical unit receives the response, obtains and examines the session parameters, and finds that the session parameters are acceptable. The primary logical unit then sends the start data traffic (SDT) request to the secondary logical unit to allow user data traffic in the LU-LU session.

Regardless of whether the session parameters are acceptable to the primary logical unit, the session is established upon receipt of the positive response. However, user data traffic is not allowed in the session until the SDT request is sent to and received by the secondary logical unit.

If VTAM is allowed to send the SDT request, the secondary logical unit may start sending data as soon as it receives the request. This means that data may flow in the session before the primary logical unit has a chance to examine the negotiable session parameters. Therefore, it is preferable for the primary logical unit to send the SDT request rather than to allow VTAM to send the request. When the primary logical unit is specified to send the SDT request, the following occur for a negotiable response from the secondary logical unit.

- First, the primary logical unit receives the negotiable response and examines the modified BIND parameters. (The session is established when the response is received.)

- Second, the primary logical unit sends the SDT request if the modified parameters in the response are acceptable. If the parameters are not acceptable, the primary logical unit issues a CLSDST macro instruction to terminate the session.

Parallel Sessions

ACF/VTAM allows multiple simultaneous sessions (parallel sessions) between the same two VTAM programs. Each of the parallel sessions between the same LU pairs are independent of each other.

Each parallel session can employ a different set of protocols and have independent session control. For example, one session could use chaining while other sessions might not use chaining, each session could support different maximum RU sizes, and some sessions could use brackets while others might not. On the other hand, all the parallel sessions may support the same protocols.

Establishing Parallel Sessions

There can be any number of parallel sessions between two VTAM programs up to the limit of the SNA addressing structure. The definition of a VTAM program must include the coding PARSESS=YES for the program to participate in parallel sessions.

Each VTAM program has one network name but can have many unique network addresses. ACF/VTAM provides for a VTAM program (logical unit) to have one secondary address and multiple primary addresses.

Figure 10-5 shows two VTAM programs with network names PROG1 and PROG2. The three lines between PROG1 and PROG2 represent three parallel sessions. The three enclosed Ps (P1, P2, and P3) in PROG1 are to indicate that

PROG1 is the primary for all three sessions. There is a unique CID for each session. The enclosed S in PROG2 is to indicate that PROG2 is the secondary logical unit for all three sessions.

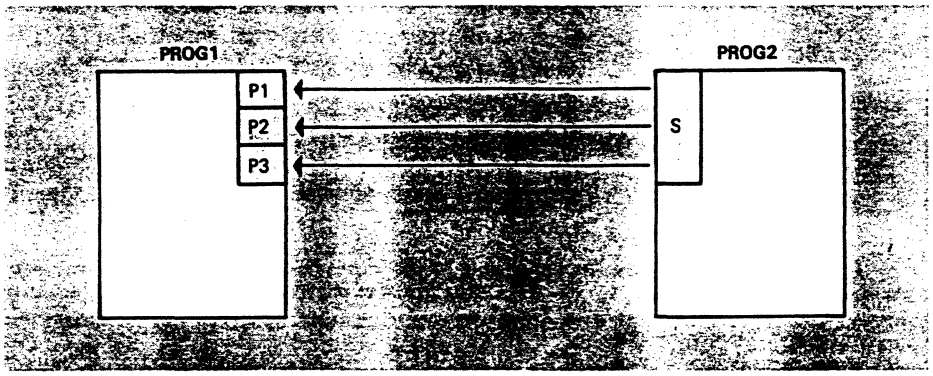

Figure 10-5. Parallel Sessions

The next description is a logon operation with PROG2 submitting a logon that drives PROG1's LOGON exit. It is assumed that one session (S-P1) already exist: Study Figure 10-6, then read the discussion that follows.

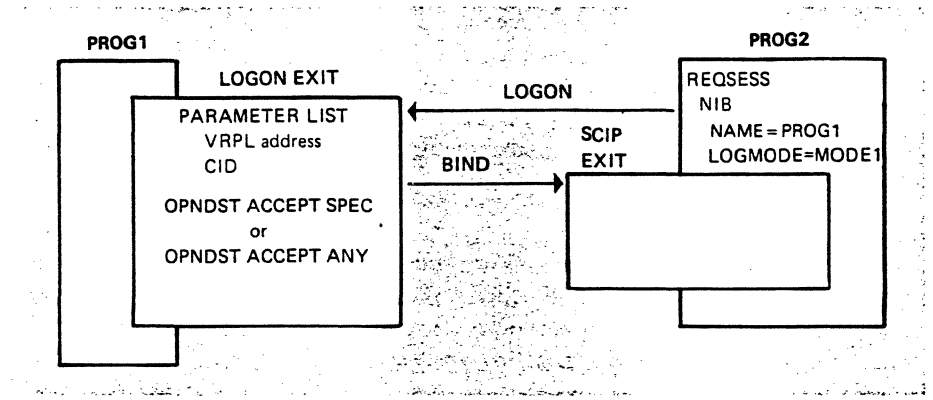

Figure 10-6. Establishing Parallel Sessions

PROG2 issues the REQSESS macro to send a logon. The NAME field of the referenced NIB directs the logon to PROG1. The LOGMODE field specifies a set of session parameters to be included in the BIND request. There is nothing that PROG2 has to do to specify that a second session is to be established during this logon operation.

The logon flows to PROG1 where it drives the LOGON exit. The LOGON exit routine doesn't necessarily have to do anything special or different just because a second session is being established. The exit can obtain and examine session parameters to determine their acceptability. It can obtain the logon message and verify the authority of the LU to logon. Then the exit can issue an OPNDST with OPTCD=ACCEPT to accept the logon that invoked the LOGON exit. The OPNDST causes the BIND to be sent to PROG2. If PROG2 returns a positive response, the session is established. VTAM places a unique CID for this second session (S-P2) in the RPL and in the NIB referenced by the OPNDST macro instruction. Now there are two LU-LU sessions between PROG1 and PROG2.

Please turn to Mini-Course 10 in your Personal Reference Guide and do Exercise 10.1.

VTAM Concepts

Mini-Course 11

Terminating Sessions Between VTAM Programs

MINI-COURSE 11. Terminating Sessions Between VTAM Programs

Introduction

A VTAM program that performs the role of a secondary logical unit must be able to receive and process session termination requests which include UNBIND and SHUTD. Also, there is a session termination protocol that may be used for a session that uses brackets protocol called the stop bracket initiation protocol. VTAM programs have the capability to use this protocol.

Primary Logical Unit Initiates Session Termination

A primary logical unit can initiate session termination by requesting VTAM's assistance or by sending a request to the secondary logical unit. Two requests can be sent to the secondary logical unit: SHUTD and SBI (stop bracket initiation).

Submitting Termination Request to VTAM

The primary logical unit requests VTAM's assistance to terminate a session by issuing a CLSDST macro, as shown in Figure 11-1.

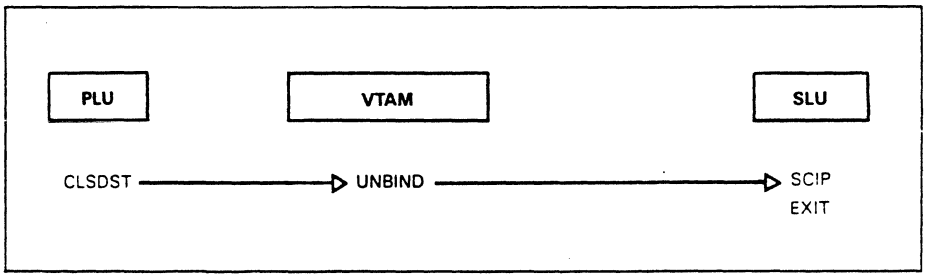

Figure 11-1. Submitting Termination Request to VTAM

The secondary logical unit must include a SCIP exit routine to receive and process the UNBIND request.

Submitting Termination Request to the SLU

Next we examine two requests that are sent to a secondary logical unit to terminate LU-LU sessions, namely the SHUTD request and the SBI request.

Sending the SHUTD Request: Figure 11-2 shows the request flow when the primary logical unit initiates session termination with the SHUTD request.

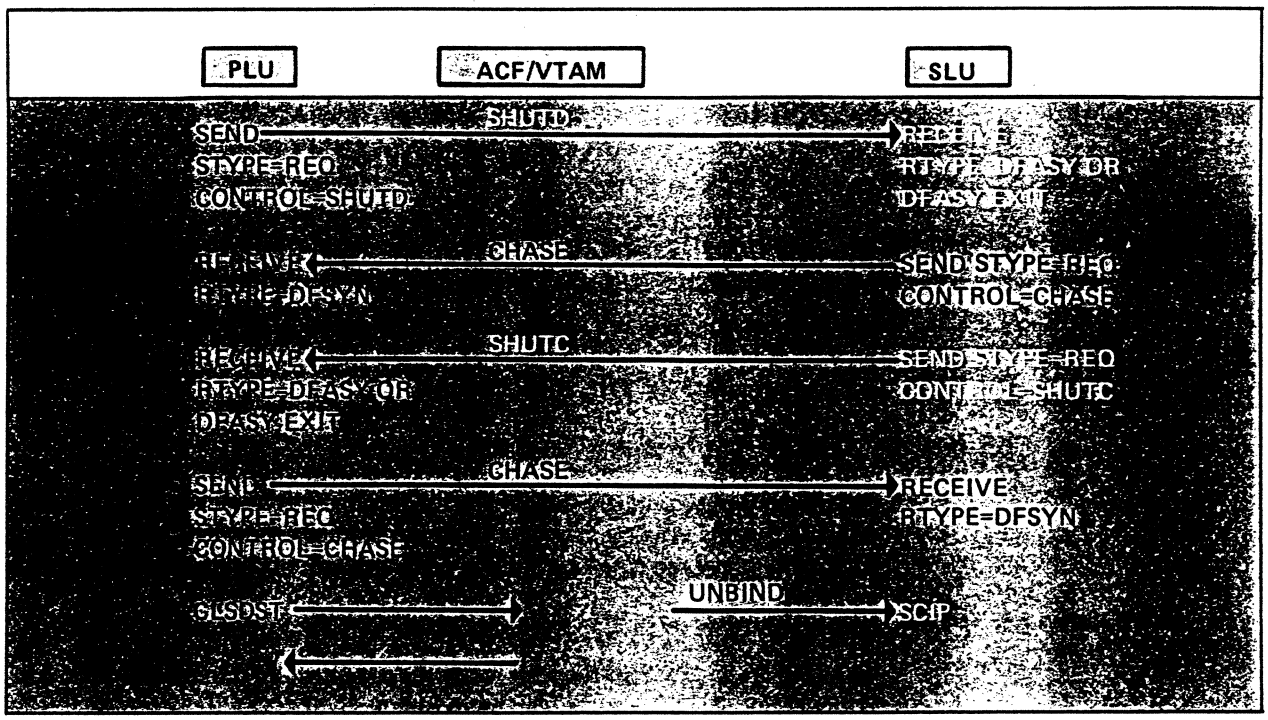

Figure 11-2. Submitting the SHUTD Request

The primary logical unit (PLU) issues a SEND macro to transmit a SHUTD request on the expedited-flow. The secondary logical unit (SLU) uses a RECEIVE macro instruction or a DFASY exit routine to accept the SHUTD request from the PLU. Upon receipt of the SHUTD request, the SLU optionally issues a SEND to transmit the CHASE request ensuring that there are no outstanding requests. The SLU then issues a SEND macro to transmit the SHUTC request to the PLU on the expedited-flow, indicating that the SLU is ready for session termination. The PLU optionally issues a SEND macro to transmit a CHASE request, ensuring that there are no outstanding requests. Then the PLU issues a CLSDST to transmit the UNBIND request to the SLU, driving the SLU's SCIP exit and causing the session to be terminated.

Sending the SBI Request: Now assume that the two session partners are using brackets protocol and will use the stop bracket initiation protocol to terminate the session. The logical place to terminate the session is at the end of a bracket. The primary logical unit can send a request asking the secondary logical unit to stop initiating brackets and when it agrees and the session is between brackets, the session can be terminated with assurance of not losing any data. Figure 11-3 shows the request sequence when the stop bracket initiation protocol is used.

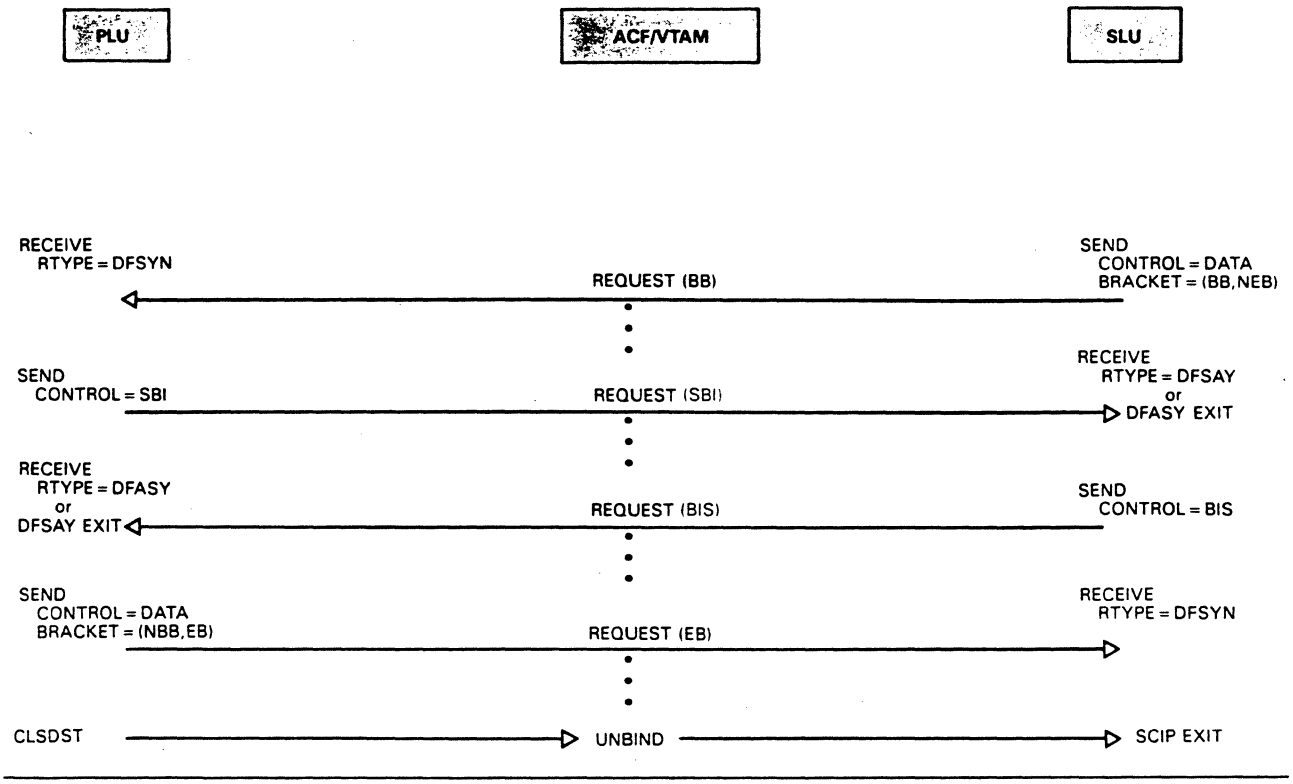

Figure 11-3. Stop Bracket Initiation Protocol

The primary logical unit is ready to terminate the LU-LU session, so it issues a SEND to transmit a control request to the secondary logical unit on the expedited flow. The CONTROL=SBI operand, specified in the RPL referenced by the SEND macro, causes the stop bracket initiation command (SBI) to be placed in the request unit (RU).

The secondary logical unit accepts the request with either a RECEIVE that accepts expedited requests or with a DFASY exit routine. The secondary logical unit can continue initiating brackets, that is, starting transactions, until it is ready to terminate the session. Then it transmits a request to the primary logical unit that contains the bracket initiation stopped (BIS) command. Now the secondary logical unit is not allowed to start another bracket. The primary logical unit then issues a CLSDST macro to transmit the UNBIND request to terminate the session.

Secondary Logical Unit Initiates Session Termination

The secondary logical unit can issue a TERM-SELF request to terminate sessions. The stop bracket initiation (SBI) request can be used to terminate sessions but will not be covered since it is the same as when the primary logical unit uses it.

Submitting Termination Request to VTAM

Figure 11-4 shows the session termination request sequence when the secondary logical unit submits to VTAM a TERM-SELF request that specifies unconditional session termination.

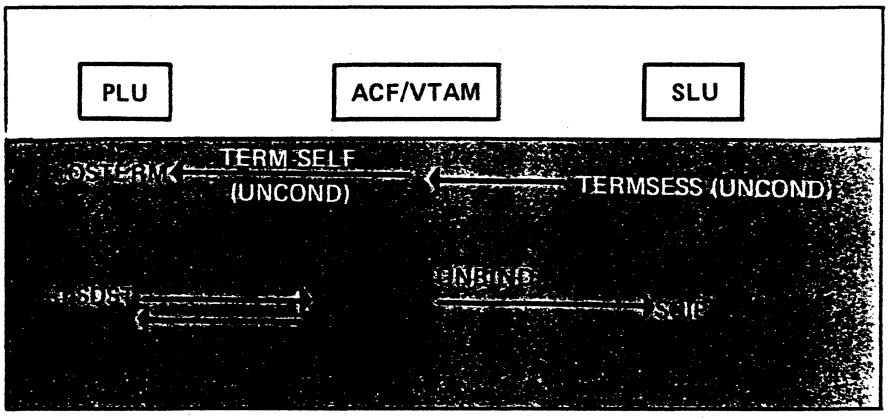

Figure 11-4. SLU Submits TERM-SELF

The secondary logical unit (SLU) issues a TERMSESS macro instruction to transmit the unconditional TERM-SELF request to the primary logical unit (PLU). VTAM schedules the PLU's LOSTERM exit routine to notify it that the SLU is to be disconnected. VTAM also sends an UNBIND request to the SLU (drives the SCIP exit). The UNBIND terminates the session but the PLU must still issue a CLSDST to clean up control blocks used in the connection.

Figure 11-5 shows the request sequence when the SLU transmits a conditional TERM-SELF request to VTAM.

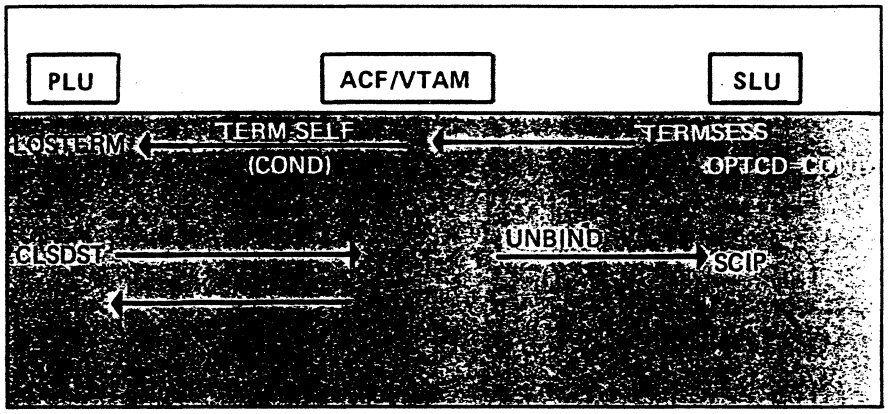

Figure 11-5. Conditional TERM-SELF Request

The SLU issues a TERMSESS macro instruction to transmit a conditional TERM-SELF to the primary logical unit. VTAM schedules the PLU's LOSTERM exit, notifying it of the request for a conditional termination. The PLU has the option of when to terminate the session. When it is ready for termination, it issues a CLSDST macro instruction, causing VTAM to send an UNBIND request which drives the SLU's SCIP exit.

Submitting Termination Request to the PLU

Figure 11-6 shows the request sequence when the SLU transmits a RSHUTD request to the PLU.

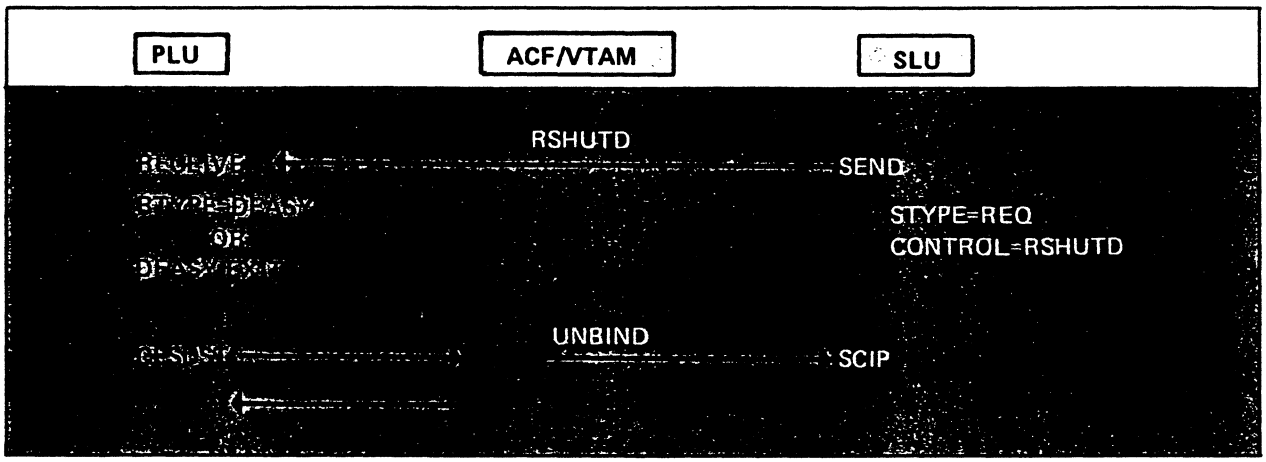

Figure 11-6. Submitting RSHUTD Request

The SLU issues a SEND to transmit the RSHUTD request on the expedited flow to the PLU requesting session termination. The PLU uses a RECEIVE macro that accepts expedited requests or a DFASY exit routine to accept expedited requests. The PLU issues a CLSDST when it is ready to terminate the session.

Terminating Parallel Sessions

A specific LU-LU session or all parallel sessions between the same two LUs can be terminated with one VTAM macro instruction. First, the CLSDST macro is used by the primary logical unit. Second, the TERMSESS macro is used by the secondary logical unit.

A primary LU uses the CLSDST macro instruction to terminate a specific session or all parallel sessions with a specific secondary logical unit. A specific session is terminated when the CLSDST is coded to include the CID that represents the session. All parallel sessions are terminated when the CLSDST is coded to include the LU name of the secondary LU.

A secondary LU uses the TERMSESS macro instruction to send a TERM-SELF request to the primary LU. This request can cause a specific session or all parallel sessions with the primary LU to be terminated. A specific session is terminated when the TERMSESS is coded to include the CID that represents the session. All parallel sessions are terminated when the TERMSESS is coded to include the LU name of the primary LU.

Please turn to Mini-Course 11 in your Personal Reference Guide and do Exercise 11.1.

VTAM Concepts

Mini-Course 12

Basics of LU-LU Communication

MINI-COURSE 12. Basics of LU-LU Communication

Introduction

The purpose of LU-LU sessions is to communicate units of information between two session partners. Units of information consist of user data, commands (control information), and acknowledgement information. Each unit of information is sequence numbered and the units may be logically grouped. This mini-course describes the units of information, how they are sequence numbered and why, and how requests can be grouped into a set called a request chain.

Types of Information

Communication between a VTAM program and a logical unit consists of exchanges of requests and responses. A request consists of either data and control information or control information alone. A response indicates whether the request was acceptable or unacceptable.

Requests

A request that includes data and control information is called a data request. A request that contains an SNA command is called a command request.

