E40

Designing Applications to Web-enable IMS

Suzie Wendler

Las Vegas, NV **September 15 - September 18, 2003**

RETURN TO INDEX

Application Design Considerations

The Environment

- Network requirements - SNA or TCP/IP

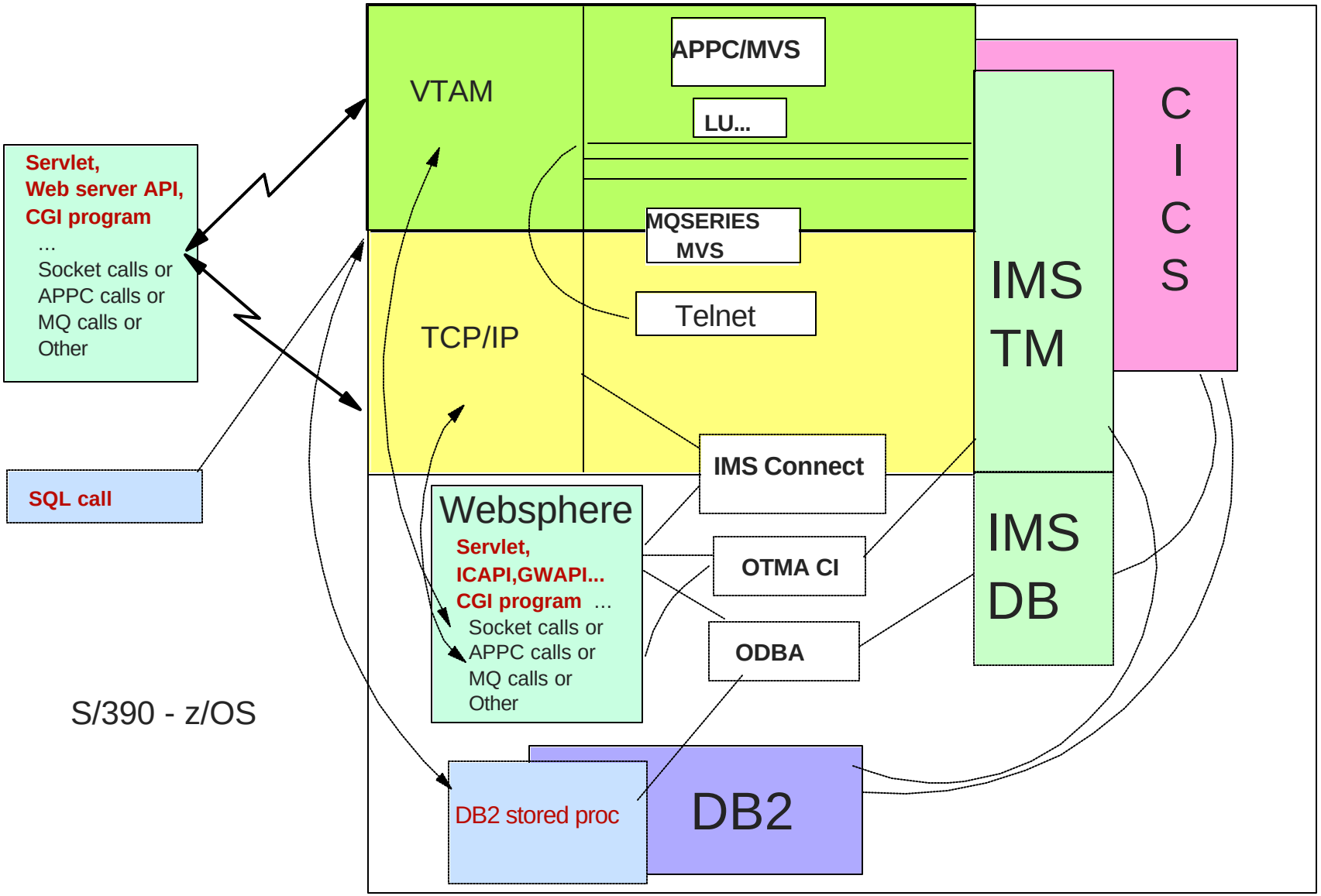
Application requirements

- Direct connection model vs. Messaging and Queuing model
- Access to transactions versus direct access to data
- Inquiry (read-only) or Update
- Simplicity or extensibility

Development requirements

- Programming language
- Skill
- Cost - Build versus Buy and Modify
  - Toolkits

Define the Environment ...



