Simulated FDX Interoperability Recommendation

December 11, 1995

William R. Greene

IBM Corporation
Networking Software Division
P.O. Box 12195
Research Triangle Park, North Carolina 27709, USA

☎ (919) 254-4770
Internet: wrg@vnet.ibm.com

Contents

Introduction 1

Definition of Terms 3

Requirements 5
Transparency 5
Performance 5
Use true full duplex capability where available 5
Safety 5

Non-requirements 7

Full Duplex Simulation Protocol 9
Protocol for CMALLC 9
Protocol for CMACCP/CMACCI 13
Protocol for CMDEAL 14
Protocol for CMFLUS 14
Protocol for CMRCV 14
Other Issues 14

Flow Diagram for CMALLC 17

Discussion 19

Alternative Solutions 21
Manage a single pipe 21
Do twin piping between application programs 21

Introduction

The CPI-C 2.0 specification describes how full duplex CPI-C conversations are established and used, assuming the underlying support of a full duplex CRM.

Although full duplex support is always present in OSI TP, such support is relatively new to the LU 6.2 CRM. Full duplex LU 6.2 implementations are not widely available today and will not be universally available in the near future.

We would like to make CPI-C full duplex conversation support available now, without waiting for LU 6.2 full duplex support.

A standard method of *simulating* full duplex conversations at the CPI-C level would allow full duplex CPI-C programs written by different authors or organizations to interoperate. If the simulation is done below the CPI-C API level, it would facilitate the migration of CPI-C programs from half duplex to full duplex LUs by enabling full duplex CPI-C application programs to be developed today on half duplex LUs and used tomorrow between full duplex LUs or between half and full duplex LUs.

This paper proposes a method of simulating full duplex CPI-C conversations by using twin opposed half duplex LU 6.2 conversations ("pipes"). A SNA service TP is used to set up the two pipes, one to be used for operations on the CPI-C send queue, and the other to be used for operations on the CPI-C receive queue. These conversations never change direction.

Definition of Terms

True full duplex Full duplex conversation support at the LU 6.2 level.

Simulated full duplex Full duplex conversation support at the CPI-C level, implemented by means of half duplex conversations at the LU 6.2 level.

FDX Descriptor A record that is exchanged between remote partners as part of the process of establishing a simulated full duplex CPI-C conversation.

Pipe A half duplex LU 6.2 conversation.

Application TP An end-user program that processes transactions, accessing the network via LU type 6.2

SNA service TP An IBM-supplied architecture-defined program that executes within an LU type 6.2 and that typically provides utility services to application TPs. For example, the CNOS (Change Number of Sessions) function is realized as a SNA service TP.

CRM Conversation Resource Manager. For CPI-C, either OSI TP or LU 6.2.

Requirements

Transparency

Full duplex simulation must support the full duplex API and semantics of CPI-C 2.0 without extensions or deviations. Thus full duplex CPI-C application programs can be developed now that will execute on future systems that support either true or simulated full duplex, with no need for recompilation or relinking.

Performance

Half duplex LU 6.2 conversation support must be used in such a way that acceptable performance is possible.

Use true full duplex capability where available

Simulated full duplex is not a substitute for true full duplex, but is a compensation used when true full duplex support is not available to both partners. When both implementations support true full duplex, it will be used rather than simulated full duplex.

Safety

In setting up a simulated full duplex conversation, there must be no unexpected flows to a remote partner, should that partner not support a compatible method of full duplex simulation.

Non-requirements

There is no requirement for a full duplex simulation to support expedited data.

The proposed design does not address the use of sync_level SYNC_POINT in a simulated full duplex CPI-C conversation.

Ease of implementation is always desirable but is not a requirement.

Full Duplex Simulation Protocol

In the following, we assume a working implementation of CPI-C that supports half duplex conversations over an LU 6.2 CRM. Therefore, we do not elaborate on return_code and other APPC-to-CPI-C mappings that do not change when moving from half to full duplex CPI-C support.

Protocol for CMALLC

Any CPI-C 2.0 application program that initiates a full duplex conversation must do so by calling CMINIT and CMSSRM(CM_FULL_DUPLEX) before calling CMALLC.