Data Requests: The main function of a VTAM network is to transmit data requests between VTAM programs and logical units. Figure 12-1 illustrates a data request.

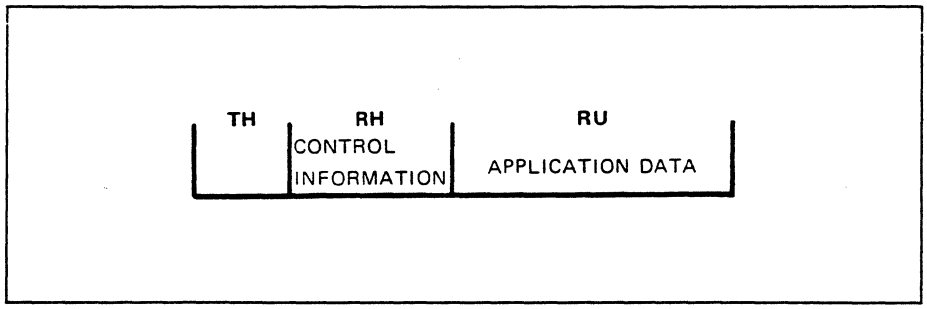

Figure 12-1. Data Request PIU

All requests are transmitted in the form of a path information unit (PIU). Figure 12-1 shows data in the request unit (RU) and control information in the request header (RH). The RH identifies the type of information that's contained in the RU, the form of response requested, and other control information. The RU itself contains application data, that is, data to be processed.

Command Requests: A command request does not include application data. A command request includes information in the RH and a command in the RU. This information is used for such things as data-flow control and shutting down the session.

Figure 12-2 shows a PIU that contains a command request. The RH contains control information and the RU contains an SNA command.

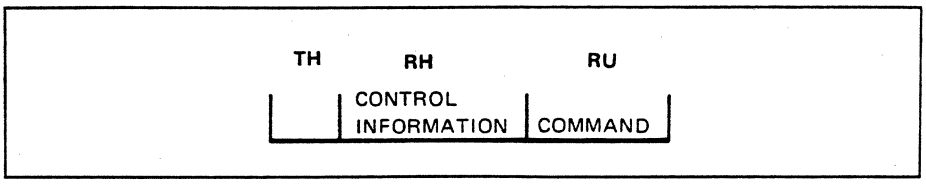

Figure 12-2. Control Request PIU

Responses

The request header (RH) of every request PIU contains an indicator that specifies whether a response is to be returned by the receiving logical unit. The RH of a data message PIU can specify a definite response, exception response, or that no response is to be returned. The RH of a PIU that contains an SNA command must specify a definite response.

The logical unit that receives a request may return a response to the request sender to indicate whether the request was acceptable. A positive response indicates that a request was acceptable, while a negative response indicates that a request was unacceptable.

When a VTAM program or a logical unit sends a request, it specifies the form of response for that request. The request sender can indicate a definite response, an exception response, or no response.

Requesting a definite response means that the sender wants an appropriate positive or negative response to be returned. The receiver of the request must return the appropriate response. Requesting an exception response means that the request sender wants a response only if a negative response is appropriate. The request receiver must return a negative response if the request is unacceptable, otherwise no response is to be returned. Finally, requesting no response means that regardless of the acceptability of the request, the receiving logical unit is not to return a response.

VTAM provides the facility for the request sender to indicate one or two responses each time it sends a message request. The responses are referred to as type one responses (DR1) and type two responses (DR2). The VTAM parameter FME indicates a DR1 response and the VTAM parameter RRN indicates a DR2 response. FME and RRN are VTAM terminology, while DR1 and DR2 are SNA terminology.

The request sender can indicate either DR1 (FME) or DR2 (RRN), or both. Whichever is indicated, the request sender must also specify either definite response or exception response. This means that there are seven possible responses: (1) definite DR1 (FME), (2) definite DR2 (RRN), (3) both definite DR1 and DR2 (FME and RRN), (4) exception DR1 (FME), (5) exception DR2 (RRN), (6) both exception DR1 (FME) and DR2 (RRN), and (7) none.

Programmable logical units can assign different meanings to DR1 and DR2 responses. A use for both could be as follows. A positive DR2 (RRN) response could signify that the message request has been received by the receiving logical unit, and that logical unit accepts responsibility for getting the message request to the end user. The end user could be an operator or a printer or some other end user. A positive DR1 (FME) response could signify that the message request has been delivered to the end user by the receiving logical unit.

LU-LU sessions may not need to distinguish between the two types of responses. In fact, most logical units only support one form of the response.

It's mandatory that both session partners agree to use the same response protocol. The response protocol to be used in a session is specified by session parameters in the BIND request that establishes the LU-LU session. While communicating, the request sender specifies the form of response to be returned. This specification is placed in the request header of the PIU that contains the request. The request receiver analyzes the information in the request unit and responds according to the response indicator in the associated RH.

Figure 12-3 shows a secondary logical unit sending a request to a VTAM program (item 1). The RH specifies a definite DR1 (FME) response.

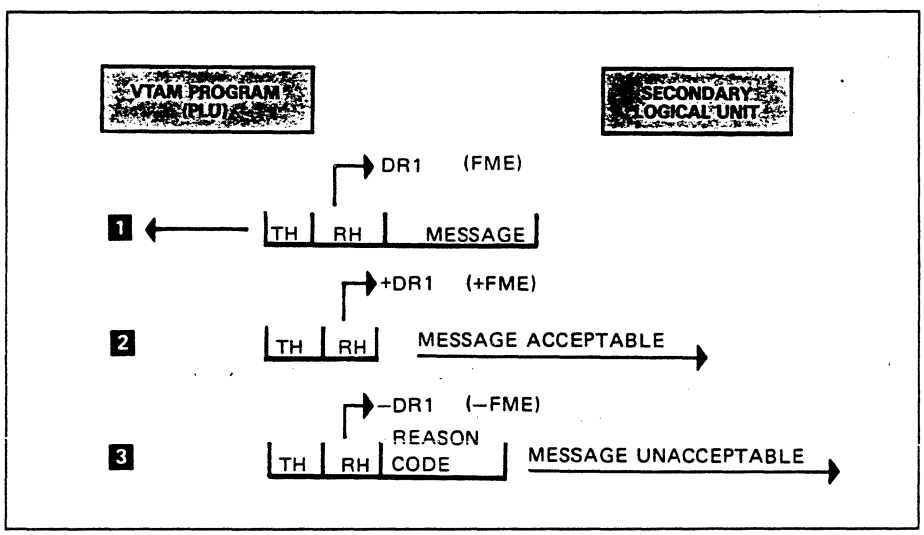

Figure 12-3. Requesting a Definite Response

The definite DR1 specification in the RH indicates that the VTAM program is to return a positive or negative response as appropriate. The VTAM program issues a SEND macro to return a positive response if the request is acceptable (item 2).

```
SEND
  STYPE=RESP
  RESPOND=(NEX,FME,NRRN)
```

The STYPE (send type) operand specifies that a response is to be transmitted and the RESPOND operand specifies the form and type of response to be returned. NEX (not exception) specifies a positive response, FME specifies that an FME (DR1) is to be transmitted, and NRRN (not RRN) specifies that an RRN (DR2) is *not* to be returned. The SEND macro causes a response PIU to be transmitted and the SEND operands cause the response indicators to be set in the associated response header (RH).

If the request received from the logical unit is unacceptable, the VTAM program issues a SEND macro to transmit a negative response (item 3).

```
SEND
  STYPE=RESP
  RESPOND=(EX,FME,NRRN)
  SSENSEO=code,
  SSENSMO=code
```

The RESPOND parameter EX (exception) specifies a negative response. Therefore, the response header of the associated response PIU will be set to specify a negative FME (DR1). A negative response also includes a code in the response unit (RU) that describes why the request was unacceptable. The VTAM program supplies the reason code by the two operands, SSENSEO and SSENSMO.

Figure 12-4 shows a secondary logical unit sending requests to a VTAM program and each request specifies an exception response.

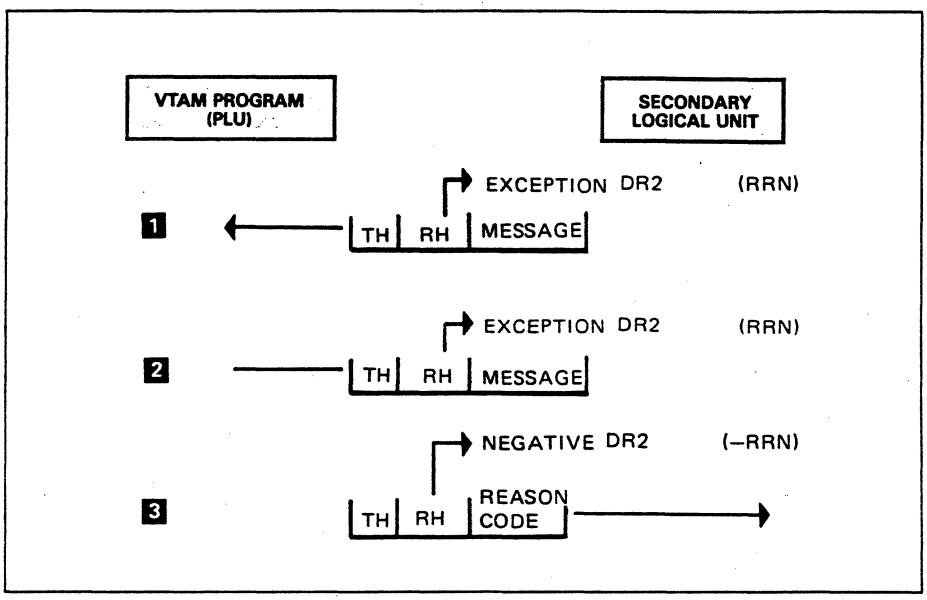

Figure 12-4. Requesting an Exception Response

Exception DR2 is specified as shown in the RH of each request PIU (items 1 and 2). The first request is acceptable to the VTAM program (item 1), therefore a response is not returned to the logical unit. The second request is unacceptable to the VTAM program (item 2), therefore the program uses a SEND macro to transmit a negative response to the logical unit (item 3).

```
SEND RPL=RPL1,
     STYPE=RESP,
     RESPOND=(EX,NFME,RRN),
     SSENSEO=code,
     SSENSMO=code
```

The RESPOND operand specifies a negative (EX) RRN (DR2) response to be transmitted to the logical unit. The SSENSEO and SSENSMO operands specify the code to be transmitted in the response unit (RU) describing why the associated request is unacceptable.

So a request sender must specify the form of response to be returned. The request sender specifies two types of information:

1. Whether a definite response, exception response, or no response is desired

2. Whether a DR1 (FME) or a DR2 (RRN), both DR1 and DR2, or no response is desired

The request receiver must respond accordingly.

Sequence Numbering

A sequence numbering technique is used to ensure that message requests arrive at their destination in the order that they are sent. This is accomplished by having VTAM assign a sequence number to each message request that is sent to a logical unit. VTAM places the sequence number in the transmission header (TH) of the PIU and provides the sending VTAM program with the sequence number.

The numbering begins with the first request sent after the LU-LU session is established. The number is increased by one for each subsequent message request. This process continues until the session is terminated, unless the VTAM program interrupts it earlier. The VTAM program might have to reinitialize the sequence numbers if a data transmission error occurs.

A logical unit assigns sequence numbers to message requests sent to the VTAM program. So you should understand that message requests in the primary to secondary flow (out bound) are sequence numbered and message requests in the secondary to primary flow (in bound) are sequence numbered. The sequence numbering for either flow is independent of the other, as illustrated in Figure 12-5.

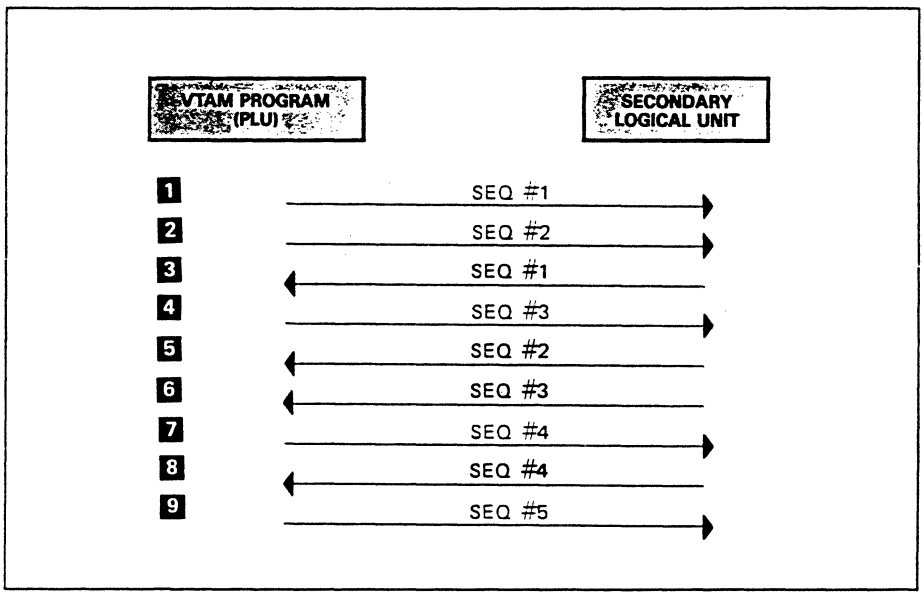

Figure 12-5. Sequence Numbering for Message Requests

For this discussion, assume that the session has just been established. Items 1,2,4,7, and 9 show message requests flowing from the VTAM program to the secondary logical unit. These message requests are assigned sequence numbers 1 through 5, in order. Items 3,5,6, and 8 show message requests flowing from the secondary logical unit to the VTAM program. These message requests are assigned sequence numbers 1 through 4, in order.

Sequence numbers are also assigned to responses. The sequence number of each response must be the same as that of the associated request. The VTAM program and the secondary logical unit are responsible for assigning sequence numbers to responses.

The VTAM program has access to the sequence number of each received request. This access is via the RPL referenced by the RECEIVE macro that accepts the

request. When VTAM receives a request path information unit (PIU) for a VTAM program, VTAM moves information from the request unit (RU), request header (RH), and the transmission header (TH) to the VTAM program. Information from the headers is placed in the RPL referenced by the RECEIVE macro. Therefore, the VTAM program can obtain the sequence number of the received request and include it in the response that is transmitted to the logical unit.

The logical unit also has access to the sequence number of received message requests. It too assigns the appropriate sequence number to all responses that it sends.

Request Chaining

Applications deal with units of work. The unit of work for one application could consist of a VTAM program sending one request to a logical unit. The unit of work for another application could consist of many requests being transmitted.

An example of a unit of work can be illustrated by an inquiry operation. Assume that a logical unit sends an inquiry request to a VTAM program. The VTAM program is to obtain various pieces of information from different data files and, as they become available, send them to the logical unit. That means that a number of message requests are sent to the logical unit. The unit of work is complete when the logical unit receives all of the message requests required to satisfy the inquiry.

The VTAM program can group these message requests into a set called a request chain. If the receiving logical unit can recognize the first and last requests in the chain, it knows that it has received the information to satisfy the inquiry. The sender (VTAM program in our example) can indicate which is the first request of the chain, the last request of the chain, and those requests between. All requests in a chain, other than the first and last requests, are called middle-of-chain requests. Figure 12-6 shows a sequence of SEND macro instructions that sends a four element request chain (items 1,2,3, and 4 at the top of the figure).

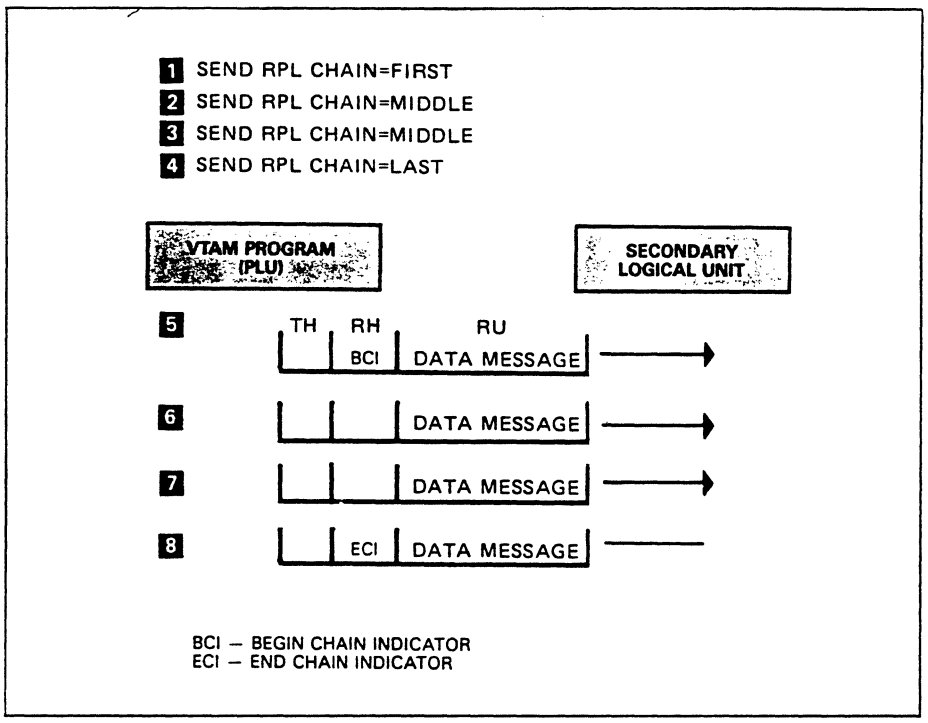

Figure 12-6. Request Chain

The RPL operand CHAIN=FIRST identifies the message request as the first request in the request chain (item 1). The CHAIN=FIRST operand causes the begin chain indicator(BCI) to be set in the RH. The RPL operand CHAIN=MIDDLE identifies the second and third message requests as middle-of-chain message requests (items 2 and 3). The begin chain indicator (BCI) and end chain indicator (ECI) are set off in the RH. The RPL operand CHAIN=LAST identifies the fourth message request as the last request of the chain (item 4) and causes the end chain indicator (ECI) to be set in the RH. VTAM sets an RH field in each PIU to identify which part of the request chain is being transmitted as shown in the lower part of Figure 12-6. The SEND macro instruction at item 1 initiates the PIU at item 5. The rest of the PIUs are initiated in order by the SENDs at items 2,3, and 4.

Each transmitted request is considered to be part of a request chain. Each transmitted request is either a single element request chain or it's a single-element of a multiple-element chain. A VTAM program identifies a single element request chain with the RPL operand CHAIN=ONLY.

Please turn to Mini-Course 12 in your Personal Reference Guide and do Exercise 12.1.

VTAM Concepts

Mini-Course 13

Communication Macros and Control Blocks

MINI-COURSE 13. Communication Macros and Control Blocks

Introduction

A VTAM program issues SEND macro instructions to transmit requests and responses to other logical units. A VTAM program issues RECEIVE macro instructions to obtain requests and responses transmitted to the program from other logical units.

Communication Macro Instructions and Control Blocks

All SEND/RECEIVE macro instructions must reference a request parameter list (RPL) control block since the RPL provides VTAM with the necessary information to process the SEND/RECEIVE request. Figure 13-1 illustrates a SEND/RECEIVE macro instruction referencing an RPL control block.

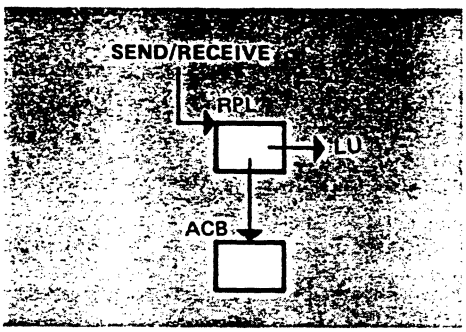

Figure 13-1. SEND/RECEIVE Macros and the RPL Control Block

As the figure shows, the RPL identifies the logical unit (LU) to which a request or response is to be transmitted, or from which a request or response is to be received. The RPL identifies the access method control block (ACB) that represents this VTAM program to VTAM.

The RPL also contains other information:

- Bracket indicators
- Response indicators
- Chaining indicators
- Sequence number
- Error codes for negative responses
- Specifications as to how VTAM is to handle the operation
- Control information for the target logical unit

The RPL control block can be generated by the RPL macro instruction. The RPL macro is coded as part of a VTAM program and when the program is assembled, the RPL control block is generated. The fields of the control block are initialized according to the values specified in the coded RPL operands. RPL control blocks may also be generated at execution time by the use of the GENCB macro instruction.

One or more RPLs can be generated for a VTAM program. The number of RPLs depend on such things as the following:

- The number of logical units that the VTAM program communicates with concurrently
- The types of operations that use an RPL

Regardless of whether one or many RPLs are used in a VTAM program, the fields of each RPL are continually modified by the VTAM program and by VTAM. The VTAM program can modify RPL fields using one of the following:

- Assembler instructions
- Manipulative macro instructions (for example, MODCB)
- RPL modifier parameters in the macro instructions that reference the RPLs

We will refer to the following skelton VTAM program in explaining how and when RPL fields are set.

VTAM Program

```
             OPNDST     RPL=RPL1
               .
               .
               .
             MVC   ...
             SEND       RPL=RPL1
               .
               .
               .
             MODCB ...
             RECEIVE    RPL=RPL1
               .
               .
               .
             SEND       RPL=RPL1,
                        BRACKET=(NBB,EB)
               .
               .
               .
RPL1         RPL        ACB=ACB1,
                        BRACKET=(BB,NEB),
                        CHAIN=ONLY,
                        NIB=NIB1
```

The RPL macro (named RPL1) is shown at the bottom. Four operands are coded and those associated RPL fields will be initialized to the coded values by the

assembly process. Certain of the other fields will be initialized to default values and the remaining fields will not be initialized.

The OPNDST macro is the first to reference RPL1 and the initial field values are used. VTAM uses the RPL values to process the OPNDST. The OPNDST sends a BIND request to establish an LU-LU session. At the completion of the OPNDST operation, VTAM sets certain fields in RPL1 to a particular value. VTAM places the CID (network addresses of the VTAM program and the logical unit) that represents the session in the ARG field of RPL1. When the ARG field contains this CID, the VTAM program can use RPL1 to communicate with the logical unit.

The next VTAM macro is a SEND and it references RPL1. RPL1 contains the appropriate CID but several other RPL fields may not be set appropriately for the SEND operation. The coding shows an assembler instruction before the SEND macro and it is used to modify a field in RPL1 in preparation for the SEND operation. If several RPL fields need to be modified, then several assembler instructions will be needed. Once the RPL fields are set, the SEND macro instruction is executed and the new RPL field values are used.

The RECEIVE macro also references RPL1 and certain RPL fields must be set to different values than for the previous SEND. Our coding example shows the MODCB macro being used to modify fields in RPL1. Depending on the number of RPL fields to be modified, several instructions may be required.

The third way to modify RPL fields is illustrated by the second SEND macro. The RPL operand BRACKET is coded in the SEND macro instruction. This causes the BRACKET field of RPL1 to be set to the coded value before the SEND operation is performed.

A VTAM program can use any combination of the three ways to modify RPL fields.

Not all RPL fields are used by each VTAM macro that references the RPL. A certain set of RPL fields are used for SEND operations, another for RECEIVE operations, and so on.

VTAM programs pass information to VTAM in control blocks. (For example the RPL), and VTAM passes information to VTAM programs in those same control blocks.

Identifying the Logical Unit

The VTAM program uses the SEND macro instruction to transmit requests and responses to a logical unit (LU). The RPL that is referenced by a SEND must contain a CID, because a SEND request is always directed to a specific logical unit. The CID must be placed in the RPL before the SEND macro instruction is executed.