Application Requirements

Direct Connection Model (transactions)

- Characteristics
  - Processing begins only if connections can be established
  - Immediate notification of problems
    - Error indicators sent in the case of failures

- Most popular types of support

  - **3270 emulation - Traditional interface**
    - SNA=EHLLAPI, TCP/IP=TN3270

  - **Program-to-Program support**
    - SNA=APPC, TCP/IP=Sockets
    - Interactive processing
    - Output messages can be sent before/after IMS syncpoint
      - Remote programs can affect whether or not commit occurs

Application Requirements ...

Messaging and Queuing Model (transactions)

- Characteristics
  - Processing occurs whether or not a connection is made
    - Assured delivery of messages (inbound/outbound) when components and/or network are available

- Support
  - MQSeries
    - Remote program is not sensitive to the network type
      - MQ provides its own high-level standard API
      - Same applications can be deployed on TCP/IP or SNA

Application Requirements ...

Direct Connection (database)

- ODBA interface (Open DataBase Access)
  - Programs that issue database calls must reside on the same MVS as IMS

Starting Simple - 3270 emulation

3270 emulation

- Straightforward and simple
  - IMS is unaware that the access is from the Web
- Traditional IMS communication model

Initial Web approaches to 3270 emulation

- Some browsers packaged 3270 emulators in the browser
- Some software mapped 3270 data streams to HTML

Java-based approach

- Applet provides 3270 emulation at the browser when needed
- Products / tools
  - Websphere Host Integration Solution
    http://www-3.ibm.com/software/webservers/hostintegration/
  - ResQNet - www.resqnet.com
  - Jacada - www.jacada.com
  - ...

Starting Simple - 3270 emulation ...

Application Considerations

- **IMS**
  - Applications execute "business as usual"
  - Connection to IMS is SLU2 through a Telnet server
  - Traditional commit model (commit-then-send)

- **Remote program**
  - Understand emulation package and any restrictions
  - Determine interaction requirements for communication
    - Emulation-only with browser front-end screens versus
    - More complex application interfaces

- **Setup and Configuration**
  - Telnet uses the concept of pooled LUs
    - Are the IMS applications sensitive to LTERM name?

Starting Simple - 3270 emulation...

Benefits

- Straightforward implementation
  - Quick and easy from an IMS perspective
- IMS continues to communicate using SNA LU2 protocols
  - Web browser does sees more than just a "green screen"
    - Customized web screens
- Easy way to web-enable existing 3270-based transactions
  - 3270 attributes and interaction

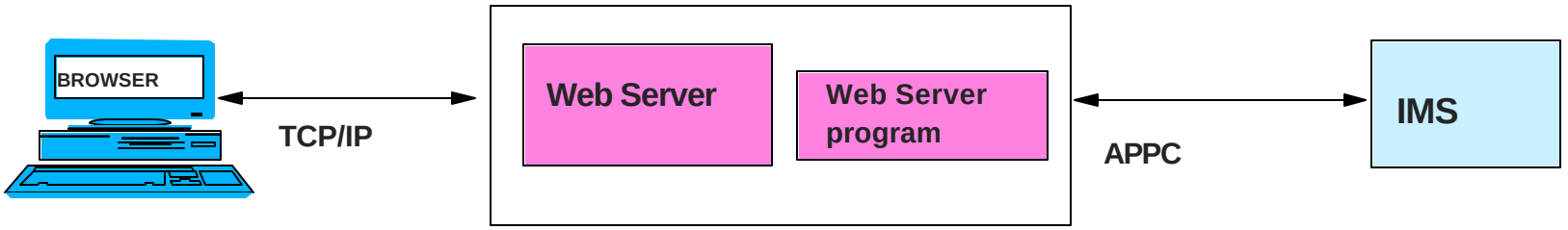
Why consider any other solution?

