SRA Computer Training Library

SNA Fundamentals

TEXT Vol. 2

Luther Bonnett

Technical Education
IBM National Accounts Division

SRA

SCIENCE RESEARCH ASSOCIATES, INC.
Chicago, Toronto, Henley-on-Thames, Sydney

A Subsidiary of IBM

SR20-8491

PREFACE

This course is designed to teach concepts of Systems Network Architecture (functions of SNA components, and usage of SNA commands. This Text and Personal Reference Guide (SR20-8492) form the nucleus of the course and a supported with a videotape.

Since this is an independent study course, the completion time may vary depe on the ability and experience of the individual student. However, most stude study time should fall in the range of 12 to 16 hours.

● ***Before continuing in this Text, turn to your Personal Reference Guide (PRG) and s with the Course Introduction section. The PRG will guide you through the course.***

Reprinted, April 1984

CONTENTS

SNA Fundamentals

Mini-Course 8

Data Flow Control on LU-LU Sessions

Printed in the United States of America.

Mini-Course 8. Data Flow Control on LU-LU Sessions

Introduction

SNA defines the end user function and the logical unit function as two separate components:

1. End user function. End users process user application data. The end user does **not** get involved in controlling or managing sessions. It gives its data to a logical unit for transmission to another end user. It also accepts data from its associated logical unit that was sent by another end user.

2. Logical unit function. The logical unit manages the exchange of data between end users. Two logical units (LUs) must be designed so that they can communicate intelligently. The LUs must be able to send and receive at the appropriate times, recognize an entity of data, and process errors. They also must be able to resynchronize communication after an error.

SNA formally identifies the required functions for two end users to exchange information and has defined indicators and commands to be used by an LU-LU session to accomplish the required functions. We refer to these communication requirements between LUs (LU-LU sessions) as protocols or the rules that each session partner must abide by to carry out the functions of an LU-LU session. The BIND request contains session parameters that identify the protocols (rules) for a session. The BIND request is sent from the primary logical unit to the secondary logical unit and if the secondary logical unit can support the specified protocols, it returns a positive response to establish the LU-LU session.

LUs transmit commands and indicators between each other to implement the protocols agreed to in the BIND request. All commands are transmitted in a request unit (RU). Indicators are transmitted in the request/response header (RH).

BIND Request

We will now take a look at the structure and content of the BIND request to gain a better understanding of its purpose and use. As we just said, the BIND request is sent by the primary logical unit to the secondary logical unit to establish an LU-LU session. The BIND request contains session parameters which describe how the session will behave.

Session parameters can be supplied by a user table called a mode table, or the primary logical unit can supply some or all session parameters. Most Logical units are designed to support a subset of LU-LU protocols. Therefore, the appropriate session parameters must be selected for two logical units to be able to go into session and communicate with each other.

The BIND request is divided into fields and these fields must be set to appropriate values before the request is sent to the secondary logical unit.

Negotiable BIND Request: Some logical units have the capability to negotiate session parameters. For example, the primary logical unit sends the BIND request to the secondary logical unit and the request contains the primary logical unit's choice for session parameters. The secondary logical unit can examine the parameters and if they are unacceptable, it can send its choice of session parameters back to the primary logical unit in a positive response. If these parameters are acceptable to the primary logical unit, it sends the start data traffic (SDT) request to allow data requests to flow on the session, otherwise the primary logical unit terminates the session. This is called a **negotiable BIND request**.

Non-negotiable BIND Request: Typically, a **non-negotiable BIND request** is used to BIND a session. A primary logical unit sends a non-negotiable BIND request to a secondary logical unit and its options are to either accept with a positive response or reject with a negative response.

Now we will discuss the structure and content of a BIND request. Refer to Figure 8-1.

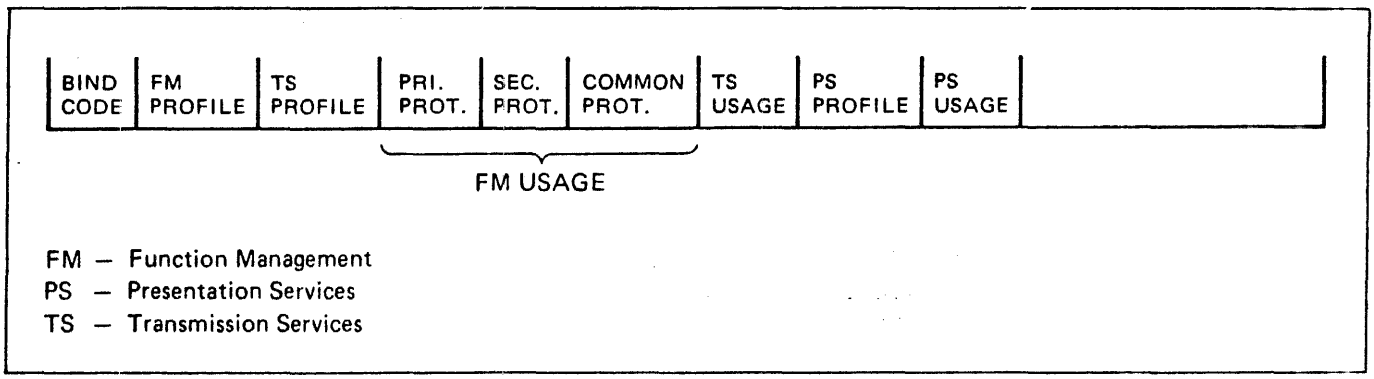

Figure 8-1. BIND Request

We said earlier that logical units usually implement a subset of protocols. There are three broad categories of protocols for LU-LU sessions:

1. Function Management protocols.

2. Transmission Services protocols.

3. Presentation Services (PS) protocols.

We will now look at the structure of a logical unit and the services that it provides. One layer, the (Transmission Control) provides services for transmitting requests and responses into the path control network. It also provides for receiving requests and responses from the path control network. The protocols that are associated with these functions are called transmission services (TS) protocols. For each LU-LU session we select a subset of these protocols and this subset is called a transmission services (TS) profile. Look at Figure 8-1 and you will see a TS profile field in the BIND request.

Other layers in the logical unit provide services which we call function management (FM) services. The protocols that are associated with these services handle communication between two logical units. We can select a subset of these protocols for a session and we call this subset the function management (FM) profile. Look at Figure 8-1 to see an FM profile field.

Another logical layer of the logical unit is called presentation services (PS). Presentation services handles communication with the end user, e.g., formatting data to be sent to the end user. Protocols associated with these services are called presentation services protocols and we can select a subset of these protocols for a session (called PS profile).

We specify an FM profile, a TS profile, and a PS profile for each session which specifies three subsets of protocols that the two logical units are capable of abiding by. However, all the protocols do not have to be used for this particular session. Other fields in the BIND request (FM usage, TS usage, and PS usage) specify exactly what protocols the primary logical unit will abide by and what protocols the secondary logical unit will abide by. Some protocols apply to the whole session while other protocols apply to an individual logical unit. For example, the primary logical unit may be specified to support definite-response protocol and the secondary logical unit may be specified to support exception-response protocol. A send/receive protocol applies to both logical units. Thus we could not have one logical unit performing in full-duplex mode and the other logical unit performing in half-duplex mode.

There are three FM usage fields: (1) primary protocol, (2) secondary protocol, and (3) common protocol. They supplement the FM profile specifications. The primary protocol field specifies protocols for the primary logical unit to abide by. The secondary protocol field specifies protocols for the secondary logical unit to abide by. Finally, the common protocol field specifies protocols that both logical units are to abide by.

The TS usage specifications supplement the TS profile specification. TS usage specifies pacing values and maximum request unit sizes.

The PS usage field supplements the PS profile specification. PS usage specifies function management header usage. It also specifies types of character strings used, and the type of code, EBCDIC or ASCII.

Now we will discuss some of the protocols that are implemented by LU-LU sessions, starting with a discussion on send/receive modes.

Normal-Flow Send/Receive Modes

In an LU-LU session both logical units are responsible for maintaining the integrity of data flow within the session. SNA defines three send/receive modes for data flow control:

1. Full-duplex.

2. Half-duplex flip-flop.

3. Half-duplex contention.

Regardless of which mode is used for a session, the two session partners (logical units) enforce the protocol. SNA describes how each session partner is to act for each send/receive mode. SNA also defines an indicator (change direction indicator) that is used by session partners to enforce the half-duplex send/receive mode.

You should keep in mind that full-duplex and half-duplex, as they pertain to an LU-LU session, are independent of the physical properties of the network. For

example, the requests that flow on this session may travel over several data links, some of which may be half-duplex. The fact that requests can flow in only one direction at a time over the half-duplex data links does not mean that the two session partners cannot send to each other at the same time. The requests flowing in opposite directions can pass each other at buffered points in the network.

Full-Duplex: First we will discuss the full-duplex send/receive protocol. In full-duplex mode, both session partners can send requests at the same time. The requests flowing in both directions are independent of each other. We will now consider the application example where inquiries are being made against an inventory data base. Figure 8-2 illustrates a configuration with a peripheral logical unit that is connected to a display terminal and a printer. LU21 is in session with LU1.

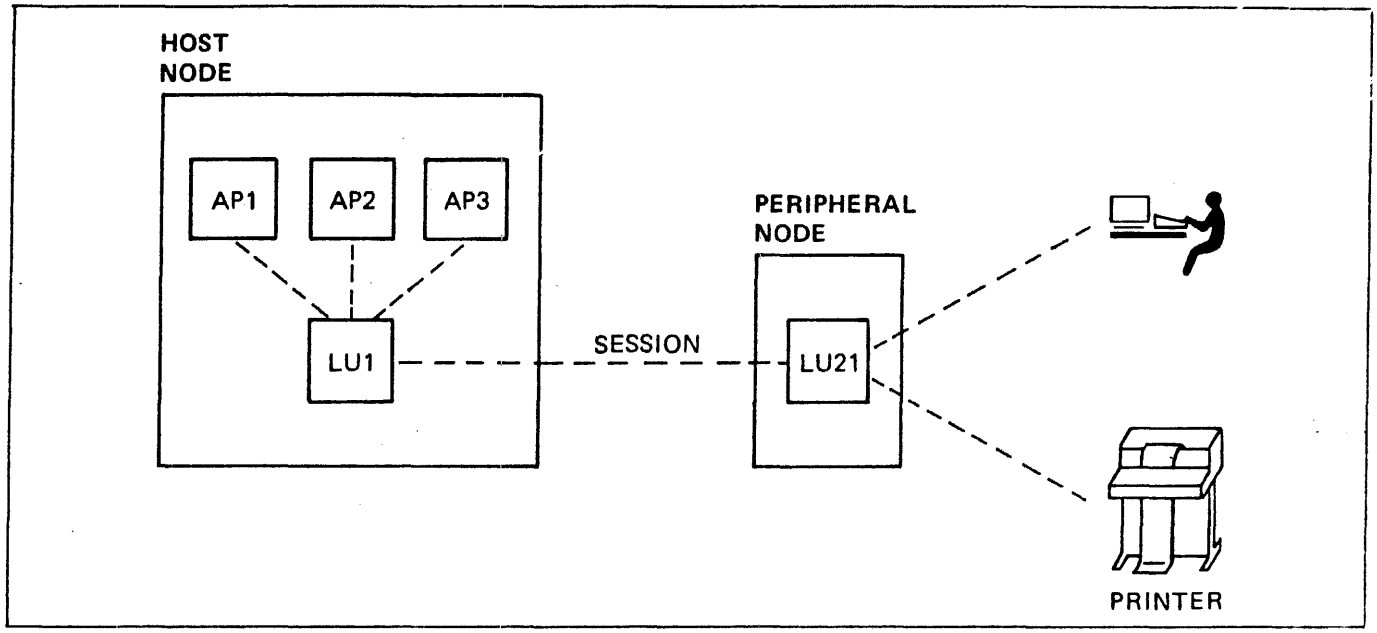

Figure 8-2. Full-Duplex Application

The host logical unit (LU1) is associated with several application programs that process inquiries against the inventory data base.

The operator at the display terminal submits inquiries one after another to the peripheral logical unit (LU) which sends them to the host logical unit. Each inquiry is given to the appropriate application program which generates a reply to be sent to the peripheral logical unit.

In full-duplex communication, the peripheral logical unit is sending requests to the host logical unit at the same time that the host logical unit is sending requests to the peripheral logical unit.

The nature of many applications, and probably most, prohibit such unrestricted exchange of data. SNA defines half-duplex protocols that allow each session partner to exert varying degrees of control over each other's ability to send data.

Half-Duplex Flip-Flop: In half-duplex flip-flop mode, only one session partner is allowed to send at a time. At any given time, one session partner is a sender and the other session partner is a receiver. The sender sends normal-flow requests and

the receiver replies with responses if the sender asks for responses. The receiver is not allowed to send requests on the normal-flow. When the sender needs to permit its session partner to start sending, it sets a change-direction indicator (CDI) in the request header (RH) of the last request that it sends. The logical unit that sends the "CDI" switches from sender status to receiver status. The logical unit that receives the request that contains a "CDI," switches from receiver status to sender status.

The two logical units continue to switch between send and receive status as required.

The session parameters, in addition to specifying that half-duplex flip-flop mode is to be used, also specify which logical unit is to be the first sender and the first receiver. Figure 8-3illustrates half-duplex flip-flop communication between a peripheral logical unit and a host logical unit. The peripheral logical unit is designated first sender. The BIND also specifies that multiple element request chains can be used.

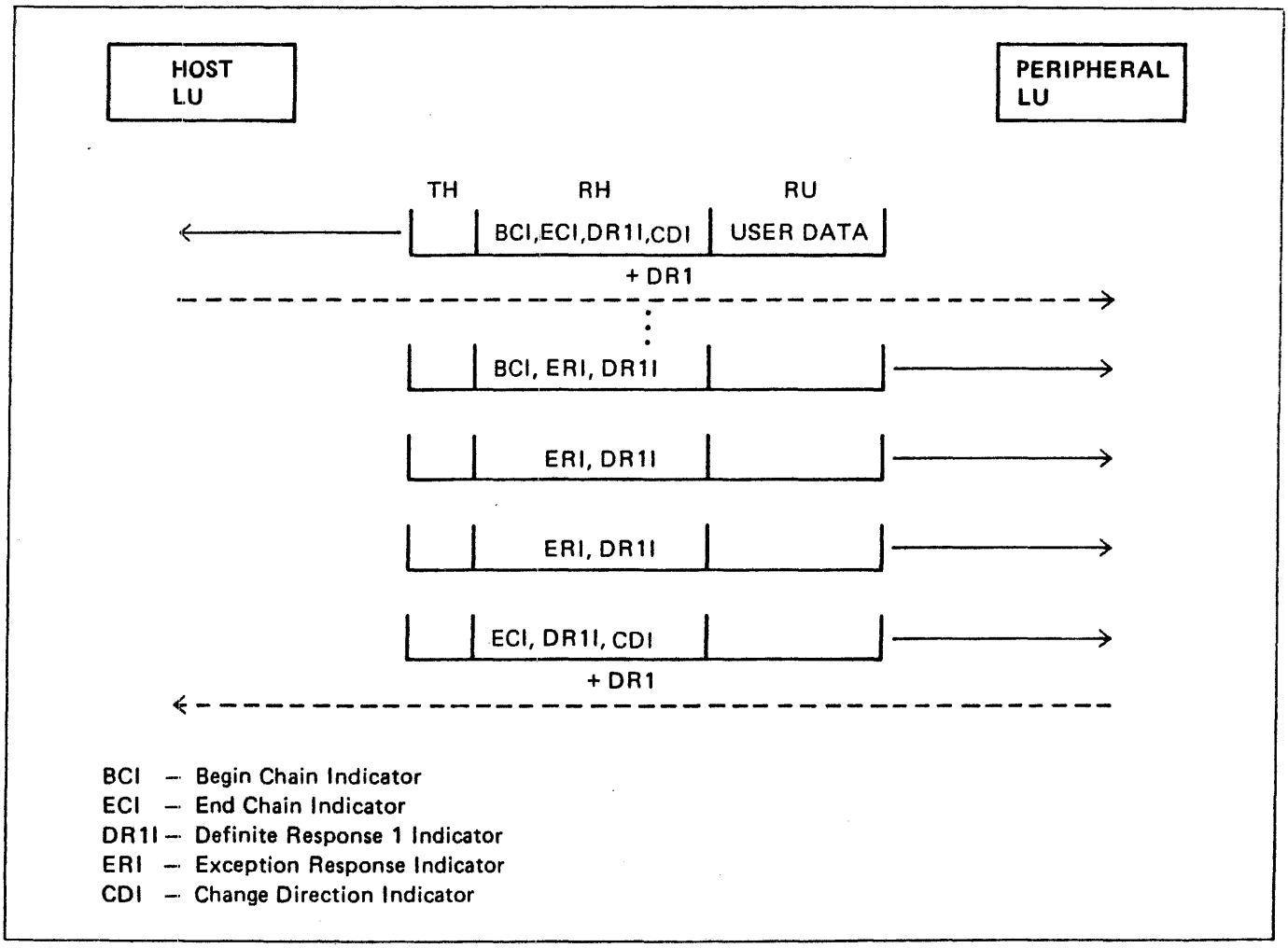

Figure 8-3. Half-Duplex Flip-Flop Mode

The peripheral logical unit (first sender) sends a request to the host logical unit. The request header (RH) contains a begin chain indicator (BCI) and an end chain indicator (ECI) which designates this as a single element request chain. The response indicator "DR1I" specifies that the receiver is to return either a positive

or a negative DR1 response, depending on whether the request is acceptable. The change direction indicator (CDI) specifies that the host logical unit is to become a sender and the peripheral logical unit is to become a receiver. Upon sending this request, the peripheral logical unit switches from sender to receiver. The host logical unit sends a positive DR1 to the peripheral logical unit indicating that the received request was acceptable and then switches from receiver to sender.

The host logical unit now sends requests until it needs to receive from the peripheral logical unit. Our figure shows the last of several multiple element request chains sent in reply to the request received from the peripheral logical unit. The chaining indicators, BCI and ECI, identify the first and last chain elements. The middle elements are identified by neither chaining indicator being set. The response indicators specify an exception DR1 for all requests except the last element of the chain. A definite DR1 is specified for the last chain element. The change direction indicator (CDI) is included in the last request element of the last chain causing the host logical unit to become the receiver and the peripheral logical unit to become the sender.

The role of each session partner continues to alternate between sender and receiver until the session is terminated.

Half-Duplex Contention: In half-duplex contention mode both session partners are in contention mode when neither partner is sending. That is, either session partner is allowed to begin to send a request chain. While one session partner is sending a request chain, the other session partner is not allowed to send a request chain on the normal-flow.

Session parameters, in addition to specifying half-duplex contention, also specify one session partner to win contentions. Contentions occur when one session partner begins to send while its partner is already sending. The contention is won by the logical unit that was designated contention winner. Figure 8-4 illustrates send/receive sequences when half-duplex contention protocol is used. The BIND request designates the secondary logical unit as the contention winner.

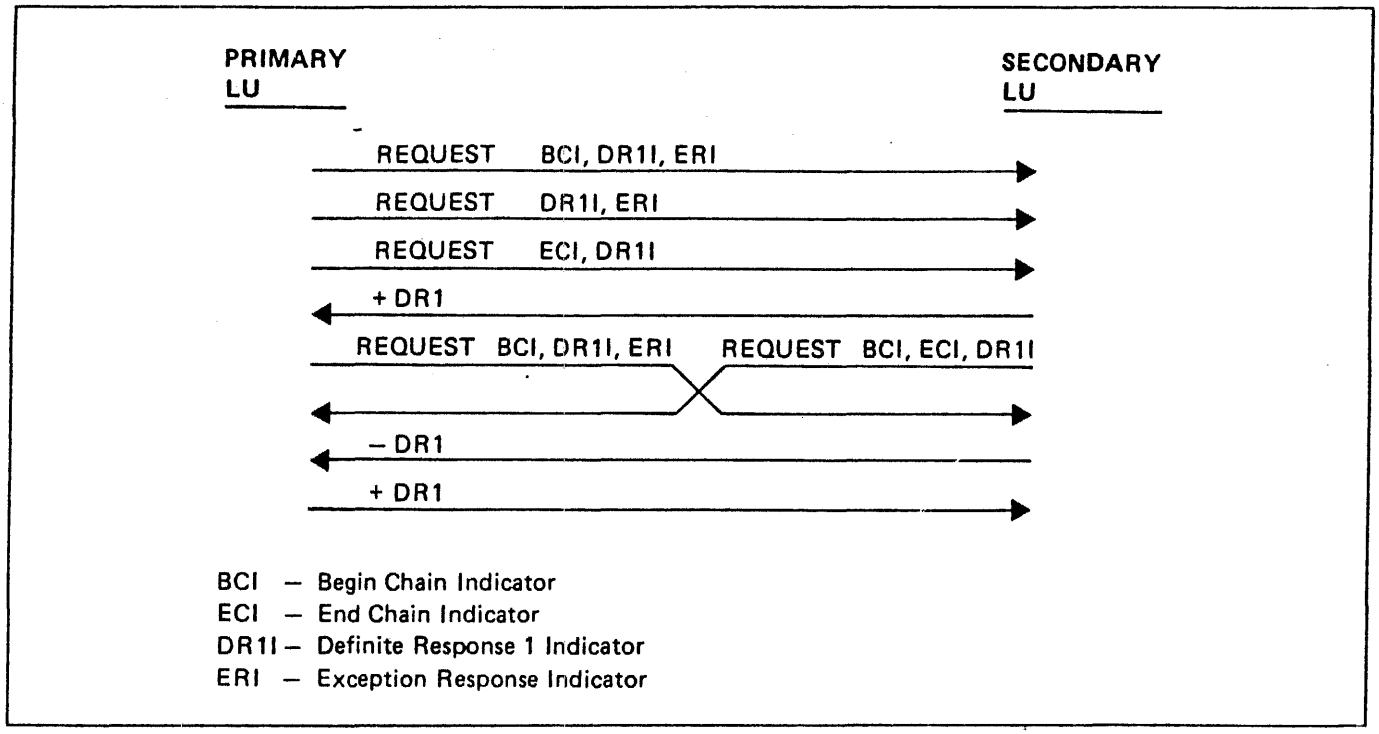

Figure 8-4. Half-Duplex Contention Protocol

In this illustration, the primary logical unit is the first to send a request chain. The secondary logical unit is in contention state (It is not sending or beginning to send.), therefore it accepts the request from the primary logical unit. Now neither logical unit is in contention state and the secondary logical unit is not allowed to send a request chain until the primary logical unit completes sending its chain.