A RECEIVE may be coded to accept requests and responses from a specific logical unit, or from any logical unit that is connected to the VTAM program. To accept input from a specific LU, the CID must be included in the referenced RPL and OPTCD=SPEC must be specified. A RECEIVE that accepts input from any connected logical unit does not have to provide a CID in the referenced RPL and OPTCD=ANY is specified. VTAM completes the RECEIVE with a request or a response from one of the connected logical units and places the CID of that logical

unit in the referenced RPL. The VTAM program can use the CID to determine which logical unit the request or response came from.

You should now realize that an appropriate CID must be included in an RPL associated with a SEND operation, and that a CID must be included in an RPL that is associated with a RECEIVE that is to accept input from a specific logical unit.

Sending and Receiving Requests

You learned earlier that a transmitted data request includes control information as well as application data. Now you will see how both application data and control data are handled as they flow through the VTAM network. The first data flow description is from the VTAM program to the logical unit.

Sending Requests

Figure 13-2 illustrates a SEND operation.

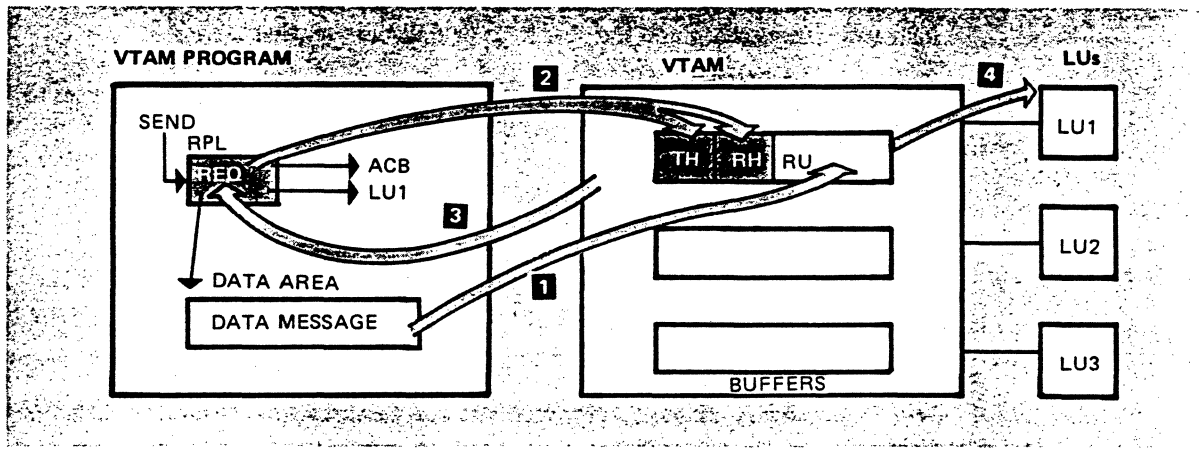

Figure 13-2. Sending Data Requests

VTAM has a number of buffers used in the transfer of requests between logical units and VTAM programs. The VTAM program doesn't have to know about the VTAM buffers, nor about the addresses, sizes, or number of buffers available. The VTAM program must only include an area to hold data that is to be sent to a logical unit.

The SEND RPL shown in Figure 13-2 references the appropriate ACB, specifies a logical unit (LU1), and includes the address of the data area. The RPL also specifies that it is sending a request (REQ), not a response.

When the SEND macro instruction is executed, VTAM moves the data message from the VTAM program's data area to a VTAM buffer (item 1). VTAM proceeds to build a PIU using information from the RPL (item 2). The CID is placed in the origin and destination fields of the transmission header (TH). Control information is placed in the request header (RH) that describes the contents of the request unit (RU) and how it is to be processed.

The PIU now resides in a VTAM buffer. VTAM can send it to the target logical unit (item 4) because the logical unit's address is in the TH.

Not only does information flow from the RPL to VTAM, but VTAM changes some of the RPL fields at the completion of a requested operation (item 3). This includes information such as error codes and the sequence number of the request.

The RPL is the main vehicle for communication between the VTAM program and VTAM.

Receiving Requests

You saw that data flow outbound from the VTAM program is initiated by a SEND macro instruction. Does that mean that inbound data flow is initiated by a RECEIVE macro instruction? The answer is no. Logical units, without prompting from a VTAM program, send requests inbound as illustrated in Figure 13-3.

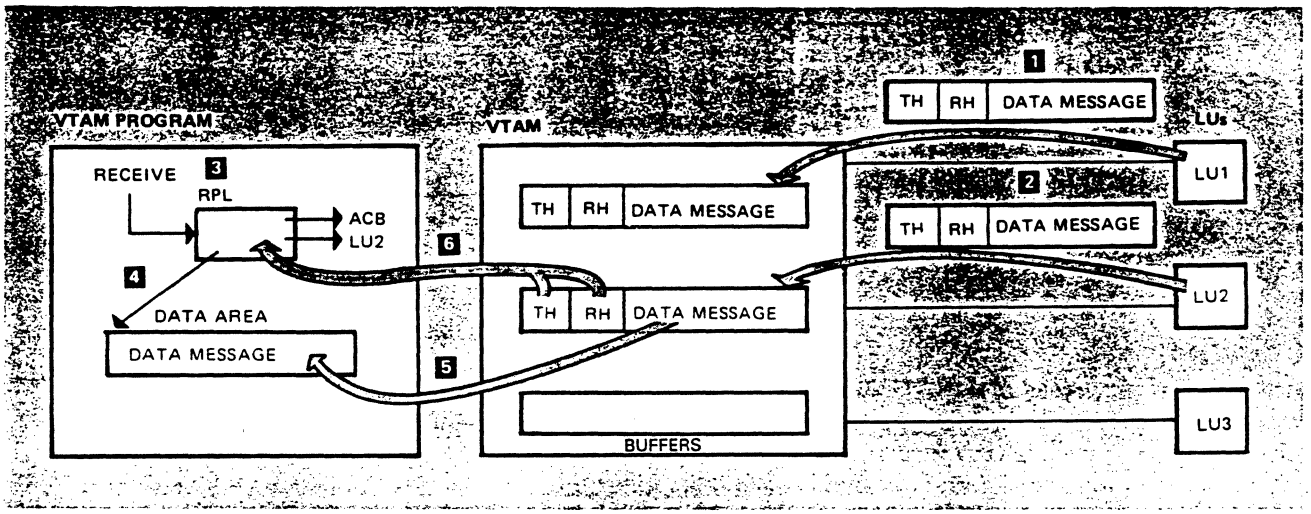

Figure 13-3. Receiving Message Requests From a Logical Unit

This figure shows VTAM receiving requests from LU1 (item 1) and from LU2 (item 2). The transmission header (TH) of each request contains the address of the destination VTAM program.

The destination VTAM program in our example shows a RECEIVE that indicates that it will accept a request from LU2 as specified in the RPL (item 3). The OPTCD field of the referenced RPL is set to the **SPEC** value

```
RPL fields
OPTCD=SPEC
```

The RPL also identifies a data area within the program that is to receive the data message from LU2 (item 4). VTAM completes the RECEIVE by placing the message in the program's data area (item 5) and placing information in the RPL (item 6).

The request from LU1 is also for this VTAM program. Another RECEIVE must be issued to obtain the request from LU1 and the RECEIVE may reference the

same RPL or a different RPL. If the same RPL is referenced, the VTAM program must place the CID of LU1 in the RPL before the RECEIVE macro instruction is executed. VTAM stores received requests until the destination VTAM program issues a RECEIVE instruction to accept it.

There is still another way that a RECEIVE macro instruction can obtain the two requests from LU1 and LU2. An RPL can be coded to accept input from any connected logical unit rather than a specific logical unit. This is shown in Figure 13-4.

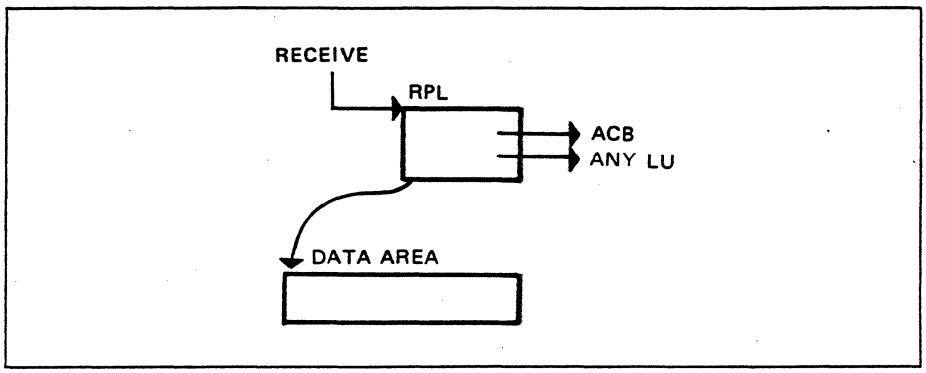

Figure 13-4. Obtaining Input From Any Logical Unit

The OPTCD field of the referenced RPL must be set to the ANY value in order to accept input from any logical unit that has a session with the associated VTAM program.

```
RPL fields

OPTCD=ANY
```

VTAM completes the RECEIVE by giving to the program the request that has been waiting the longest. VTAM also places the CID of the logical unit in the RPL so the VTAM program will know which logical unit sent the request. Each time that the RECEIVE is issued, it may obtain input from a different logical unit.

Sending and Receiving Responses

A VTAM program uses a SEND macro instruction to transmit a response. A VTAM program accepts a response from a logical unit with the following:

- A RECEIVE macro instruction
- An RPL referenced by a SEND macro instruction
- A response exit routine

Sending Responses

Figure 13-5 illustrates a VTAM program transmitting a response.

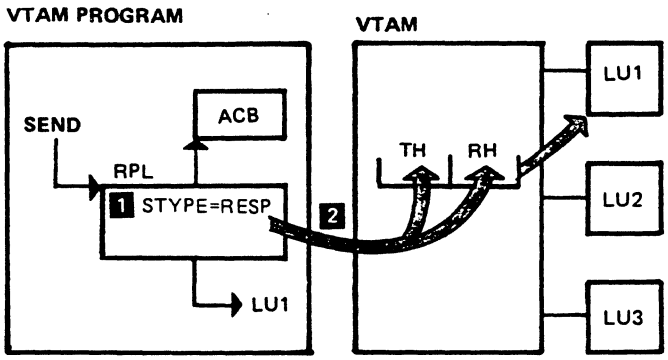

Figure 13-5. Transmitting a Response

The RPL specifies that a response (STYPE=RESP) is to be sent (item 1) and that the response is to be sent to LU1. You should notice that the RPL doesn't specify a data area. VTAM obtains all response data from the RPL and places it in a PIU (item 2) to be transmitted to LU1. All response information must be placed in the RPL by the VTAM program before the SEND is issued.

Receiving Responses

There are three ways that a VTAM program can obtain a response: (1) with the RPL that is referenced by the SEND macro instruction that sent the request, (2) by a RECEIVE macro instruction, and (3) by a RESP exit routine.

The method for obtaining a response depends on when the SEND RPL is freed to be used in another operation. An operand in the RPL is used to specify when VTAM is to free the RPL. POST=RESP says that VTAM is to free the RPL when a response is returned. POST=SCHED says that VTAM is to free the RPL before a response is returned.

The SEND RPL will obtain the response when POST=RESP is used. This is illustrated in Figure 13-6. Study the figure and then read the discussion that follows.

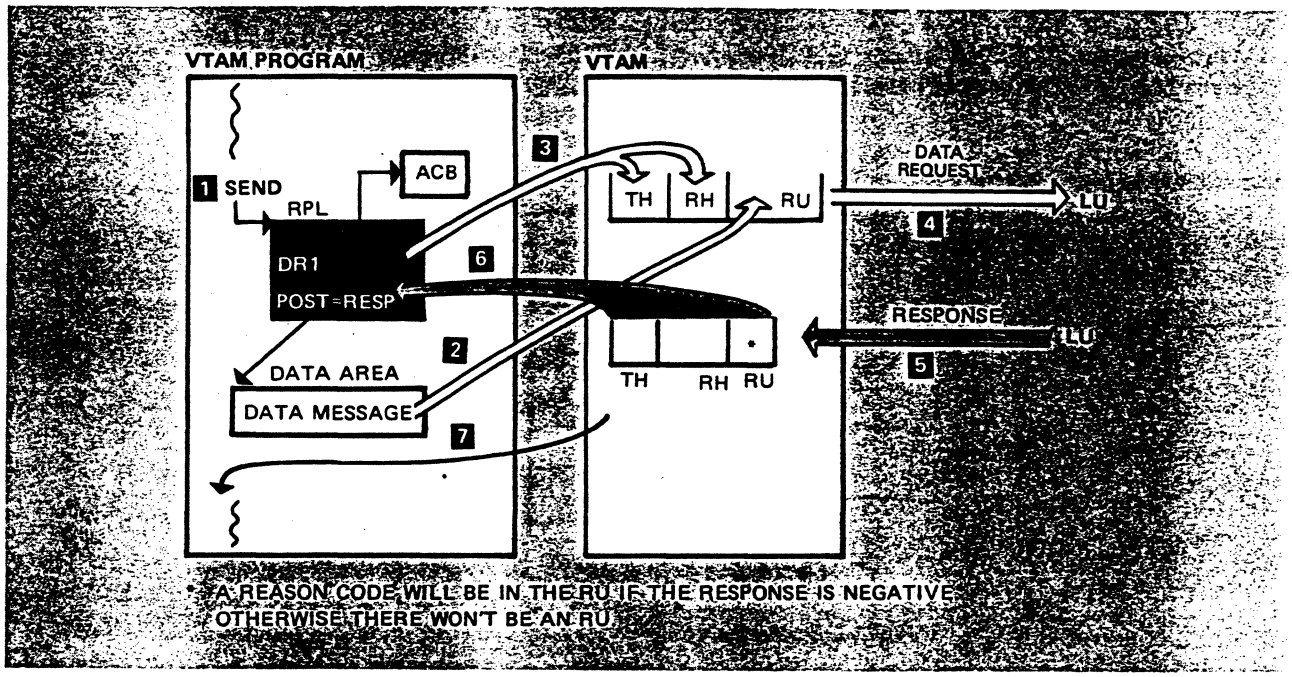

Figure 13-6. Obtaining a Response With a SEND RPL

The SEND macro instruction is executed to transmit a request to a logical unit (item 1). The RPL specifies POST=RESP and definite DR1.

The data message (item 2) and RPL information (item 3) are used by VTAM to build a PIU. VTAM transmits the PIU to the specified logical unit (item 4) and then receives a response from that logical unit (item 5). VTAM places the response information in the RPL referenced by the SEND (item 6). VTAM completes the operation and gives control to the VTAM program (item 7). The program can access the RPL to obtain the returned response information.

When is a RECEIVE macro instruction used to obtain a response? A RECEIVE macro instruction is used to obtain a response when the associated SEND macro instruction specifies POST=SCHED. POST=SCHED causes VTAM to free the SEND RPL before the response is returned. This means that the RPL will not be available to accept a response. Therefore, a RECEIVE macro instruction may be used to obtain the response as illustrated in Figure 13-7. (A response exit routine could be used instead of a RECEIVE.) Study the figure and then read the discussion that follows.

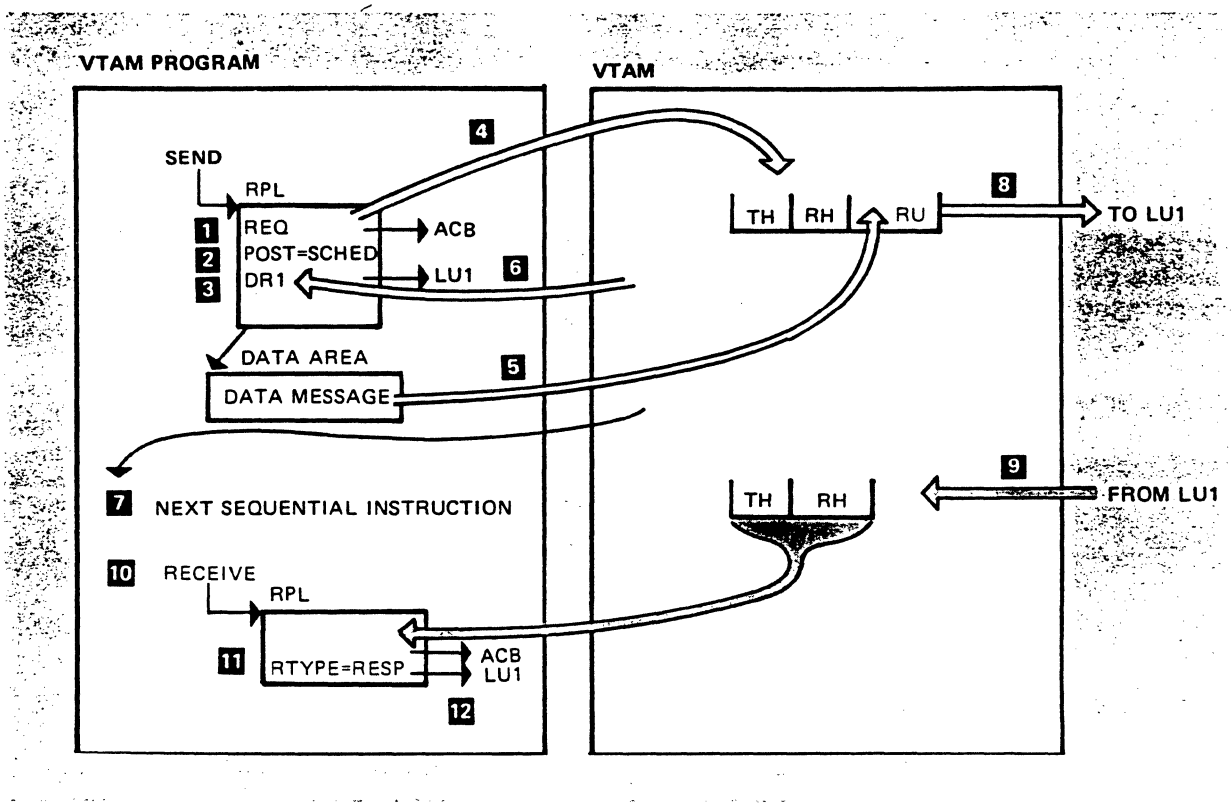

Figure 13-7. Obtaining a Response With a RECEIVE Macro Instruction

The SEND RPL specifies that a request is to be sent (item 1), specifies POST=SCHED (item 2), and indicates a definite DR1 (item 3). VTAM builds the PIU (item 4 and item 5), makes a few checks, and posts feedback information in the RPL (item 6). VTAM then gives control to the next sequential instruction in the VTAM program.

The SEND operation is now complete. The RPL is free to be used in another operation. The VTAM program can continue executing (item 7). Concurrently, the request is transmitted to LU1 (item 8) and LU1 returns a response (item 9). VTAM cannot place the response in the SEND RPL because it has been freed for use with other operations. VTAM looks for a RECEIVE that will accept responses or for a response exit routine. VTAM finds a RECEIVE in this example (item 10). The referenced RPL specifies that this RECEIVE will accept a response (item 11) from LU1 (item 12). VTAM moves the response data to the RPL.

The programmer could include a response exit routine to obtain responses instead of the RECEIVE. In that case, VTAM schedules the exit routine when a response is received from a logical unit. The routine determines which logical unit sent the response and then takes appropriate action.

Types of Flow

The exchange of requests during a session is divided into two flows, **normal-flow** and **expedited-flow**. Certain requests travel on the normal-flow and certain requests travel on the expedited-flow. In each direction of flow between session partners (primary-to-secondary and secondary-to-primary), the normal-flow requests and the expedited-flow requests are independently sequence numbered or identified.

Requests that travel on the expedited-flow can change the state of the normal-flow. For example, certain expedited-flow requests quiesce data traffic on the normal-flow, reset sequence numbers on the normal-flow, and terminate the LU-LU session. So control requests on the expedited-flow impose useful flow-controls on end-user traffic that travels on the normal-flow. Traffic on the expedited-flow is not restricted, allowing expedient and timely control over traffic on the normal-flow.

The normal-and expedited-flows on one session are independent of normal-and expedited-flows on other sessions.

Normal-Flow

Requests that are transmitted on the normal-flow arrive at their destination in the order in which they were sent. All data requests and some command requests are always transmitted on the normal-flow.

Expedited-Flow

A request that is transmitted on the expedited-flow can pass a request that is transmitted on the normal-flow, as illustrated in Figure 13-8. Study the figure and then read the discussion that follows.

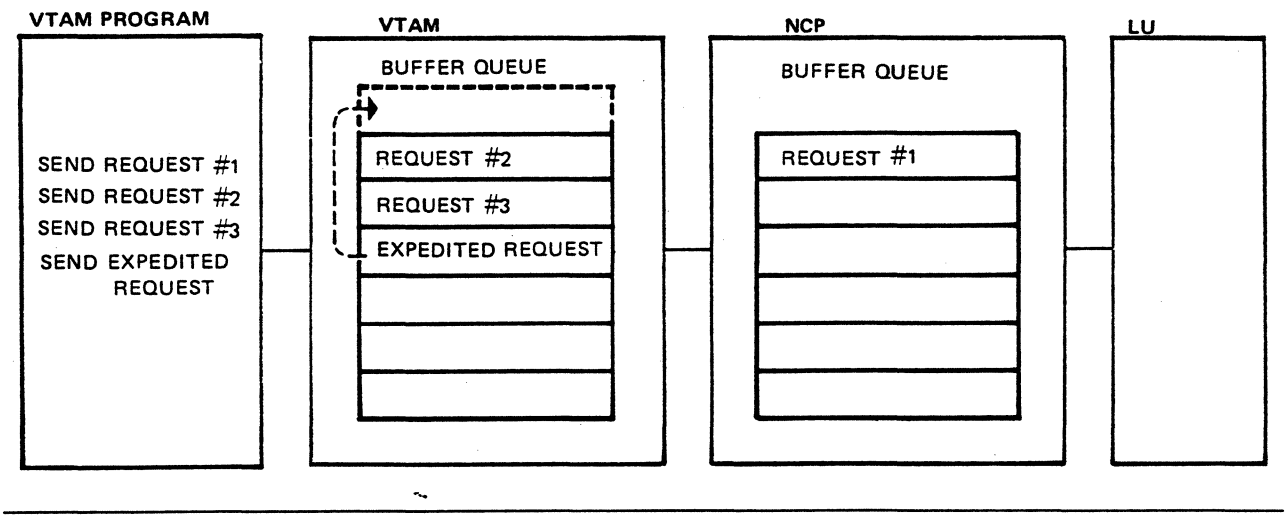

Figure 13-8. Sending on the Expedited-Flow

Message requests numbered 1,2, and 3 are sent on the normal flow. First, VTAM moves data request #1 to its own buffer and then sends it to the appropriate NCP where it is held in an NCP buffer. It is at the top of the buffer queue. The data request at the top of a buffer queue is the next one to be sent to the logical unit.

Data request #2 is sent next. VTAM puts the data request in one of its buffers and this request is now at the top of the VTAM buffer queue. Next, data request #3 is moved to a VTAM buffer. It is second in the queue behind request #2.

While data request #1 is still in an NCP buffer and data requests #2 and #3 are still in VTAM buffers, the VTAM program sends a request that is to be transmitted on the expedited-flow. The request is moved to a VTAM buffer and is put at the top of the queue. This means that the expedited-flow request will be the next one to be sent by VTAM.

There are two queue points where requests on the expedited-flow can pass requests on the normal-flow:

- Within VTAM
- Boundary function within NCPs

Certain SNA commands are transmitted on the expedited-flow. These commands are used for error recovery and for session control. Error recovery and session control require immediate communication between the VTAM program and the logical unit.

Please turn to Mini-Course 13 in your Personal Reference Guide and do Exercise 13.1.