- Need for greater extensibility and a solution based on something other than 3270 emulation

Direct Connection - SNA (APPC)

Benefits

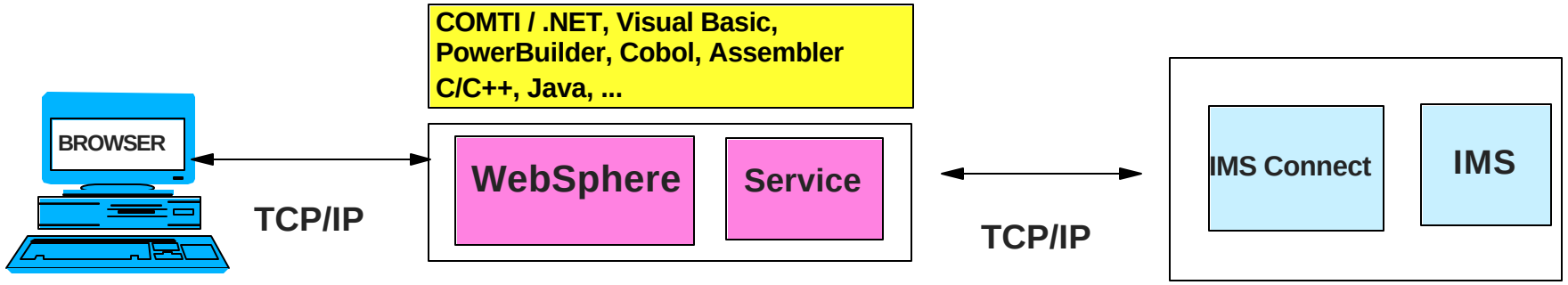
- IMS TM supports APPC natively with a choice of capabilities
  - Implicit - IMS functions as the partner program
    - IMS applications continue to use DL/I calls
  - Explicit - the IMS application program directly controls the communication sequence using APPC calls
  - Synchronization levels (none, confirm, syncpoint)
  - Commit modes (synchronous, asynchronous)
- The remote program uses a standard well-defined interface
  - Connectivity using APPC has been for many years
- Several existing solutions available from IBM and other vendors



Direct Connection - TCPIP

Benefits

- IMS Connect provides the capability for existing IMS applications to be invoked using standard TCP/IP socket calls
  - The capability is flexible and extensible for capacity and performance requirements
  - Exit interfaces are provided for modification and tailoring to a specific environment's needs
  - IMS application programs do not have to be modified
    - Continue to use the DL/I call interface
- Remote Programs
  - Have a direct connection capability to the IMS environment
  - Are provided with a documented, standard interface



Messaging and Queuing - MQSeries

Benefits

- MQSeries provides a programming interface that can be deployed across multiple platforms on different types of networks
- Adapter
  - Uses the ESS interface
  - The IMS application uses explicit MQ calls to get/put messages in MQ with syncpoint coordination in IMS
  - Calls to MQ, DB2 and IMS in the application are considered one UOW
- Bridge
  - Uses the OTMA interface
  - Coordinates message transfer between the MQ queue and the IMS message queue
  - Allows the application to use DL/I to access the messages

Application Design

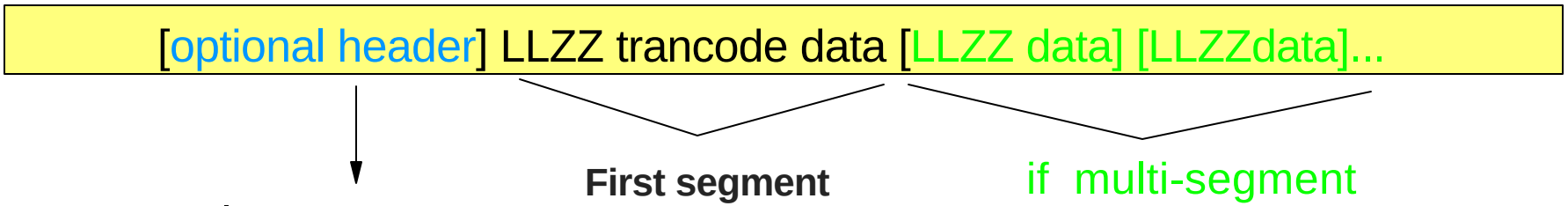
Design considerations that pertain to:

- APPC applications
- OTMA clients
  - IMS Connect for TCP/IP socket applications
  - MQSeries,
  - OTMA CI
  - ...

Note: the subsequent pages provide overall design considerations when comparing the different protocols/clients. They do not go into the details of application design.