Once the secondary logical unit receives the last request chain element and returns a response to the primary logical unit, it goes into contention mode along with the primary logical unit. Both logical units are now allowed to begin to send a request chain as illustrated in Figure 8-4. Since the secondary logical unit is the designated contention winner, it rejects the primary logical unit's request chain by returning a negative response.

The primary logical unit receives the request from the secondary logical unit after having sent a request. The primary logical unit queues the request for processing. When the negative response is received, the primary logical unit goes into receiver state and dequeues the request and processes it.

The contention winner reverts to contention state when it sends the last element of the request chain. The contention loser reverts to contention state when it receives the last element of the request chain.

Bracket Protocol

The bracket protocol helps the user to separate one unit of work from another unit of work. A unit of work consists of related sequences of requests that flow between pairs of logical units. For example, a terminal operator submits an inquiry to a data base program and the reply is displayed on the terminal. The inquiry and the sequence of requests that make up the reply are a unit of work.

It would be unacceptable for unrelated messages to be displayed on the terminal at the same time the reply to the inquiry is being displayed. The brackets protocol facility can prevent this from happening. Brackets identify the beginning and the end of a unit of work. This is called a bracket. A bracket includes the first request through the last request of related request sequences or units of work.

SNA defines indicators and commands to implement brackets protocol. Two indicators that reside in the request header (RH) are used to identify the beginning and the end of a bracket:

1. Begin bracket indicator (BBI). The begin bracket indicator is set in the request header of the first request of a bracket.

2. End bracket indicator (EBI). The end bracket indicator is set in the request header of the last request of a bracket.

Command requests provided to manage each bracket include:

1. BID. In an LU-LU session that supports brackets, one of the logical units is defined as a **first speaker** and the other logical unit is defined as the **bidder**. The first speaker is allowed to initiate a bracket by sending a request that contains a begin bracket indicator. The bidder is required to get permission from the first speaker to initiate a bracket. The bidder can use the BID request to get permission to initiate a bracket. Later we will see a second way for a bid to initiate a bracket.

2. Ready to receive (RTR). The first speaker sends the RTR request to the bidder. This lets the bidder know that it is allowed to initiate a bracket. We will discuss when this request is used, shortly.

Figure 8-5 illustrates the use of bracket indicators to identify the beginning and the end of a bracket. It is assumed that the secondary logical unit is the first speaker. That is, it is allowed to initiate a bracket without getting permission from the primary logical unit.

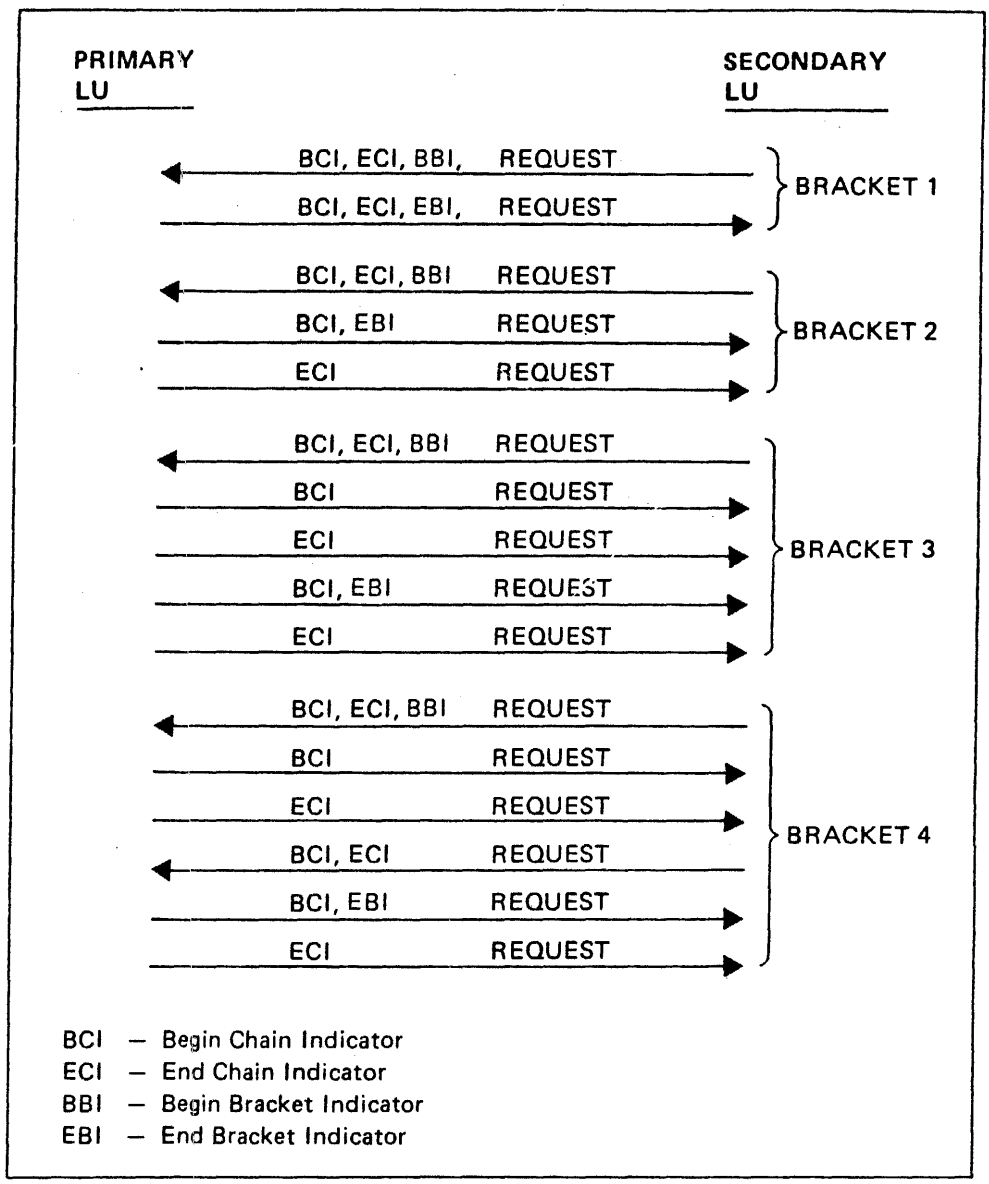

Figure 8-5. Identifying a Bracket

Response protocols and send/receive protocols are not illustrated here. However, keep in mind that both protocols are defined for all sessions.

Four sets of related sequences of message units are identified as brackets (referred to as brackets 1, 2, 3, and 4) in this figure. Bracket 1 consists of two requests, one sent by the secondary logical unit and one sent by the primary logical unit. We can assume that the request sent by the secondary logical unit is an inquiry and the request sent by the primary logical unit is the reply to the inquiry. Each request is a single element chain indicated by the chaining indicators, BCI and ECI. The secondary logical unit initiates the bracket by including the begin bracket indicator (BBI) in the request header of its request. The primary logical unit terminates the bracket by including the end bracket indicator (EBI) in the request header of its request. The two requests, identified as a bracket, are a unit of work. The inquiry request and the reply request complete the transaction.

Bracket 2 includes three requests, an inquiry sent by the secondary logical unit and two requests from the primary logical unit that make up the reply. The reply is a two element request chain. The end bracket indicator (EBI) is set in the first chain element even though the bracket is not terminated until the last element of the chain is transmitted. The end bracket indicator as well as the begin bracket indicator is always included on a first chain element.

Bracket 3 shows a reply that consists of two request chains and each chain contains two elements. The end bracket indicator is included in the first element of the last chain of the bracket.

Bracket 4 shows that a bracket can consist of multiple request exchanges between the primary logical unit and the secondary logical unit. The secondary logical unit submits the inquiry which initiates the bracket and receives a partial reply by a two element request chain. The secondary logical unit submits another request asking for additional data. The primary logical unit sends a reply which completes the inquiry, therefore it ends the bracket.

These bracket examples should make the point that a bracket consists of related sequences of requests. Two logical units participate in a dialogue until the inquiry, transaction, or unit of work is completed. This dialogue can consist of one or more request chains sent by each logical unit and the request chains can be single and/or multiple element chains.

Now that we know how the bracket indicators, BBI and EBI, are used to initiate and terminate brackets, we will define two bracket terms in preparation for further discussion on implementing brackets:

1. In-bracket (INB). Once a bracket has been initiated by the begin bracket indicator and before the bracket is terminated by the end bracket indicator, the LU-LU session is considered to be in a state called in-bracket (INB). A bracket cannot be initiated by a logical unit when the session is already in the in-bracket state.

2. Between-bracket (BETB). An LU-LU session is in a between-bracket (BETB) state if a bracket has not been initiated or if the last bracket has been terminated. When a session is in the between-bracket state, the first speaker can initiate a bracket and the bidder can bid to initiate a bracket.

Both logical units of an LU-LU session must provide logic that manages bracket protocol (called bracket state manager). Each bracket state manager must keep track of whether the session is in the in-bracket state or in the between-bracket state. The two bracket state managers participate in the implementation of bracket protocol.

Bidding: Figure 8-6 shows an LU-LU session that is to use brackets.

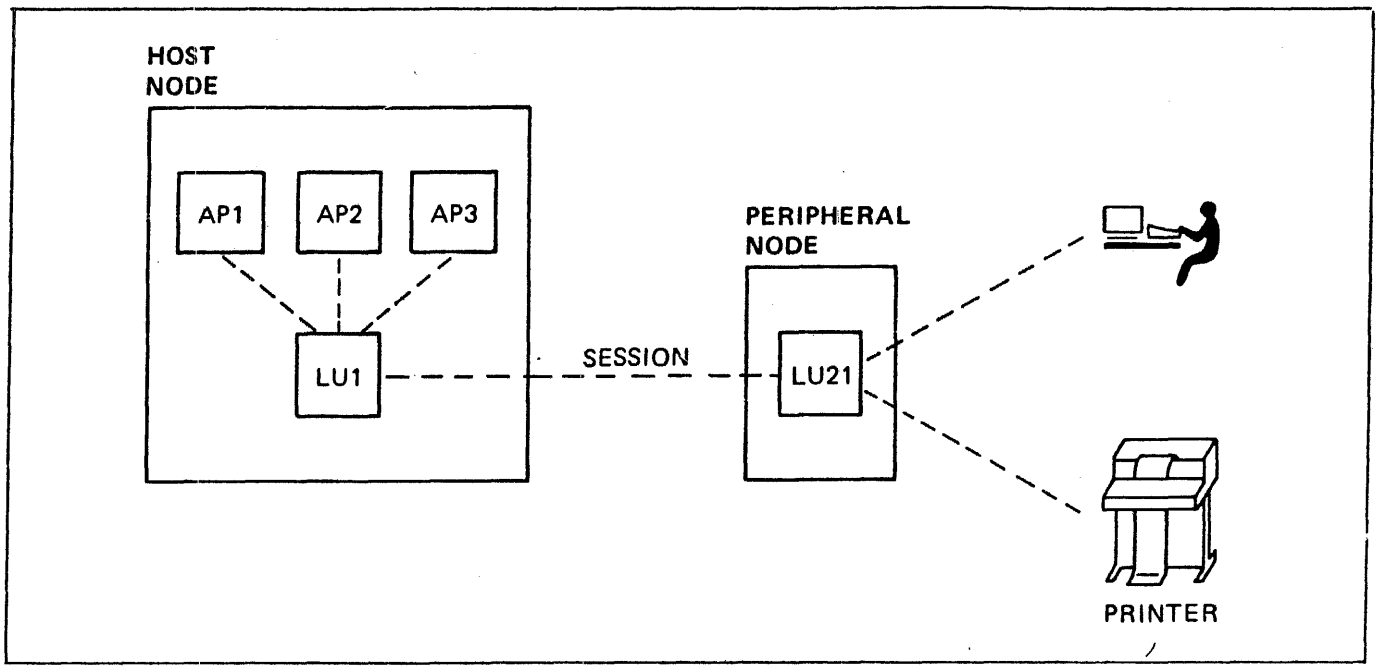

Figure 8-6. Bidding

The BIND request that established the session specified bracket protocol. It specified LU21 (secondary logical unit) as the first speaker, and specified that the primary logical unit (LU1) be allowed to end the bracket. Session parameters can specify that either or both logical units can end the bracket. Typically, the host logical unit is specified to end the bracket.

LU21 is allowed to send a request any time the session is not in the in-bracket state. This includes a begin bracket indicator to initiate a bracket. LU1 must ask the first speaker (LU21) for permission to initiate a bracket, called bidding. LU1 can bid two different ways.

1. Send a BID request to the first speaker. Normally the first speaker replies with a positive response indicating that LU1 can initiate a bracket. A positive response to the BID request indicates that LU21 will not begin a bracket, but will wait on LU1 to begin a bracket. LU1 begins a bracket by sending a data request chain containing a begin bracket indicator. The bracket must be terminated by LU1, because LU1 is defined as the one to send the end bracket indicator.

   LU21 can reject the BID request from LU1 with a negative response. The negative response will contain one of two sense codes to indicate why the request was rejected. One sense code specifies that the BID request is rejected and that LU1 can BID again latter. The other sense code specifies that the BID request is rejected and also specifies that LU21 will notify LU1 when it is allowed to begin a bracket. LU21 sends the ready to receive (RTR) request to LU1 notifying LU1 that it is now allowed to begin a bracket. All logical units are not implemented to send an RTR request, therefore, they would send the negative response that informs the bidder to bid later.

2. LU1 can also bid by sending a request that contains a begin bracket indicator. The request must specify definite response because LU1 is bidding with the request and needs the response which specifies whether the bid was successful.

One possible drawback to this method of bidding is that a data request is usually much larger than a BID request. If the chances of being rejected are high, then the longer request chain will require more overhead.

Typically, the first speaker only rejects a bid when it is in the in-bracket state. For example, assume that LU1 has just sent a request to LU21 that contains an end bracket indicator (EBI). The EBI terminates the bracket and both bracket state managers are in a between-bracket state. LU1 (bidder) can send a BID request to LU21 with some assurance that it will be accepted. However, LU21 may send a request at the same time that LU1 sends the BID request. LU21's bracket state manager stores a status of in-bracket at the time it sends the request, therefore, it rejects the BID request when it is received.

Figure 8-7 illustrates the use of the BID request and the RTR request. LU21 is defined as the first speaker and LU1 is allowed to end brackets. Study the figure, then read the discussion that follows.

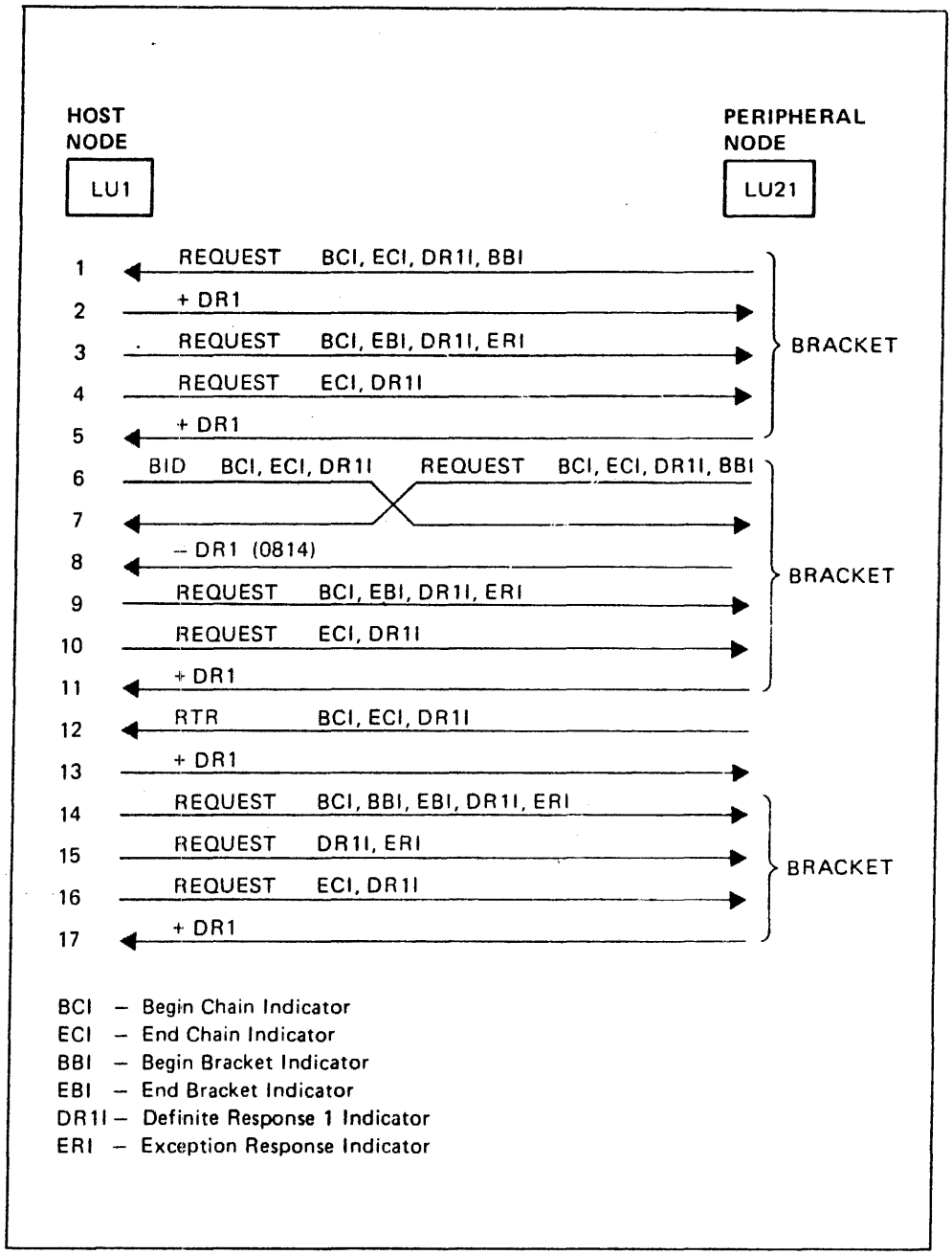

Figure 8-7. Use Of The BID and RTR Request

Items 1 through 5 make up one bracket that was initiated by LU21. After the first bracket is terminated, LU1 sends a BID request to LU21, but LU21 is sending a request to LU1 at the same time (items 6 and 7). The moment that LU21 sends its request it goes into in-bracket state because it is the first speaker. Therefore, LU21 rejects the BID request and returns a negative response with a sense code of "0814." This sense code indicates that the BID was rejected and it also indicates that LU21 will send an RTR request to notify LU1 that it can initiate a bracket.

LU1 receives the data request before it receives the negative response to its BID request. The received request causes LU1 to go into in-bracket state because the first speaker always wins any contention. Now the session is in the in-bracket state.

LU1 generates a reply to the request and sends a two element request chain to LU21 and terminates the bracket (items 6 through 11).

Next, LU21 sends the RTR request to LU1 indicating that it can initiate a bracket (item 12). LU21 will not initiate another bracket until LU1 initiates a bracket or declines the opportunity to initiate a bracket. When LU1 receives the RTR request, it can either initiate a bracket or it can decline by sending a negative response to LU21. If LU1 sends a negative response to LU21 then LU21 can initiate a bracket and LU1 must bid to initiate a bracket.

Our example has LU1 sending a data request to LU21 and initiating a bracket. The bracket consists of a three element request chain.

Bracket Termination Rules: A BIND request that specifies bracket protocol also specifies one of two bracket termination rules.

1. Bracket termination rule 1 (conditional termination). If BBI and EBI appear on the same chain, the bracket is unconditionally terminated when the last request is processed. For example, the primary logical unit sends a definite response chain containing BBI and EBI to the secondary logical unit. The bracket ends for the primary logical unit when it sends the last element of the chain. The bracket ends for the secondary logical unit when it receives and processes the request. The secondary logical unit returns a positive response but the response has no effect on the termination of the bracket.

   For the chain that contains only EBI (no BBI), bracket termination is controlled by the form of response requested (definite, exception, or no response). If a definite response is requested, the bracket is not terminated until the positive response is processed. However, if a negative response is returned for one of the requests, termination of the bracket varies depending on whether the negative response occurred for the last request or other than the last request. A negative response for the last request (marked definite response) causes the bracket to be continued. There is an option to either continue or terminate the bracket if a negative response occurs for a request other than the last request (marked exception response).

   If the chain requests exception response or no-response, the bracket is terminated unconditionally when the last request of the chain that has EBI in its first element is processed.