VTAM Concepts

Mini-Course 14

Data Flow Control

MINI-COURSE 14. Data Flow Control

Introduction

A VTAM program may be in session with a number of logical units. Each logical unit may have different functional capabilities. The VTAM program may have to perform different operations in order to maintain data flow with each logical unit. For example, one logical unit may be able to send and receive multiple element request chains. Assuming that the application calls for request chaining, the VTAM program would have to contain the logic to send and receive multiple element request chains. At the same time, the program may be in session with another logical unit that cannot send or receive multiple element request chains, in which case the VTAM program would not have to support the chaining protocol for that session.

Each session may perform a different application. One session might perform a batch operation, while another session might perform an interactive operation, and so on. In the batch operation, the VTAM program might receive all the time. Therefore, the VTAM program would not be concerned about sending data requests. On the other hand, the VTAM program would have to send and receive data requests when performing an interactive operation. That means that the VTAM program and logical unit require a method to control when the VTAM program is allowed to send and when the logical unit is allowed to send.

The complexity of maintaining control of the data flow between a VTAM program and a logical unit depends on the functional capability of the logical unit and the application to be performed. A VTAM program and a connected logical unit must agree on protocol rules that each will abide by in order to maintain data flow. Each session might agree on a different group of protocol rules. The application to be performed dictates which protocol rules are required for this session. Therefore, it makes sense to combine protocol rules into sets; each set is appropriate for a particular session. A set of protocol rules is identified by a set of session parameters.

Supplied with each VTAM system is a logon mode table that contains sets of session parameters. The user may require combinations of session parameters that differ from those in the IBM-supplied logon mode table. The user can define and include one or more logon mode tables that contain combinations of session parameters of the user's choice. The VTAM program is a third source of session parameters; the program can generate sets of session parameters.

Session parameters identify protocol rules for controlling data flow in a session. The following protocols, specified by session parameters for use by the session, will be covered:

- Full-duplex communication
- Half-duplex communication
- Bracket protocol
- Changing the direction of data flow
- Quiescing data flow

Selecting Session-Communication Rules

Before discussing rules for controlling data flow in a session, notice how a VTAM program and a logical unit agree on a set of rules for data flow.

Session partners must agree on matters such as these:

- Will multiple element request chains be used?
- Will communication be in full duplex mode or will it be in half duplex mode?
- Will bracket protocol be used?
- Which session partner will be responsible for handling recovery should an error occur?

How does a VTAM program and a logical unit select a set of session parameters? You already know that there are three sources of session parameters:

- IBM-supplied logon mode table
- User-supplied logon mode table(s)
- The VTAM program

The logical unit can specify in a logon (by logmode name) a set of session parameters. The VTAM program, upon receipt of the logon, can examine the specified session parameters to determine if they're acceptable. If they are acceptable, the VTAM program issues an OPNDST macro to send a BIND request and establishes the LU-LU session. VTAM includes the session parameters in the BIND command. The logical unit checks these session parameters and returns a positive response if the parameters are acceptable. Otherwise the logical unit returns a negative response and the session is not established.

The VTAM program always has the ability to specify the set of session parameters to be included in the BIND request. The program can allow the ones specified in a logon to be used, specify a set in a logon mode table, or specify that session parameters located within the program's storage area are to be used.

Next we will examine some of the LU-LU session protocols that are established by the BIND request.

Full-Duplex Communication Protocol

A VTAM program and a logical unit can communicate in full-duplex mode, which imposes few restrictions on the two logical units. Both logical units are allowed to send at anytime, regardless of whether the other is sending. The following inquiry example illustrates full-duplex communication.

This example involves banking customers who want to find out their savings account balances. A bank teller receives a request for an account balance from a customer and submits the request from a terminal station. The request is formatted by the associated logical unit and sent to a VTAM program which gives the request to the appropriate application processing program. The application program accesses the customer's savings account records, determines the account balance, gives the information to the VTAM program and the VTAM program transmits the

reply to the logical unit. The logical unit, in turn, sends the information to the terminal station.

The bank teller doesn't have to wait for the reply to one request before submitting another request. The teller may submit many requests before the first reply is returned. This is full-duplex communication.

Although a VTAM network allows this kind of unrestricted full-duplex operation, the nature of many (and probably most) applications prohibits such unrestricted exchange of data. For that reason, VTAM provides methods of communication that allow the VTAM program and the logical unit to control each other's ability to send to each other.

Half-Duplex Communication Protocol

Half-duplex communication protocol is a way of restricting when session partners can send to each other. Half-duplex communication allows only one session partner to send at a time. Thus, at any given time, one session partner is a sender and the other session partner is a receiver. The session parameters in the BIND command specify which session partner can send first and which partner has priority to send when both issue requests at the same time.

A session may agree to use half-duplex flip-flop protocol or half-duplex contention protocol. The protocols are enforced by the VTAM program and the logical unit. VTAM does not enforce the protocols but it provides support for them in the VTAM macro instructions.

Half-Duplex Flip Flop Communication

Half-duplex flip-flop protocol allows only one session partner to send at any given time. In addition to specifying half-duplex flip-flop protocol, session parameters in the BIND request specify which session partner is the first speaker (that is, which partner is allowed to send first). We'll assume that the secondary logical unit is the first speaker.

When the LU-LU session is established, the logical unit can start sending requests to the VTAM program. Once the logical unit exhausts the information that it has to send, the change direction indicator (CDI) is turned on in the last request to notify the VTAM program that it is allowed to send. Once the VTAM program becomes the sender, the logical unit becomes the receiver. The sender maintains control and the receiver cannot send requests until the sender signals the receiver to start sending. Figure 14-1 illustrates half-duplex flip-flop communication. Study the figure, then read the discussion that follows.

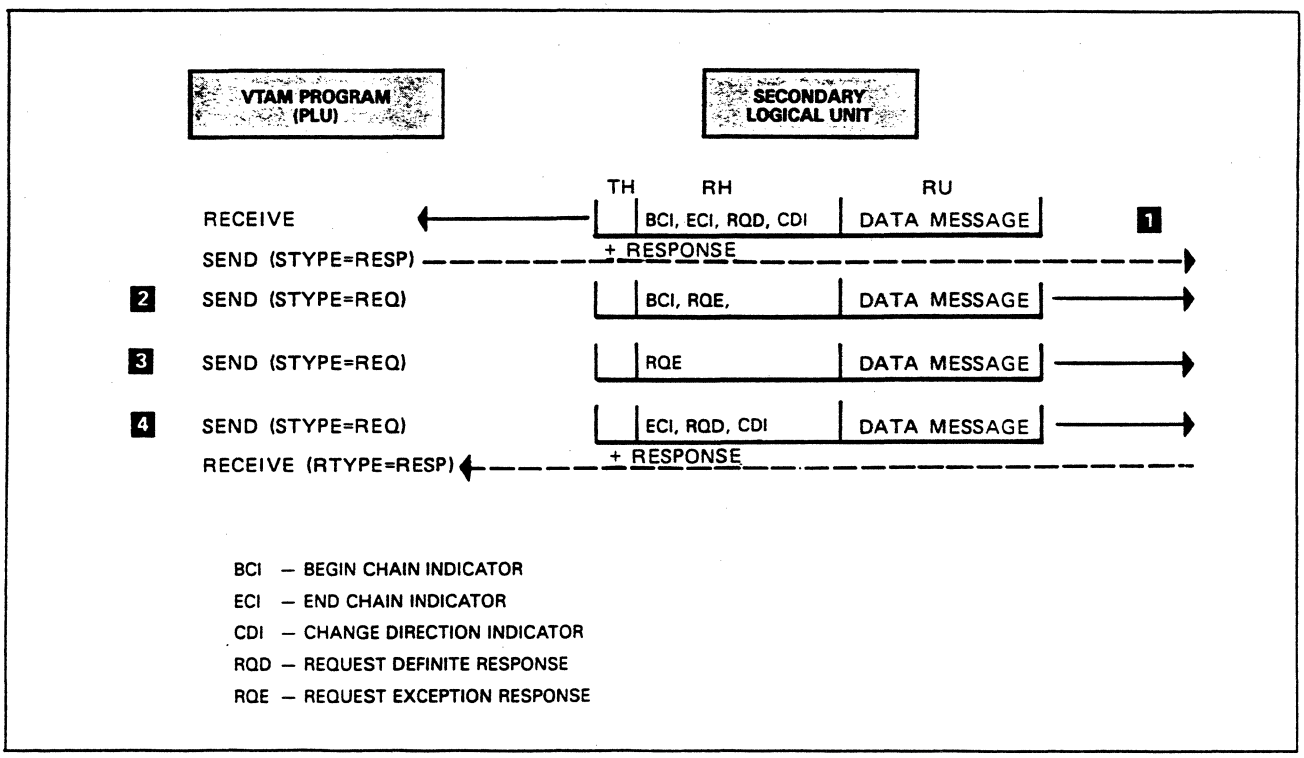

Figure 14-1. Half-Duplex Flip-Flop Communication

Since the logical unit is the first speaker, it is first to transmit a request (item 1). The indicators in the request header (RH), begin chain indicator (BCI), and end chain indicator (ECI) indicate that this is a single element request chain. A definite response is requested as indicated by the RQD notation. The change direction indicator (CDI) notifies the VTAM program that it can now send. The VTAM program accepts the request with a RECEIVE macro and the data in the request unit (RU) is placed in a specified data area within the program. The information from the TH and the RH is placed in the RPL referenced by the RECEIVE. The program can test the appropriate RPL fields to determine that it received a single element request chain, that a DR1 response must be returned, and that the VTAM program can now send.

The VTAM program now becomes the sender and the logical unit becomes the receiver. The VTAM program sends a three element request chain (items 2, 3, and 4). The third element contains a change direction indicator (CDI) to notify the logical unit that it can send. The role of each session partner continues to alternate between send and receive until the session is terminated. This protocol works only because both session partners enforce it.

The session partner that is currently a receiver is restricted from sending normal-flow requests. However, it is free to send responses and expedited-flow requests.

Half-Duplex Contention Communication

The other half-duplex protocol, half-duplex contention, is similar to half-duplex flip-flop in that only one session partner is allowed to send at a time. However, the method of determining which partner may send is different.

With contention mode, the VTAM program (primary logical unit) and the secondary logical unit are in contention to send when the session is established. The first one to attempt to send becomes the sender and the other the receiver. Both session partners return to contention status when the last element of a request chain is sent and received. This is true whether single element or multiple element request chains are transmitted.

What happens if both session partners send to each other at the same time? A session parameter in the BIND request specifies one of the session partners to win a contention. Let's assume that the secondary logical unit is specified to be the winner of a contention. If both session partners try to send at the same time, the logical unit's request will be accepted by the VTAM program, but the logical unit sends a negative response to reject the VTAM program's request.

The half-duplex contention protocol might be used when both session partners intermittently have data to send. Either partner can send without first getting permission to send.

Like the half-duplex flip-flop protocol, this protocol is enforced by the two session partners.

Bracket Protocol

The bracket protocol facility is used to identify a unit of work that must be completed before another unit of work is started. An inquiry from a logical unit to a VTAM program is one example of a unit of work where bracket protocol can be used. Once the inquiry is submitted, the logical unit wants to handle only the data associated with the inquiry. Completion of the inquiry could consist of a number of interchanges between the logical unit and the VTAM program.

For example, the logical unit sends an inquiry request to the VTAM program and the VTAM program gives the inquiry to the appropriate application program to process against a data base. The application program generates a reply and gives it to the VTAM program for transmission to the logical unit. The reply could result in several requests being sent to the logical unit. Assuming that the requests do not satisfy the inquiry, the logical unit sends another request asking for more data. Eventually the transaction or unit of work is completed.

In this example, even though there are multiple transmissions, all the transmissions are related to the initial inquiry. The logical unit isn't supposed to submit another inquiry before the current one is complete. Nor is the VTAM program supposed to transmit a request that isn't related to the present inquiry. The two session partners must agree on some method of control to ensure that the unit of work or transaction isn't interrupted by another unit of work or transaction.

Bracket protocol provides the facility to control the data flow in this manner. A begin bracket indicator (BBI) and an end bracket indicator (EBI) are used to start and end a bracket, respectively. These two indicators are transmitted in the request header (RH) of a request.

Figure 14-2 illustrates the use of bracket protocol. The figure shows the PIUs that are transmitted between the VTAM program and the logical unit. Study the figure and then read the discussion that follows.

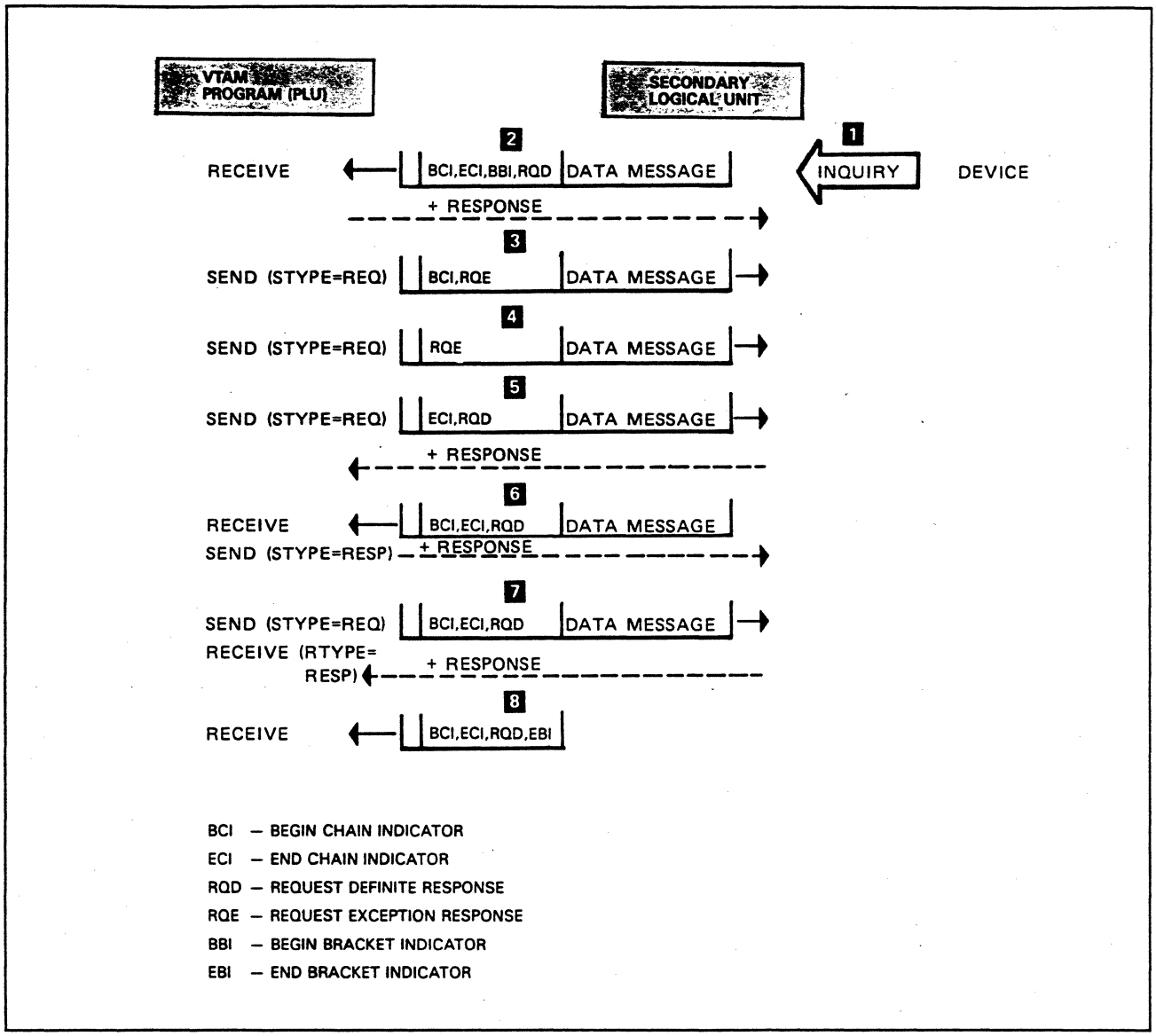

Figure 14-2. Bracket Protocol

An operator enters an inquiry from a work station (item 1). The logical unit transmits the request to the associated VTAM program (item 2). The RH indicates that this is a single element chain (BCI,ECI), and it indicates that this request begins a bracket (BBI). The VTAM program receives the request, and now the session is considered to be in an in-bracket state. Another bracket cannot be started until this bracket has been terminated.

The VTAM program gives the inquiry to the appropriate application program to process against a data base. The application program's reply results in the VTAM program returning three requests (items 3, 4, and 5). The three requests are transmitted in the form of a three-element request chain. The logical unit receives the three requests and determines that it needs more information. So it sends another request to the program asking for more information (item 6). This reply results in one request being transmitted to the logical unit (item 7).

The logical unit determines that this request completes the inquiry. Now the bracket can be terminated. The logical unit terminates the bracket by including an end bracket indcator (EBI) with the next request (item 8). Because there is no data to be sent, the PIU containing the EBI will have no RU. The session now is considered to be in a between-bracket state. Another bracket can now be started.

Session parameters in the BIND command specify which session partner is allowed to begin a bracket and which is allowed to end a bracket.

First Speaker: The logical unit or the VTAM program is referred to as *first speaker*. The first speaker is allowed to begin a bracket without requesting permission from its session partner.

Bidder: The VTAM program or the logical unit is referred to as the *bidder*. The bidder must request and receive permission from the first speaker to begin a bracket. The BID request is used by the bidder to obtain permission to begin a bracket.

Bidding to Begin a Bracket

We'll assume that the logical unit is the frst speaker and that the VTAM program is the bidder. Also, the logical unit is defined to end a bracket. The logic of this application requires that the logical unit identify each unit of work or transaction by beginning a bracket. However, there are occasions when the VTAM program needs to initiate a transaction, which means that the VTAM program will need to begin a bracket.

The VTAM program requests permission from the logical unit to begin a bracket by sending a BID request. The logical unit returns a positive response to accept the request and a negative response to reject the request. If the session is in the in-bracket state, the logical unit cannot give permission immediately. Permission can be given when the bracket has ended. The logical unit can convey this message to the program by including a predefined sense code in the negative response that is returned to the program. The sense code says that the program cannot begin a bracket now, but that the logical unit will send notification when the bracket can be started.

The session partners continue to process the present bracket to completion. At that time, the logical unit notifies the program that it can start a bracket. The logical unit does this by sending the ready to receive (RTR) request to the VTAM program. Upon receipt of the RTR request, the VTAM program may begin a bracket.

The BID and RTR requests are transmitted in the RU of a path information unit (PIU).

Figure 14-3 illustrates a session using bracket protocol. Study the figure and then read the following explanation.

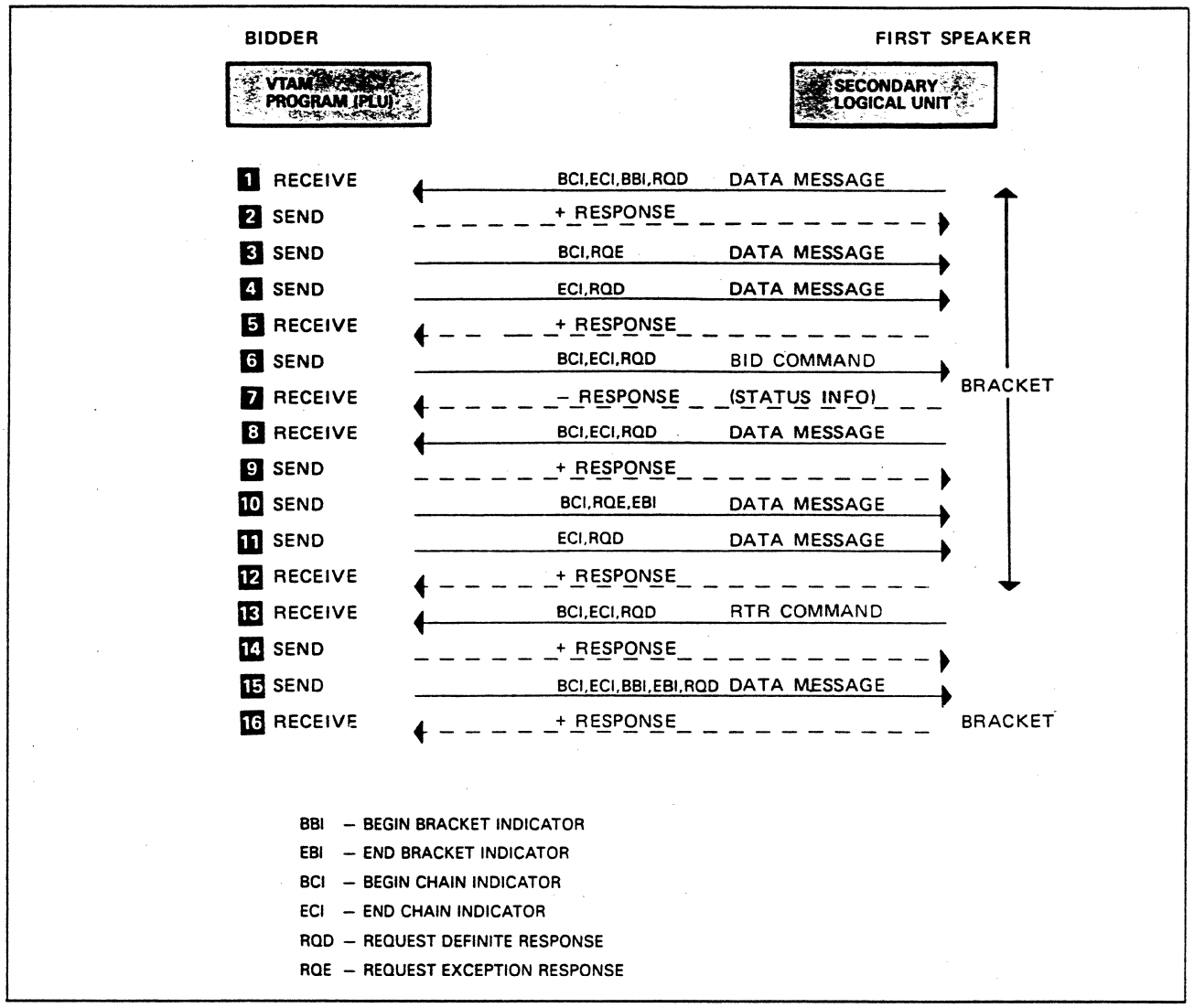

Figure 14-3. Bracket Protocol (BID and RTR Usage)

The logical unit sends a request and begins a bracket at item 1. The VTAM program replies with a two-element request chain (items 3 and 4). The VTAM program now wants to send a request that is unrelated to the bracket. It requests permission to start a bracket by sending the BID request to the logical unit (item 6). The logical unit receives the BID but will not give permission to start a bracket, because the present bracket has not been completed. The logical unit sends a negative response to inform the program that it cannot start a bracket at this time (item 7). Status information included in the negative response lets the program know that later it will be given permission to start a bracket.

The processing of the bracket continues to completion (items 8 - 12). The logical unit then sends an RTR request to the program, which tells the program that it can begin a bracket (item 13). The program sends one request and indicates that the single request begins and ends the bracket (item 15).

A bracket spans one or more request chains transmitted in both directions. That is, a bracket may consist of several interchanges of request chains by two session partners. A session partner begins a bracket by including the begin bracket

indicator (BBI) on the first chain element of the bracket. A session partner signals termination of a bracket by including the end bracket indicator (EBI) in the last request chain of the bracket. If the last request chain contains multiple elements, the BB indicator must be included in the first element of that chain. The bracket is not terminated until the last element of the chain has been transmitted and received.