Messages

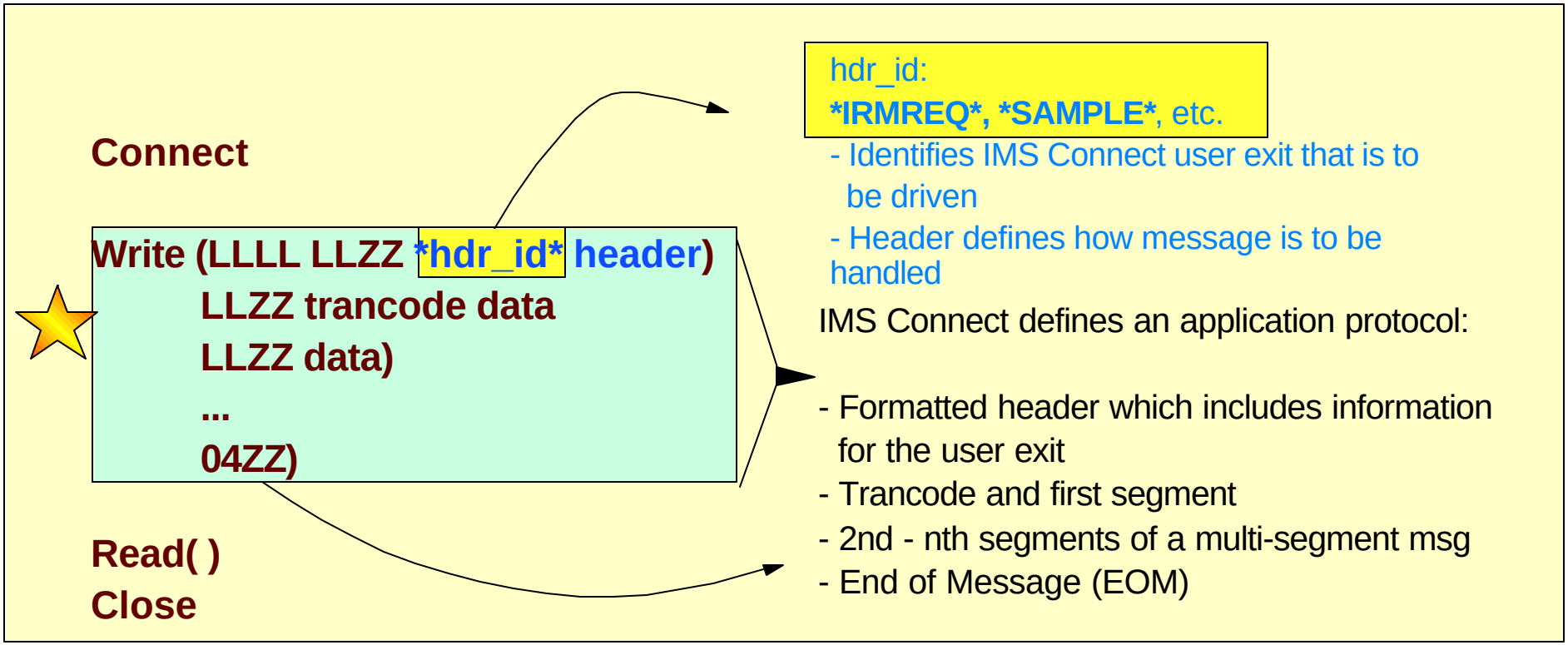
Input Message Format



- Headers
  - Unique to Application environment
    - APPC - parameters in the Allocate verb
    - OTMA CI - parameters in the calls
    - MQSeries - MQIIH
    - IMS Connect - *IRMREQ*, *SAMPLE*, ...
- Shield the remote application from having to understand and create IMS message headers
- Provide a way to influence the interaction with IMS
  - Commit mode and synclevels, Overrides (lterm, etc.), Timeouts...

Messages ...

IMS Connect



Sending just one write with all the data improves performance over multiple writes

To prevent multiple writes from causing performance problems with IMS Connect, make sure the IMS Connect PORT statement specifies NODELAYACKS

Messages ...

IOPCB output reply messages

- Single segment: ISRT IOPCB (LLZZ data)
- Multi-segment: ISRT IOPCB (LLZZ data), ISRT IOPCB (LLZZ data)...
- Multiple messages: ISRT - PURGE, ISRT - PURGE, ...
  - APPC
    - Synchronous: First message is sent synchronously, subsequent messages are sent asynchronously
    - Asynchronous: All messages are sent asynchronously
  - OTMA
    - Send-then-commit: Messages are sent as one multi-segment message
      - Purge is ignored, only one output message per commit scope
    - Commit-then-send: Messages are sent as separate messages

Data Sensitivity

MFS is not invoked

- Remote programs send/receive data in the "raw" form used by the IMS application program in the IOAREA
  - Determine what this data looks like

3270 attributes

- Understand how they are used, if at all, by the IMS application
- If necessary, they can be sent as data in the data stream
  - Remote program(s) will need to deal with this
    - E.g., highlight, color, redisplay of data, ...
  - Can add complexity to the remote program
- May be a reason for a specific application to be web-enabled using a solution like Host On Demand

Data Sensitivity ...