2. Bracket termination rule 2 (unconditional termination). The bracket is terminated unconditionally when the last request of the chain whose first chain element contains EBI is processed. This happens regardless of the form of response requested.

● ***Please turn to Mini-Course 8 in your Personal Reference Guide and read the summary.***

SNA Fundamentals

Mini-Course 9

Suspending Data Flow on LU-LU Sessions

Printed in the United States of America.

Mini-Course 9. Suspending Data Flow on LU-LU Sessions

Introduction

A logical unit can request its session partner to suspend transmission by sending a data flow control (DFC) request to the partner. A logical unit sends one of these requests for one of two reasons:

1. The logical unit is ready to terminate the session.

2. The logical unit needs to temporarily suspend normal-flow traffic from its partner.

The quiesce and shutdown protocols can be used to suspend session traffic temporarily or permanently:

1. The quiesce protocol can be used to temporarily suspend traffic on the normal-flow either in preparation to terminate the session or because a session partner is running low on some resource,such as a buffer pool.

2. The primary logical unit can use the shutdown protocol to suspend its partner's normal-flow traffic in preparation for an orderly shutdown of the session.

Quiesce Protocol

The quiesce protocol provides a means for a logical unit to stop its partner from sending normal-flow requests. Requests on the expedited-flow are not affected. This protocol may be used for several reasons, e.g., a session partner wants to terminate the session, or a session partner is unable to handle the rate at which requests are being sent by the other logical unit.

Typically, the quiesce protocol is used to **temporarily** stop the flow of normal-requests in one direction. Either logical unit can quiesce its session partner. SNA defines three command requests to implement the quiesce protocol:

1. Quiesce at end of chain (QEC).

2. Quiesce complete (QC).

3. Release quiesce (RELQ).

Figure 9-1 illustrates the use of these requests by a session partner to quiesce the other partner.

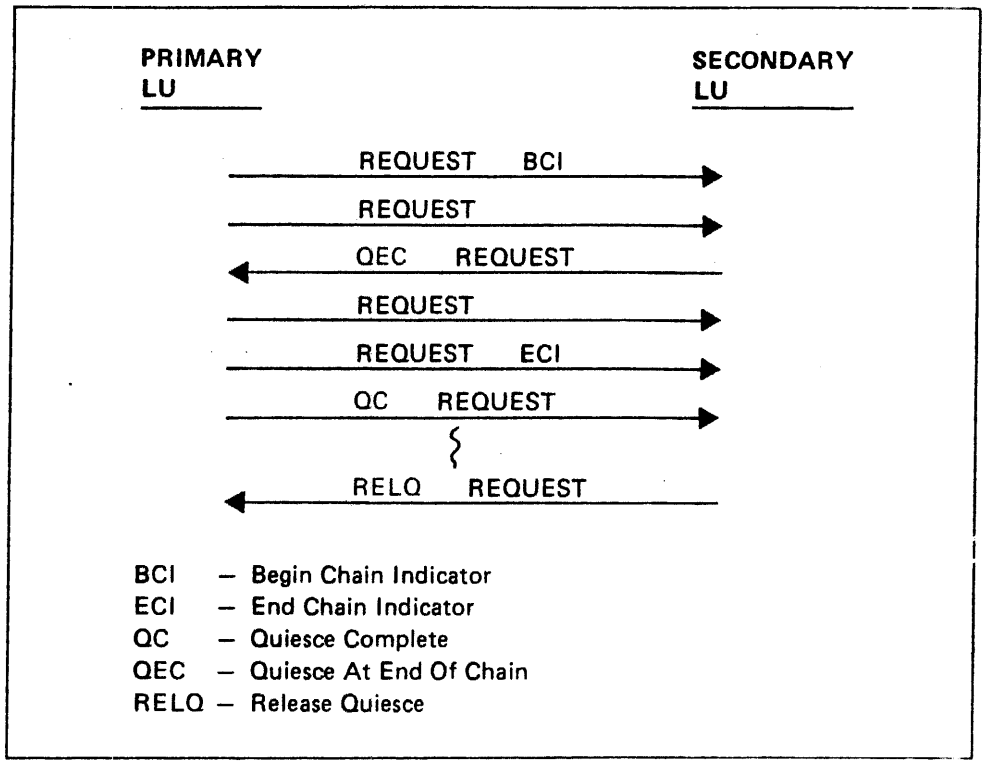

Figure 9-1. Quiesce Protocol

The primary logical unit is sending request chains to the secondary logical unit and the secondary logical unit wants to quiesce the primary logical unit until the received requests have been processed. The secondary logical unit sends the QEC request on the expedited-flow to quiesce the primary logical unit.

Once the primary logical unit receives the QEC request and sends the last element of the present request chain, it is **not** allowed to send another request on the normal-flow (except the QC request). After sending the last element of the present request chain, the primary logical unit sends the QC request to inform the secondary logical unit that it is quiesced. A quiesced logical unit can send responses on the normal-flow and can send requests and responses on the expedited-flow.

Some time later, when the secondary logical unit is ready to receive more requests on the normal-flow, it sends the RELQ request to remove the primary logical unit from the quiesced state.

Shutdown Protocol

The shutdown protocol is useful for the situation where a secondary logical unit initiates the session, starts transactions, and submits a logoff for session termination. In this situation, there are occasions when the primary logical unit wants to terminate the session.

We will use the example where an IMS logical unit has sessions with many peripheral logical units and the IMS master terminal operator submits a command to terminate IMS. IMS needs to terminate all of its LU-LU sessions, but it is not appropriate to send an UNBIND request to a secondary logical unit while there is work being done on the session.

IMS can send the SHUTDown (SHUTD) request to each secondary logical unit requesting them to stop communicating on the normal-flow. Each secondary logical unit, when it completes a unit of work, should send the SHUTdown-complete (SHUTC) request to the primary logical unit.

Once a SHUTC request is processed, the secondary logical unit is not allowed to send requests on the normal-flow. (It may send responses and expedited-flow requests.) The primary logical unit will send the UNBIND request to each secondary logical unit upon completion of each SHUTC request. Figure 9-2shows an orderly shutdown sequence. Study the figure, then read the discussion that follows.

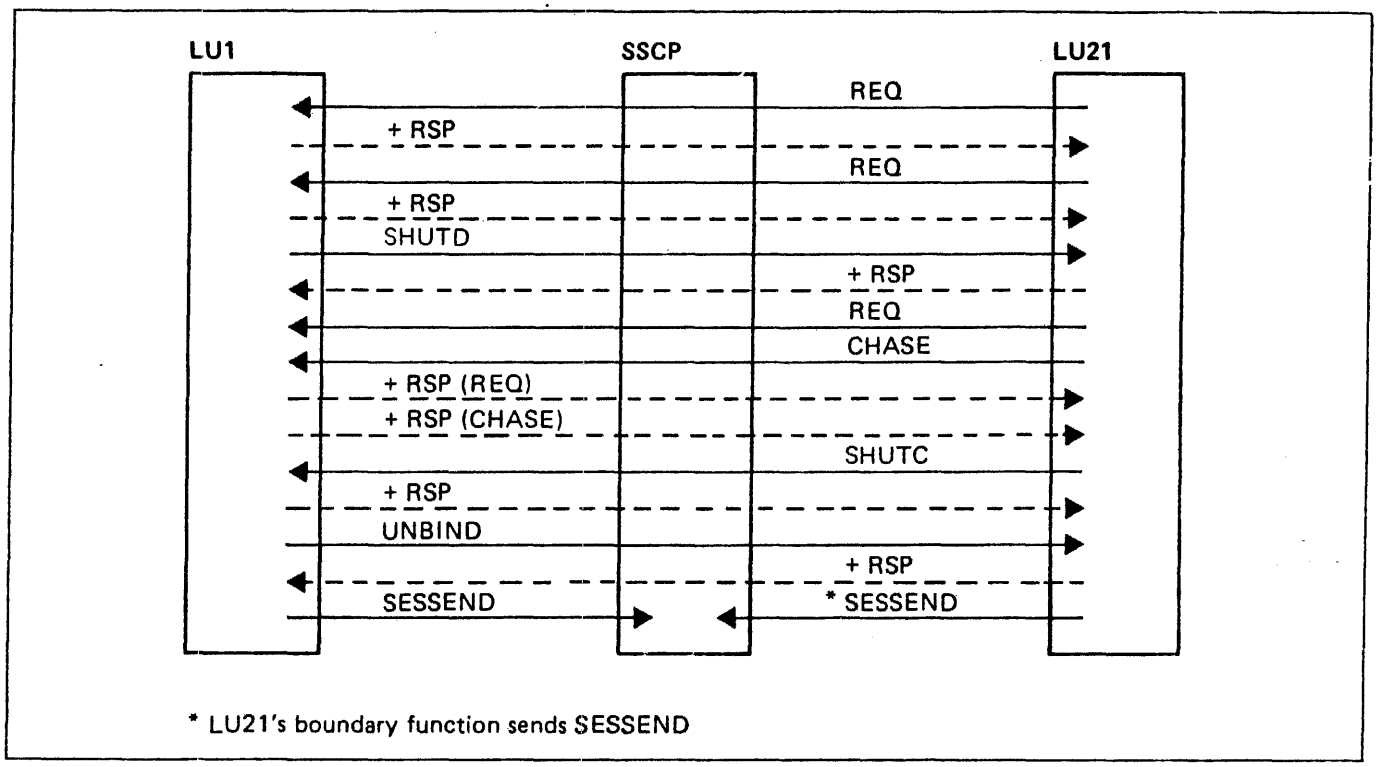

Figure 9-2. Shutdown Protocol

This orderly session closedown process is **not** assisted by the SSCP. When LU1 (primary logical unit) is ready to close the session down, it sends the SHUTD request to LU21. LU21 acknowledges the acceptance of the request with a positive response. LU21 notifies LU1 with the SHUTC request when it is ready for the session to be terminated.

Non-programmable secondary logical units are ready for session termination if they are not in the process of sending or receiving a request chain or waiting for a response, or if they are not in bracket state.

Programmed secondary logical units that have the capability, may use the CHASE request to verify whether they are ready for the session to be terminated. The secondary logical unit uses the CHASE request to determine if there are any outstanding requests to be processed. LU21 sends the CHASE request to LU1 and LU1 returns a positive response for the CHASE request only after all requests received from LU21 have been processed to completion. LU21 knows that there are no outstanding requests when the positive response for the CHASE request is received.

When LU21 is ready for the session to be terminated it sends the SHUTC request to LU1. LU21 is quiesced and is not allowed to send requests on the normal-flow. Now LU1 knows that there are no requests flowing on the session and that there are no requests to be processed. LU1 sends the UNBIND request to LU21 and LU21 returns a positive response and the session is terminated. LU1 notifies its SSCP that the session with LU21 is terminated with the SESSEND request. LU21's boundary function sends the SESSEND request to the SSCP.

Although not typical, LU1 can send the RELQ request to LU21 while it is quiesced which allows it to send requests on the normal-flow again.

● *Please turn to Mini-Course 9 in your Personal Reference Guide and read the summary.*

SNA Fundamentals

Mini-Course 10

LU-LU Data Streams

Printed in the United States of America.

Mini-Course 10. LU-LU Data Streams

Introduction

User application data is transmitted between logical units (therefore, between end users) in request units. The data is represented in either EBCDIC or ASCII interface codes. Typically, the EBCDIC interface code is used although there are some terminals that can accept and transmit data that is represented in the ASCII interface code. For this discussion we will assume that all data is represented in the EBCDIC interface code.

The data stream that flows between two logical units may be a **3270 data stream** or an **SNA character string (SCS)**. Most logical units in the 3270 family support the 3270 data stream. Peripheral logical units that support the SNA character string include the 8100, 3767, 3600, 3630, 3770, S/32, S/34, S/38, Series 1, and certain 3270s. SCS is also used in sessions between programmed logical units such as, CICS and IMS. The graphic characters in the two character strings are essentially the same but the defined control characters are different. The control functions defined by SCS are appropriate for printers, displays, storage mediums, and word processors.

A data stream can contain control information in addition to user data. The control information can be used for such things as:

- Formatting of data.
- Specifying the destination device for the data.
- Compression and expansion of data.
- Specifying the application program that is to process the data.

Some of the control information can be interspersed within the data stream at appropriate places and other control information must precede the user data. SNA has defined several headers, called **function management headers (FMHs)**, that are included in request units and are processed by logical units. Each FMH has a specific format and is used to convey specific kinds of information to a destination logical unit. This provides a standard for logical units to communicate control information to each other.

Building The Request Unit

A data stream that is printed contains control characters that format the printed page and the data streams that are displayed contain control characters that format the display screen. Either the sending logical unit inserts the control characters or the receiving logical unit inserts the control characters. Typically, a non-programmable logical unit (secondary logical unit) is not capable of inserting control characters at the appropriate places, therefore, the sending logical unit (primary logical unit) must insert the characters.

A secondary logical unit in a type 1 or type 2 node may be associated with several devices, such as a printer, display, or a diskette. Data sent from the primary logical unit may be for any one of the devices and the logical units make that decision.

The primary logical unit can include control information in the request unit along with the user-data to specify the destination device.

Figure 10-1 illustrates the flow of user-application data from a primary logical unit that resides in a type 5 subarea node to a secondary logical unit that resides in a type 1 or type 2 peripheral node.

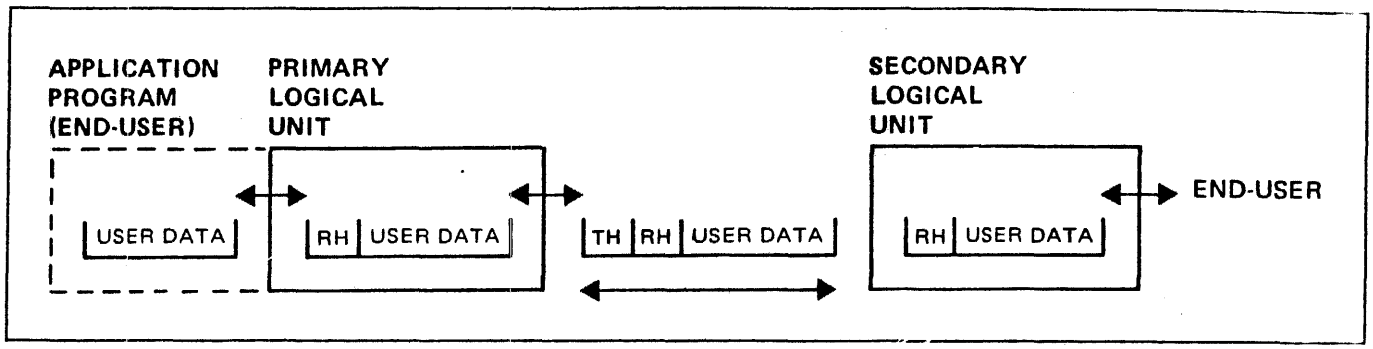

Figure 10-1. User-Application data stream

The user application program gives data to the primary logical unit to be transmitted to the secondary logical unit and the secondary logical unit is to deliver the data to its end user. Our figure does not identify the end user. This means that the data could go to a display or printer device for an operator, a disk or diskette, a card device, a tape device, or if the secondary logical unit resides in a word processing device, the information may be printed on a page.

If we assume that the secondary logical unit is associated with a single display device, chances are that the display screen is to be formatted rather than to display a continuous string of characters. In which case, the user data that goes to the display device must include control information in addition to the graphic data to be displayed. For example, three columns of data are to be displayed:

Part Name	Part Number	Price
Hand Saw	102367	$ 12.95
Hammer	103752	$ 9.95
Tape	117583	$ 14.99
	o	
	o	
	o	

As you can surmise some control information must be sent to the display device to format the display. Spaces must be included between the three fields and a new line must be started at the appropriate place. The control information can be included in the data stream at the appropriate places to format the display.

What resource in this LU-LU session supplies the control information? What resource causes user-data to be presented to the end user in the appropriate format? This function is called presentation services and can be performed by the sending end user, the sending logical unit, or the receiving logical unit. If we are

going to have the end user perform the application function and nothing else, then one of the logical units must provide presentation services. For example, CICS and IMS logical units provide presentation services for formatting data on display screens, called Basic Mapping Support and Message Formatting Services.

If our secondary logical unit resides in a 3270 controller, the primary logical unit must provide presentation services. The primary logical unit includes control information in the user data stream at the appropriate places.

Selecting Data Stream Types

A data stream to a 3270 device can be either a 3270 data stream or an SNA character string (SCS) depending on which data stream the logical unit supports. Control characters are different in the two data streams, therefore the primary logical unit must know which data stream to use to provide presentation services for the secondary logical unit. The logical units find out at BIND time what data stream is to be used. One field in the BIND request identifies the logical unit type. One of the definitions of a logical unit type indicates the type of data stream. Logical unit types are discussed in the Mini-Course entitled "Logical Unit Types."

We now know that a user data stream contains user data and probably control information for presenting the data to the end user in the appropriate format. Presentation services may be provided by the sending end user, the sending logical unit, or the receiving logical unit. Presentation services capability within terminal logical units vary from almost none to considerable. Programmed logical units can have minimum to maximum capability based on the implementation.

Terminal Logical Units With Multiple End Users

We will now consider an LU-LU session in which a secondary logical unit is associated with three devices (end users) as shown in Figure 10-2.

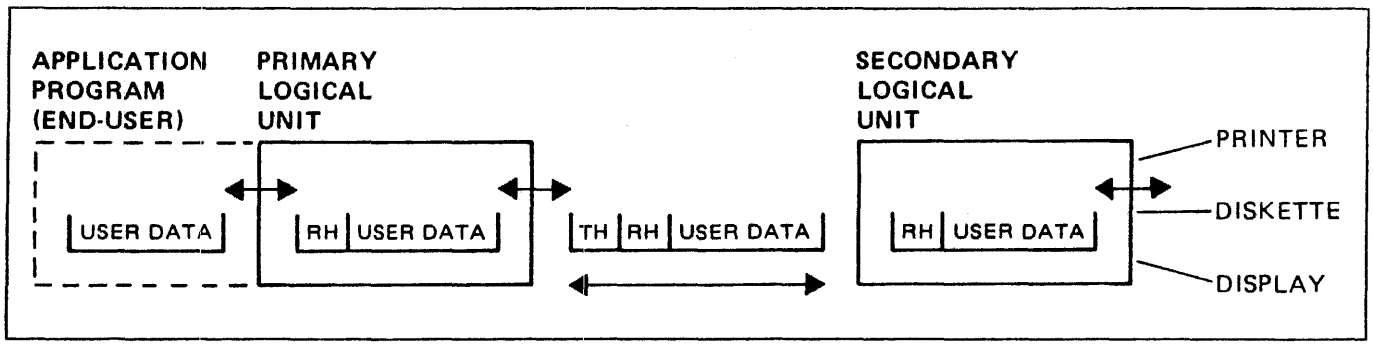

Figure 10-2. Terminal LU With Multiple End Users

Requests sent by the primary logical unit may be for any of the three end users (printer, diskette, display). Each request chain received by the secondary logical unit must be given to the appropriate device. Information included in the data stream is used in selecting the end user.

Function Management Headers (FMHs): SNA has defined several header formats to assist session partners in making decisions of this nature. These headers are called function management headers (FMHs). Each FMH is defined to provide particular kinds of information for the destination logical unit.

Not all logical units support function management headers (FMHs) but for those LU-LU sessions that do, data transmissions can look like those in Figure 10-3.

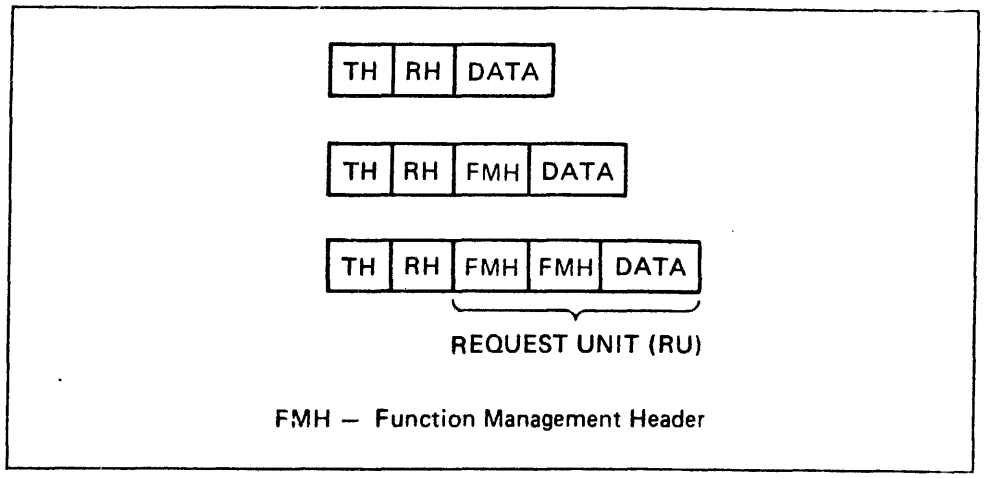

Figure 10-3. Data Streams With FMHs

A BIND field in the BIND request specifies whether FMHs can be used in the LU-LU session. Assuming that FMHs are allowed, not all requests will contain an FMH. A receiving logical unit recognizes the presence of an FMH by two request header (RH) indicators (request category indicator and format indicator). The request category indicator specifies a function management data (FMD) request and the format indicator specifies that the request unit is formatted. On an LU-LU session, a formatted FMD request contains a function management header (FMH).

An FMH appears first in a request unit and each FMH type has a specific format. A logical unit can determine whether a second FMH is included by inspecting the concatenation field of the first FMH. Figure 10-3 shows two concatenated FMHs.