Quiescing Data Request Flow

Quiescing or temporarily stopping the flow of data requests in one direction is another way to control data flow within a session. Three commands can be used to achieve this control:

- Quiesce at end of chain (QEC)
- Quiesce complete (QC)
- Release quiesce (RELQ)

Let's consider one example of why a session partner might need to quiesce data request flow. A logical unit may want to stop data request flow from the VTAM program temporarily because the request is too fast to handle or because the logical unit needs to do some local processing before more data is received.

How is quiescing accomplished? Assume that the VTAM program is presently sending and the logical unit wants to tell the program to stop sending for a while. The logical unit sends the QEC request to the VTAM program, thereby requesting the VTAM program to stop sending after it completes the request chain that is being sent.

The VTAM program sends the remainder of the current request chain and then sends the QC request to the logical unit. This says that the VTAM program is in a quiesced condition. In effect, the VTAM program and the logical unit switch roles. That is, the VTAM program becomes the receiver and the logical unit becomes the sender. The logical unit may or may not send requests. What the logical unit does depends on why the VTAM program was quiesced. In any case, the VTAM program expects to eventually receive the RELQ request to release it from the quiesced condition. The RELQ request causes the VTAM program and the logical unit to revert back to their previous roles. The VTAM program again becomes the sender while the logical unit becomes the receiver.

Figure 14-4 illustrates the quiesce data flow control function.
Study the figure and then read the following discussion.

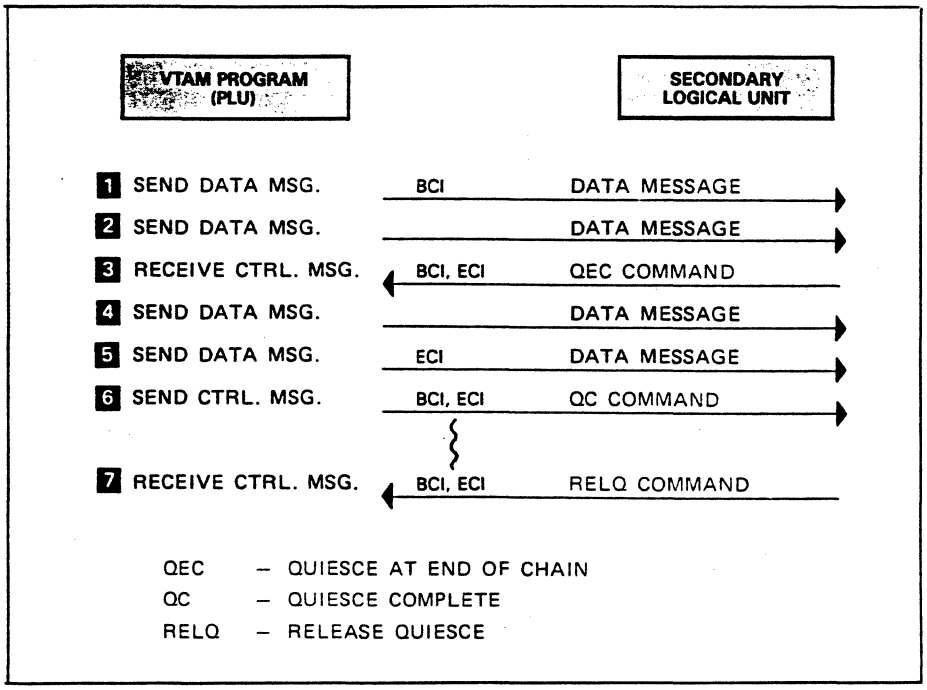

Figure 14-4. Quiesce Data Flow Control

This figure shows a VTAM program sending a multiple element request chain to a logical unit, at which time the logical unit needs to quiesce the program. The logical unit sends the quiesce at end of chain (QEC) request to the VTAM program (item 3). The program receives the QEC request, but continues sending the request chain (items 4 and 5). The program sends the quiesce complete (QC) request when it has completed sending the request chain (item 6). The QC request informs the logical unit that the quiesce is complete. That is, the VTAM program is in a quiesced condition and is ready to receive if the logical unit wants to send a request. The logical unit wanted to temporarily stop request flow from the VTAM program so that it could process the backlog. After the logical unit processes the backlog, it sends the release quiesce (RELQ) request. to the VTAM program (item 7). This removes the VTAM program from the quiesced condition, allowing the program to start sending again.

Data Flow Control

The data flow protocols just discussed are implementations of systems network architecture (SNA). The architecture defines the restrictions imposed on each session partner: it defines when and what a session partner can send, and it defines the indicators and commands to be used to implement each protocol.
The protocols just discussed are enforced by the two session partners. Therefore you need to know what a VTAM program can send and receive when using any of these protocols.

What follows is a general discussion of what the VTAM program can send and receive for the various protocols. First, let's consider the transmission of data requests.

Sending Data Requests

Figure 14-5 illustrates the sending of a data request. Study the figure and then read the discussion that follows.

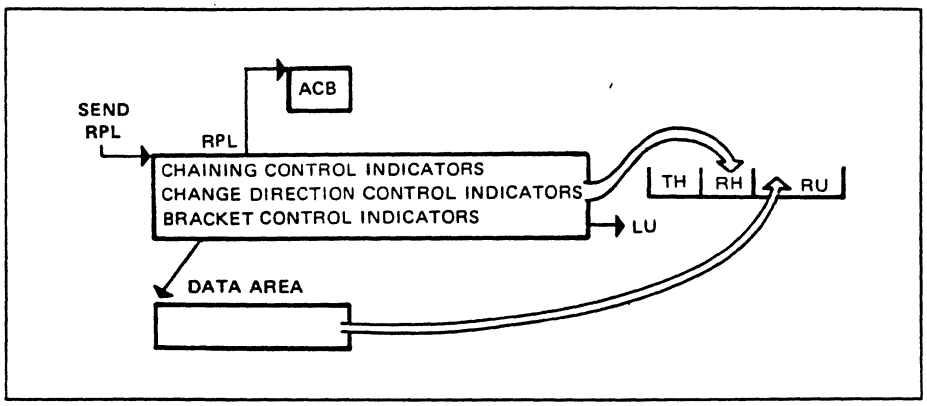

Figure 14-5. Sending a Message Request

The VTAM program issues a SEND macro instruction to transmit a data request. The SEND must identify an RPL to be used for this operation. The program must place the appropriate information in the RPL before the SEND is executed.

The RPL can contain information such as the following:

- Chaining indicators
- Change direction indicators
- Bracket control indicators

VTAM takes all of these control indicators from the RPL and places them in the RH of the request. It is the responsibility of the logical unit to examine these indicators and take appropriate action.

Sending Command Requests

Command requests are transmitted by the SEND macro instruction. This is illustrated in Figure 14-6. Study the figure and then read the discussion that follows.

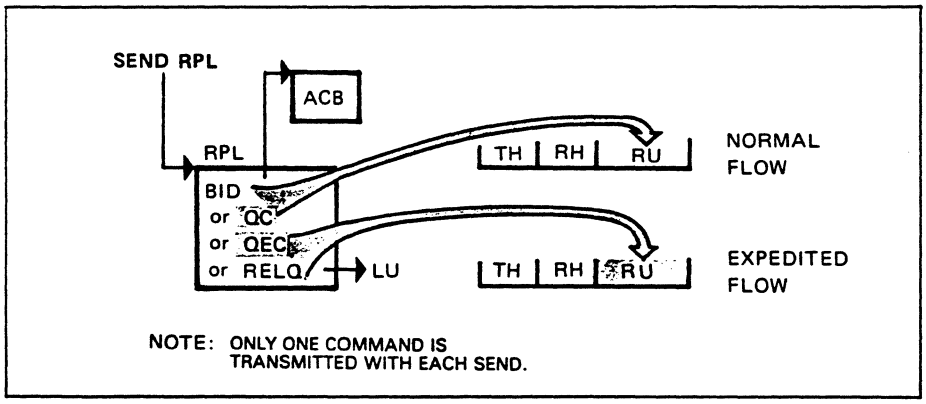

Figure 14-6. Sending Command Requests

A SEND macro instruction can transmit one SNA command. A command is transmitted on either normal-flow or expedited-flow. VTAM takes care of transmitting a command on the appropriate flow (normal or expedited). The QEC

and RELQ commands are transmitted on the expedited-flow while the BID and QC commands are transmitted on the normal-flow.

Receiving Command Requests

Figure 14-7 illustrates a VTAM program receiving command requests.

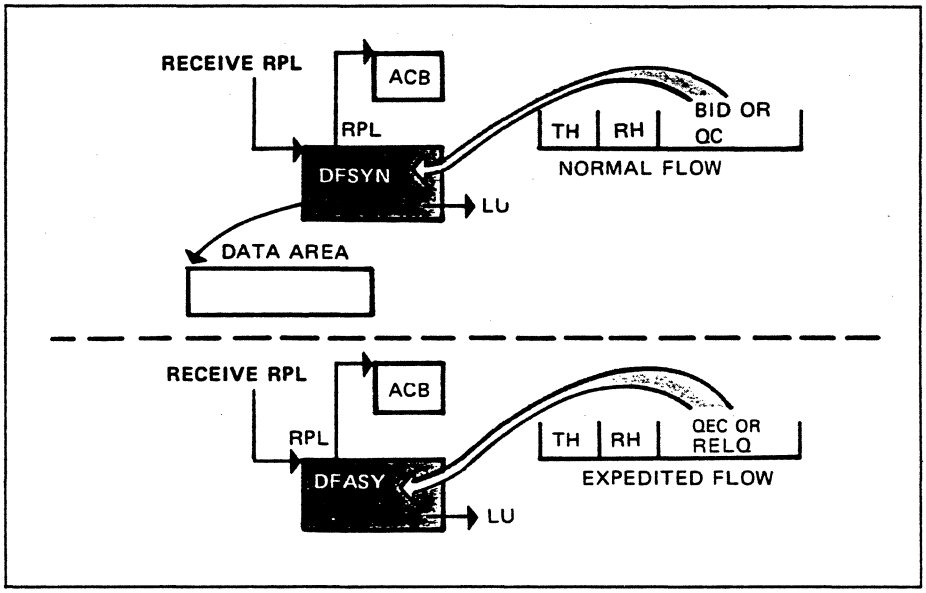

Figure 14-7. Receiving Command Requests

The upper part of Figure 14-7 shows a RECEIVE macro instruction that will accept normal-flow requests as specified by the RPL parameter DFSYN. Because there is no data message, nothing will be placed in the data area specified by the RPL. The RECEIVE macro instruction shown at the bottom of the figure will accept an expedited-flow request as specified by the RPL parameter DFASY. The RPL does not specify a data area because an expedited-flow request does not contain a data message.

When and what can a VTAM program send for the various protocols? The following descriptions concentrate on this question.

Sending and Receiving When Quiesced

What is a quiesced VTAM program allowed to send and receive? It can receive all requests and responses while quiesced. It can send responses and expedited-flow requests, but not normal-flow requests.

Sending and Receiving in Half-Duplex Mode

A VTAM program participating in half-duplex flip-flop communication can receive all types of information at any time. It can send responses and expedited-flow requests at any time. Normal-flow requests can be sent only when the program is in sender status. Either the VTAM program or the secondary logical unit becomes the sender at the time the session is established. Session parameters specify which is the sender.

Both session partners can send at any time when communicating in half-duplex contention mode. Session parameters specify which partner will win a contention when both partners send simultaneously.

Sending and Receiving When Using Brackets

Bracket protocol does not restrict a session partner from sending and receiving normal-flow requests, expedited-flow requests, and responses. It specifies that only one bracket can be active within a session at a given time. Also, it specifies which session partner can begin or end a bracket.

Please turn to Mini-Course 14 in your Personal Reference Guide and do Exercise 14.1

VTAM Concepts

Mini-Course 15

VTAM Macro Execution Sequence
and Error Notification

MINI-COURSE 15. VTAM Macro Execution Sequence and Error Notification

Introduction

In this mini-course, we will first consider the transfer of control between the VTAM program and VTAM. Then we will look at how VTAM notifies the program about an error.

Transfer of Control

Control is transferred from the VTAM program to VTAM when the program issues a VTAM macro instruction. The program regains control at different times, according to whether a request is handled synchronously or asynchronously.

Synchronous Request Handling

Synchronous request handling means that VTAM returns control to the next sequential instruction (NSI) in the VTAM program only after the requested operation has completed. Program execution is halted until VTAM determines that the operation has completed. Figure 15-1 shows a VTAM program issuing a SEND macro instruction that is to be handled synchronously. The SYN parameter in the macro instruction indicates synchronous.

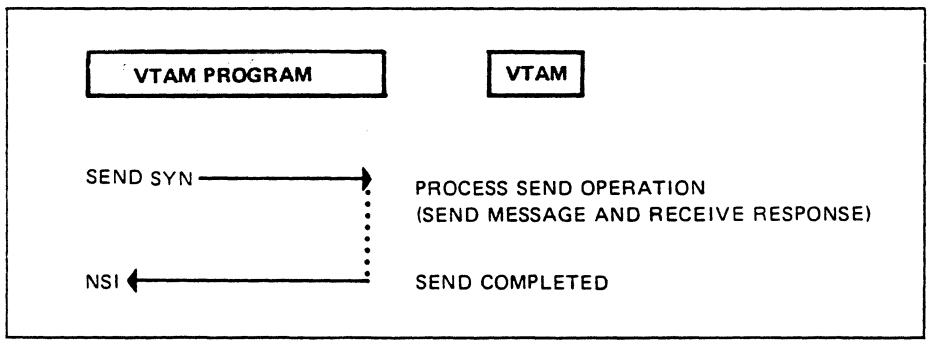

Figure 15-1. Synchronous SEND Request

The SEND macro instruction causes control to be given to VTAM. VTAM processes the request and returns control to the program only when the SEND operation is completed. Control is returned to the next sequential instruction.

Asynchronous Request Handling

Asynchronous request handling means that VTAM returns control to the next sequential instruction in the VTAM program as soon as VTAM has accepted the request, **not** when the requested operation has been completed. Accepting the request consists of screening the request for errors. While the requested operation is being completed by VTAM, the VTAM program is free to initiate other I/O transactions or processing. For example, a VTAM program might issue a RECEIVE macro instruction and indicate that it is to be handled asynchronously; while the input operation is being completed, the VTAM program can begin to do other processing or communicate with another logical unit.

How does VTAM notify a VTAM program that an asynchronous operation has completed? That depends on specifications within the VTAM program. A VTAM program can specify for each request either an event control block (ECB) or an RPL exit. VTAM will post the ECB complete (when provided), or schedule the RPL exit (when provided) upon completion of the requested operation. First we'll discuss requests that specify an ECB and then the use of RPL exit routines.

Using an ECB: Figure 15-2 illustrates an asynchronous (ASY) SEND request that specifies an ECB.

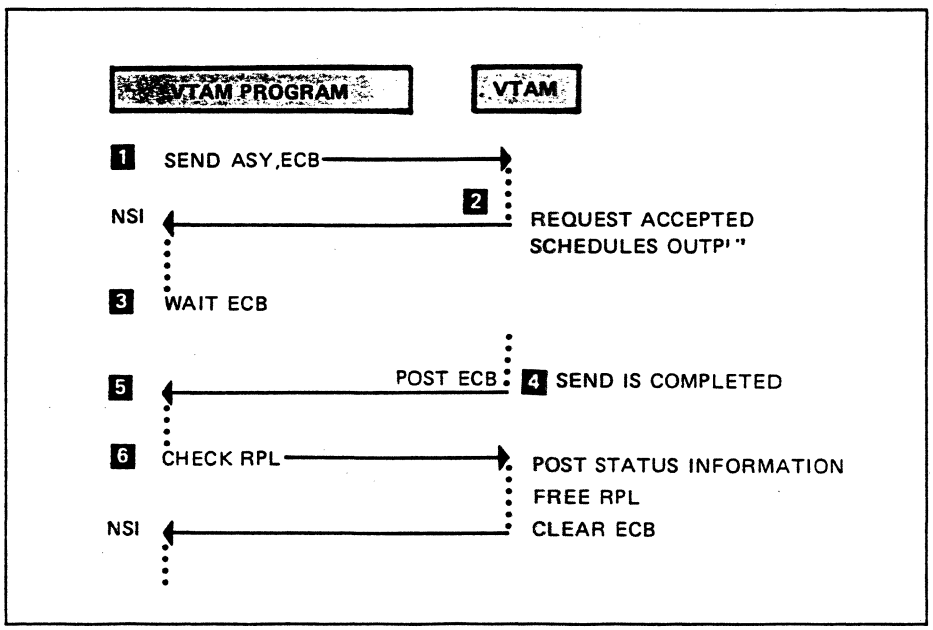

Figure 15-2. Asynchronous SEND Request (ECB)

The SEND causes control to be given to VTAM (item 1). VTAM screens the request for errors and returns control to the next sequential instruction (item 2) following the SEND. The VTAM program continues processing concurrently with the SEND operation. The program issues a WAIT ECB when it needs to know the result of the SEND operation (item 3). Upon completion of the SEND (item 4), VTAM posts the ECB and returns control to the instruction following the WAIT (item 5).

The VTAM program must issue a CHECK macro instruction to cause VTAM to post status information, free the RPL, and clear the ECB (item 6). Both the RPL and the ECB can now be reused. VTAM then returns control to the VTAM program at the instruction following the CHECK macro instruction.

The VTAM program can use one WAIT macro instruction for a combination of VTAM and non-VTAM requests that use ECBs. For example, a VTAM program can issue three VTAM asynchronous requests and three requests for DISK I/O. By issuing one WAIT for all six ECBs, the program can continue processing when any one of the six operations completes.

Using an RPL Exit Routine: Now we'll consider an asynchronous SEND that specifies an RPL exit routine. Study the asynchronous SEND operation shown in Figure 15-3 and then read the discussion that follows.

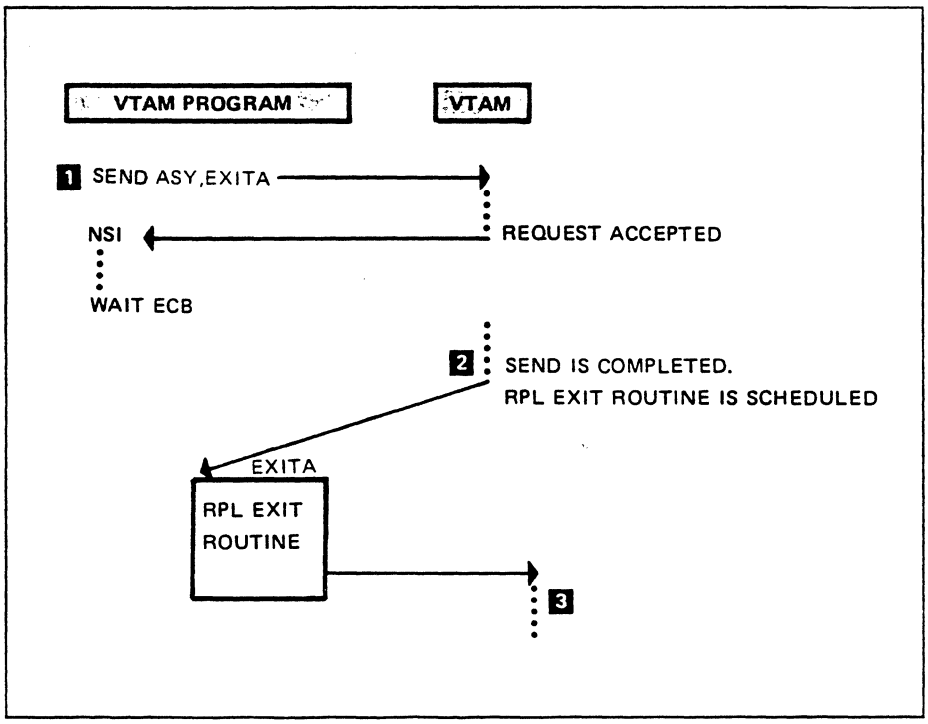

Figure 15-3. Asynchronous SEND Request (RPL Exit Routine)

The SEND macro instruction specifies an RPL exit routine at the label EXITA (item 1). This routine is to be given control when the SEND operation completes.

The SEND causes control to be given to VTAM (item 1). VTAM screens the request for errors and returns control to the next sequential instruction following the SEND. The VTAM program continues processing concurrently with the SEND operation. When the program has no more data to process, it issues a WAIT on an ECB. VTAM program execution is suspended until an outstanding operation completes.

When the SEND operation completes, VTAM schedules the RPL exit routine EXITA. The user-written RPL exit routine executes, then returns to VTAM. VTAM returns control to the VTAM program when another operation completes.

Freeing the RPL and Clearing the ECB

An RPL becomes active when the associated macro instruction is issued. The RPL cannot be used in another operation until it is set inactive by the CHECK macro instruction.

VTAM CHECKs the RPL at the completion of a synchronous operation, setting the RPL inactive. Therefore you should *not* CHECK an RPL that was used in a synchronous operation. CHECKing an inactive RPL gives a logical error.

The VTAM programmer is responsible for CHECKing an RPL that was used in an asynchronous operation. When should the CHECK be issued? That depends on whether the operation used an ECB or an RPL exit routine.

When an ECB is specified, you have two options as to when to CHECK an RPL: (1) after a WAIT or (2) without WAITing on the associated ECB. Normally a WAIT is included to WAIT on the ECB. Execution falls through the WAIT after

the requested operation completes and the ECB is posted. You would CHECK the appropriate RPL after the WAIT. An alternative is to CHECK the RPL without a WAIT on the associated ECB. This could be inefficient since execution is suspended until the operation completes. Upon completion of the operation, the ECB is posted and the CHECK operation can complete. The CHECK clears the ECB, frees the RPL (marks the RPL inactive), and posts feedback information.

When an RPL exit routine is specified, normally the CHECK is issued in the routine. The CHECK marks the RPL inactive and posts feedback information.

Responded and Scheduled SEND Requests

You have read the description of synchronous and asynchronous request handling for requests that specify either an ECB or an RPL exit routine. A synchronous or asynchronous request indicates the action that the program can take while waiting for the request to complete. Synchronous or asynchronous can be specified in any RPL-based request.

There are two other parameters that may be specified in a SEND request: responded and scheduled. They identify when the SEND is considered complete.

Responded Output

Responded output is specified in a SEND macro instruction by including POST=RESP in the referenced RPL. VTAM considers a responded SEND complete when a response is returned from the logical unit. The RPL and the output data area specified in the SEND macro instruction are not reuseable until a response has been received from the logical unit. If the response indicates that an error occurred, the data is still available for retransmission.

Since a responded SEND completes when a response is returned, RPL coding that specifies a definite response should be included when a responded SEND is specified.

Scheduled Output

Scheduled output is specified in a SEND macro instruction by including POST=SCHED in the referenced RPL. Requesting a response is optional. A SEND macro instruction that includes POST=SCHED is considered complete as soon as the request has been scheduled for transmission and the output data area is free. The SEND and the referenced RPL are no longer associated with the data being transmitted. You must provide a way, other than the SEND RPL, to obtain response information if a response is requested. You can do one of two things to obtain response information: (1) include a RECEIVE macro instruction that specifies RTYPE=RESP, or (2) include a response exit routine. Both of these will accept incoming responses. You will read a few SEND examples for both responded and scheduled output.

Synchronous Responded SEND

Figure 15-4 illustrates a synchronous SEND operation that specifies responded output. This means that the SEND operation is to be considered complete when the response is returned to the VTAM program.