Sensitivity to MOD/LTERM name in the IOPCB

- Default LTERM name
  - APPC - partner_lu name
  - OTMA - Tpipe name
- Default MOD name
  - blanks
- To override defaults
  - APPC - implement DFSLUEE0 (LU 6.2 Edit Exit Routine)
  - OTMA - specify value in message prefix (differs per OTMA client)
    - OTMA CI: in the verb parameters
    - MQ: in the MQIIH header
    - IMS Connect: in the message exits or in the header

Timing Considerations

Timing out

- How long should a process wait?
  - What action should be taken when a timeout occurs?
- Setting timeout values
  - APPC
    - In IMS: APPCIOT value in DFSDCxxx
      - Times out waits on the IMS side
    - In remote program, this is based on
      - Verb used: receive, prepare_to_receive, etc.
      - Specific implementation: post_on_receipt, blocking wait, etc.
      -
  - OTMA
    - MQ program - MQGET with timeout or wait interval
    - IMS Connect - timeout value in the configuration
      - IMS Connect client program - timer value in header
      - PQ54020/ PQ66542 / PQ67660

IMS Connect - Timers

- **Provides a greater level of granularity for timeout settings**
  - IRM_TIMER value in IRM header
  - Time values:
    - no wait, wait indefinitely, .01-95 sec, 1-60 sec, 1-60 min
  - Specified by the client program and affects
    - RESUME TPIPE
    - SEND ACK/NAK
    - SEND of data
  - Also affects:
    - HWSIMSO0, HWSIMSO1, HWSJAVA0,
    - HWSSMPL0, HWSSMPL1

IMS Connect - Timers ...

- **When setting value, consider appropriate wait time for IMS to return data to IMS Connect**
  - Defaults
    - RESUME TPIPE and associated ACK: 0.25 seconds
    - All other SENDs: HWSCFGxx TIMEOUT value
- **Each client SEND can specify a different value**
  - Guidelines
    - SEND of trancode+data or data only, or SEND of ACK associated with RESUME TPIPE
      - Set value to reflect the wait time in IMS
        - Do not use X'E9' - no wait
    - SEND of ACK/NAK associated with last output message
      - Set value to X'E9' - no need to wait
    - SEND of RESUME TPIPE
      - Value depends on AUTO, NOAUTO or SINGLE option

Synchronization levels

- **These levels control the interaction between the IMS environment and the remote/calling program**
  - **None** - assumes the partner received the message
    - No acknowledgement required
  - **Confirm** - requests acknowledgement of message receipt
    - Allows greater integrity and interaction
  - **Syncpoint** - implements the support for distributed commit
    - Ensures all partners go through commit/backout
    - Supported by APPC and OTMA in IMS
      - Ability to use support depends on environment of calling program

Commit Scopes and Implications

Asynchronous (APPC) - Commit-then-Send (OTMA)

- Reflects traditional IMS processing model
  - Message sent as a result of a successful commit
  - Input and output messages are enqueued
  - Has restrictions on transaction types
    - **APPC requests cannot access:**
      - **Response mode, Conversational, or IFP transactions**
    - **OTMA requests cannot access:**
      - **Conversational or IFP transactions**

Synchronous (APPC) - Send-then-Commit (OTMA)

- APPC/OTMA can access all transaction types
- Remote program waits for a reply
  - **Terminating tran without a reply can result in DFS2082 msg**
  - **IMS sends the output reply before commit**
- When used with synchronization level = confirm
  - Remote program controls when/if the commit occurs in IMS
    - **Impacts dependent region occupancy and database locks**
- Must be used if synchronization level = syncpoint

Asynchronous Output

- **Asynchronous output support**
  - Alternate TP PCBs (ALTPCB) messages
  - Queued commit-then-send reply messages (IOPCB) that could not be sent back on the original connection
- **APPC**
  - Destination is specified in CHNG call or through use of LU6.2 descriptors
  - Remote program is written to ACCEPT the connection, RECEIVE the message, and ACK back to IMS
- **MQSeries**
  - Requires OTMA exits
    - Pre-Routing Exits (DFSYPRX0)
    - Destination Resolution Exit (DFSYDRU0)
      - Specifies destination and MQ header