On the initiating system, the implementation of CMALLC must take the following actions:[1]

1. If true full duplex is not supported on the local LU, proceed to step 2. Otherwise, issue a full duplex APPC Allocate verb. If the return code from the Allocate verb indicates that the partner LU does not support true full duplex, proceed to step 2. Otherwise, execute no further steps.
2. Issue a half duplex APPC Allocate verb, specifying as the partner TP the SNA service TP X'28F0F0F3', and specifying a sync_level of confirm.[2] Specify the mode_name, partner_LU_name, and other parameters[3] as if the Attach were to flow directly to the partner TP. The APPC conversation can be either mapped or basic.

   If the Allocate verb fails, return an appropriate CPI-C return_code to CMALLC, and do not execute any more steps. If it completes successfully, store the returned APPC conversation ID in the CPI-C conversation control block for later use as this side's send pipe.

---

[1] All steps are numbered in chronological order, regardless of which system executes them. The flows are graphically depicted in “Flow Diagram for CMALLC” on page 17.

[2] CPI-C full duplex conversations always have a sync_level of CM_NONE. The fact that the CPI-C implementation uses LU 6.2 conversations having a sync_level of confirm is hidden from the CPI-C user.

[3] Security information flows on the Attach in the usual way, except that the destination TP is the SNA service TP rather than the application TP. The remote system saves this information for use in automatically starting the application TP at step 9. If the application TP is operator-started, the stored security information is used to determine whether the TP is subsequently allowed to successfully execute a CMACCP or CMACCI.

3. Construct a *FDX Descriptor* having the following format and content:

Field	Size in bytes	Contents
is_returning[4]	4	X'00000000'
correlator_length	1	length of ***correlator***
correlator	0-8	a value that provides the CPI-C implementation access to the conversation control block used internally by CMALLC
accepting_TP_name_length	1	length of ***accepting_TP_name***
accepting_TP_name	0-64	partner application TP name
	6-78 total	

Note: The LL (and, for mapped conversations, GDS_ID) fields are not shown here or discussed further in the text that follows.

4. Issue an APPC Send_Data verb, specifying the FDX Descriptor as the buffer to be sent.

   If the Send_Data verb fails, return an appropriate CPI-C return_code to CMALLC, and do not execute any more steps.

5. Issue an APPC Confirm verb. This flushes the FDX Descriptor (and possibly the Attach) and blocks until one of the following occurs:

   - The remote SNA service TP responds with a Confirmed at step 19. In this case, the remote system will have executed steps 6-12 and step 18.
   - The remote LU rejects the Attach from step 2 with a sense code of X'084C0000', indicating that the SNA service TP for CPI-C full duplex simulation is not present on the remote system and that therefore the remote system does not support simulation of CPI-C full duplex conversations. In this case, the Confirm verb returns a primary return code of AP_ALLOCATION_ERROR and a secondary return code of AP_TRANS_PGM_NOT_AVAIL_NO_RETRY. Deallocate[5] the LU 6.2 conversation allocated at step 2, return a return_code of CM_SEND_RCV_MODE_NOT_SUPPORTED to CMALLC, and do not execute any more steps.
   - The remote SNA service TP issues an APPC Deallocate(ABEND) or Deallocate(ABEND_SVC),[6] depending on whether the conversation allocated at step 2 is basic or mapped.[7] Return a return_code of

---

[4] The ***is_returning*** field is at offset zero from the FDX Descriptor.

[5] NS/DOS does not deallocate the LU 6.2 conversation here, but rather unbinds the session.

[6] This could occur if, say, the remote system found at step 10 that system resources were inadequate for the second pipe to be allocated.

[7] What if the remote SNA service TP issues an APPC Deallocate(ABEND_TIMER)?

CM_ALLOCATE_FAILURE_NO_RETRY to CMALLC and do not execute any more steps.

The remote SNA service TP must take the following actions:

6. Issue an APPC Receive_Allocate verb in order to receive the Attach sent at step 2.

   If the Receive_Allocate verb fails, do not execute any more steps. Otherwise, store the returned APPC conversation ID for later use as this side's receive pipe.

7. Issue an APPC Receive_and_Wait verb in order to receive the FDX Descriptor sent at step 4, providing a large enough buffer and requesting a sufficient number of bytes to receive the largest possible FDX Descriptor (78 bytes, not counting the LL or GDS_ID fields in the header).

   If the Receive_and_Wait verb fails, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic, and do not execute any more steps.