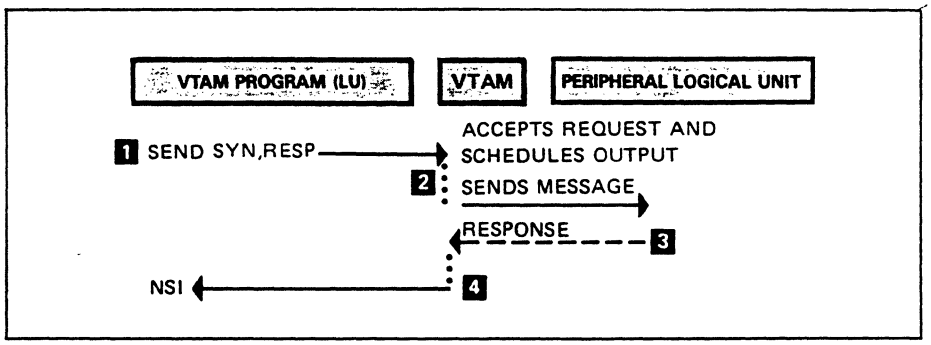

Figure 15-4. Synchronous Responded SEND

The SEND macro instruction (item 1) causes control to be given to VTAM. VTAM screens the request for errors, schedules the request to be transmitted, and transmits the request to the peripheral logical unit (item 2). The peripheral logical unit receives the message and returns a response (item 3). VTAM posts the response information in the RPL and returns control to the next sequential instruction (item 4) in the VTAM program. The program can now examine the response information.

Synchronous Scheduled SEND

Figure 15-5 illustrates a synchronous SEND operation that specifies scheduled output. The SCHED parameter directs VTAM to consider the operation complete as soon as the operation is scheduled. As a result, the VTAM program regains control as soon as VTAM schedules the request to be transmitted. Study the figure, then read the discussion that follows.

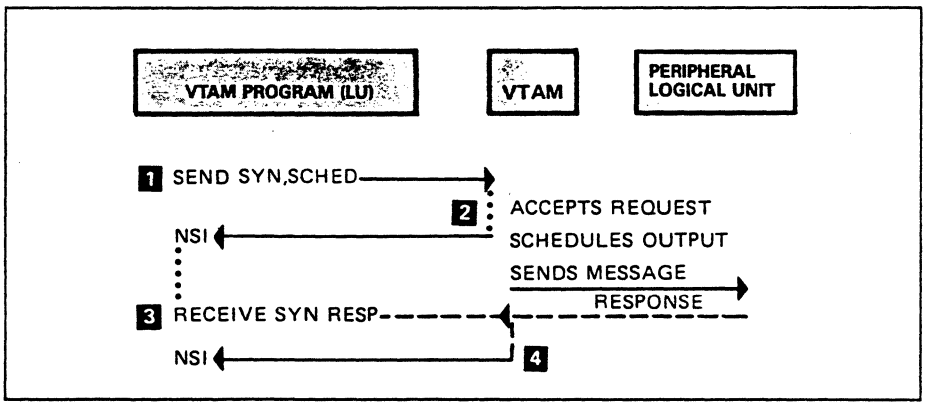

Figure 15-5. Synchronous Scheduled SEND

The SEND macro instruction (item 1) causes control to be given to VTAM. VTAM screens the request for errors, and schedules the request to be transmitted. VTAM posts the RPL and returns control to the next sequential instruction (item 2) in the VTAM program. The program can now examine information in the RPL and do other processing.

VTAM considers the operation complete now that the request has been scheduled for transmission and the RPL may be reused. VTAM transmits the request and accepts the response when it is returned from the peripheral logical unit. What does VTAM do with the response, given that the SEND operation has already completed? The VTAM program must make provisions to obtain the response

information from VTAM. The program can issue a RECEIVE that specifies RTYPE=RESP as shown in the figure (item 3). A synchronous RECEIVE is used in the example for the sake of simplicity. VTAM completes the RECEIVE by placing the response information in the RECEIVE RPL and returns control to the next sequential instruction (item 4) in the VTAM program.

The VTAM program can use a response exit routine to obtain the response information rather than an inline RECEIVE macro instruction. The program can maintain a single response exit routine (identified in ACB) to handle all responses, or a response exit routine can be maintained for a specific logical unit (identified in NIB). Figure 15-6 illustrates the use of a response exit routine.

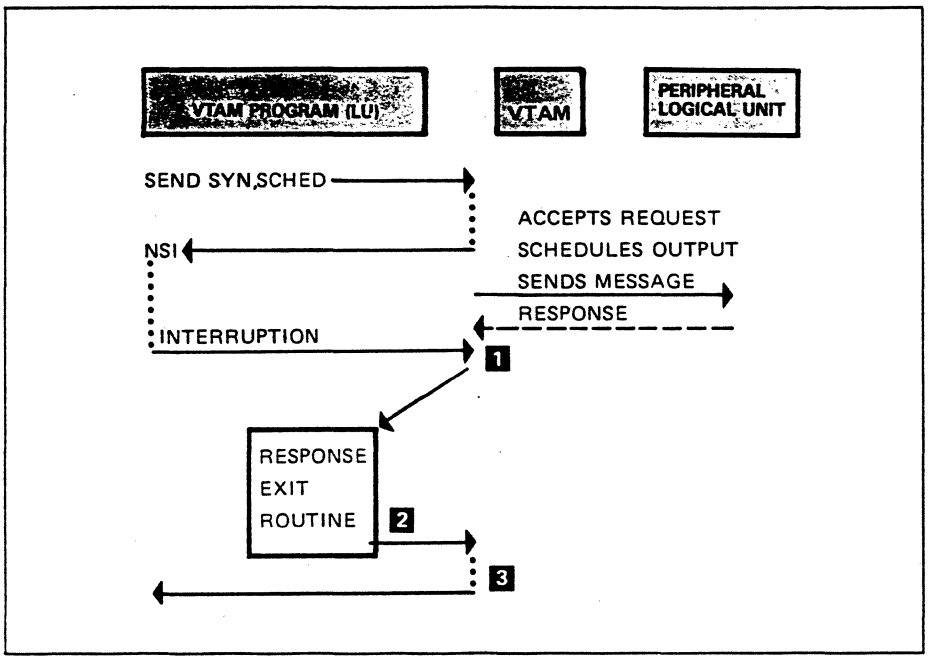

Figure 15-6. Specifying a Response Exit Routine

The SEND operation in this figure is the same as the one in Figure 15-5, up to the point where the logical unit returns the response to VTAM. Assuming that main-line code is executing, VTAM interrupts the VTAM program and gives control to the response exit routine (item 1).

VTAM makes the response information available to the exit routine. The routine processes the information and returns control to VTAM (item 2). VTAM then returns control to the program at the point where it was interrupted (item 3).

Asynchronous Responded SEND

Figure 15-7 illustrates an asynchronous SEND operation that specifies responded output. This directs VTAM to consider this SEND operation complete when the response is returned to the VTAM program. The SEND also specifies an ECB.

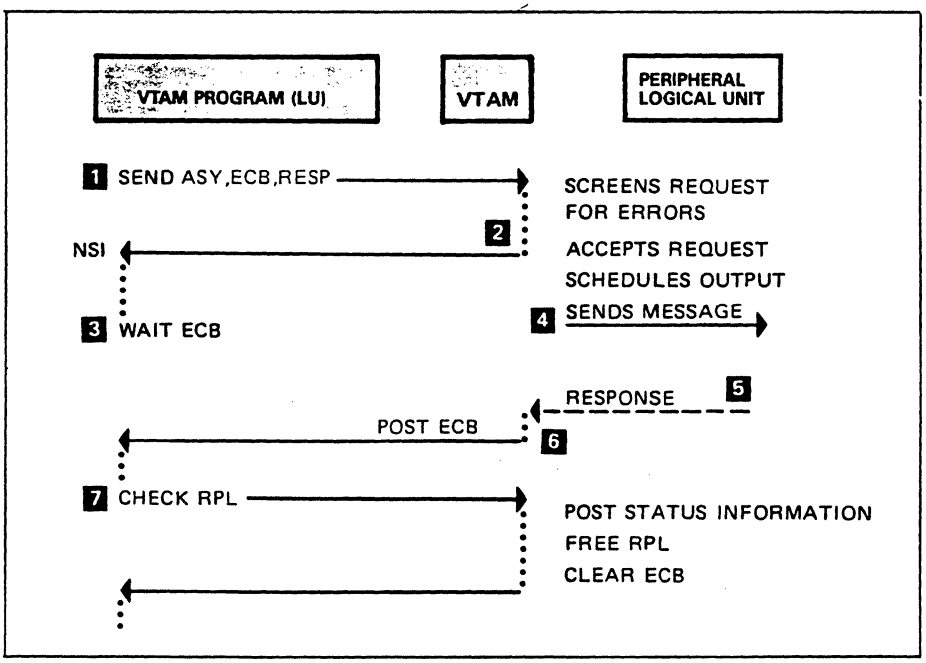

Figure 15-7. Asynchronous Responded SEND

The SEND macro instruction causes control to be given to VTAM (item 1). VTAM screens the request for errors and returns control to the VTAM program at the next sequential instruction (item 2). The RPL cannot be reused because the operation is not yet complete, but the program can continue executing. Concurrent with VTAM program execution, VTAM schedules the request to be transmitted and transmits it (item 4). At some point during execution, the program issues a WAIT on the ECB (item 3).

The peripheral logical unit receives the request and returns a response (item 5). VTAM then posts the ECB and gives control to the VTAM program (item 6).

The VTAM program issues a CHECK macro instruction to request VTAM to post status information, free the RPL, and clear the ECB (item 7).

Asynchronous Scheduled SEND

Figure 15-8 illustrates an asynchronous SEND operation that specifies scheduled output. This directs VTAM to consider the SEND operation complete when the request is scheduled to be transmitted. The SEND also specifies an ECB.

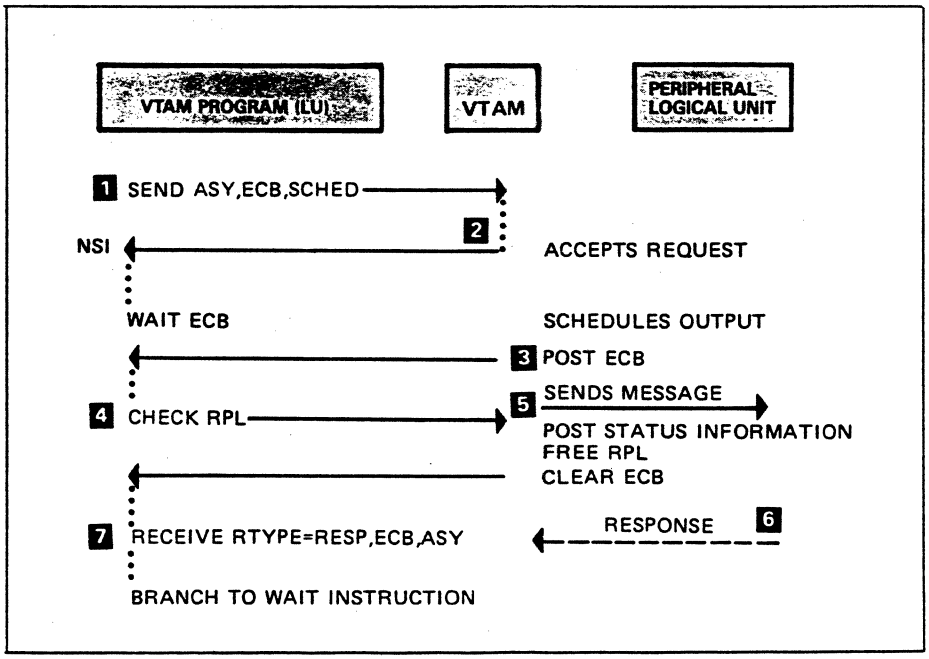

Figure 15-8. Asynchronous Scheduled SEND

The SEND macro instruction in Figure 15-8 causes control to be given to VTAM (item 1). VTAM screens the request for errors and returns control to the next sequential instruction in the VTAM program (item 2). The program can continue processing but the operation is not yet considered complete.

VTAM schedules the request to be transmitted and posts the ECB (item 3). Execution in the VTAM program falls through the WAIT, and the program CHECKs the RPL (item 4). As a result, VTAM posts status information, frees the RPL, and clears the ECB. VTAM returns control to the VTAM program and the program continues processing. Concurrently, VTAM sends the request (item 5) and the peripheral logical unit returns a response (item 6). The program has an outstanding RECEIVE that accepts responses (item 7). VTAM completes that RECEIVE with the returned response.

Error Notification

Next you will see how VTAM notifies the VTAM program of any errors that occur when a request is processed. Error notification varies according to whether a request is handled synchronously or asynchronously. For synchronous request handling, error conditions are reported when the request has been completed and control is returned to the VTAM program.

For asynchronous request handling, error conditions are reported in two stages. When control is first returned to the VTAM program, VTAM indicates whether it has accepted or rejected the request. When an accepted request has been completed, VTAM either posts the specified ECB or schedules the designated RPL exit routine. VTAM returns information about the completion of the requested operation when the VTAM program issues a CHECK macro instruction.

Information about success or failure is sent to the VTAM program in register 15 and, under some circumstances, in register 0. If the request is rejected or the operation is unsuccessful, VTAM normally places additional information in return code fields of the RPL and may invoke one of the two types of error-handling exit

routines that the user can furnish (LERAD and SYNAD). The LERAD routine handles logic errors and the SYNAD routine handles non-logic errors.

VTAM automatically invokes the appropriate routine when an error occurs during a synchronous operation. This is *not* the case for asynchronous operations. For asynchronous operations, the LERAD or SYNAD exit routine is entered at either of two times: (1) if the request has been rejected or (2) if the request completes unsuccessfully. An exit is driven when a CHECK macro instruction is issued. The invoked routine has access to register 0 and RPL fields that contain information regarding the specific cause of error. Register 15 contains the address of the exit routine.

Now we'll discuss how VTAM notifies the VTAM program about errors that occur during ACB-based macro operations or RPL-based macro operations. Figure 15-9 shows VTAM providing error information to both types of macro instructions.

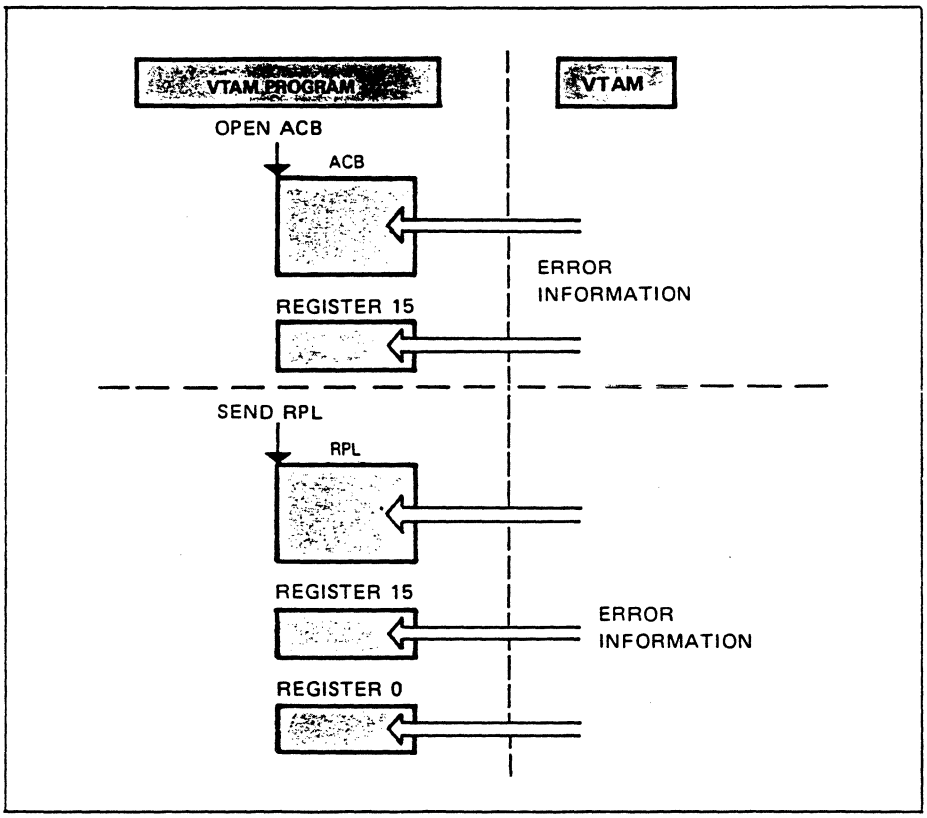

Figure 15-9. Error Notification

The upper part of the figure shows VTAM returning error information for an ACB-based macro operation. VTAM places error information in the ACB and a nonzero value in register 15 for OPEN ACB and CLOSE ACB macros. Register 15 will contain a zero if there is no error, otherwise it will be nonzero.

The lower part of the figure shows VTAM returning error information for an RPL-based macro instruction. Error information is placed in the RPL and in registers 0 and 15. Register 15 will contain a zero if there is no error, otherwise it will be nonzero. Register 0 will contain a code that indicates the error category.

OPEN and CLOSE Macro Instructions

This description concerns the OPEN and CLOSE macro instructions. The OPEN macro instruction identifies a VTAM program to VTAM, while the CLOSE macro instruction disassociates the program from VTAM.

Control is returned to the next sequential instruction following completion of the OPEN/CLOSE operation. First consider the OPEN operation. The VTAM program needs to know if the OPEN operation completed successfully. If not, the program needs to know why not.

The OPEN macro instruction causes control to be given to VTAM. VTAM processes the request, posts any error information in the ACB, sets register 15 to a nonzero value if an error occurs, and returns control to the instruction following the OPEN. Figure 15-10 illustrates how to test for an OPEN error.

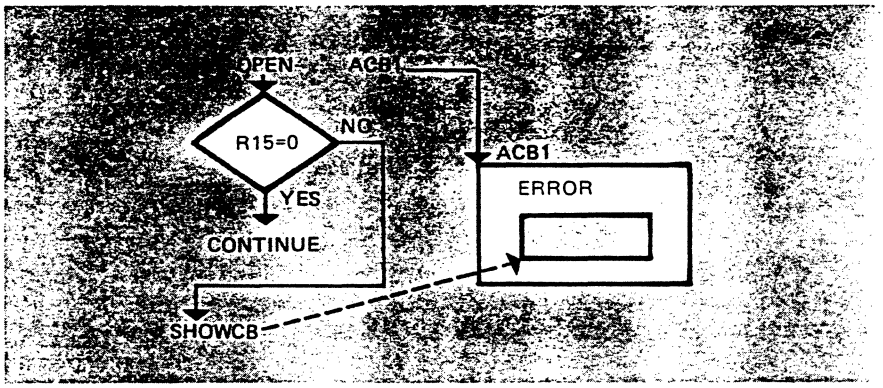

Figure 15-10. OPEN Errors

The VTAM program can test register 15 to determine if there was an error (register 15 is nonzero). The ACB error field will contain an error code if there was an error. The VTAM program can examine the appropriate ACB field to determine the reason for failure.

RPL-Based Macro Instruction

For synchronous RPL-based operations, the VTAM program is notified automatically of an error when the operation completes. Figure 15-11 illustrates error notification for a synchronous operation. Error exits are not used in this example. Study the figure and then read the discussion that follows.

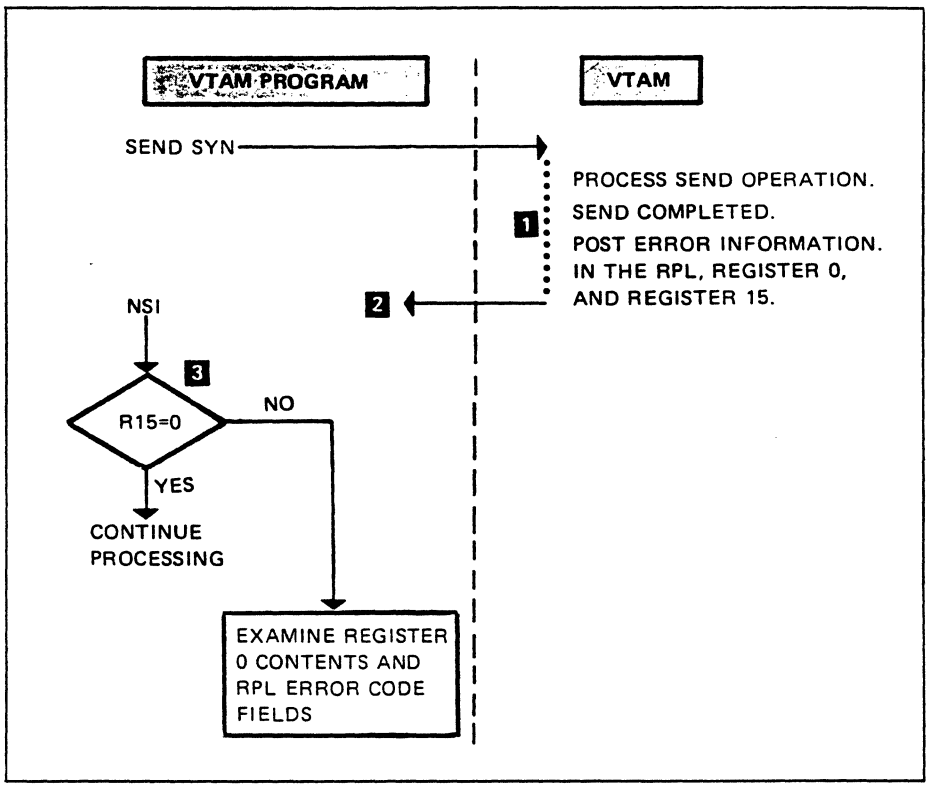

Figure 15-11. Error Notification For a Synchronous SEND Operation

VTAM posts error information in the RPL, register 0, and register 15 upon completion of the operation (item 1). VTAM returns control to the next sequential instruction (item 2). The program tests register 15 to find out if there was an error (item 3). If there was an error, then register 0 and RPL error code fields can be examined to determine what caused the problem.

Error notification for asynchronous request handling is different. VTAM can post error information at two different times for an asynchronous request. The first time that an error may be posted is when VTAM screens a request for acceptability. VTAM posts an error if it rejects the request. The second time that an error may be posted is when the requested operation completes. Figure 15-12 shows an asynchronous SEND that specifies an ECB. The figure shows when error information may be posted by VTAM and when it may be examined by the VTAM program. Study the figure and then read the discussion that follows.