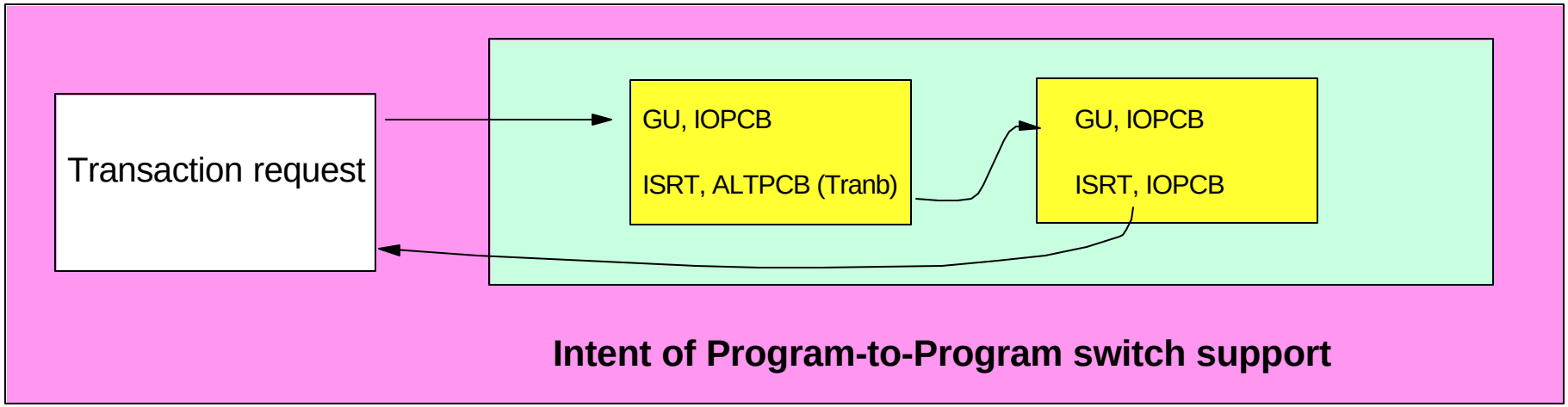
Asynchronous Output ...

- **IMS Connect**
  - IMS environment - IMS V7/V8
    - IMS application ALTPCB destinations
      - Specify a destination = tpipe name = client id
    - IMS OTMA Exits needed for ALTPCB output
      - Prerouting Exit Routine (DFSYPRX0)
      - Destination Resolution Exit Routine (HWSYDRU0)
  - Remote client environment
    - Retrieve messages

RESUME TPIPE - specify client id
- specify request type (single, noauto, auto)
RECEIVE - receive first output msg
ACK - acknowledge receipt of first msg
RECEIVE - receive second output message
ACK ...

Program-to-Program Switches

Transfer responsibility of replying to the IOPCB



- IMS transaction chain
  - Can involve multiple transactions e.g., A -> B -> C ....
  - Can be sent across an MSC link

- The Remote transaction request can be
  - Asynchronous / Commit-then-send
  - Synchronous / Send-then-commit

Program-to-Program Switches ...

Asynchronous conversations/commit-then-send requests

- Remote program design accounts for a reply to be sent asynchronously or expects no reply