8. Decode the received FDX Descriptor.

   If what is received does not match the description of a FDX Descriptor given at step 3, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic, and do not execute any more steps. Otherwise, after determining that the ***is_returning*** field is X'00000000', continue with steps 9 through 20.

9. If the ***accepting_TP_name*** field corresponds to a TP that is to be automatically started, it is started at this time.[8] If the TP is to be automatically started, the security information that flowed on the Attach received at step 6 is used to determine whether that TP's privilege level would allow it to accept the Attach.

   If an attempt is made to automatically start the TP, but the specified TP name is not valid, or if the TP cannot be started for some other reason,[9] deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic, and do not execute any more steps.

10. Issue a half duplex APPC Allocate verb, specifying as its partner TP the SNA service TP X'28F0F0F3', and specifying the same sync_level, conversation_type (mapped or basic), mode_name, and partner_LU_name that were returned by the Receive_Allocate call at step 6.

    If the Allocate verb fails, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic, and do not execute any more steps. Otherwise, store the returned APPC conversation ID for later use as the send pipe for the CPI-C application program (possibly started at step 9) that should eventually execute CMACCP or CMACCI.

11. Of the fields in the received FDX Descriptor, change only the ***is_returning*** field, to X'00000001', then issue an APPC Send_Data verb, specifying the FDX Descriptor as the buffer to be sent, but specifying the number of bytes to send so that only the ***is_returning***, ***correlator_length***, and ***correlator***, but not the ***accepting_TP_name*** and ***accepting_TP_name_length***, will be sent.

---

[8] But see "Other Issues" on page 14.

[9] Deallocate(AP_TRANS_PGM_NOT_AVAIL_RETRY) might be more appropriate in this case.

If the Send_Data verb fails, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; do likewise for the LU 6.2 conversation allocated at step 10; and do not execute any more steps.

Note that the ***correlator*** is not used here but is merely sent back to the originating side unchanged. It is used at step 15 to associate the second pipe with the first.

12. Issue an APPC Confirm verb. This flushes the FDX Descriptor and blocks until step 17.

    If the Confirm verb fails, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; do likewise for the LU 6.2 conversation allocated at step 10; and do not execute any more steps.

The local SNA service TP on the initiating system must take the following actions:

13. Issue an APPC Receive_Allocate verb (*cf.* step 6) in order to receive the Attach sent at 10.

    If the Receive_Allocate verb fails, do not execute any more steps.

14. Issue an APPC Receive_and_Wait verb (*cf.* step 7), providing a large enough buffer and requesting a sufficient number of bytes to receive the largest possible FDX Descriptor (78 bytes, not counting the LL or GDS_ID fields in the header).[10]

    If the Receive_and_Wait verb fails, deallocate the LU 6.2 conversation accepted at step 13 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic, and do not execute any more steps.

15. Decode the received FDX Descriptor (*cf.* step 8).

    If what is received does not match the description of a FDX Descriptor given at step 3 or if the ***correlator*** field does not correspond to a valid CPI-C conversation, deallocate the LU 6.2 conversation accepted at step 13 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; and do not execute any more steps. Otherwise, on finding that its ***is_returning*** field is X'00000001', use the ***correlator*** field to access the conversation control block used in the original CMALLC call. Store the APPC conversation ID returned at step 13 in the conversation control block for later use as this side's receive pipe, then continue with steps 16 and 17.

16. Issue an APPC Receive_and_Wait verb, specifying that zero bytes of data are to be received, in order to receive the Confirm sent at step 12. This call is only necessary because the LU 6.2 specification [TPRM] does not provide for both data and status to be returned on a single Receive_and_Wait (otherwise, the Confirm could be received at step 14).

    If the Receive_and_Wait verb fails, deallocate the LU 6.2 conversation accepted at step 13 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; do likewise for the LU 6.2 conversation allocated at step 2; and do not execute any more steps.

17. Issue an APPC Confirmed verb. This unblocks the remote SNA service TP, which is awaiting the completion of its outstanding Confirm verb, issued at step 12.

---

[10] This is because the SNA service TP cannot know before receiving the FDX Descriptor whether it has just received an initial Attach (as at step 6) or the matching return Attach (in which case the accepting TP information is omitted as at step 11).