We will now take a look at our LU-LU session that is repeated in Figure 10-4.

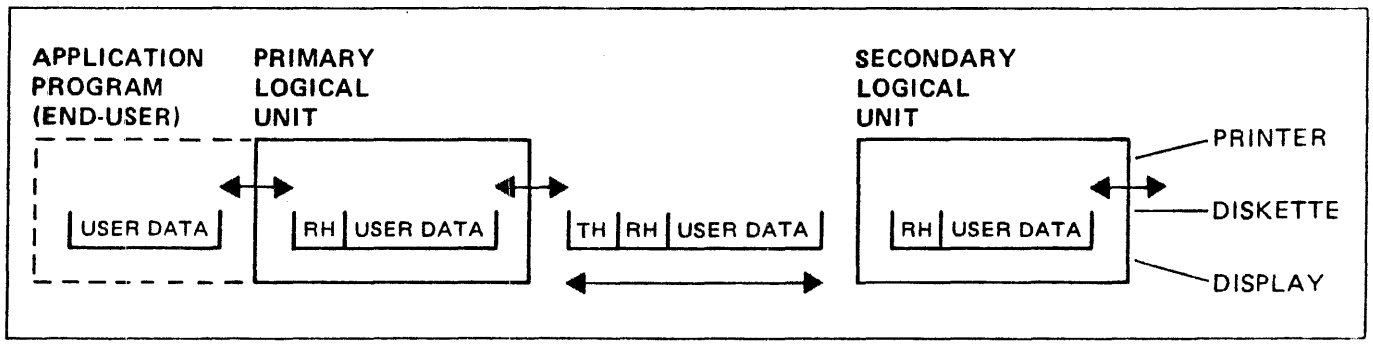

Figure 10-4. Secondary LU With Multiple End Users

Several kinds of information can be included in the FMH sent to this secondary logical unit. An FMH-1 can specify the destination device (printer, diskette, display) for the data. Requests for the diskette could include FMH-2s which specify data management activities that include:

- Creating a data set.
- Scratching a data set.
- Erasing data set records.
- Adding records.
- Replacing records.

Compression and Compaction

A sending logical unit could compress or compact a data stream to shorten network transmissions. The receiving logical unit expands or decompacts the data stream before giving the data to the end user.

Compression of data eliminates gaps, empty fields, or redundancies. One technique is to select one character, called the prime compression character, and replace repetitive sequences of that character with a string control byte (SCB). Unless otherwise specified, a blank (X'40') is the prime compression character. Figure 10-5 shows a request sequence sent by the primary logical unit in which the prime compression character is a blank.

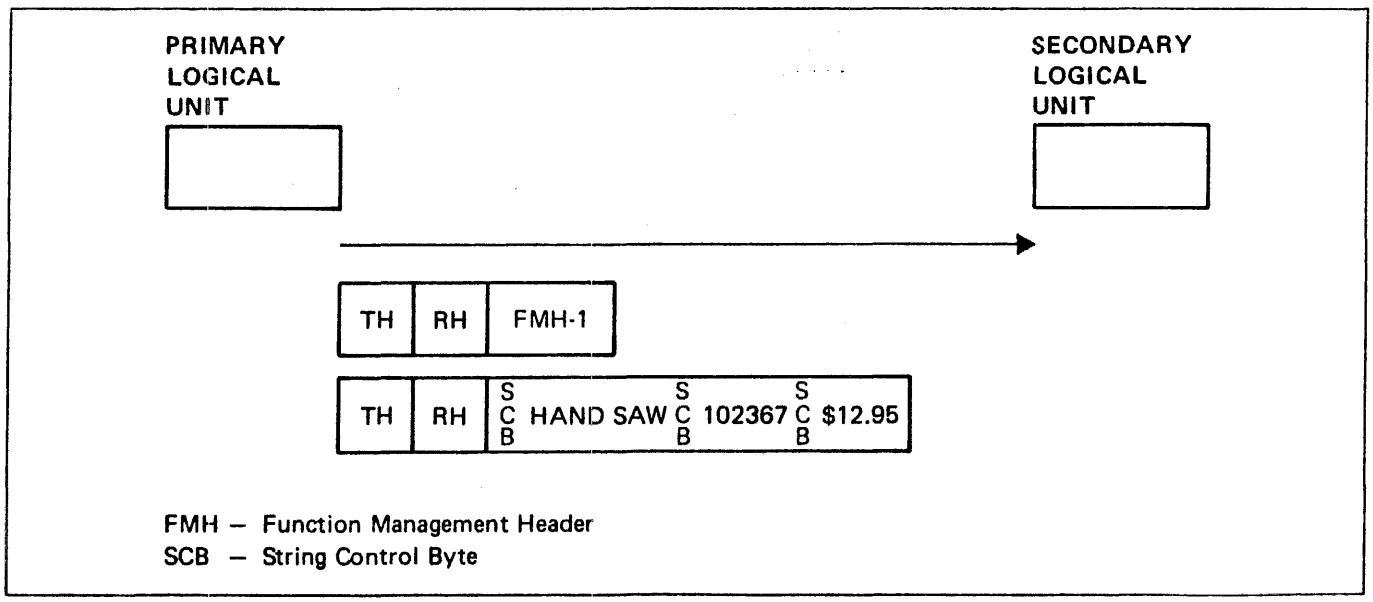

Figure 10-5. Compression

The sending logical unit uses the FMH-1 to notify the receiving logical unit that the request unit contains compressed data. For a prime compression character other than blank, the sending logical unit sends an FMH-2 or FMH-3 to identify the character. SCBs are included in the data stream to specify the number of repetitive blanks that are compressed. Once compression has started, all following requests received by the secondary logical unit are considered to contain compressed data until an FMH-1 is received to specify the end of compressed data.

Compaction is used to shorten the length of transmissions. A technique used is to compact two bytes of data into one byte. Up to 16 characters (called master characters) can be specified to be compacted. The sending logical unit compacts the data and sends it to the receiving logical unit which decompacts the data before giving it to the end user. The sending logical unit must notify the receiving logical unit that the data is compacted and what characters may be compacted. An FMH-1 is included in the data stream to indicate that the data is compacted. A compaction table is sent via an FMH-3 to the secondary logical unit to identify master characters that may be compacted. String control bytes (SCBs) must also be included in the data stream to identify the beginning of each block of compacted data.

Data Stream For Two Programmed LUs

Two programmed logical units (e.g., CICS and IMS) can use FMH-4, FMH-5, FMH-6, FMH-7, and FMH-10 to communicate control information to each other. For example, a sending CICS logical unit can include an FMH-5 in a request unit that tells the receiving CICS logical unit which application program is to process this transaction. An FMH-7 is used to supply additional information on a previously sent negative response. An FMH-10 is sent to prepare the session for a sync point.

This Mini-Course has mentioned only some of the many uses of function management headers. But you now know that they exist, you know what resources use them, you know that they reside in request units that flow on LU-LU sessions, and you are aware of some of their uses.

● *Please turn to Mini-Course 10 in your Personal Reference Guide and read the summary.*

SNA Fundamentals

Mini-Course 11

Logical Unit Types

Printed in the United States of America.

Mini-Course 11. Logical Unit Types

Introduction

The logical unit is the means by which end users communicate with other end users[1]. A logical unit provides the following services within an LU-LU session:

- End user services. The logical unit must be able to accept data from an end user and also give data to an end user in the appropriate format. One category of end user service is called **session presentation services**. Session presentation services formats data that is sent to the end user (e.g., displays, printers, disk, tape, and card punches).

- Data flow control services. A pair of logical units must be able to maintain the integrity of the data flow within their session. Some of the functions supported by data flow control services include:

  - Send/Receive modes (full-duplex and half-duplex).

  - Chaining.

  - Brackets.

  - Response options.

  Earlier we discussed a set of SNA requests (called data flow control requests) which are used by data flow control services to assist in maintaining the integrity of the data flow within the session.

- Transmission control services. Transmission control routes requests and responses into the path control network and receives requests and responses from the path control network. Transmission control keeps track of the status of the session, helps pace the flow of requests within it, assures the correct sequencing of requests within the session, and can encipher and decipher end user data.

Figure 11-1 illustrates the SNA layers within a logical unit that provide end user services, data flow control services, and transmission control services.

---

1 We previously defined an end user as either an application program or a person. Information that is destined for a printer, disk, or tape is for use by a person who is the end user. In this discussion, we will refer to a printer or storage medium (e.g., disk or tape) as an end user.

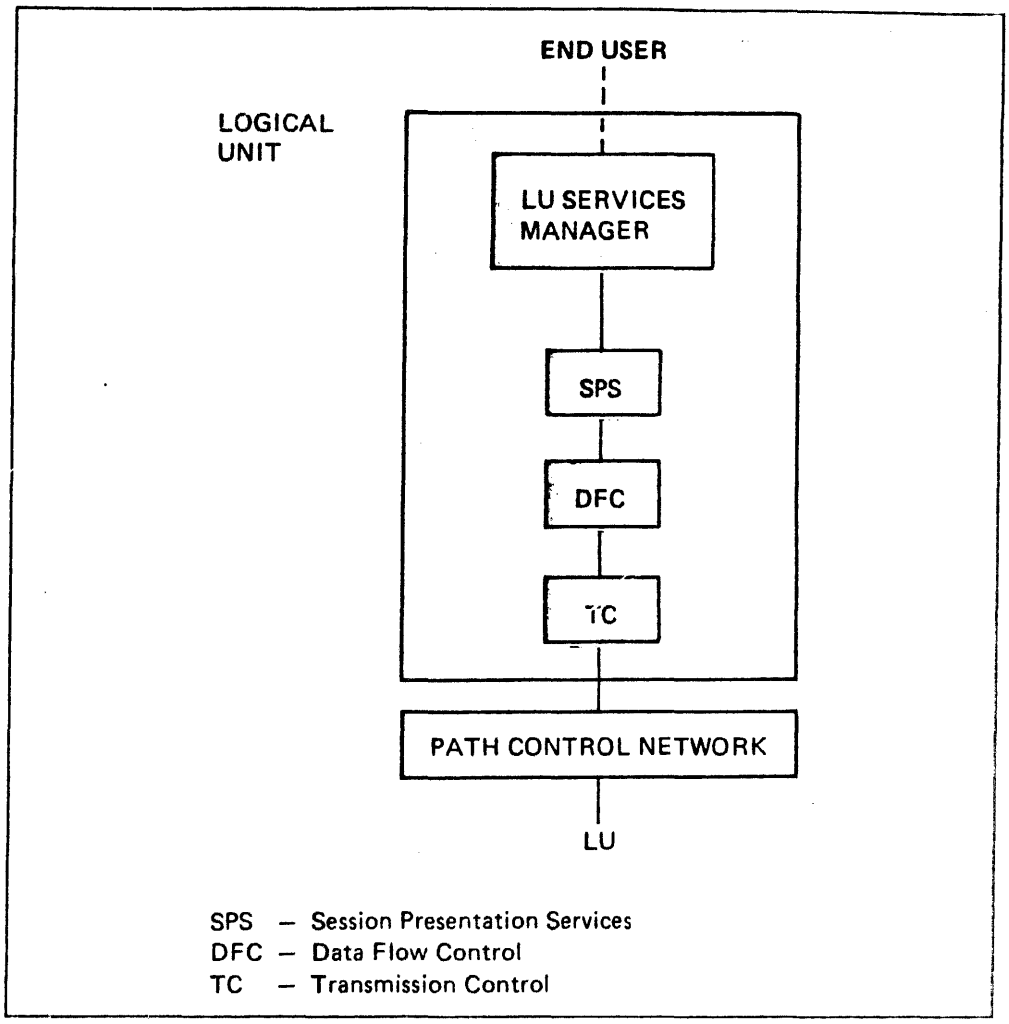

Figure 11-1. Logical Unit Services

The logical unit services manager controls the end user activities. It is responsible for sending data to and receiving data from the end user.

The information flow is from an end user to the LU services manager, from the LU services manager to data flow control services, to transmission control services, and into the path control network. A request travels through the path control network to a destination logical unit. Transmission control services within that logical unit receives the request from the path control network, processes it and gives it to data flow control services. Data flow control services processes the request and gives the data to session presentations services to be formatted for the end user. The LU services manager handles the transfer to the end user.

The formatting by session presentation services can be done at either the origin logical unit or the destination logical unit.

SNA Profiles

We have discussed some of the SNA functions provided by logical units within an LU-LU session. Some of the SNA functions are provided by **session presentation services**, some are provided by **data flow control services**, and some are provided by **transmission control services**.

SNA defines more functions than are needed or can be used by some end users. Consider an operator using an IBM 3767 Communications terminal to communicate with an application program in a host subarea node. The logical unit in the 3767 terminal only needs to support a subset of SNA functions. Certain LU functions are mandatory for this logical unit, some are optional, and some are not supported.

Each SNA product provided by IBM supports a subset of SNA functions. SNA defines each subset of SNA functions as a profile by assigning a profile number. SNA defines three profile categories as illustrated in Figure 11-2.

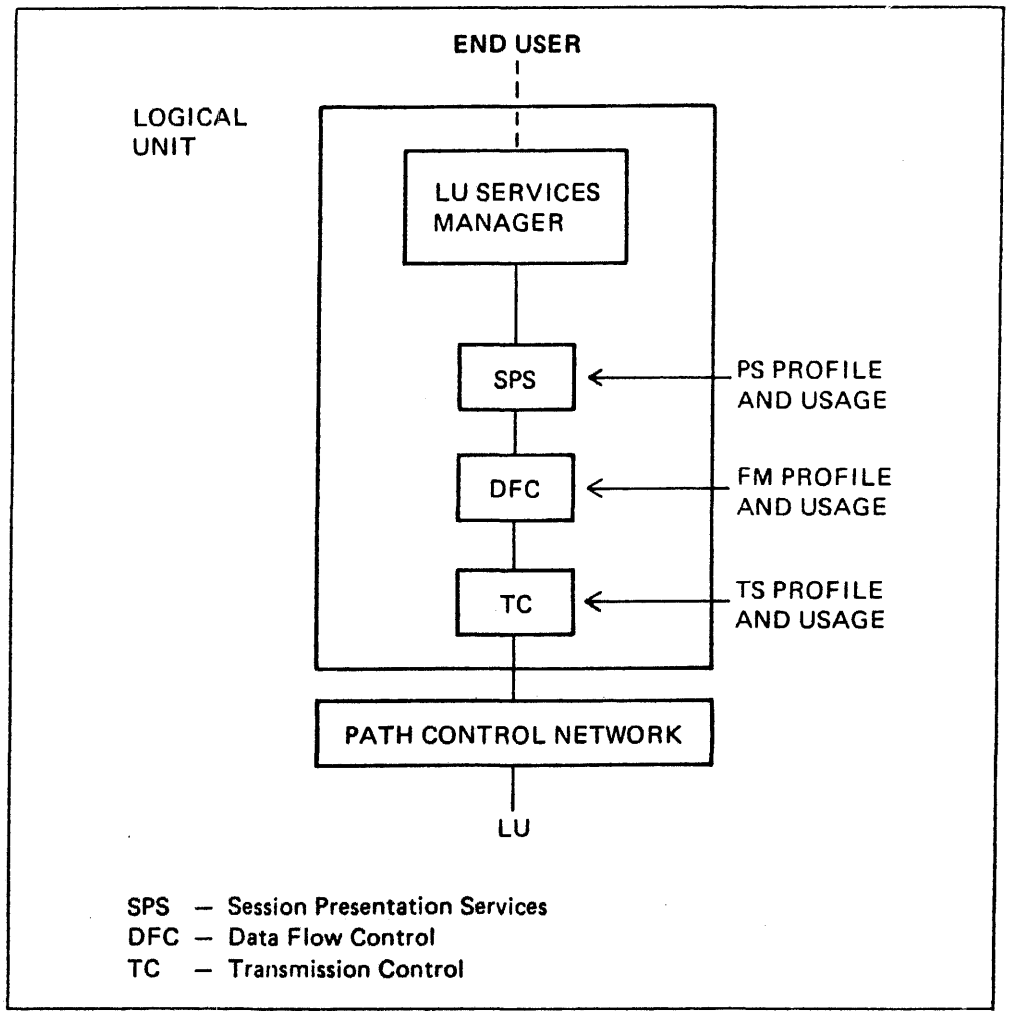

Figure 11-2. BIND Profiles

1. Presentation services (PS) profile. This subset of SNA functions is implemented by the session presentation services component of the logical unit. The functions that can be selected include:

- Function management (FM) header usage.
- SNA character string usage.
- Code to be used (e.g., EBCDIC or ASCII).

2. Function management (FM) profile. This subset of SNA functions is implemented by the data flow control (DFC) services component of the logical unit. The functions that can be selected include:
   - Send/receive modes (half-duplex or full-duplex).
   - Request and response modes.
   - Brackets and chaining rules.
3. Transmission services (TS) profile. This subset of SNA functions is implemented by the transmission control services component of the logical unit. The functions that can be selected include:
   - Pacing.
   - Allowable session control requests.
   - Request unit size.
   - Cryptography options.

SNA defines several FM profiles, PS profiles, and TS profiles. The BIND request that establishes an LU-LU session must specify an appropriate profile (FM, PS, and TS) for the two logical units. We will take a look at an FM profile (FM profile 3) to get an idea of the functions selected.

FM Profile 3

Both session partners use immediate response mode.

Both session partners support the following
data flow control requests:

CANCEL
SIG
LUSTAT
CHASE
SHUTD
SHUTC
RSHUTD
BID and RTR (only if brackets are used)

FM usage, TS usage, and PS usage further define the functions that are supported by a pair of logical units for their LU-LU session.

Fields in the BIND request are set to select profiles. Figure 11-3 shows the format of the BIND command.

Byte	Field Description
0	Identifies this RU as a BIND command
1	Negotiable or non-negotiable BIND
2	FM Profile
3	TS profile
4-7	FM usage primary protocols secondary protocols common protocols
8-13	TS usage pacing maximum RU size
14	PS profile (identifies type of LU-LU session)
15-25	PS usage primary logical unit FM headers Data streams
26-27	Cryptography
28-n	names and end user data

Figure 11-3. BIND Command Format

The fields located in bytes 0 through 13 were discussed in the Mini-Course entitled "Data Flow Control on LU-LU Sessions" and earlier in this Mini-Course. The presentation services (PS) profile field (byte 14) identifies logical unit types. We are going to discuss logical unit types next.

The fields located in bytes 15-25 specify function management (FM) header usage, if any, and identify data streams to the destination end user. Bytes 26 and 27 specify whether cryptography is supported, and if it is supported, which cryptography options. Bytes 28-n contain the BIND sender's name and any user data.

SNA products are classified according to the SNA functions that they support. The logical unit(s) that reside in each product assume(s) that classification. The classification type of each logical unit designates a particular subset of SNA functions that the product can perform when its logical unit is in session with a logical unit in another product.

SNA defines logical unit types 0 through 7. The logical unit type designation is a convenient way of classifying SNA products according to the subsets of SNA functions they perform. It is useful in selecting a set of SNA products to use in a particular application.

Some SNA products support more than one logical unit type. For example, the logical unit provided by CICS can participate in LU-LU sessions with logical unit types 0 through 6. It can participate in LU-LU sessions with peripheral logical units and subarea logical units. In contrast, a logical unit in an IBM 3767 Communication Terminal is classified as a type 1 logical unit and can participate in an LU-LU session with another logical unit that supports the same protocols.

Logical unit types are characterized by their associated profiles. Figure 11-4 lists each logical unit type and the allowable subsets that can be specified by the TS, FM, and PS profiles.

PS Profile (Session Type)	FM Profile	TS Profile	FM Profile PS Characteristics
0	2,3,4,7	2,3,4,7,18	Any option desired
1	3,4	3,4	SNA character string FM Headers usage Data processing media supported
2	3	3	3270 data stream No FM headers Display support
3	3	3	3270 data stream No FM headers Printer support
4	7	7	SNA character string FM headers usage Data processing and word processing support
5			Reserved
6	4	18	SNA character string or field format FM headers usage Program-to-program support for data bases, files, queues, and programs
7	7	7	5250 Data Stream FM headers usage Display support

Figure 11-4. Logical Unit Types

Type 0 Logical Unit: Type 0 logical units are implementation-dependent and do not fall within the groupings of profiles defined by SNA. IBM 3271 Information Display Systems (models 11 and 12) are defined to SNA networks as type 0 logical units.

Type 1 Logical Units: Type 1 logical units are for application programs that communicate with terminals. They support single or multiple devices in interactive, batch, or distributed processing environment. Devices may include consoles, printers, diskettes, disks, and card units. This session uses the SNA character string (SCS). Three types of function management (FM) headers (FMH-1, FMH-2, and FMH-3) are allowed for this session. The IBM 3767 Communication Terminal is an example of a type 1 logical unit. It does not support FMHs.

Type 2 Logical Units: Type 2 logical units are for application programs that communicate with single display devices in an interactive environment. The session uses the SNA 3270 data stream and operates in half-duplex flip-flop mode.

Function management (FM) headers are not supported. The IBM 3274 controller contains type 2 logical units.

Type 3 Logical Units: Type 3 logical units are for communication between application programs and a single printer (without a keyboard) using the SNA 3270 data stream. Function management (FM) headers are not supported. The printer logical unit can accept multiple element chains but does not send multiple element chains. The printer logical unit operates only in half-duplex flip-flop mode and it returns a change direction indicator (CDI) immediately when the primary logical unit sends a CDI. A logical unit in an IBM 3274 controller that is associated with a 3287 printer can be bound as a type 3 logical unit.