Synchronous conversations / send-then-commit requests

- Remote program waits for a reply that is sent back via the IOPCB

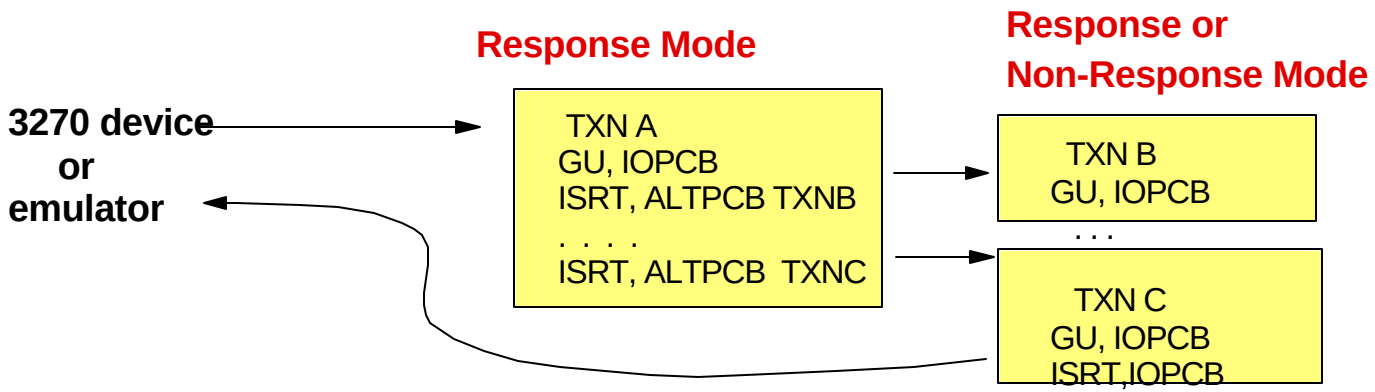
- Considerations:
  - Are the ALTPCBs defined as express?
    - IOPCB replies from express PCB trans do not satisfy the synchronous APPC/OTMA request
  - Does the receiving tran spawn more than one transaction?
    - Are the switched-to transactions response/non-response mode?

- Depending on the protocol used, behavior may differ

Program-to-Program Switches ...

Pgm-to-pgm switch and synchronous requests

- **3270 devices/emulators**
  - Synchronous equates to response-mode



Remote Device either:

A. Receives response message from any spawned transaction that responds to the IOPCB

OR

B. HANGS, if no transaction responds to the IOPCB
   - DFS2802 sent only if TXN A does not respond AND does not spawn another transaction

Program-to-Program Switches ...

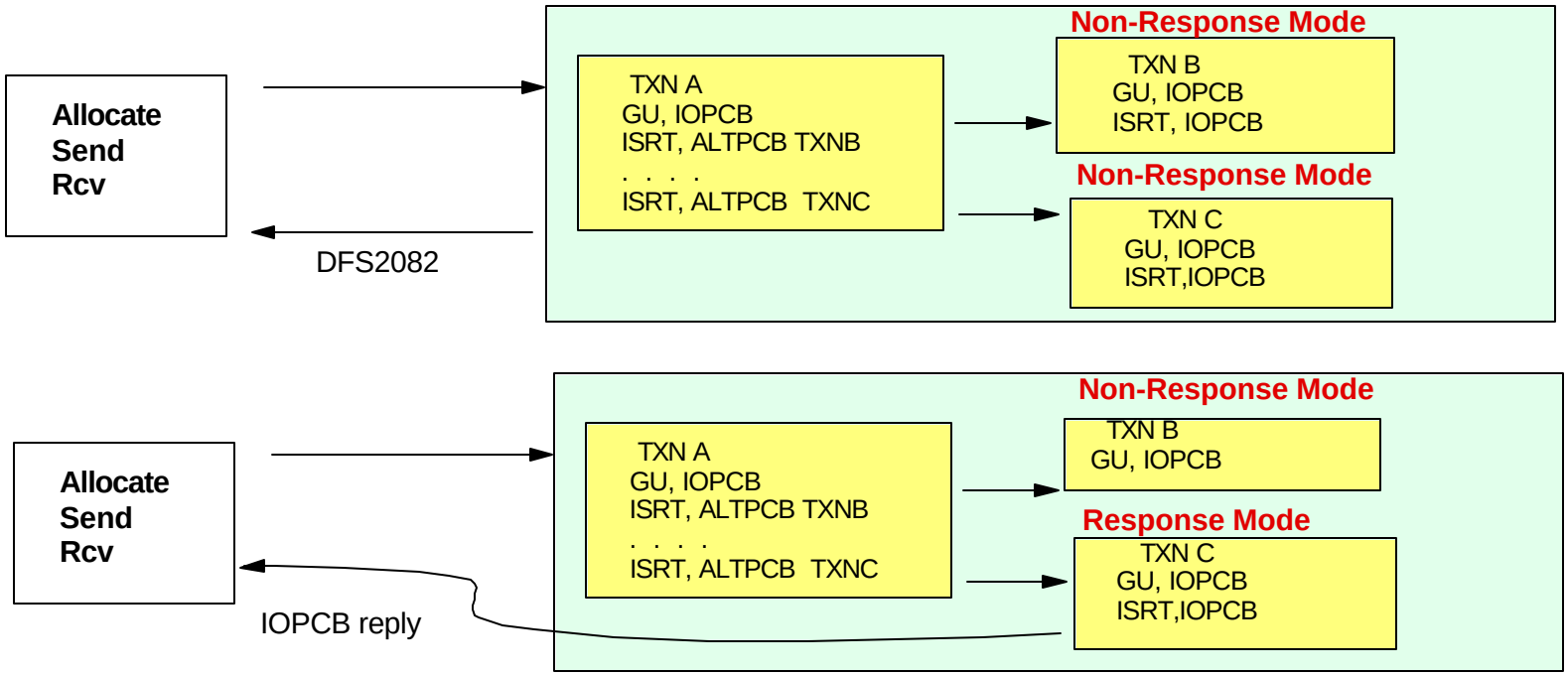
Pgm-to-pgm switch and synchronous requests