If the Confirmed verb fails, deallocate the LU 6.2 conversation accepted at step 13 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; do likewise for the LU 6.2 conversation allocated at step 2; and do not execute any more steps.

Now the remote SNA service TP, having completed its Confirm from step 12, must take the following actions:

18. Issue an APPC Receive_and_Wait verb (*cf.* step 16), specifying that zero bytes of data are to be received, in order to receive the Confirm sent at step 5. This call is only necessary because the LU 6.2 specification [TPRM] does not provide for both data and status to be returned on a single Receive_and_Wait (otherwise, the Confirm could be received at step 7).

    If the Receive_and_Wait verb fails, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; do likewise for the LU 6.2 conversation allocated at step 10; and do not execute any more steps.

19. Issue an APPC Confirmed verb (*cf.* step 17). This unblocks the application program on the initiating system, which is awaiting the completion of its outstanding Confirm verb, issued at step 5.

    If the Confirmed verb fails, deallocate the LU 6.2 conversation accepted at step 6 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; deallocate the LU 6.2 conversation allocated at step 10 by issuing an APPC Deallocate(ABEND) if that conversation is mapped, or Deallocate(ABEND_SVC) if basic; and do not execute any more steps.

20. Pass the conversation control block for the simulated full duplex conversation to the CPI-C implementation, which will store it until the partner TP on this side (possibly started at step 9) executes a CMACCP or CMACCI call.

The CPI-C implementation on the initiating side uses the APPC conversation IDs obtained at steps 2 and 13 for operations on its Send and Receive queues, respectively. The remote CPI-C implementation uses the APPC conversation IDs obtained at steps 10 and 6 for operations on its Send and Receive queues, respectively.

The side that initially has send permission on each pipe retains send permission for the life of the pipe.

Protocol for CMACCP/CMACCI

Any CPI-C 2.0 application program that accepts a full duplex conversation will do so by calling either CMACCP or CMACCI. For a simulated full duplex CPI-C conversation to already be established, the protocol for CMALLC (see “Protocol for CMALLC” on page 9) must have been completely executed. In particular, step 20 allows a simulated full duplex CPI-C conversation to be waiting. No further flows result from executing CMACCP/CMACCI in this case.

Note that CMACCP and CMACCI will accept true full duplex (if possible), simulated full duplex, or half duplex CPI-C conversations.

Protocol for CMDEAL

For a simulated full duplex CPI-C conversation, if the deallocate_type is CM_DEALLOCATE_ABEND, then after completing an APPC Deallocate(ABEND_PROG) on the send pipe, issue an APPC Deallocate(ABEND_PROG) on the receive pipe. Otherwise, issue a APPC Deallocate only on the send pipe, using the appropriate deallocate_type.

Protocol for CMFLUS

In most cases, a CMFLUS maps directly to an APPC Flush verb. But for a simulated full duplex CPI-C conversation with a conversation_type of CM_BASIC_CONVERSATION and a fill_type of CM_FILL_BUFFER, issue an APPC Confirm verb instead.

If the program has started, but not finished, sending a logical record, issuing an APPC Flush verb will result in a state check. We may be unable to fully achieve transparency here.

Protocol for CMRCV

For a simulated full duplex CPI-C conversation, whenever an APPC Receive_and_Wait or Receive_Immediate issued by CMRCV receives a Confirm indicator, immediately issue an APPC Confirmed verb. Should the latter fail, return a return_code of CM_PRODUCT_SPECIFIC_ERROR to CMRCV.

If CMRCV issues an APPC Receive_and_Wait or Receive_Immediate and gets a primary return code of AP_DEALLOC_ABEND*, deallocate the LU 6.2 conversation used as the send pipe, or unbind the underlying session, if the conversation or session is still present.

Other Issues

For a CPI-C implementation that supports nonblocking, any one of the APPC verbs issued during CMALLC or CMACCI could return a primary APPC return code of AP_OPERATION_INCOMPLETE. Such a CPI-C implementation must be prepared to return a CPI-C return_code of CM_OPERATION_INCOMPLETE and to subsequently complete these outstanding APPC verbs as part of the completion of the outstanding CPI-C call.

An implementation may choose to automatically discard Attaches or to reject a CMACCP with a return_code of CM_PROGRAM_STATE_CHECK after a certain period of time has elapsed.