Type 4 Logical Units: Type 4 logical units are for data communication between two terminal logical units or between an application program and a terminal that supports single or multiple devices. Type 4 logical units are similar to type 1 logical units.

Type 4 logical units do not require a network with a separate SSCP if the session partners handle required SSCP functions. If the logical units are part of a network with an SSCP, both logical units must be in session with the SSCP before there can be an LU-LU session.

Function management (FM) headers are allowed and the SNA data stream is used. The type 4 logical unit can use the controls for data processing and word processing provided by the SNA data stream.

The IBM 6670 Information Distributor is an example of a type 4 logical unit.

Type 6 Logical Units: Type 6 logical units are for data communication between application subsystems. CICS and IMS are examples of type 6 logical units. A CICS logical unit can establish a session with another CICS logical unit or with an IMS logical unit. An IMS logical unit can establish a session with another IMS logical unit as well as with a CICS logical unit.

Type 7 Logical Unit: A type 7 logical unit is for an application program that communicates with a single display terminal in an interactive environment, for example, a session involving an application program in a System/34 and an IBM 5251 Display Station.

LU Type 2 Example

Now that we have discussed logical unit (LU) types, the profiles that characterize LU types, and the BIND request that establishes sessions between equivalent LU types, we will discuss a session example whose session partners are type 2 logical units. We will use CICS and a 3274 controller as our type 2 logical units (see Figure 11-5).

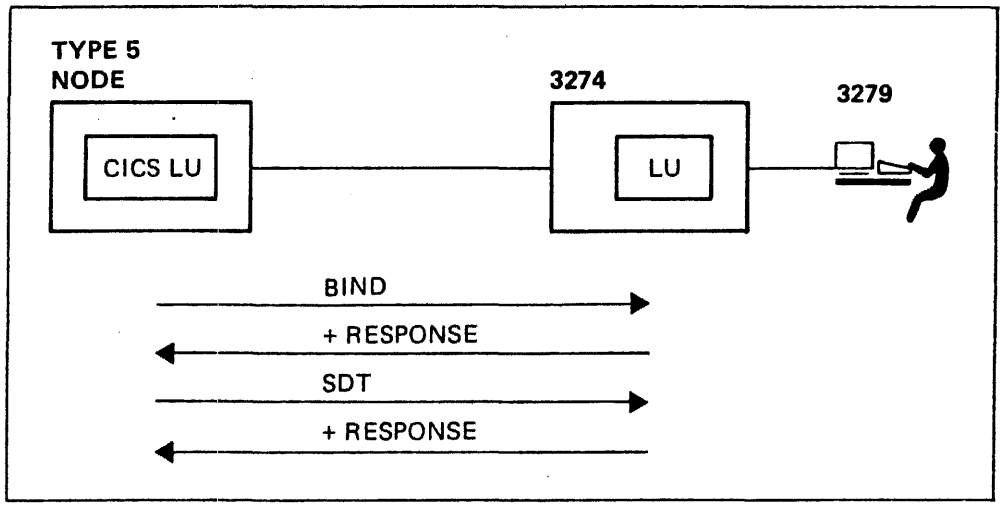

Figure 11-5. Type 2 Logical Unit Example

The BIND request that is sent by CICS to the 3274 logical unit must specify appropriate session parameters for type 2 logical units as shown in Figure 11-6.

Byte	Description
0	BIND code (X'31')
1	Nonnegotiable BIND
2	FM profile 3
3	TS profile 3
4-7	**FM Usage**
4	Primary Protocols Chaining Request Control Mode Chain Response Protocol Compression Indicator Send End Bracket Indicator
5	Secondary Protocols Chaining Request Control Mode Chain Response Protocol Compression Indicator Send End Bracket Indicator
6-7	Common Protocols FM Header Usage Brackets Usage Alternate Code Set Send/Receive Mode Recovery Responsibility Contention Winner/Loser
8-13	TS Usage Pacing Maximum RU Size
14	PS Profile LU Type 2
15-25	PS Usage Screen Size

Figure 11-6. BIND Request For Type 2 Logical Unit Session

Byte 0 identifies this as a BIND request. Byte 1 specifies that this is a nonnegotiable BIND. Negotiable BIND is not valid for a type 2 logical unit.

Byte 2 specifies the function management (FM) profile for this session (FM profile 3). This profile specifies that CICS and the 3274 logical unit use immediate response mode, that is, responses are always returned in the order that requests are received. When request 1 is received before request 2, the response to request 1

must be returned before the response to request 2. This profile also specifies that the following data flow control (DFC) requests can be used by both session partners.

```
CANCEL
SIGNAL
LUSTAT
CHASE
SHUTD
SHUTC
RSHUTD
BID
RTR
```

Both session partners can send CANCEL and SIGNAL requests. Only CICS sends the requests, CHASE, SHUTD, and BID. Only the 3274 logical unit sends the requests, LUSTAT and SHUTC. RTR is not used by the type 2 3274 logical unit.

Byte 3 specifies transmission services (TS) profile 3. This profile allows pacing from primary-to-secondary and secondary-to-primary and specifies that data traffic is controlled by the CLEAR and SDT requests. A BIND request and an associated positive response establishes a session in the data-traffic-inactive state allowing only certain control requests to flow on the session. The SDT request must be sent by CICS to the 3274 logical unit to change the session to "data traffic active" state. Then traffic, including user-data, can flow on the normal-flow. CICS can send the CLEAR request to reset the session to the data-traffic-inactive state. Error recovery and resynchronization of sequence numbers are performed when the session is in the data-traffic-inactive state.

Bytes 4, 5, 6, and 7 provide FM usage for CICS and the 3274 logical units. FM usage is supplemental information to the FM profile. Byte 4 specifies protocols for CICS (the primary logical unit) and byte 5 specifies protocols for the 3274 LU (the secondary logical unit). Bytes 6 and 7 specify protocols that are common to both logical units.

Byte 4 specifies protocols for the primary logical unit (CICS).

- The primary logical unit (CICS) may send multiple-element request chains.
- Immediate request mode is used which means that only one definite response can be outstanding at a time. The response must be received before the primary logical unit can send another request.
- The primary logical unit may ask for exception-only responses, definite responses, or exception or definite responses.
- The primary logical unit cannot send compressed data.
- The primary logical unit will send the end bracket indicator if brackets are used.

Byte 5 specifies protocols for the secondary logical unit (the 3274 logical unit).

- The secondary logical unit may send multiple-element request chains.
- Immediate request mode is used which means that only one definite response can be outstanding at a time. The response must be received before the secondary logical unit can send another request.
- The secondary logical unit may ask for exception-only responses, definite responses, or exception or definite responses.
- The secondary logical unit cannot send compressed data.
- The secondary logical unit **cannot** send the end bracket indicator.

Bytes 6 and 7 specify protocols common to both logical units.

- FM headers cannot be used.
- Brackets are used.
- Alternate code set (e.g., ASCII) may be used by CICS and the 3274.
- The send/receive mode must be half-duplex flip-flop.
- CICS is responsible for error recovery.
- The 3274 logical unit is the brackets first speaker.
- Any contentions (simultaneous transmissions by CICS and the 3274 logical unit) are resolved in favor of the 3274 logical unit. CICS accepts the transmission from the 3274 logical unit but the 3274 logical unit rejects the transmission from CICS.

Bytes 8 -13 (TS usage) supplement the information supplied by the TS profile field. TS usage specifies whether pacing is to be used from primary-to-secondary and from secondary-to-primary, and and if pacing is to be used, it specifies the pacing count.

TS usage also specifies the maximum request unit (RU) size that can travel from primary-to-secondary and the maximum RU size that can travel from secondary-to-primary.

Byte 14 (PS profile) specifies logical unit type 2.

Bytes 15-25 (PS usage) supplement the PS profile field. PS usage specifies screen size of the display associated with this secondary logical unit.

Please turn to Mini-Course 11 in your Personal Reference Guide and read the summary

SNA Fundamentals

Mini-Course 12

Multiple Domains

Printed in the United States of America.

Mini-Course 12. Multiple Domains

Introduction

A single domain network includes one SSCP. That SSCP owns and controls all the SNA resources in the network. The SSCP assists its logical units to establish LU-LU sessions.

A multiple domain network includes two or more SSCPs, that is, two or more SNA access methods. A multiple domain network may include one host node or multiple host nodes. For example, ACF/VTAM and ACF/TCAM can reside in the same host node. Also, a host node that is controlled by VM can support multiple ACF/VTAMs and/or multiple ACF/TCAMs.

Some SNA resources in an SNA network can be owned by more than one SSCP at the same time. Other SNA resources can be owned by only one SSCP at a time and logical units fall in this category. When two logical units are owned by different SSCPs, both SSCPs must communicate with each other to establish a session between the two logical units. We will discuss how this is accomplished.

Subareas and Domains

A **domain** is made up of an SSCP and the SNA resources that it controls. An SSCP controls physical units, logical units, and links. An SNA domain includes one or more **subareas**. A subarea is a collection of SNA resources directly controlled by a type 5 subarea node (SSCP) or a type 4 subarea node (network control program - ACF/NCP/VS).

Figure 12-1 shows two SNA domains, each containing two subareas. For easy reference we will call one domain A and the other domain B.

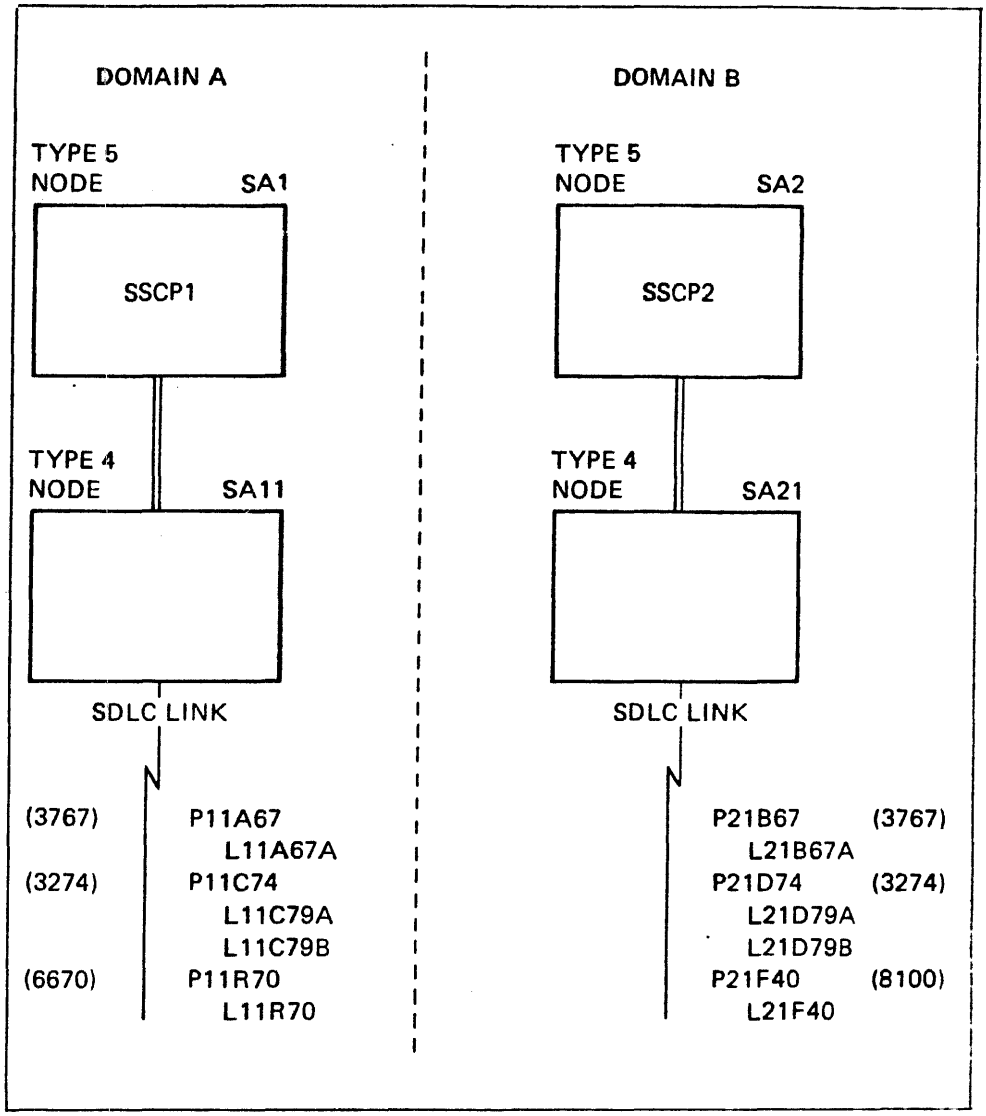

Figure 12-1. SNA Domains

Domain A includes a type 5 subarea node (subarea 1) and a type 4 subarea node with attached peripheral nodes (subarea 11). Domain B includes a type 5 subarea node (subarea 2) and a type 4 subarea node with its attached peripheral nodes (subarea 21).

Domain A and domain B are independent, non-attached domains. There is no way that logical units in domain A can communicate with logical units in domain B. There are several benefits of connecting domains to form one SNA network rather than leaving them as independent networks. One of the main benefits is that end users associated with peripheral logical units in each domain can have access to host application programs in all domains. Also, application programs associated with host logical units in each domain can have access to host application programs in all domains.

Another benefit of multiple domain networks is that there are more similar network components in the network. These provide more backup for failing components. For example, in a multiple domain network that includes two or more host nodes and one of the host nodes fails another host node may be able to take over the resources controlled by the SSCP in the failed host node. SNA resources need a controlling SSCP in order to participate in LU-LU sessions. Also, applications that were running in the failed host node may be able to be started in a backup host node.

Now we will discuss how to connect multiple domains to form a multiple domain network. Also, we will discuss what must be done to allow logical units in one domain to communicate with logical units in another domain.

Connecting Domains

Domains can be connected by SDLC links or by processor channels. Figure 12-2 illustrates the ways that domain A can be connected to domain B.

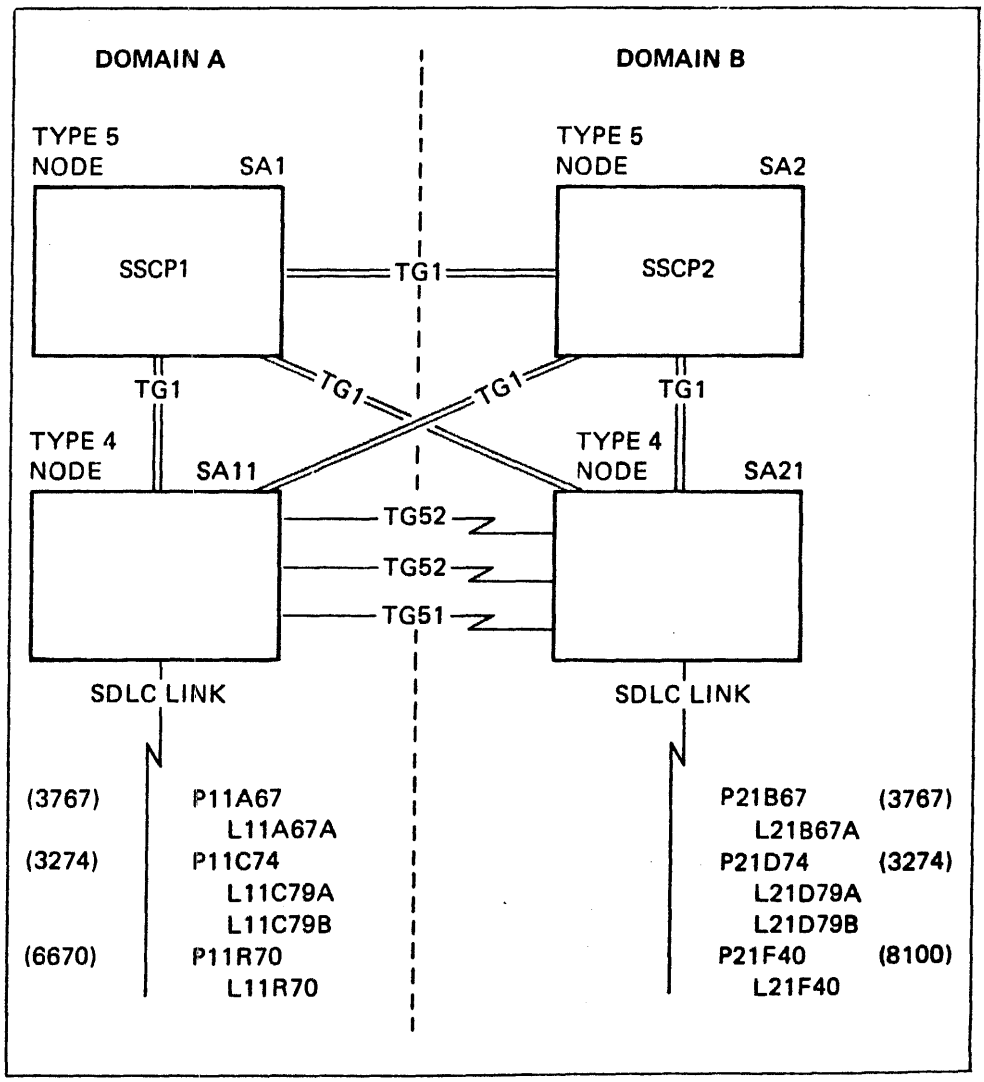

Figure 12-2. Connecting Domains

SDLC links connect the two type 4 subarea nodes. For this type of connection the components of domain A and domain B can be located in the same room, in different buildings, or in different geographical locations. Another connection is that both host nodes are connected to both type 4 subareas nodes by processor channels. With this type of connection the four nodes must be located close to each other. The other connection is the processor channel that connects the two host nodes. Here the two nodes must be close to each other.

Any or all of the connections shown in Figure 12-2 can be used to connect domains. One processor channel or a single SDLC link can be used to connect the two domains.

Establishing SSCP-SSCP Sessions

An SSCP manages the process of establishing LU-LU sessions between logical units controlled by that SSCP. When a logical unit in one domain is to establish an LU-LU session with a logical unit in another domain, both controlling SSCPs manage the process of establishing the session. There must be an SSCP-SSCP session between the two SSCPs before they can handle the process of establishing LU-LU sessions.

An SSCP has two logical components:

1. One logical unit is called the SSCP. It has a unique network address. We have already discussed the functions of the SSCP.

2. The other logical component is called a cross domain resource manager (CDRM) and it has a unique network address. The SSCP manages sessions within its domain. Cross domain resource managers manage the process of establishing and terminating LU-LU sessions between logical units in different domains, called cross domain sessions.

Figure 12-3 identifies a CDRM for each domain.

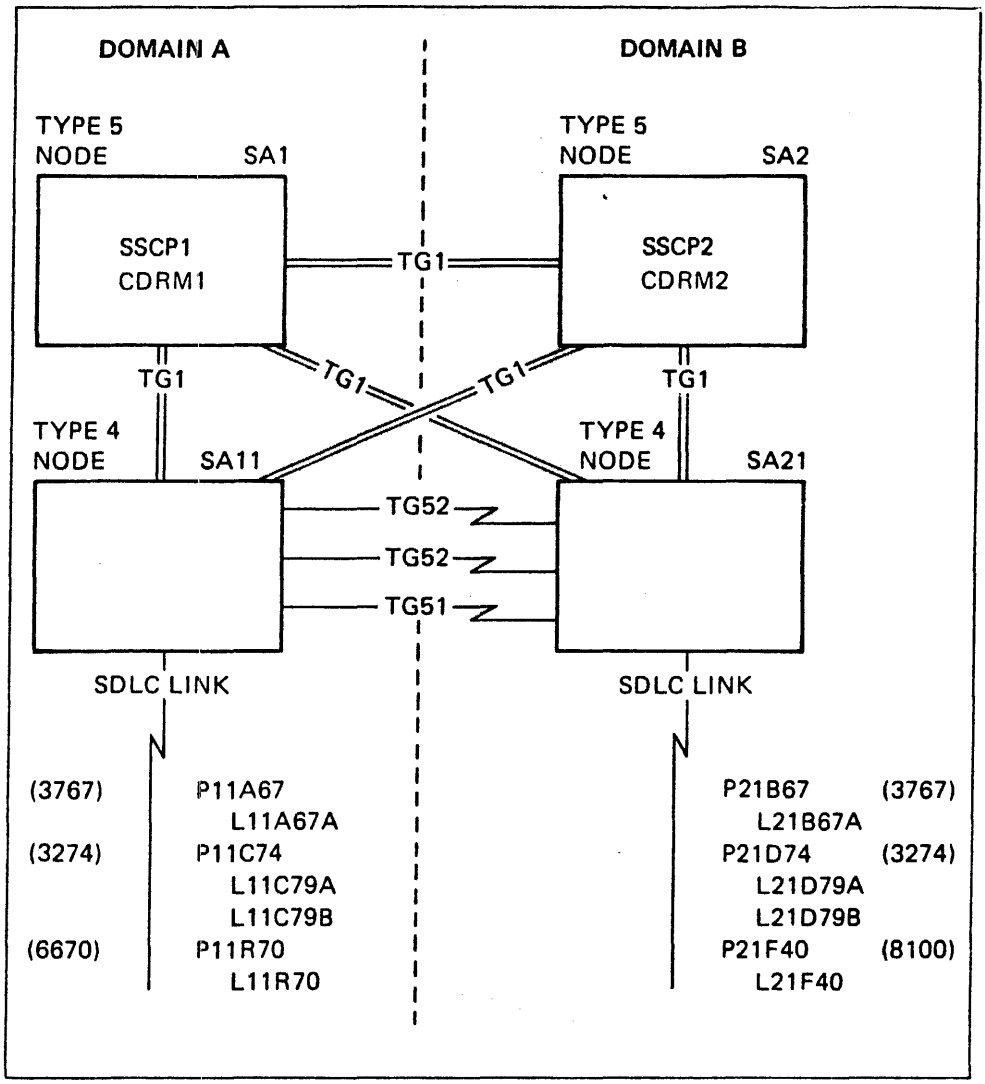

Figure 12-3. Cross-Domain Resource Managers (CDRMs)

The network definer must define to each domain its own cross domain resource manager (CDRM) as well as CDRMs in other domains that the CDRM will communicate with. We will assume that the name CDRM1 is defined in both domains to represent the cross domain resource manager in subarea 1 and that the name CDRM2 is defined in both domains to represent the cross domain resource manager in subarea 2.