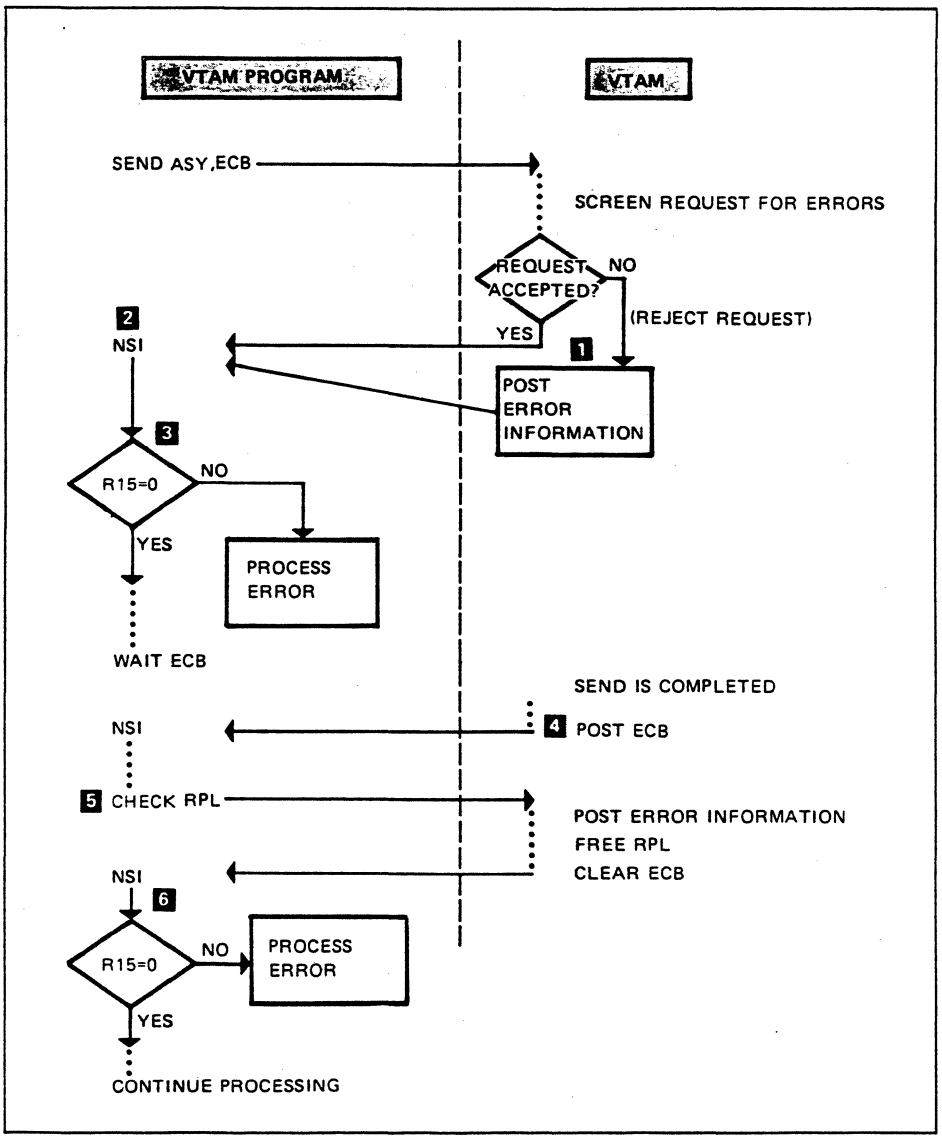

Figure 15-12. Error Notification For an Asynchronous SEND Operation

VTAM receives the SEND request and screens it for errors. If there is an error, VTAM does not process the request. Instead, it posts error information in the RPL, in register 15, in register 0 (item 1), and then returns control to the next sequential instruction (item 2). The program tests register 15 (item 3), detects an error, and processes the error.

Now go back to the top of the figure and this time assume that VTAM accepted the SEND request. VTAM returns control to the program (item 2), and the program tests register 15 for an error (item 3). There is no error because VTAM accepted the request. The request completes, VTAM posts the ECB, and then returns control to the program (item 4). At some time, the VTAM program issues a CHECK macro instruction to cause VTAM to post error information, free the RPL, and clear the ECB (item 5). Control is returned to the next sequential instruction where the program will make its error checks (item 6).

The VTAM program could use an RPL exit routine rather than an ECB. In that case, the sequence of events would be the same up to the point where the request is

completed. Rather than post the ECB, VTAM would schedule the RPL exit routine. The exit routine could issue the CHECK macro instruction and process the error (if any).

LERAD and SYNAD Exit Routines

The LERAD and SYNAD exit routines can be used for error notification and processing. Both types of routines are used the same way, except that they are used for different types of errors. The LERAD is invoked if a logic error occurs and the SYNAD routine is invoked if a non-logic error occurs. Let's see what happens with the LERAD exit routine as the example. The sequence is exactly the same if the SYNAD routine is invoked.

VTAM invokes the LERAD exit routine when a logic error occurs. Register 0 and the RPL are available at entry to the routine. Therefore the routine has available the necessary information to process the error.

Consider how the LERAD routine fits in a synchronous operation. You know that control isn't returned to the VTAM program until the operation is completed. That means that VTAM receives a VTAM request, screens it for errors, and schedules the LERAD exit routine if there is a logical error. Otherwise VTAM processes the request. If a logical error occurs at request completion, VTAM also schedules the LERAD exit routine. In either case the LERAD routine processes the error and returns to VTAM. VTAM then returns control to the next sequential instruction in the VTAM program. This means that when the VTAM program regains control, the error routine has already been executed if there was an error.

Figure 15-13 illustrates the use of the LERAD exit routine in an asynchronous operation. Study the figure and then read the discussion that follows.

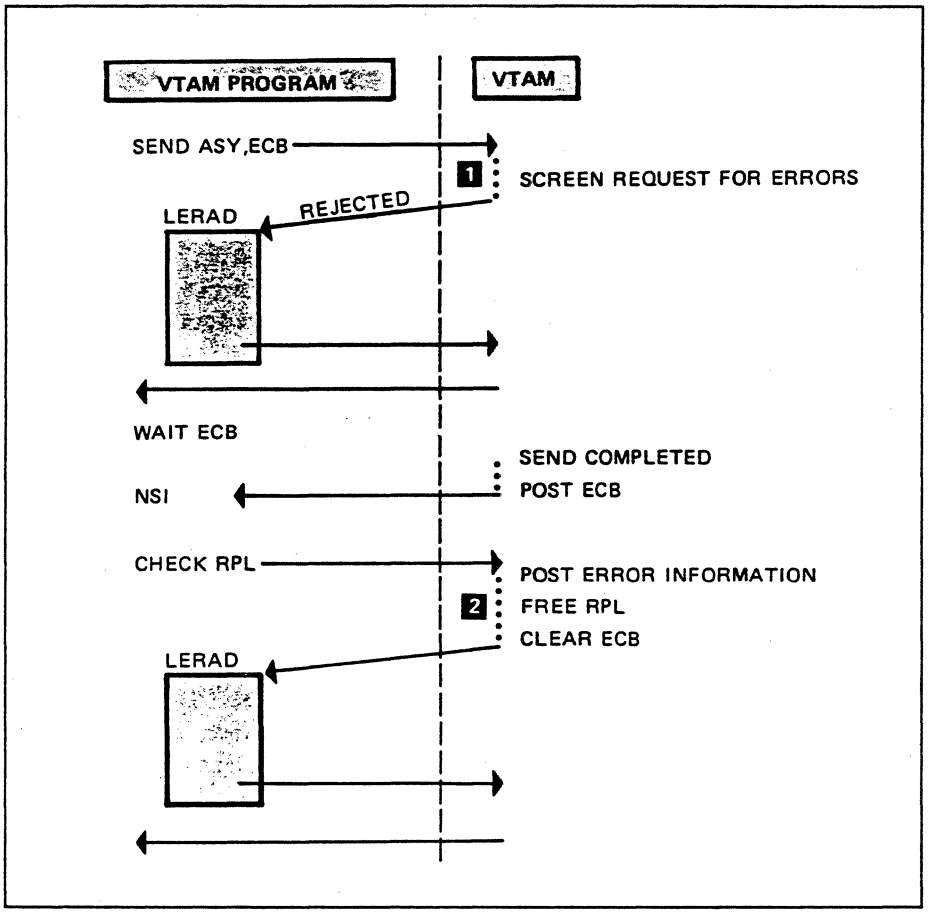

Figure 15-13. LERAD Exit Routine

The figure shows that the LERAD exit routine can be scheduled for execution if the request was rejected (item 1). VTAM automatically schedules the routine at this time.

The lower part of the figure shows that the LERAD routine also can be invoked when an accepted request completes (item 2). The CHECK macro instruction causes VTAM to schedule the LERAD exit routine when a request completes with a logic error. So we can say that CHECK invokes the LERAD exit routine. CHECK also invokes the SYNAD exit routine when a request completes with a non-logic error.

NOTE: Most logic errors are not recoverable because it is unlikely that the error exit could correct a programmer error.

Please turn to Mini-Course 15 in your Personal Reference Guide and do Exercise 15.1.

VTAM Concepts

Mini-Course 16

Multiple Domain Communication

MINI-COURSE 16. Multiple Domain Communication

Introduction

Everything that you learned about single domain networks in the previous mini-courses is applicable to multiple domain networks. This mini-course discusses those things that are unique to multiple domain networks.

As you will recall, a single domain network includes one SNA access method (ACF/VTAM in our examples). The access method (VTAM) monitors and controls all of the network resources. A multiple domain network includes two or more SNA access methods, for example, ACF/VTAM and ACF/TCAM. In this mini-course we will only be concerned with VTAM networks.

Each VTAM establishes ownership and controls a portion of the network. A domain consists of a VTAM and the resources that are owned and controlled by that VTAM. A multiple domain network typically consists of multiple host processors with a VTAM in each processor. However, a host processor that runs under control of VM can support multiple VTAMs.

VTAM domains may be connected to each other by processor channels or by SDLC links. Once VTAM domains are interconnected by links and appropriate network definitions have been installed, logical units in one domain can establish sessions and communicate with logical units in other domains.

You will read how domains are interconnected, the relationship between domains and subareas, network definitions required for resources in one domain to communicate with resources in other domains, and how cross-domain sessions are established and terminated.

Domains

Figure 16-1 illustrates a network that consists of three domains. Study the figure, then read the discussion that follows.

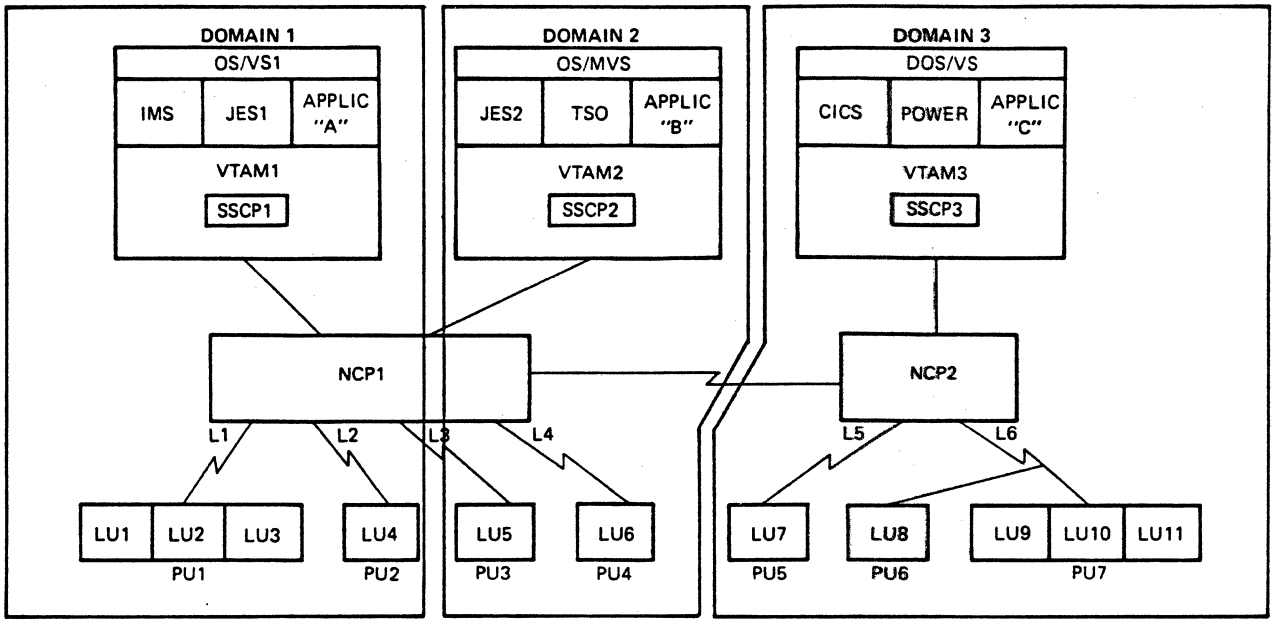

Figure 16-1. Three Domain Network

Domain 1 consists of a host subarea node with a VTAM access method and a type 4 subarea node with an NCP. The resources in VTAM1's domain include: IMS, JES1, APPLIC "A", NCP1, links L1 and L2, PUs PU1 and PU2, and LUs LU1, LU2, LU3, and LU4. All of these resources are defined to VTAM1, therefore VTAM1 can activate and control them.

Domain 2 consists of a host subarea node with a VTAM access method and a type 4 subarea node with an NCP. The resources include: JES2, TSO, APPLIC "B", NCP1, links L3 and L4, PUs PU3 and PU4, and LUs LU5, and LU6. These resources are defined to VTAM2, therefore VTAM2 can activate and control them.

As you can see, NCP1 resides in domain 1 and in domain 2, which means that an NCP can have multiple simultaneous owners. In addition to NCPs, non-switched SDLC links can have multiple simultaneous owners. All other resources can have only one owner at a time.

Domain 3 consists of a host subarea node with a VTAM access method and a type 4 subarea node with an NCP. The resources include: CICS, POWER, APPLIC "C", NCP2, links L5 and L6, PUs PU5, PU6, and PU7, and LUs LU7, LU8, LU9, LU10, and LU11. These resources are defined to VTAM3, therefore VTAM3 can activate and control them.

Connecting Domains

Figure 16-1 illustrates the different ways of interconnecting domains. There are two connections between domains 1 and 2.

1. Processor channel. The two host processors are connected by a processor channel.

2. Shared NCP. Both host processors are channel-attached to the communications controller which is controlled by NCP1. Therefore, the two domains are connected by the shared NCP.

Domain 3 is connected to domains 1 and 2 by an SDLC cross-subarea link, the link that connects NCP1 and NCP2.

So a domain can be connected to another domain by one of the following:

1. Processor channel

2. SDLC cross-subarea link

3. Shared NCP

Establishing Ownership

This is a review of how a VTAM access method establishes ownership of resources that are defined to VTAM. Use domain 1 (Figure 16-1) as the example.

The three host logical units (IMS, JES1, and APPLIC "A") issue an OPEN macro to identify themselves to VTAM1, which, in effect, establishes three sessions:

1. SSCP1-IMS

2. SSCP1-JES1

3. SSCP1-APPLIC "A"

Now the three resources are under control of SSCP1 (VTAM1).

SSCP1 establishes ownership and control of the other logical units (LUs) physical units (PUs), and links with the following requests:

- NCP1 with an ACTPU request
- Each link (L1 and L2) with ACTLINK requests
- Each PU (PU1 and PU2) with a CONTACT and ACTPU requests
- Each LU (LU1, LU2, and LU3) with ACTLU requests

SSCP1 must establish ownership of the resources in the following sequence:

- NCP1 (PU.T4)
- Links
- Physical Units (PU.T1s and PU.T2s)
- LUs

Now assume that VTAM1, VTAM2, and VTAM3 have established ownership and control over all of their resources. LU1, LU2, LU3, and LU4 may establish sessions and communicate with IMS, JES1, and APPLIC "A" in domain 1. LU5 and LU6 may establish sessions and communicate with JES2, TSO, and APPLIC "B" in domain 2. And LU7, LU8, LU9, LU10, and LU11 may establish sessions and communicate with CICS, POWER, and APPLIC "C" in domain 3.

Cross-Domain Resources

Now that the three domains are interconnected by links, the logical units in each domain should be able to establish sessions and communicate with logical units in the other two domains. In order for logical units in domain 1 to initiate sessions with logical units in the other two domains, the logical units in domains 2 and 3 (called cross-domain resources to domain 1) must be defined to VTAM1 in domain 1. The same is true for domains 2 and 3. For logical units in domain 2 to initiate sessions with logical units in domains 1 and 3, the logical units in those two domains must be defined to VTAM2 in domain 2. And the logical units (LUs) in domains 1 and 2 must be defined to VTAM3 for the LUs in domain 3 to initiate sessions with the LUs in domains 1 and 2.

Cross-Domain Resource Managers

You know that the logical component in VTAM that is the central point of control for the network is called the systems service control point (SSCP). Among other things, the SSCP assists logical units to establish sessions with other logical units in the domain. For example, a peripheral logical unit submits a logon to its SSCP requesting a session with a VTAM program. Both logical units must be in session with the SSCP for the session to be established.

Now consider what happens when that same peripheral logical unit submits a logon to its SSCP requesting a session with a VTAM program in another domain. Unless the SSCP knows about the VTAM program in the other domain, the logon request will fail.

In addition to being aware of the logical unit in the other domain, the SSCP must be in communication with the SSCP that owns that logical unit for the cross-domain session to be established. A session must be established between the two SSCPs for the logical units in one domain to establish sessions with logical units in the other domain.

An SSCP has two logical components. One is called the SSCP and it manages the initiation and termination of sessions within its domain. The other is called the cross domain resource manager (CDRM) and it handles the initiation and termination of cross-domain LU-LU sessions. Figure 16-2 shows a CDRM in each domain.

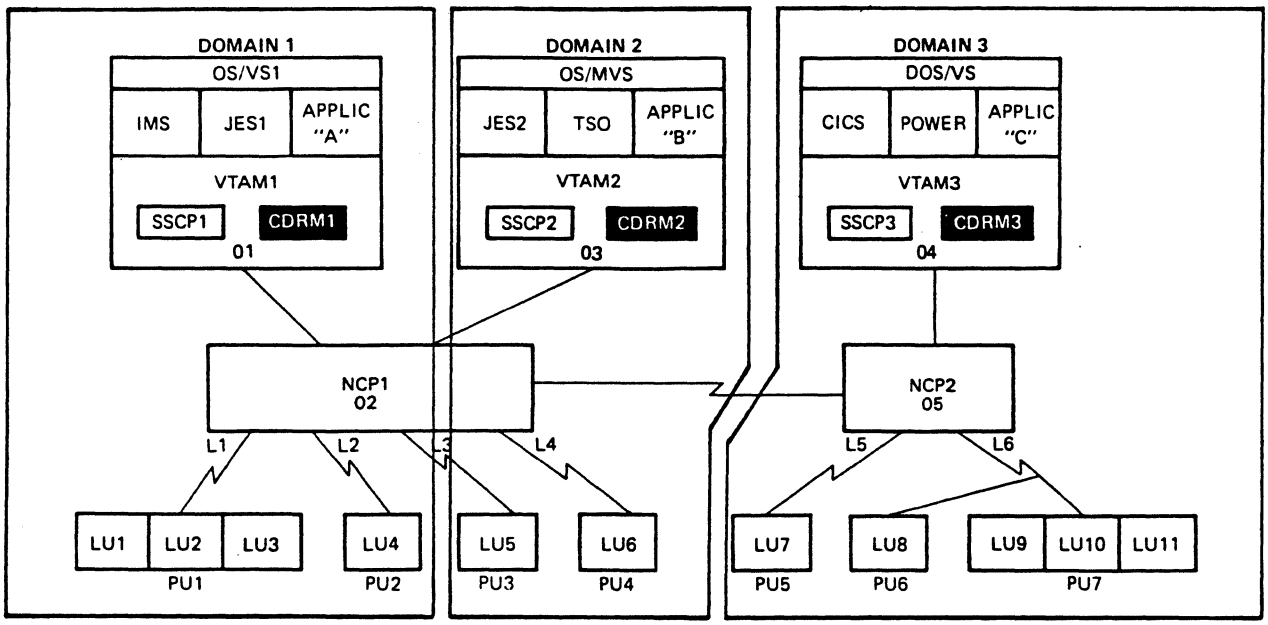

Figure 16-2. Cross-Domain Resource Managers (CDRMs)

Establishing Cross-Domain Sessions

Defining Cross-Domain Resource Managers (CDRMs)

The network definer must define to each domain its own cross domain resource manager (CDRM) as well as CDRMs in other domains that this CDRM will communicate with. CDRMs may be defined in a single definition list or in multiple definition lists. Here is a definition list for domain 1.

```
         VBUILD  TYPE=CDRM
CDRM1    CDRM    SUBAREA=01,ISTATUS=ACTIVE
CDRM2    CDRM    SUBAREA=03,ISTATUS=INACTIVE
CDRM3    CDRM    SUBAREA=04,ISTATUS=INACTIVE
```

This CDRM list shows that CDRM1 is automatically activated (ISTATUS=ACTIVE) when VTAM1 activates this list. This CDRM list also shows that CDRM2 and CDRM3 are initially inactive. They can be activated by network operator commands. CDRM1 is called the host CDRM, because it resides in the domain of these definitions. CDRM2 and CDRM3 are called external CDRMs, because they reside in domains external to this domain.

A domain is willing to participate in cross-domain sessions when its host CDRM is active.

The following defintions are for domain 2.

```
         VBUILD  TYPE=CDRM
CDRM1    CDRM    SUBAREA=01,ISTATUS=INACTIVE
CDRM2    CDRM    SUBAREA=03,ISTATUS=ACTIVE
CDRM3    CDRM    SUBAREA=04,ISTATUS=INACTIVE
```

CDRM2 is the host CDRM while CDRM1 and CDRM3 are external CDRMS. The only difference between these definitions and the ones for domain 1 is the ISTATUS definition. Typically, ISTATUS=ACTIVE is defined for the host CDRM.

The following definitions are for domain 3.

```
         VBUILD  TYPE=CDRM
CDRM1    CDRM    SUBAREA=01,ISTATUS=INACTIVE
CDRM2    CDRM    SUBAREA=03,ISTATUS=INACTIVE
CDRM3    CDRM    SUBAREA=04,ISTATUS=ACTIVE
```

Once the CDRM in each domain is active, CDRM-CDRM sessions can be established. A CDRM-CDRM session must be established before cross-domain LU-LU sessions can be established.

Defining Cross-Domain Resources

Now read about the definition of cross-domain resources (logical units not owned by the domain). The network illustrated in Figure 16-2 is the basis for this discussion.

Looking at the figure, you should understand that all logical units in domains 2 and 3 are cross-domain as far as SSCP1 is concerned. The same reasoning applies to domains 2 and 3.

In order for the resources (LU1, LU2, LU3, LU4, JES1, IMS, and application "A") in domain 1 to initiate a session with the resources in domains 2 and 3, the resources in those domains must be defined to SSCP1 as cross-domain resources. These resources can be grouped together to form one definition set, or there can be several definition sets, each defining one or more resouces. For this discussion, assume that all cross-domain definitions for an SSCP are grouped together, and refer to that group of definitions as a cross-domain resource (CDRSC) set. Now take a look at a partial CDRSC set for SSCP1.

```
         VBUILD TYPE=CDRSC
JES2     CDRSC  CDRM=CDRM2,ISTATUS=ACTIVE
TSO      CDRSC  CDRM=CDRM2,ISTATUS=ACTIVE
B        CDRSC  CDRM=CDRM2,ISTATUS=ACTIVE
LU5      CDRSC  CDRM=CDRM2,ISTATUS=ACTIVE
LU6      CDRSC  CDRM=CDRM2,ISTATUS=ACTIVE
LU7      CDRSC  CDRM=CDRM3,ISTATUS=ACTIVE
CICS     CDRSC  CDRM=CDRM3,ISTATUS=ACTIVE
C        CDRSC  CDRM=CDRM3,ISTATUS=ACTIVE
         o
         o
         o
```

Activating this set of definitions allows logical units in domain 1 to initiate sessions with logical units in the definition list. For example, LU1 in domain 1 can initiate a session with CICS in domain 3 because CICS is defined as a cross-domain resource in domain 1. However, LU1 cannot initiate a session with POWER in domain 3 because POWER is not defined as a cross-domain resource in domain 1.

Cross-domain resources must be defined in the domain that initiates the session. For example, LU1 in domain 1 initiates a session with CICS in domain 3. The network definer must define CICS as a cross-domain resource in domain 1. CDRM1 in domain 1 must know the subarea of the CDRM in CICS's domain to send a session setup request. Optionally, LU1 may be defined as a cross-domain

resource in domain 3. Definitions for the host CDRM in domain 3 (CDRM3) can specify that that domain automatically define LU1 as a cross-domain resource when a session request is received from that LU. If the definitions do **not** specify automatic definition of cross-domain resources, then the network definer must include a cross-domain definition for LU1 in domain 3 and for any other LUs that request sessions with LUs in domain 3.

Now that the CDRMs and CDRSCs are defined, you will see how cross-domain sessions are established. CDRM-CDRM sessions must be established first, then cross-domain LU-LU sessions can be established.

Establishing CDRM-CDRM sessions

Now CDRM1 in domain 1 will establish a session with CDRM3 in domain 3 (see Figure 16-3).