The following decisions are left to the discretion of CPI-C implementers:

- Where the SNA service TP for CPI-C full duplex simulation resides and how it gets started
- How application TPs get started
- How and when CPI-C conversation control blocks are built, and how they are identified by the FDX Descriptor's ***correlator*** field

- The method whereby conversations set up by the SNA service TP on the accepting side are stored, awaiting a matching CMACCP/CMACCI

Flow Diagram for CMALLC

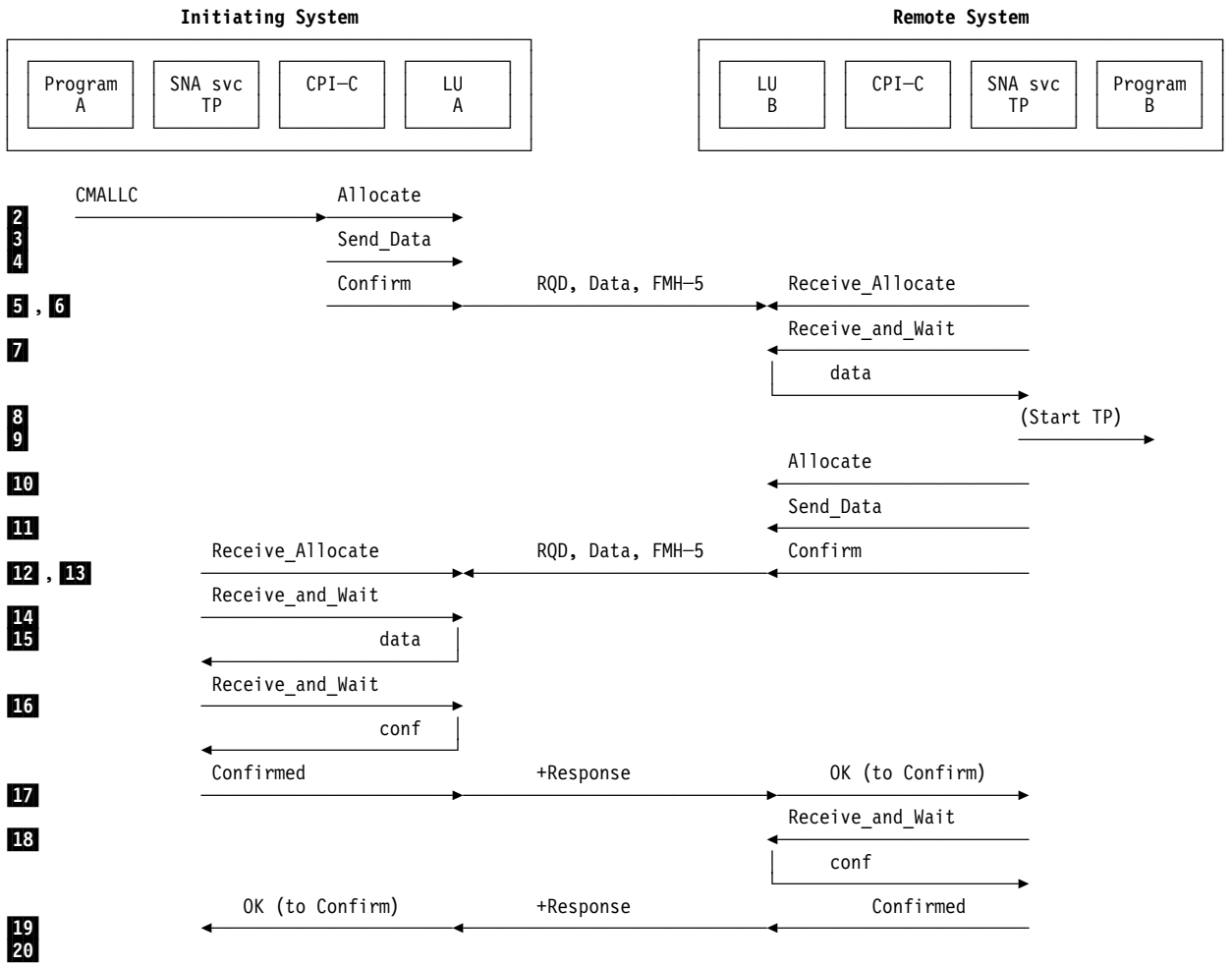

Figure 1. Establishing a Simulated FDX Conversation (Program A allocating to Program B).