Domain A	**Domain B**
CDRM1 SUBAREA 1	CDRM1 SUBAREA 1
CDRM2 SUBAREA 2	CDRM2 SUBAREA 2

The above definitions represent definitions in both domains. Domain A recognizes CDRM1 as the name of its CDRM because the definition specifies that it resides in subarea 1. The other CDRM is external to this domain because it is defined to be in another subarea. In domain A CDRM1 is called the host CDRM and CDRM2

the external CDRM. In domain B CDRM2 is called the host CDRM and CDRM1 the external CDRM.

In domain A CDRM1 must be active before a session can be established with CDRM2. In domain B CDRM2 must be active before a session can be established with CDRM1. That is, a host CDRM must be active before it can go into session with an external CDRM.

The CDRM definitions can specify that the host CDRM be activated when the SNA access method is initialized. The host CDRM can also be activated by the network operator submitting an activate command to the SSCP. We will assume that both host CDRMs are active. Now we need to establish a session between CDRM1 and CDRM2. The activation process can be initiated in either domain. We will assume that the network operator in domain A submits an activate command that specifies that a session be established between CDRM1 and CDRM2. Figure 12-4 shows the command request that establishes the CDRM1-CDRM2 session.

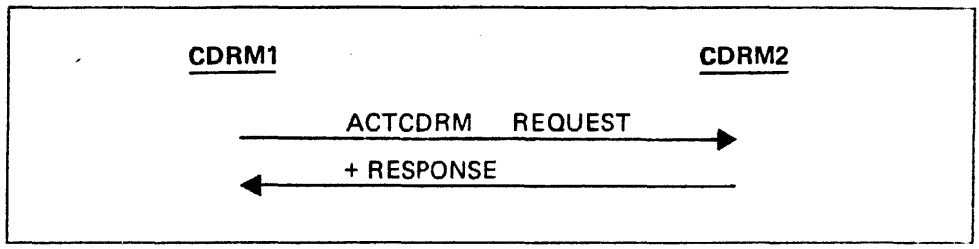

Figure 12-4. Establishing CDRM-CDRM Session

CDRM1 sends an ACTivate Cross Domain Resource Manager (ACTCDRM) request to CDRM2. If CDRM2 is active and if CDRM1 is defined to CDRM2, then CDRM2 returns a positive response to establish the CDRM1-CDRM2 session. Now the two CDRMs can handle the process of establishing and terminating LU-LU sessions between logical units in their domain and logical units in the other domain.

We have not discussed virtual routes (VRs) and explicit routes (ERs), but at least one VR and one ER must be defined between subarea 1 and subarea 2 in order for the CDRM1-CDRM2 session to be established. Our configuration (Figure 12-3) includes several paths between the two subareas. There are five channel transmission groups (TGs) and two SDLC link TGs (TG51 and TG52). Any of the paths can be used for the CDRM1-CDRM2 session as long as ERs and VRs are defined over those paths and the session initiation process selects one of the routes.

Establishing Cross Domain LU-LU Session

Now that the two CDRMs are communicating, we can establish cross domain LU-LU sessions. We will assume that logical unit "L21D79A" in domain B is to establish a session with logical unit "CICS" in domain A (Figure 12-5).

At this time, we know that all the SNA resources in subareas 2 and 21 are defined to and controlled by SSCP2. Also, the SNA resources in subareas 1 and 11 are defined to and controlled by SSCP1. SSCP2 and CDRM2 know nothing about the logical units in domain A. Also, SSCP1 and CDRM1 know nothing about logical units in domain B.

Now assume that L21D79A submits a logon request to its SSCP (SSCP2) as shown in Figure 12-5.

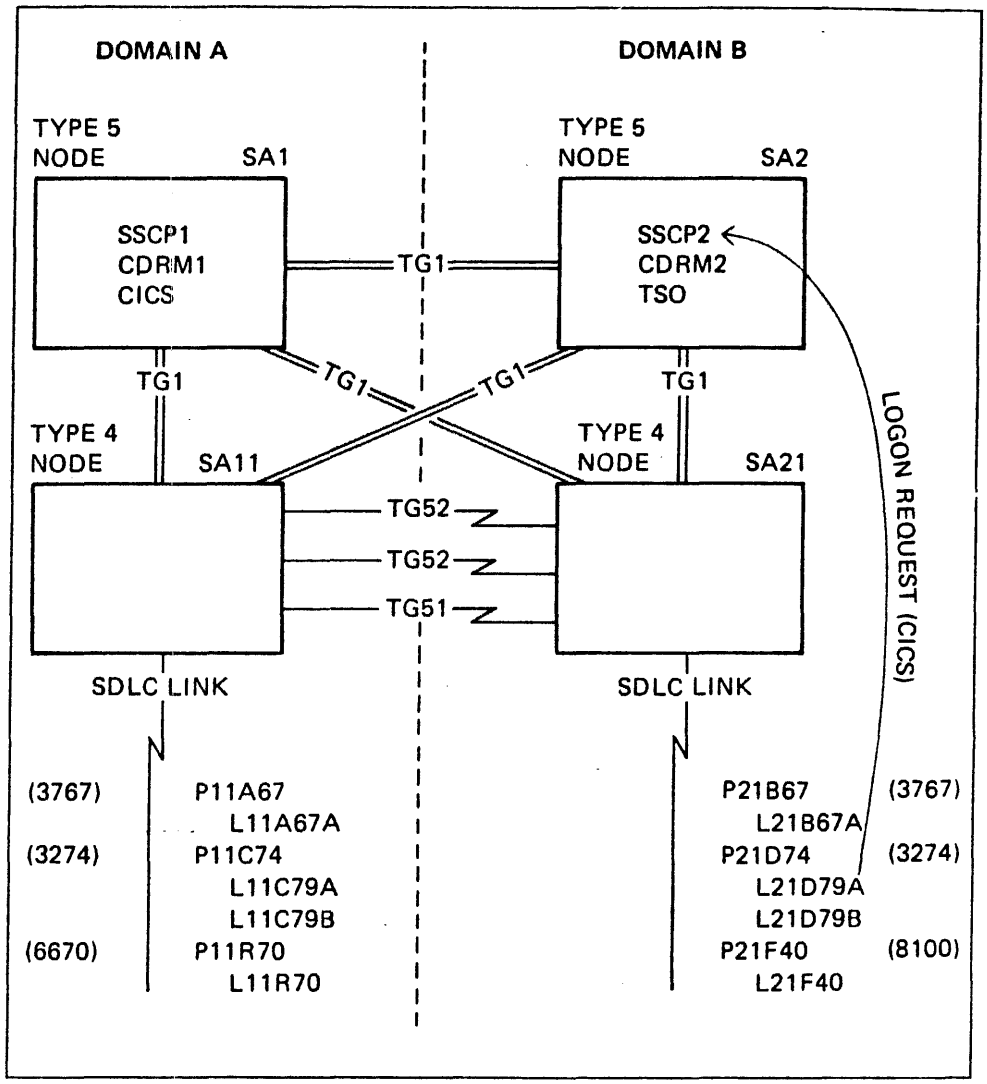

Figure 12-5. Logon Cross Domain

The logon request asks for a session to be established between L21D79A and CICS. Since CICS is not defined to SSCP2, it gives the logon request to CDRM2. CDRM2 finds out that CICS is not defined to it and the logon request is rejected. In the domain that initiates an LU-LU session, we must define what are called cross-domain resources. A cross-domain resource is a logical unit that is owned and controlled by another SSCP.

In order for L21D79A to establish a session with CICS, CICS must be defined to CDRM2 as a cross-domain resource. CDRM2 must know in which subarea CICS resides so that CDRM2 can send requests to the CDRM in that subarea.

Domain 2

CICS CDRSC CDRM1

The definition above specifies that CICS is a cross domain resource and that CDRM1 is the owner of that resource. CDRM2 knows about CDRM1 (There is a CDRM1-CDRM2 session.) and knows its subarea.

Now we will try the logon request from L21D79A again. SSCP2 receives the logon request and since SSCP2 does not own that resource, it assumes that CICS is a cross-domain resource, and gives the logon request to CDRM2. CICS is defined to CDRM2, therefore CDRM2 can initiate the session cross domain.
Figure 12-6shows the request sequence to establish a session between L21D79A in domain A and CICS in domain B. Study the figure, then read the discussion that follows.

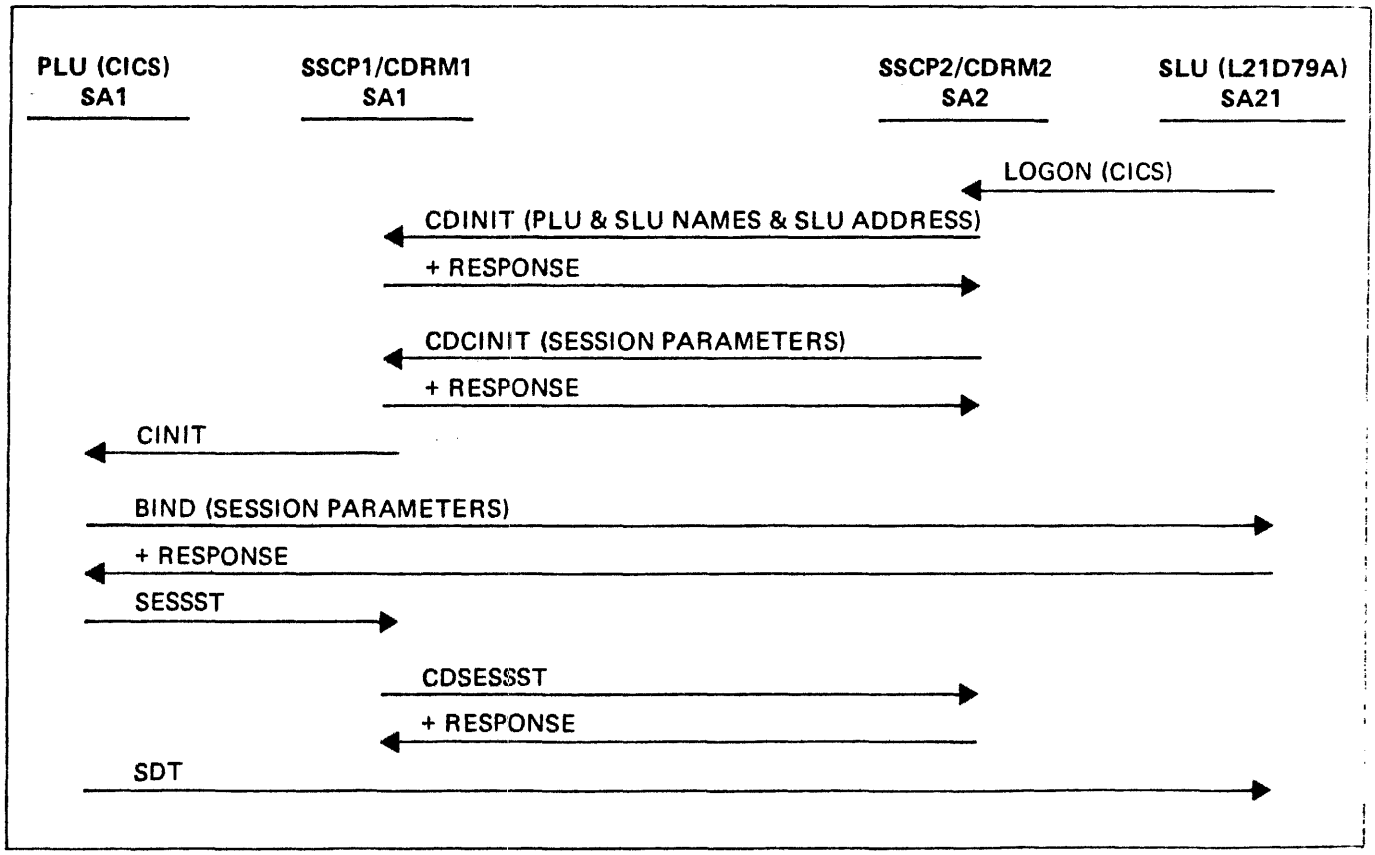

Figure 12-6. Request Sequence to Establish a Cross-Domain LU-LU Session

SSCP2 receives the logon request and gives it to CDRM2. CDRM2 determines that CICS is owned by CDRM1 in subarea 1 and sends the Cross-Domain INITiate (CDINIT) request on the CDRM1-CDRM2 session to CDRM1. The CDINIT request informs CDRM1 that CDRM2 would like a session established between L21D79A (in subarea 21) and CICS.

CDRM1 returns a positive or negative response to accept or reject the CDINIT request. If CDRM1 is allowing sessions with L21D79A, and if CICS is active and accepting logons, and if there is a route defined from subarea 1 (CICS's subarea) to subarea 21 (L21D79A's subarea) then CDRM1 returns a positive response.

CDRM2 now sends the Cross-Domain Control INITiate (CDCINIT) request to CDRM1. The CDCINIT request includes the session parameters that specify the protocols for the CICS-L21D79A session. These BIND parameters were obtained from a mode table associated with L21D79A.

CDRM1 accepts the CDCINIT request by returning a positive response and then notifies CICS about the logon request with a Control INITiate (CINIT) request. The CINIT request contains the network addresses of L21D79A and CICS, the secondary logical unit name (L21D79A), and the session parameters that were sent by CDRM2.

CICS now has enough information to decide whether to BIND the session CICS rejects logon requests from logical units that are not defined to CICS.

Typically, CICS supplies session parameters rather than using the ones from the CINIT request.

Assuming a route is defined and is available between subareas 1 and 21 and that the CICS-L21D79A session can be assigned to that route, then CICS sends a BIND request to L21D79A. The BIND request does **not** flow to CDRM2, it flows to L21D79A. If the session parameters are acceptable to L21D79A, it returns a positive response to establish the session, otherwise a negative response is returned to reject the session.

Once CICS receives the positive response to the BIND, it notifies its CDRM (CDRM1) with a SESSion STarted (SESSST) request that the CICS-L21D79A session has been established. CDRM1 then sends a Cross-Domain SESSion STarted (CDSESSST) request to CDRM2 notifying it that the CICS-L21D79A session has been established. Now both owning SSCPs know that the session is established.

Our discussion on the two domain network applies to a network that consists of many domains. Each domain must define its own cross-domain resource manager (CDRM) as well as any other CDRM with which it is to communicate. A host CDRM must be active before it will go into session with external CDRMs. Cross-domain resources must be defined in the domain that initiates a session. If peripheral logical units always logon to host logical units, then host logical units must be defined as cross-domain resources in the domains of the peripheral logical units.

Host logical units can initiate cross-domain sessions the same as peripheral logical units. If CICS in domain A initiates a session with L21D79A in domain B, L21D79A must be defined as a cross-domain resource in domain A.

Once the CICS-L21D79A session is established, the two logical units can communicate as though they were in the same domain. Neither SSCP/CDRM gets involved in the communication between the two logical units. An SSCP gets involved if one of the logical units fails or if a resource in the route used by the session fails. Also, SSCPs can assist in a session termination process.

Terminating a Cross-Domain LU-LU Session

Now we will terminate the session between CICS and L21D79A. Either of the logical units can initiate the termination process or a network operator can initiate the process. The session can be terminated with or without assistance from the SSCPS.

We will discuss one method that does not require SSCP assistance and one method that does. Figure 12-7 shows the request sequence to terminate the CICS-L21D79A session with SSCP assistance. Study the figure, then read the discussion that follows.

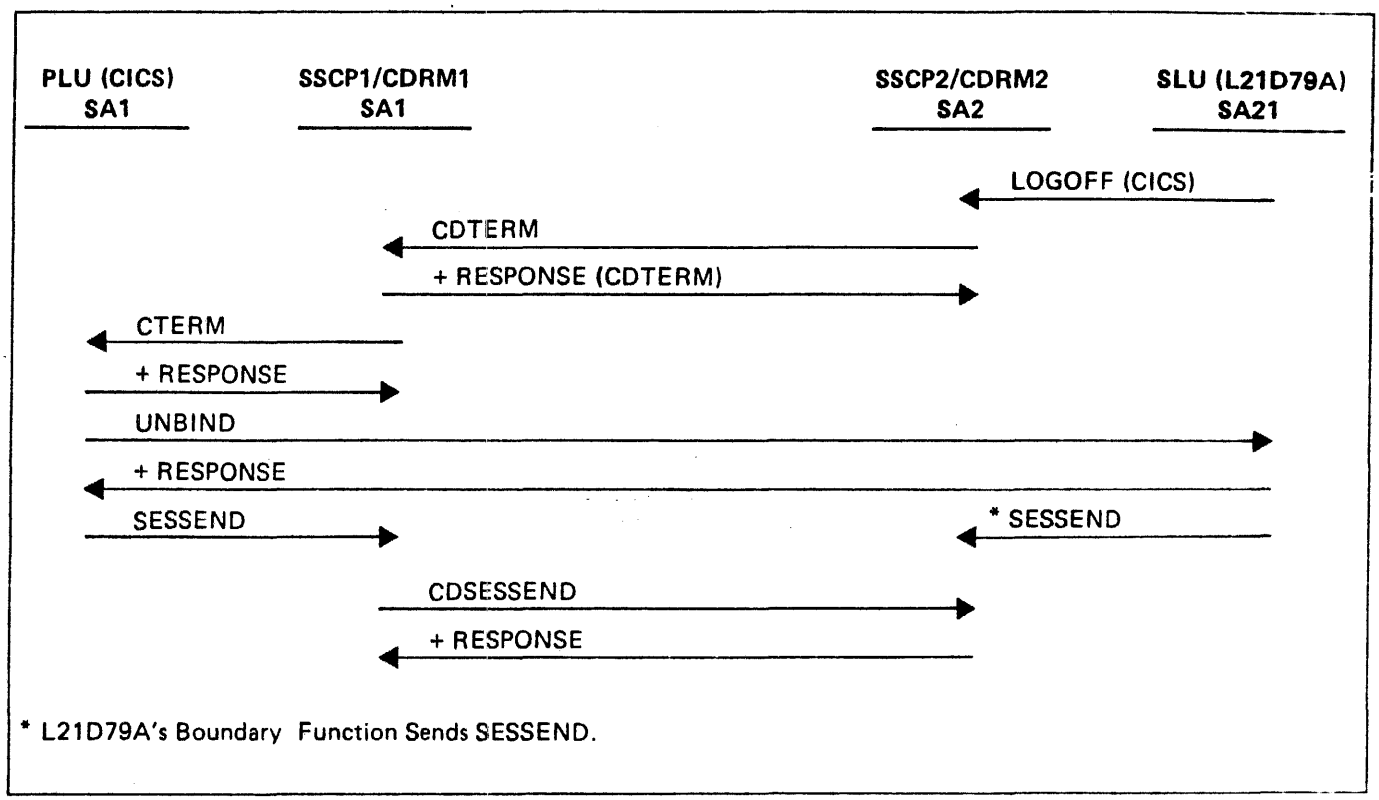

Figure 12-7. Terminating a Cross-Domain LU-LU Session With SSCP Assistance

L21D79A submits a logoff to its owner (SSCP2). SSCP2 gives the request to CDRM2 and it sends a Cross-Domain TERMinate (CDTERM) request to CDRM1. The request is given to SSCP1 and it notifies CICS with a Control TERMinate (CTERM) request. When CICS reaches a logical termination point, it sends the UNBIND request to L21D79A. CICS and L21D79A send a SESSion ENDed (SESSEND) request to their SSCP notifying it that the CICS-L21D79A session has been terminated. CDRM1 which is located in the domain of the primary logical unit sends a Cross-Domain SESSion ENDed (CDSESSEND) request to notify CDRM2 that the CICS-L21D79A session has been terminated.

Figure 12-8 shows the request sequence to terminate a cross-domain LU-LU session without SSCP assistance.