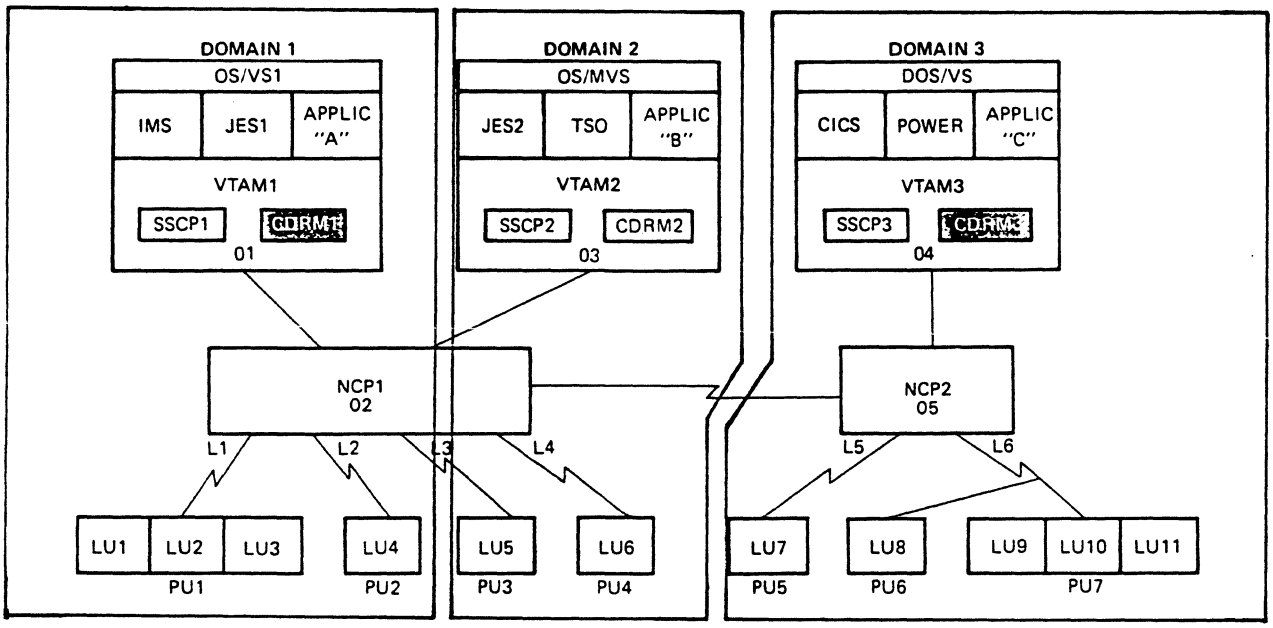

Figure 16-3. CDRM1 and CDRM2

The CDRM definition for domains 1 and 3 are shown below. Study the definitions, then continue.

```
CDRM DEFINITION LIST FOR DOMAIN 1

         VBUILD  TYPE=CDRM
CDRM1    CDRM    SUBAREA=01,ISTATUS=ACTIVE
CDRM2    CDRM    SUBAREA=03,ISTATUS=INACTIVE
CDRM3    CDRM    SUBAREA=04,ISTATUS=INACTIVE

CDRM DEFINITION LIST FOR DOMAIN 3

         VBUILD  TYPE=CDRM
CDRM3    CDRM    SUBAREA=04,ISTATUS=ACTIVE
CDRM1    CDRM    SUBAREA=01,ISTATUS=INACTIVE
CDRM2    CDRM    SUBAREA=03,ISTATUS=INACTIVE
```

When the CDRM definition list for domain 1 is activated, CDRM1 is activated, while CDRM2 and CDRM3 are not activated. When the CDRM definition list for domain 3 is activated, CDRM3 is activated, while CDRM1 and CDRM2 are not activated. Both domain 1 and domain 3 perceive their own CDRM as active, but perceive the other CDRMs as inactive.

To initiate the CDRM1-CDRM3 session, the following network-operator command is issued to SSCP1 in domain 1.

```
VARY NET,ACT,ID=CDRM3
```

SSCP1 receives the VARY command and searches its tables for the name CDRM3. When CDRM3 is found, SSCP1 determines that it is a cross-domain resource manager (CDRM) and gives control to CDRM1 so it can activate CDRM3. The definition for CDRM3 (see above CDRM definition list for domain 1) shows that it is located in subarea 03 and is inactive as far as domain 1 is concerned. Now CDRM1 will issue an activate cross-domain resource manager (ACTCDRM) request. This will establish the CDRM1-CDRM3 session. The request flow is shown in Figure 16-4.

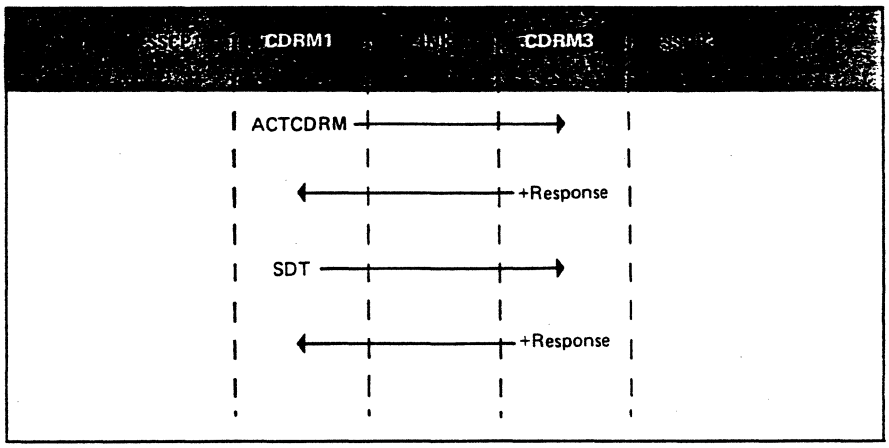

Figure 16-4. Establishing a CDRM-CDRM Session

The figure shows the ACTCDRM request flowing from CDRM1 to CDRM3. Upon receipt of the ACTCDRM request, CDRM3 verifies that the sending CDRM is valid by comparing the sender's subarea number with the list of subarea numbers in its CDRM definition list(s). In this case, the sending CDRM (CDRM1; subarea 01) is valid (see above CDRM definition list for domain 3). CDRM3 returns a positive response to CDRM1, and the CDRM1-CDRM3 session is established. CDRM3 is marked as active in domain 1 and CDRM1 is marked as active in domain 3. CDRM1 sends the SDT request to place the session in the data-traffic-active state.

The CDRM1-CDRM3 session could have been initiated from domain 3 as well as from domain 1.

Now that there is a CDRM1-CDRM3 session, LU-LU sessions can be established between the two domains.

The following describes a session formation between LU1 in domain 1 and CICS in domain 3; assume that LU1 sends a logon (INIT-SELF) to CICS. Figure 16-5 illustrates the logon flow as well as the data flow path between LU1 and CICS after the LU-LU session is established. Study the figure, then read the discussion that follows.

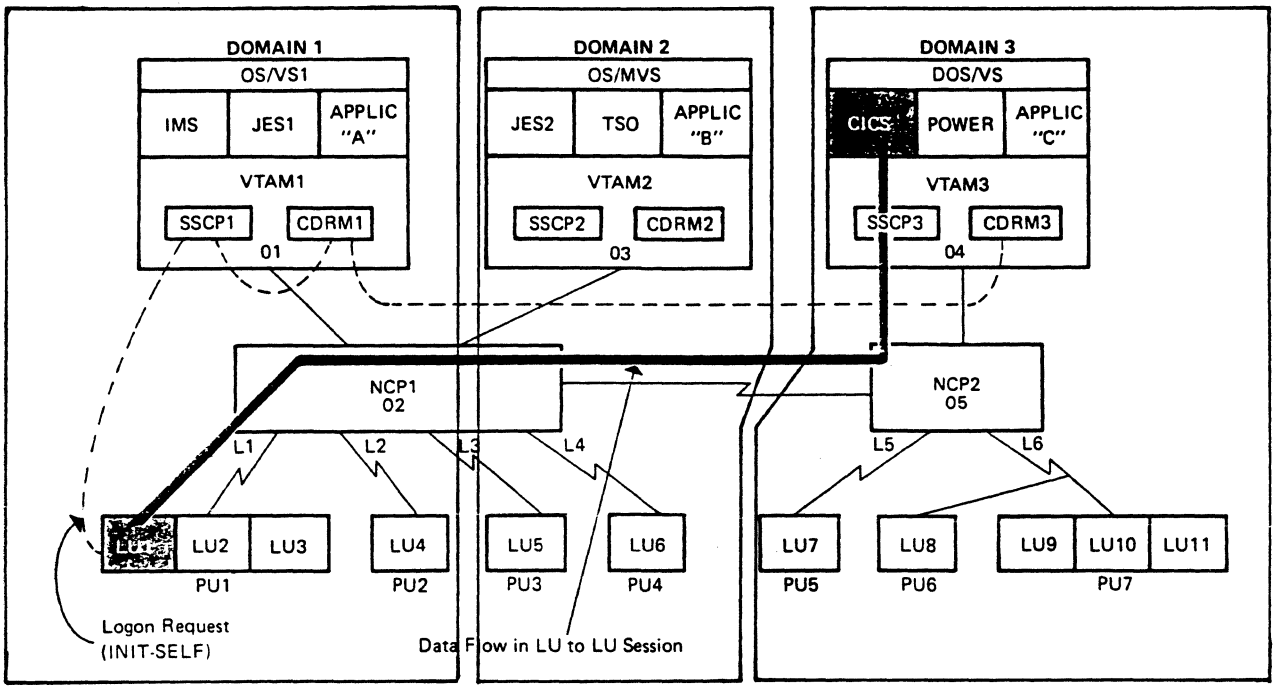

Figure 16-5. Cross-Domain Logon

LU1 is owned by SSCP1, therefore the logon request must flow to SSCP1. LU1 includes the symbolic name CICS in the INIT-SELF request but SSCP1 has no knowledge of CICS since it does not reside in its domain. Therefore SSCP1 gives the logon request to CDRM1 and since CICS is defined as a cross-domain resource, CDRM1 processes the logon.

CDRM1 communicates with CDRM3 to determine if the session can be established. CICS must be active and accepting logons, and the class of service specified in the logon must be available for route selection. Once the logon request has been validated the logon is sent to CICS and CICS issues an OPNDST macro to send the BIND request to LU1 and the CICS-LU1 session is established.

Figure 16-6 illustrates the request flow to establish the CICS-LU1 cross-domain session. Study the figure before you read the discussion that follows.

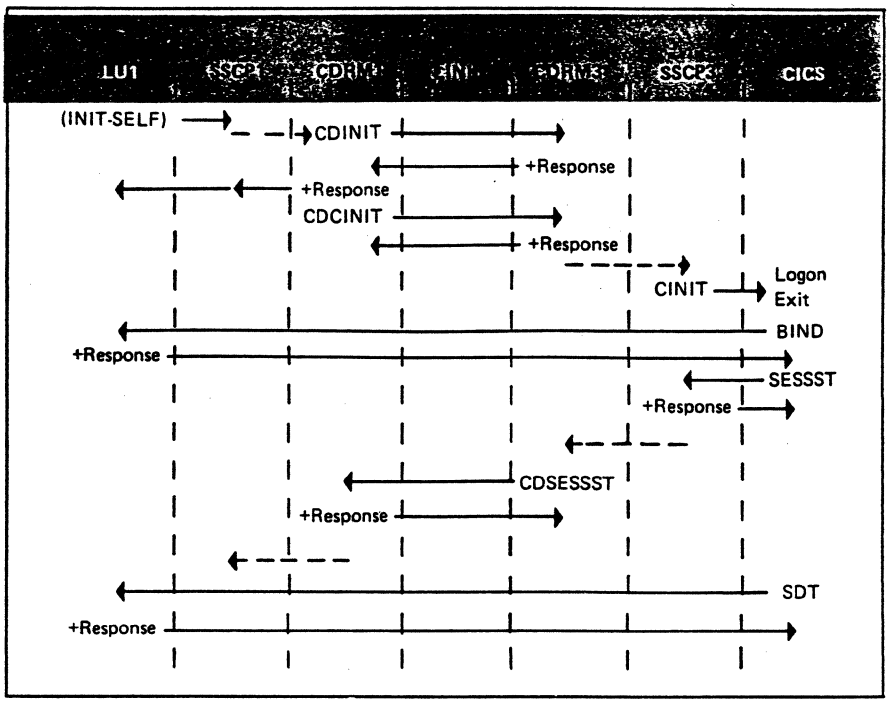

Figure 16-6. Establishing Cross-Domain LU-LU Session

SSCP1 receives the logon from LU1 and searches its local resource definition set(s) for the name CICS. When the name is not found, control is given to CDRM1, which searches its cross-domain resource sets(s) and finds CICS. The search reveals that CICS's cross domain resource manager (CDRM) is CDRM3, located in subarea 04. Now CDRM1 will communicate with CDRM3 to verify the validity of the logon.

CDRM1 sends the SNA request cross-domain initiate (CDINIT) to CDRM3. The CDINIT request contains the secondary LU name (LU1), the primary LU name (CICS), and the class of service name. Upon receipt of the CDINIT request, CDRM3 determines that CICS is active and accepting logons, and determines that the specified class of service for the LU-LU session is available. Therefore, CDRM3 returns a positive response to CDRM1. A positive response is returned to LU1 to complete the INIT-SELF request.

Now CDRM1 sends the cross-domain control initiate (CDCINIT) request which contains the session parameters as specified by LU1 and the logon message, if any. CDRM3 receives the CDCINIT and returns a positive response if CICS is still active and accepting logons.

Next, CDRM3 gives control to SSCP3. SSCP3, in turn, invokes CICS's logon exit, providing the control initiate (CINIT) request unit. The CINIT contains all the logon information sent over from domain 1. The rest of the session initiation is handled by the two LUs, the same as for a single domain network.

CICS issues an OPNDST macro to send the BIND request to LU1, and a positive response from LU1 establishes the session. Both SSCPs (SSCP1 and SSCP3) need notification that the session was established. The primary LU (CICS) notifies

SSCP3 that the session was established by sending the session started (SESSST) request to SSCP3. SSCP3 gives control to CDRM2 to notify domain 1 that the session was established. CDRM3 sends the cross domion session started (CDSESSST) request to CDRM1, which in turn notifies SSCP1. Now both SSCPs and both LUs are aware of the LU-LU session.

The SDT request is sent to place the session in a data traffic active state.

Terminating Cross-Domain Sessions

You have seen how to establish CDRM-CDRM sessions and cross domain LU-LU sessions. Now let's see how the sessions are terminated.

Terminating Cross Domain LU-LU Sessions

Figure 16-7 illustrates two LU-LU sessions: (1) LU1-CICS, and (2) LU2-APPLIC "C" The figure also shows LU2 sending a TERM-SELF request to APPLIC "C"

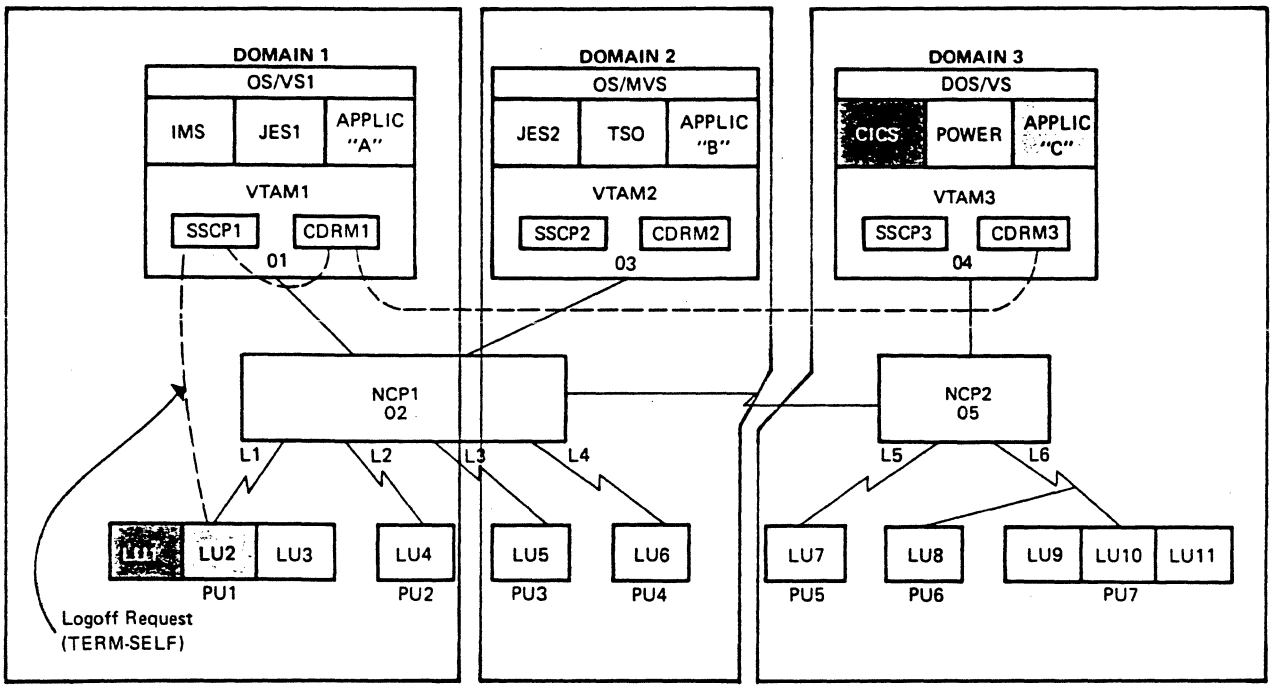

Figure 16-7. Two Cross-Domain LU-LU Sessions

Like the logon (INIT-SELF), the logoff (TERM-SELF) is directed to the SSCP that owns the logical unit, SSCP1 in this example.

Figure 16-8 illustrates the sequence of requests to terminate the session. Study the figure and then continue.

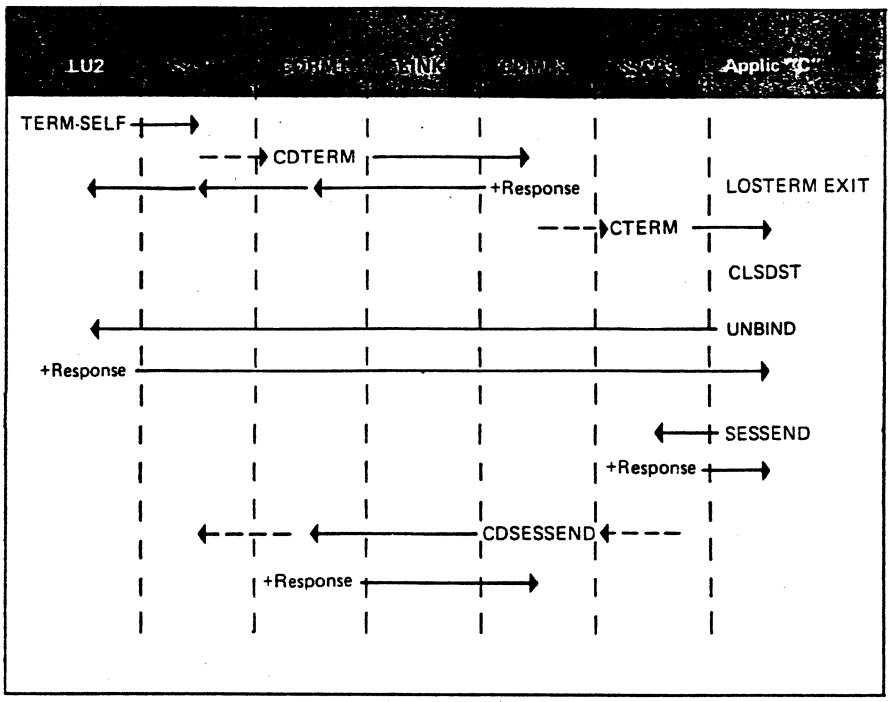

Figure 16-8. Terminating LU-LU Sessions (TERM-SELF)

SSCP1 receives the TERM-SELF request, determines that the request is not directed to a resource in this domain, and gives control to CDRM1. CDRM1 transmits the cross-domain terminate (CDTERM) request to notify CDRM3 of the logoff request. The contents of the CDTERM request identify the primary and secondary LUs as well as the type of logoff. CDRM3, in turn, gives control to SSCP3 to forward the logoff to application "C". SSCP3 drives the LOSTERM exit of application "C", providing a control terminate (CTERM) request. The CTERM request supplies the network address of LU2 as well as the type of terminate request.

APPLIC "C" issues a CLSDST macro instruction when it is ready to terminate the session. The CLSDST causes an UNBIND request to be sent to LU2, terminating the session. Now both SSCPs must be notified that the session has been terminated. APPLIC "C" sends the session ended (SESSEND) request to notify SSCP3 that the session has been terminated. APPLIC "C" sends the SESSEND request after it receives a positive response for the UNBIND request.

Now SSCP1 must be notified. SSCP3 gives control to CDRM3 and it transmits the cross-domain session ended (CDSESSEND) request to CDRM1 notifying that domain of the terminated session. CDRM1, in turn, notifies SSCP1. The boundary function for LU2 in NCP1 also sends a SESSEND request to SSCP1. This is done in case the CDRM1-CDRM3 session has been terminated and the CDSESSEND request cannot be sent to SSCP1. If SSCP1 is not notified that the LU2-CICS session has been terminated, it will not allow another session with LU2.

Terminating CDRM-CDRM Sessions

This is the sequence of commands to terminate the CDRM- CDRM session.

Previously, a CDRM1-CDRM3 session was established (see Figure 16-7). Now the session will be terminated. Assume that there are no LU-LU sessions between domains 1 and 3 at this time.

The following network operator command is issued to SSCP1 in domain 1 to cause the CDRM1-CDRM3 session to be terminated:

```
VARY NET,INACT,ID=CDRM3
```

SSCP1 receives the VARY command and gives control to CDRM1 since this is a cross-domain session. Figure 16-9 shows the request sequence to terminate the session.

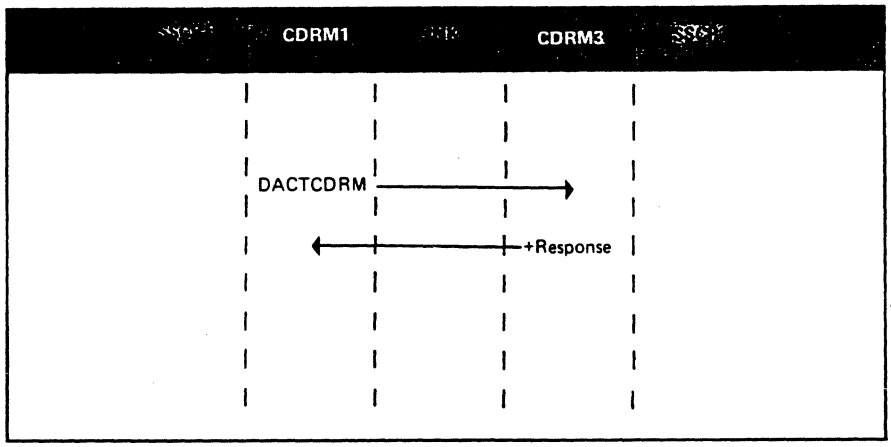

Figure 16-9. Terminating a CDRM-CDRM Session

CDRM1 transmits the deactivate cross domain resource manager (DACTCDRM) request to CDRM3, terminating the session.

If LU-LU sessions exist between domains 1 and 3 when the network operator command (VARY NET,INACT,ID=CDRM3) is submitted, the CDRM1-CDRM3 session will not terminate immediately. However, the LOSTERM exit of each LU that has a cross-domain session is scheduled, notifying the LU that the CDRM-CDRM session is to be terminated. As soon as all the LU-LU sessions are terminated, the CDRM1-CDRM3 session terminates.

Please turn to Mini-Course 16 in your Personal Reference Guide and do Exercise 16.1.