Discussion

In the proposed design, we address the stated requirements in the following ways:

- Transparency

  We use the unchanged full duplex API of CPI-C 2.0.

- Performance

  To achieve acceptable performance, we allocate a pair of LU 6.2 conversations to serve as the send and receive pipes.

- Use true full duplex capability where available

  If true full duplex support is available on both partners, we use it. If true full duplex support is available only on the initiating side, we soon discover this fact and attempt to simulate a full duplex connection.

- Safety

  Using a SNA service TP allows early detection of whether a remote partner is capable of compatibly simulating full duplex conversation support. Because we never allocate or send an FDX Descriptor directly to a application TP, we can never allocate to a half duplex CPI-C program by mistake, nor send an FDX Descriptor where it does not belong.

  The SNA service TP for simulation of full duplex CPI-C conversations does exactly one thing: set up simulated full duplex CPI-C conversations. Therefore, it is always safe for a CPI-C implementation to attempt to allocate a simulated CPI-C full duplex conversation by allocating to it. A partner mismatch will not go undetected.

Other aspects of the design:

- Use of sync_level Confirm

  On the initiating side, using sync_level Confirm allows us to easily block the initiating TP until the SNA service TP has finished setting up its receive pipe. Thus, when that Confirm completes successfully, the simulated full duplex conversation has been completely established. On the remote side, Confirm enables us to easily block the SNA service TP until it has finished setting up that side's send pipe.

- Use of ***correlator***

  The use of a ***correlator*** is a simpler and less error-prone method of associating send pipes with receive pipes than the use of a scheme where arbitrary TP names are assigned and mapped (particularly if TP names must be unique within a system).

- Modest LU 6.2 requirements

  The proposed design requires no changes to the LU 6.2 architecture such as might be required by, say, a new GDS ID or a lengthened FMH-5 Conversation Correlator field.

  A minimal implementation of LU 6.2 is sufficient to implement the proposed design. Specifically, only the following APPC calls are used to add full duplex simulation to an existing CPI-C implementation over LU 6.2:

  Allocate
  Confirm
  Confirmed

Deallocate
Receive_Allocate
Receive_and_Wait
Send_Data

Except for Receive_Allocate, only the APPC verbs in the base set are used, not the more exotic ones like Register_Local_TP_Name.

Alternative Solutions

Manage a single pipe

One alternative to twin pipes that has been tried is to have the CPI-C implementation allocate and manage a single half duplex LU 6.2 conversation, turning the line around at appropriate times. This alternative has shown poor performance, with the single pipe becoming a bottleneck.

Although setting up twin pipes is more involved than setting up a single pipe, managing the pair is simpler than making a single pipe do double duty.

Do twin piping between application programs

There is nothing to stop a CPI-C application program from implementing an application-specific protocol. Indeed, twin piping between pairs of TPs has been done in a variety of ways for as long as full duplex support has been lacking in LU 6.2. Setting up twin pipes at the application level requires no changes to the CPI-C implementation. This approach is not discussed further here.

A slightly different approach is to have the CPI-C implementation rather than the application programs set up the twin pipes but to direct all flows between CPI-C application programs rather than to use a SNA service TP. The primary advantage of this approach is that it requires no changes to the method of automatically starting TPs.

Application-level protocols have significant disadvantages:

- Transparency is not preserved, since
  - CMACCP cannot be used to accept a simulated full duplex CPI-C conversation. The CPI-C implementation must know, after accepting a half duplex LU 6.2 Attach, whether to receive an FDX Descriptor and then to help establish the second pipe.

    Since a CMACCP call cannot be preceded by another CPI-C call for the same conversation, only CMACCI could be used. One non-transparent protocol implementation allowed CMACCI to be preceded by CMSSRM(CM_FULL_DUPLEX), but the CPI-C 2.0 architecture disallows this.
  - CMALLC cannot complete until the remote partner has executed a matching CMACCI. These semantics are very different from that of CPI-C 2.0.
- Safety is not maintained. TPs must be matched, or the protocol for establishing full duplex conversations will fail. In particular, it is possible for an accepting TP to accept as a half duplex CPI-C conversation an Attach that was intended as one of the twin pipes and subsequently to receive an unexpected FDX Descriptor as data.