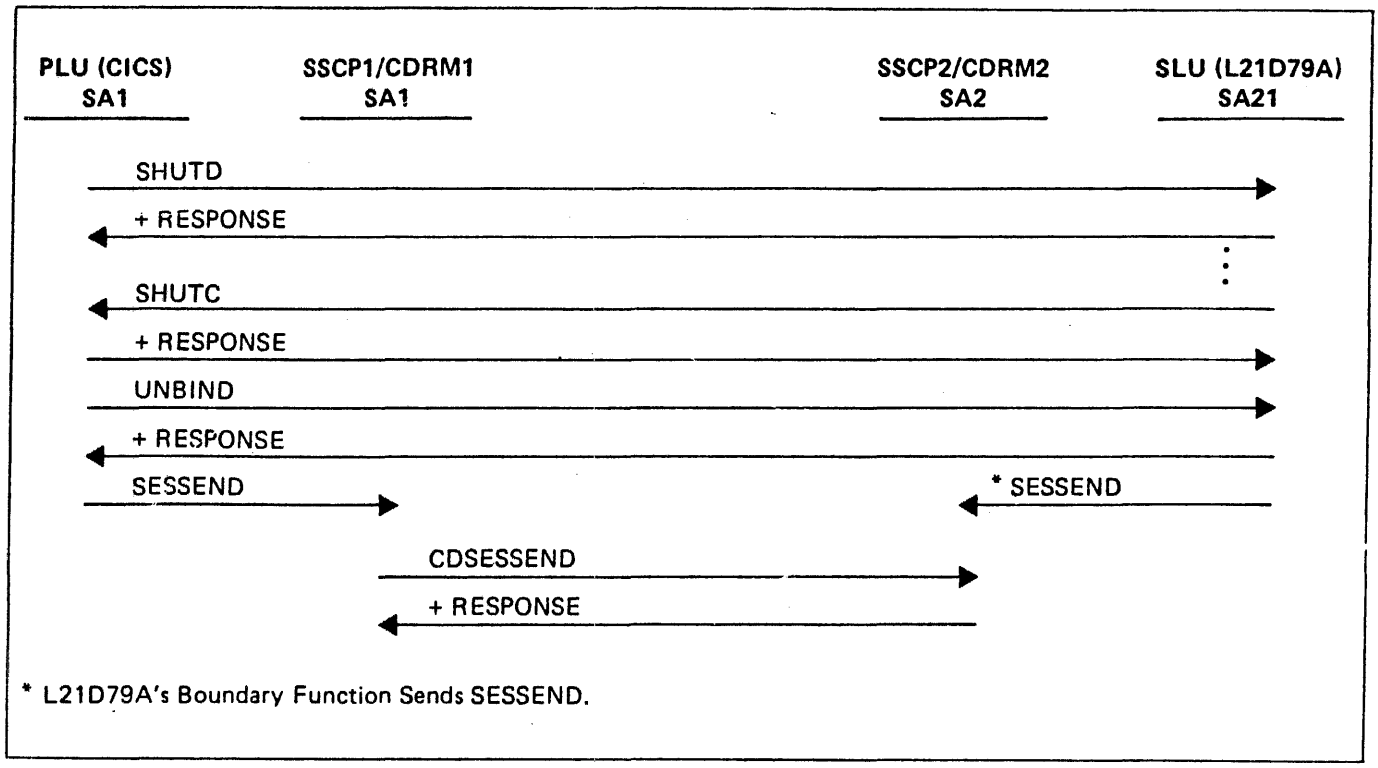

Figure 12-8. Terminating a Cross-Domain LU-LU Session Without SSCP Assistance

When CICS is ready to terminate the session with L21D79A it sends the SHUTDown (SHUTD) request. If the session is in the "between bracket" state, and if L21D79A is not sending or receiving a request chain or waiting for a response, L21D79A sends the SHUTdown Complete (SHUTC) request to CICS indicating that it is ready for the session to be terminated. CICS sends the UNBIND request to terminate the session. Then both SSCPs are notified that the CICS-L21D79A session has been terminated.

Shared Control

Up to this point we have assumed that the SNA resources in subareas 1 and 11 are defined to SSCP1 but not to SSCP2. We have also assumed that the SNA resources in subareas 2 and 21 are defined to SSCP2 but not to SSCP1. Figure 12-9 shows two domains based on these assumptions.

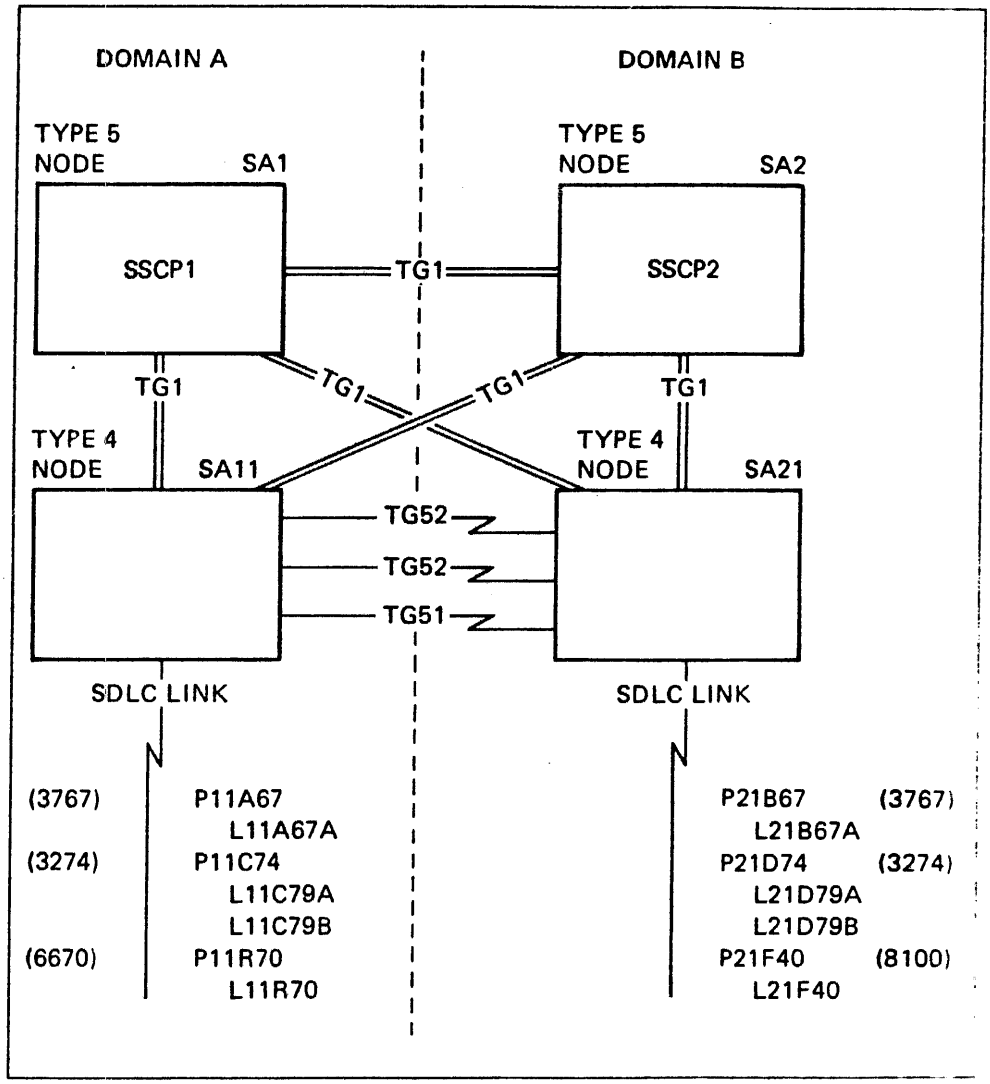

Figure 12-9. Two Domains

Now we will say that the SNA resources in subarea 11 are defined to SSCP2 and that the SNA resources in subarea 21 are defined to SSCP1.

It is now possible for SSCP1 to activate and control all SNA resources in subareas 1, 11, and 21. Also, SSCP2 can activate and control all resources in subareas 2, 11, and 21. An SSCP **cannot** become an owner of SNA resources in another host subarea node.

The following SNA resources can have multiple owners:

- Type 4 subarea nodes (ACF/NCP/VS).
- Non-switched SDLC links.
- Link stations.

All other SNA resources can have only one owner at a time.

The two domains as shown in Figure 12-9 have SSCP1 owning and controlling the resources in subareas 1 and 11 and SSCP2 owning and controlling the resources in subareas 2 and 21. That means that both SSCPs have activated all the physical units, logical units, links, and link stations shown in their domain.

Now we will have both SSCPs obtain ownership of resources in other domains. The network operator in domain 1 submits activate commands to SSCP1 to activate the type 4 subarea node in subarea 21 and the attached peripheral link. Once these resources are activated by SSCP1 they have two owners, SSCP1 and SSCP2, and they reside in both domains. The network operator in domain B submits activate commands to SSCP2 to activate the type 4 subarea node in subarea 11 and the attached peripheral link. Now these two resources have two owners and reside in both domains.

The peripheral physical units (PUs) and logical units (LUs) can have one owner at a time, therefore, they can be in only one domain at a time. If SSCP1, for example, sent an ACTivate Physical Unit (ACTPU) request to P21F40 in subarea 21, the type 4 subarea node in subarea 21 would send a negative response to SSCP1 rejecting the ACTPU request. Type 4 subarea nodes maintain a record of ownership of their peripheral PUs and LUs. Their maximum ownership count is 1.

We will assume that SSCP2 deactivates L21F40 and P21F40. Once the two resources are deactivated, they are not in either domain, that is, they have no owner. The type 4 subarea node (subarea 21) reduces the ownership count of P21F40 to 0. Now an ACTPU request from SSCP1 to P21F40 will be successful. Once P21F40 has been activated by SSCP1 it is in SSCP1's domain as shown in Figure 12-10.

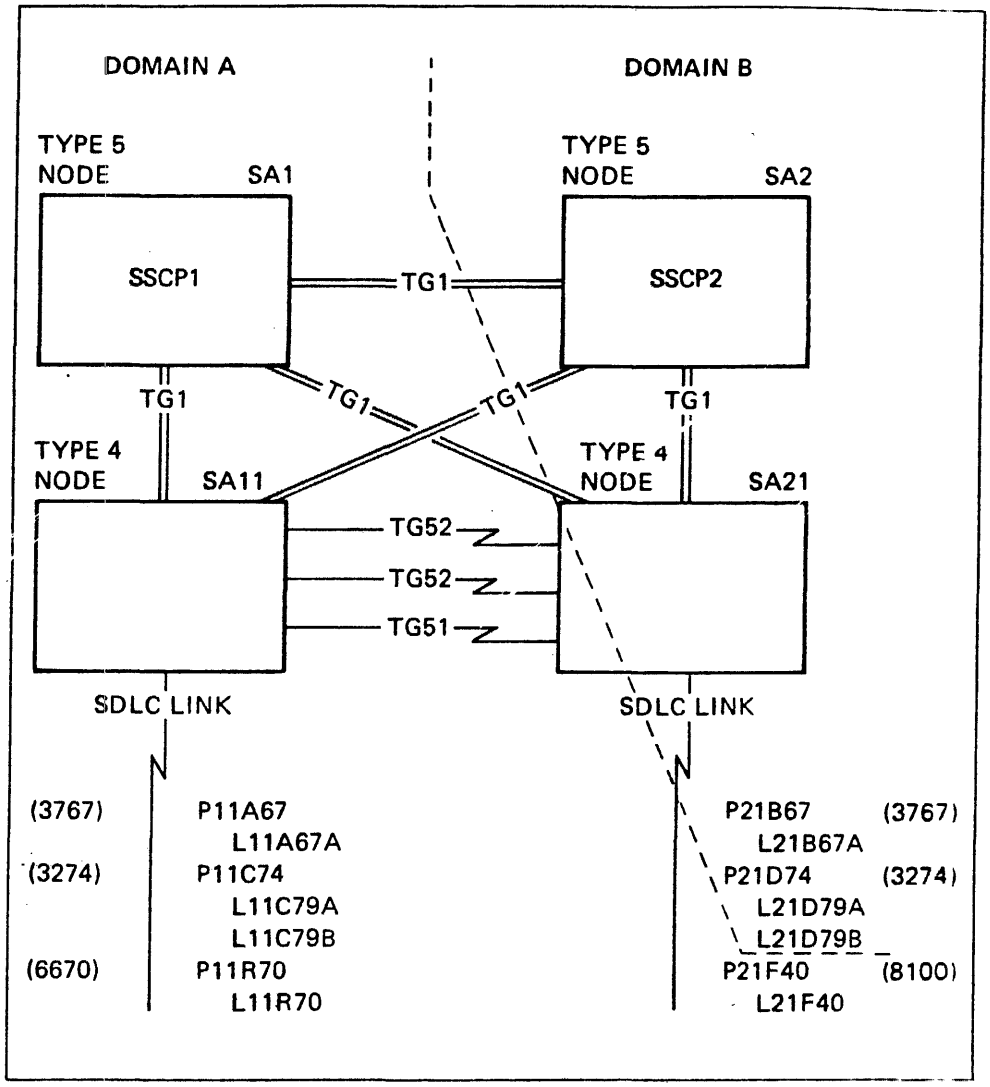

Figure 12-10. Modifying Domains

Although L21F40 is not activated it is effectively in SSCP1's domain. Once its associated PU is activated by an SSCP, only that SSCP can activate that LU. Next, we have SSCP1 send an ACTLU request to establish ownership of L21F40.

Now let's assume that L21F40 is to logon to a logical unit in subarea 2, but where does L21F40's logon go to? Does it go to SSCP1 or SSCP2? --------The logon flows to SSCP1 because SSCP1 is the new owner. Therefore this is a cross-domain logon because the primary logical unit is in subarea 2 (domain B). If L21F40 submits a logon for a logical unit in subarea 1, that logon is a same-domain logon.

We will now go back to our original domain boundaries as shown in Figure 12-9. We will assume that the peripheral logical units in subarea 21 are in cross-domain sessions with CICS in subarea 1. Also, L11C79A and L11C79B in subarea 11 are in same-domain session with CICS. L11A67A in subarea 11 is in cross-domain session with IMS in subarea 2. Now assume that the host node in subarea 2 fails (See Figure 12-11). What happens to all cross-domain sessions and the resources owned by SSCP2?

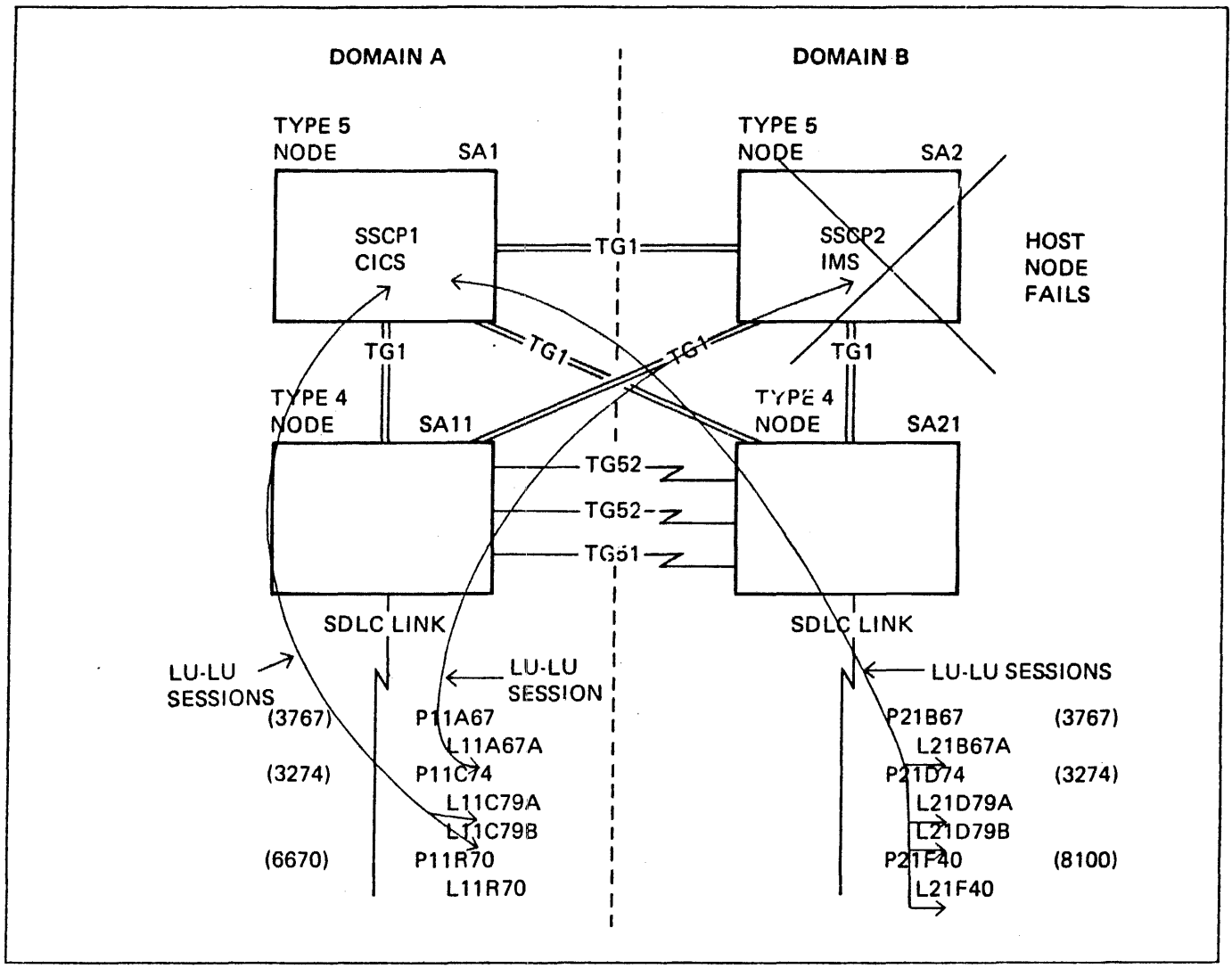

Figure 12-11. Host Node Fails

The cross-domain sessions from subarea 21 to subarea 1 can continue even though the physical units and logical units lose their owning SSCP. Their type 4 subarea node reduces their ownership count to 0. The cross-domain session from subarea 11 to subarea 2 is lost because the host node in subarea 2 failed. The same domain sessions from subarea 11 to subarea 1 continue.

The cross-domain sessions from subarea 21 to subarea 1 can continue as long as there are no failures that affect the sessions. Physical units and logical units need an owner for error recovery. If one of the cross-domain sessions failed because of an error, that peripheral logical unit would not be able to participate in another session because it has no owner. All logical units require an owning SSCP to assist in establishing a LU-LU session. Also, if one of the cross-domain sessions was terminated by either of the session partners, the peripheral logical unit would not be able to participate in another LU-LU session.

SSCP1 can take over ownership of the peripheral physical units (PUs) and logical units (LUs) in subarea 21 by first sending an ACTPU request to their type 4 subarea node to establish ownership of that node. Then SSCP1 can send ACTPU requests and ACTLU requests to the PUs and LUs respectively. When ownership of the PUs and LUs is established, some of the cross-domain sessions can continue and some can not, depending on the level of support in the SNA access method and in the peripheral resources. An example is, ACTPU ERP and ACTLU ERP, and if the physical units and their associated logical units support the erp option, then their LU-LU sessions will not be terminated when an SSCP establishes ownership.

Once SSCP1 takes over ownership of the peripheral resources in subarea 21, those logical units can establish and terminate sessions with logical units in subarea 1. Obviously, sessions cannot be established with IMS because its host node failed. However, if it is practical to start IMS in subarea 1, then all of the peripheral logical units (LUs in subareas 11 and 21) will have access to CICS and IMS.

When the host node in subarea 2 has been restored to working order, SSCP2 can reenter the network. SSCP2 can establish ownership of both type 4 nodes without any disruption. However, there can be disruption if SSCP2 is to establish ownership of the peripheral PUs and LUs in subarea 21.

For those PUs and LUs that are now owned by SSCP1. SSCP1 must deactivate those PUs and LUs which it now owns so that SSCP2 can activate them. If any of the LUs are in session, the sessions will be terminated. The practical thing to do is to wait until each logical unit terminates its LU-LU session, then SSCP1 can deactivate the LUs and associated PUs allowing SSCP2 to activate them.

If some of the PUs and LUs in subarea 21 are not owned by SSCP1, then SSCP2 can activate them. If any of the LUs are in session and if either the PU or LU does not support ACTPU ERP, then the LU-LU session will be terminated.

● *Please turn to Mini-Course 12 in your Personal Reference Guide and read the summary.*

SNA Fundamentals

Mini-Course 13

Error/Status Reporting and Processing

Printed in the United States of America.

Mini-Course 13. Error/Status Reporting and Processing

Introduction

Network errors, e.g., link errors, are reported to the owning SSCP and the SSCP sends an error-message to the network operator. The physical unit in each node reports errors concerning the resources managed by that physical unit. The error reports flow on the SSCP-PU session.

Recovery by the network is possible for some errors and exception conditions. The SSCP is responsible for recovering resources when errors are reported to it. For errors such as a failed link, the SSCP does not try to recover, it sends a message to the network operator. Once the link is repaired the network operator issues activation requests to activate the link and other resources that were affected by the failed link. For some error conditions, the SSCP attempts to recover the affected resources.

Network errors can affect existing sessions (SSCP-PU, SSCP-LU, and LU-LU). A failing component that is used by an LU-LU session, or any other session, causes that session to be terminated.

Errors that are reported on an LU-LU session do not indicate a failing component in the path between the two logical units. Logical units use responses to report to their session partner about the acceptability of received requests. If indicated in the received request, a receiving logical unit returns a response (negative or positive) to its session partner. A negative response indicates that information in the transmission header (TH), request header (RH), or in the request unit (RU) was not acceptable. For example, the request sequence number in the TH was not one higher than the sequence number of the previous received request. The RH could contain a begin bracket indicator (BBI) which indicates that a bracket is to be started. If the session is already in the in-bracket state, the BBI is an error and the receiving logical unit returns a negative response indicating that a bracket state error has been violated. An error violation in the RU could be a function management header (FMH) that is the wrong format, contains wrong information, or is the wrong FMH type.

Another example in which a negative response is returned is when a logical unit receives data that is to be sent to its printer and the printer is presently out of forms. This is an exception condition rather than an error. Once the printer has forms, requests can flow from the sending logical unit to the logical unit that is associated with the printer.

Recovery on an LU-LU session is possible for some errors and exception conditions. Usually the primary logical unit is responsible for recovery activity. The LU-LU session is terminated for those errors that are not recoverable. Once the error condition has been corrected, the LU-LU session can be reestablished.

We will now discuss network errors that are reported to the owning SSCP and errors that occur on an LU-LU session.

Reporting Errors on the SSCP Session

We have said that errors and exception conditions are reported by physical units on the SSCP-PU session. The SSCP has the responsibility for doing something about the error. The network configuration shown in Figure 13-1will be used for this discussion.

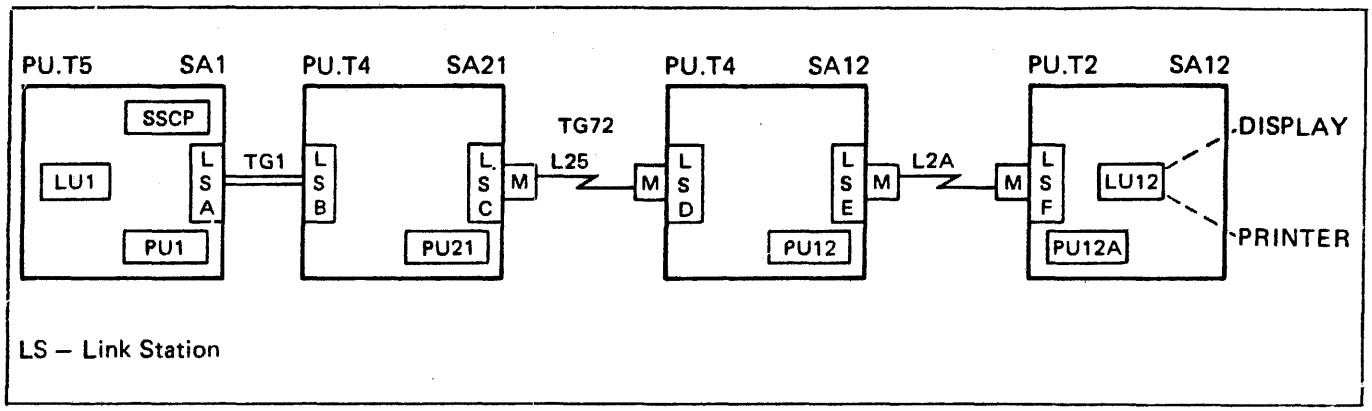

Figure 13-1. Identifying Error Types

The network is defined and is operating correctly and the following sessions exist:

- SSCP-PU21.
- SSCP-PU12.
- SSCP-PU12A.
- SSCP-LU1.
- SSCP-LU12.
- LU1-LU12.

Component failures can occur in any node. The physical unit in each node manages its links and link stations and reports errors for those resources. A node can fail or a link that connects nodes can fail. For example, PU12's node fails and PU21 in subarea 21 will detect the failure. PU21's link station "LSC" polls link station "LSD" and there is no response. Each link station is defined to timeout after a specified time duration which indicates that contact has been disrupted with the physical unit. Upon timeout, LSC's physical unit (PU21) sends an INOPerative (INOP) request to its owning SSCP to report the lost-contact.

The SSCP knows from its network status tables that contact with PU12A is also lost. The SSCP does not consider this a permanent error, it tries to reestablish contact with PU12. The SSCP sends a CONTACT request to PU21. PU21 causes link station "LSC" to send a link control contact request to link station "LSD." If the node is still not operational, LSD will not respond and LSC will timeout. LSC will continue to send the contact request to LSD until it responds or until the network operator issues an deactivate command that causes PU21 to stop the contact procedure.

The SSCP's network resource status tables maintain a desired status for each resource. The desired status for links, link stations, PUs and LUs is active when

those resources are active. For our situation where PU12's node failed, the desired status for link station "LSD," PU12, link L2A, link station "LSF," PU12A, and LU12 is active. Therefore the SSCP tries to reactivate all of those resources.

We will now assume that all network resources are active, there is a session between LU1 and LU12, and that requests and responses are being sent on the session. We will also assume that link errors occur on link L2A that connects PU12 and PU12A. Keep in mind that this type of error is reported to the SSCP not to a logical unit. The request that is to be transmitted across link L2A will either make it error-free or it will not make it at all.

Link station "LSE" transmits a request (path information unit) over link "L2A" and link station "LSF" detects an error. LSF reports the error to LSD and LSD retransmits the request. The network definer supplies definitions that specify the number of transmission retries for each link station. If the request cannot be transmitted error-free to LSE in the specified number of retries, LSD assumes a permanent failure and PU12 sends an INOP request to its SSCP indicating a permanent failure of link "L2A." The SSCP reports the failure to the network operator and does not attempt recovery because it is a permanent error.

The SSCP also notifies all logical units that have LU-LU sessions across the failed link. In our example, LU1 must UNBIND the session to cleanup the status maintained by the SSCP for this session. Once the link is operational, the network operator can issue the appropriate activation commands to activate the affected resources. The SSCP sends a CONTACT request to PU12 to contact link station "LSF." Then an ACTPU request is sent to PU12A and an ACTLU request is sent to LU12. The ACTPU and ACTLU requests will reset any session status in the peripheral node that was left over from the previous session. Next, the LU1-LU12 session can be reestablished.

The only error reporting to LU1 was by the owning SSCP indicating that the session route was inoperative. What happened to the request that could not be transmitted across link "L2A"? That request was lost and LU1 is responsible for resending it.

Reporting Errors on the LU-LU Session

A request sent by a logical unit and received by its session partner is unchanged. Errors that occur in that part of the network between the two logical units do not change requests flowing between the logical units. Requests arrive at their destination unchanged or they don't arrive at all. So if a received request is unacceptable to a logical unit it is because the sending logical unit put something in the request that is unacceptable.

The contents of the transmission header (TH), request header (RH), and/or the request unit (RU) may not be acceptable to the receiving logical unit. The receiving logical unit sends a negative response containing sense information to notify its session partner that the request is unacceptable and indicates why it is unacceptable. The logical unit that sent the request is responsible for sending an error-free request so it is responsible for correcting the error.

Certain errors occur because of logic errors in the logical unit. A programmed logical unit (LU) could send an invalid request to its session partner because of a coding error in the LU. For that type of error, the only option is to UNBIND the session. Someone must correct the program code in the logical unit.

Other errors do not bring the session down. For example, a programmed logical unit is sending data to a peripheral logical unit to be printed. The printer is out of forms and the peripheral logical unit cannot accept the data, therefore the peripheral logical unit returns a negative response to the sending logical unit indicating that the printer is out of forms. This is not a permanent error, once the operator puts forms in the printer data can be sent to the peripheral logical unit for printing. If the peripheral logical unit has the capability, it will send a logical unit status (LUSTAT) request to its session partner indicating that data can now be sent to the printer because forms have been put in the printer. If the LUSTAT request is not used then data requests can be sent after a time delay.

A negative response includes four bytes of sense data in the response unit (RU). The first byte specifies an error category which is defined by SNA.

```
X'00'     User sense data only.
X'08'     Request reject.
X'10'     Request error.
X'20'     State error.
X'40'     RH usage error.
X'80'     Path error.
```

The second byte, a modifier value, is defined by SNA and it specifies why the associated request was unacceptable. The third and fourth bytes are user-defined and are useful for reporting errors that aren't defined by SNA. When user defined sense information is included, the category is X'00' and the modifier is X'00'.

Error on a Multiple Element Request Chain

We have said that for some negative responses the only option is to UNBIND the session and fix the logical unit. Other negative responses report a type of exception condition that is temporary and can be remedied without taking the session down. We discussed the negative response when the printer was out of forms. Now we will look at recovery when an exception condition occurs while a multiple element request chain is being sent.

Figure 13-2 shows our network configuration and we can assume that LU1 is sending a ten element request chain to LU12.

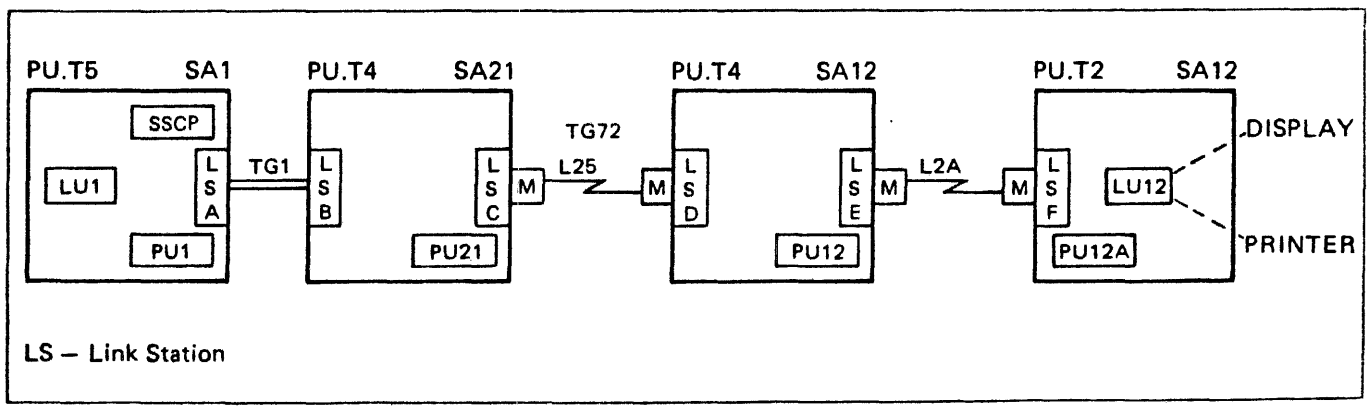

Figure 13-2. Sending a Multiple Element Request Chain

LU12 finds the third request unacceptable and returns a negative response to LU1 indicating why the request is unacceptable. By the time that the response arrives, LU1 has sent requests four and five. The receiving logical unit purges all chain elements received after detecting the error, including the error request.

Assuming that the session can recover from this error, the sending logical unit (LU1) has options when it receives the negative response. It can send the remainder of the chain elements and then perform recovery activities or it can terminate the chain without sending the remaining chain elements and then perform recovery activities.

Recovery can not be accomplished while the session is in chain state. A request must be sent to LU12 that contains the end chain indicator (ECI) to terminate the chain. LU1 can terminate the chain by sending an FMD request whose request header contains ECI or it can send a CANCEL request whose request header contains ECI. Once the chain is terminated recovery can be performed.

The request chain is the recovery entity, not just the request element that was in error. LU1 must send the ten request elements. Assume that LU12 was sending each request to a printer as it was received, which means that the first two requests of the chain were printed. What does LU12 do with the first two requests that are resent by LU1? That depends on the capability of LU12. If LU12 is a programmed LU, it could have the capability to purge the first two requests and send the remaining requests to the printer. If LU12 is implemented by hardware, it may not have the capability of purging the first two requests. Therefore the sending logical unit (LU1) must include appropriate controls in the data stream to position the printer forms for printing the entire request chain.

Synchronizing Two Logical Units

We know that data requests on the normal-flow are sequence numbered. Each logical unit maintains a send and a receive sequence count. The primary logical unit's send sequence count must match the secondary logical unit's receive sequence count and vice versa. A receiving logical unit returns a negative response if it receives a request whose sequence number doesn't match its receive sequence number.

If sequence numbers get out of sync, they must be resynchronized before data traffic can resume on the session. One way is to reset all sequence counts to zero and then resume session data traffic. This can be accomplished by the primary logical unit sending the CLEAR request to the secondary logical unit, in which case the primary logical unit is responsible for making sure that no requests are lost. The CLEAR request puts the session in the data-traffic-reset state which does not allow traffic on the normal-flow. Therefore the primary logical unit must send the start data traffic (SDT) request to put the session in data-traffic-active state.

Some programmed logical units maintain a log of sequence numbers for maintaining session traffic integrity. For example, two IMS logical units maintain send and receive sequence counts that can be used for synchronizing the session when there is a failure. Also, these sequence counts can be used for checking synchronization when the LU-LU session is established each time.

The configuration in Figure 13-3 shows LU1 in one host subarea node and LU2 in another host subarea node. The two logical units are in session with each other. The send and received counts shown above each logical unit indicate that LU1 has sent 120 requests and LU2 has received 120 requests. Also, LU2 has sent 20 requests to LU1 and LU1 has received 20 requests.

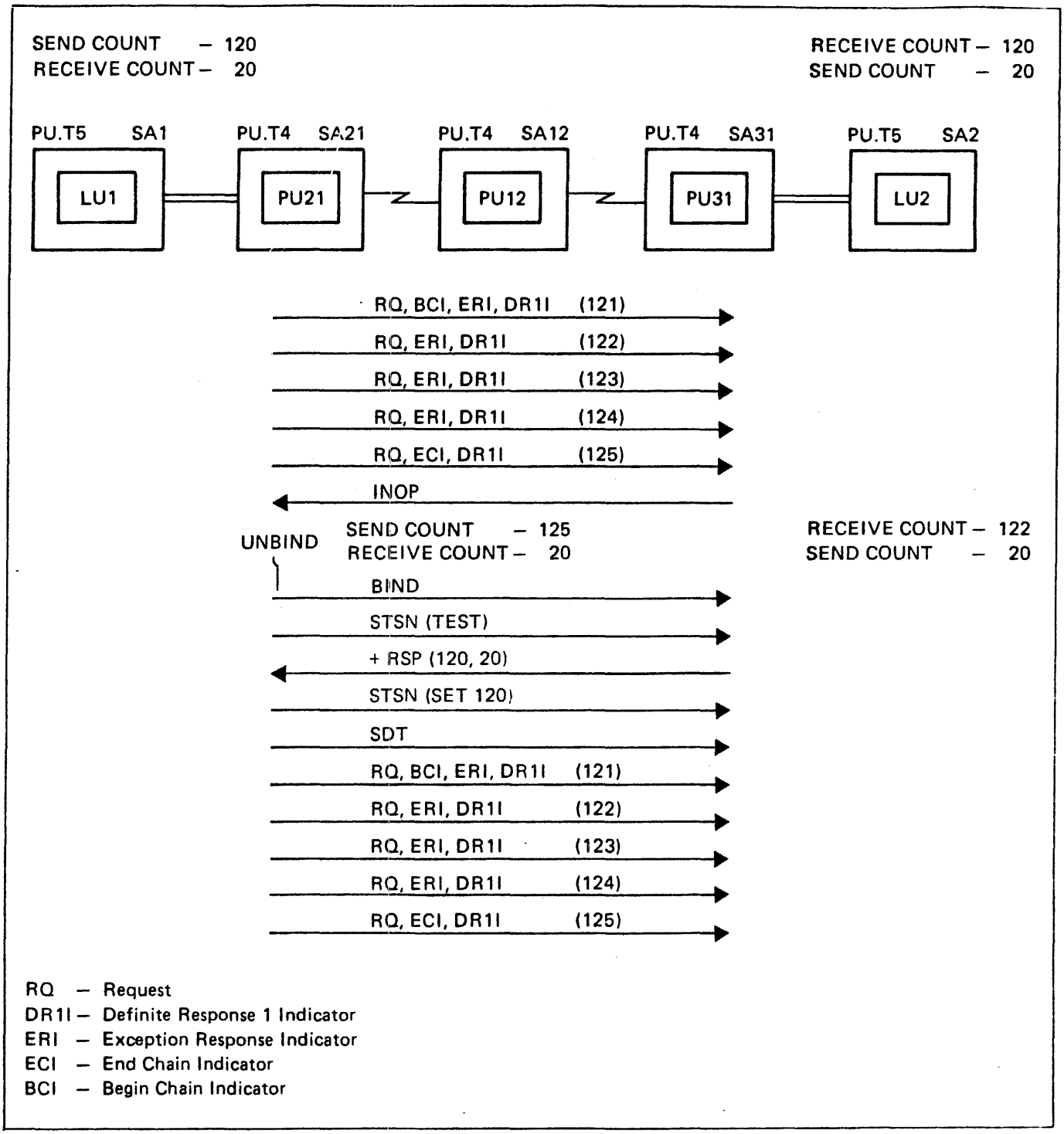

Figure 13-3. Using STSN Request

LU1 is sending a five element request chain to LU2 as shown below LU1. The requests are queueing up in the PU12 node because there are link errors on the link connecting PU12 and PU31. PU12 is attempting to transmit the request with sequence number 121 across the link. After maximum retries PU12 sends an INOP request to the SSCP in subarea 1 indicating a permanent failure of the link. PU31 sends an INOP request to the SSCP in subarea 2 indicating a link failure.

The LU1-LU2 session is terminated because the route is no longer available. LU1 now has a send count of 125 and LU2 still has a receive count of 120. LU1 did not get a positive acknowledgement for the last request chain sent.

After the link is repaired and the network resources are activated, the LU1-LU2 session is established. Assuming that LU1 is the primary logical unit, it can send a set and test sequence number (STSN) request to LU2 asking for its send and receive sequence count. LU2 returns its send and receive count in the positive response. Now LU1 knows which request LU2 last processed and LU1 knows that the five element request chain must be sent again. LU1 must set the sequence counts so that they match. This is done by the STSN request. The STSN request specifies that the sequence numbers be set to 120 and 20. Now LU1's send sequence count is 120 and its receive count is 20. LU2's receive sequence count is 120 and its send sequence count is 20. LU1's send sequence count and LU2's receive sequence count could have been set to 125 if LU1 had been coded to do that.

Once the sequence numbers are set to match, LU1 sends the SDT request to LU2 to change the data-traffic state of the session from data-traffic-reset state to data-traffic-active state. Now the five element request chain can be sent to LU2.

You should understand from this discussion that the primary logical unit can reset all sequence number counts to 0 by sending the CLEAR request to the secondary logical unit. For some secondary logical units this is the only possible way to resynchronize. If send and receive sequence number counts are to be tested or changed, the session must be in data-traffic-reset state which does not allow function management data (FMD) and data flow control (DFC) requests to flow. The primary logical unit changes the session from data-traffic-active state to data-traffic-reset state by sending the CLEAR request to the secondary logical unit. Then the primary logical unit uses the STSN request to test and/or change send and receive sequence number counts.

Reporting Status Information and Signaling The Session Partner

Logical units can communicate with each other on the normal-flow and on the expedited-flow. Logical units can always send on the expedited-flow but they cannot always send on the normal-flow. A logical unit that is in a receiver state or is quiesced, can only send responses on the normal-flow. Any request must be on the expedited-flow.

Generally we think of the normal-flow being used for transmission of end user data. The two session partners can use certain data flow control requests (e.g., BID, RTR, and CANCEL) to manage end user traffic. A logical unit can also report status information to its session partner on the normal-flow. This can be done with a data request or a LUSTAT request can be used.

In general, LUSTAT is used to report about exception conditions, failures, and error recovery conditions for a local device of a logical unit. For example, there is a session between a host logical unit and a 3274 logical unit and that logical unit is associated with a 3279 display. The terminal operator turns power off to the 3279 display and the associated logical unit sends a LUSTAT request to the host logical unit. The LUSTAT request contains a code that indicates power off. If the operator turns power on to the display, the associated logical unit sends another LUSTAT request to the host logical unit and the code in the request indicates that the component is now available.

The request unit (RU) format of the LUSTAT request allows four bytes of status information. The RU can contain either end user information or logical unit status information. If the first two bytes of the RU are 0, the low order two bytes carry end user information and may be set to any value.

SNA has defined several logical unit status information codes. The code for the powered off status is X'0831xxxx'. The value "0831" indicates the powered off condition, and the last two bytes (represented by "xxxx") identify the resource that is powered off. (In this case it specifies that the logical unit component, display, is powered off, and includes the address of the component (3279 display)).

A logical unit that is in a receiver state or is quiesced, cannot send requests on the normal-flow, therefore it cannot send the LUSTAT request. Once the logical unit is released from the quiesced state or changed from receiver to sender state, it can send the LUSTAT request.

A logical unit that is in the receiver state and would like to be in the sender state can send the SIG request to its session partner requesting the session partner to send a change direction indicator (CDI).

SIG is an expedited request that carries a four-byte value in its request unit (RU). The first two bytes are the signal code and the last two bytes are user-defined information. SNA has defined three signal codes:

1. X'00010000' Request to send.

2. X'0002uuuu' Assistance requested. User information is required in the bytes represented by "uuuu" to indicate what assistance is desired.

3. X'0003uuuu' Intervention required. User information is required in the bytes represented by "uuuu" to indicate what assistance is required.

A logical unit that is presently in the receiver state can send the SIG request containing X'00010000' to its session partner which requests the session partner to send the change direction indicator (CDI). Assume that a host logical unit is in session with a 3274 logical unit and the logical unit is associated with a 3279 display. The 3274 logical unit is in receiver state and the operator wants to send information to the host logical unit. The operator depresses the ATTN key and the associated logical unit sends A SIG request containing the code X'00010000' to the host logical unit requesting the host logical unit to send a change direction indicator (CDI).

When we talk about the user supplying code information for the SIG request or LUSTAT request, we are talking about the implementer of the logical unit. For example, IBM builds the 3274 Information Display System, which means that the IBM developers decided what SNA protocols and user defined codes to support.

The person or persons who code a program that implements a logical unit decides what SNA protocols and user codes will be supported. For example, CICS can go into to session with a wide range of terminals and application subsystems. It must be able to support the protocols and user codes that those terminals and subsystems support.

● *Please turn to Mini-Course 13 in your Personal Reference Guide and read the summary.*