- **APPC - based on the APPCASY=Y | N (PQ17309/PQ19930)**
  OTMA - based on the OTMAASY=Y | N (PQ57868)
  - Relies on accurate transaction specification
    - Response mode versus Non-Response mode
    - Way to control which spawned transactions are eligible to reply to the synchronous request
  - Affects APPC synchronous/OTMA Send-then-Commit messages
- APPCASY is specified in DFSDCxxx
  OTMAASY is specified in DFSPBxxx
    - Global specification

Program-to-Program Switches ...

Pgm-to-pgm switch and synchronous requests...

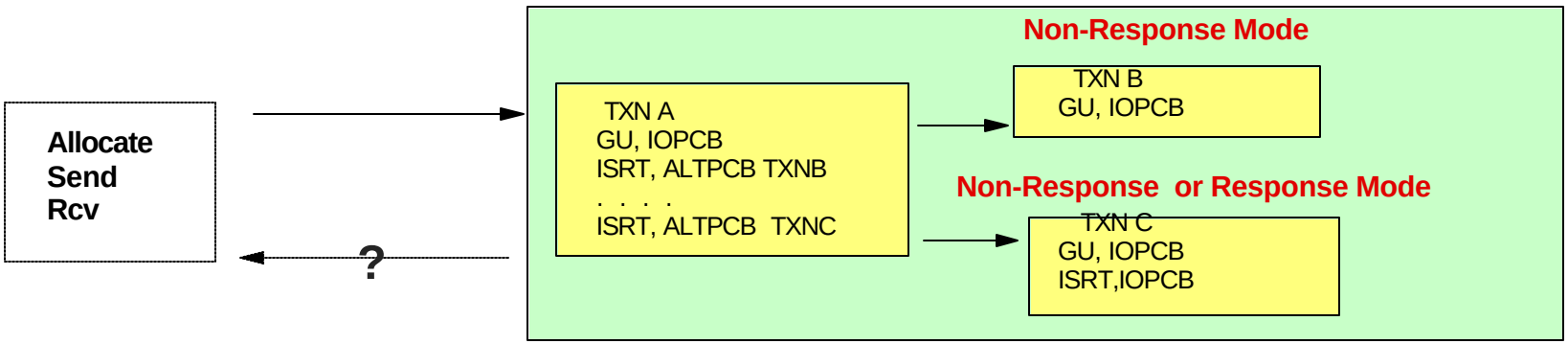
- APPC: APPCASY=Y (default)
  OTMA: OTMAASY=Y (needs to be set if desired)
  - Responsibility for synchronous reply is given to response-mode transaction, else DFS2082



Program-to-Program Switches ...

Pgm-to-pgm switch and synchronous requests

- APPC: APPCASY=N (needs to be set if desired)
  OTMA: OTMAASY=N (default for OTMA)
  - Responsibility for the synchronous reply is given to the first spawned transaction that goes through commit



Check to make sure PQ37780 is in your system to prevent a hang condition:
- Without PQ37780, if TXNB completes first, then the request will hang since there is no iopcb reply
- With PQ37780, if TXNB completes first, then a DFS2082 is sent
- If TXNC completes first, the IOPCB reply is sent

IMS Conversational Transactions

IMS Conversational transaction processing

- Access to IMS Conversational transactions is supported
- Remote program keeps track of the interaction
  - Maintains synchronous / send-then-commit requests
  - Follows the rules provided by the environment
    - APPC and OTMA clients differ
    - Cannot use certain commands to control the conversation, e.g., /HOLD, /RELEASE, etc.
      - May impact online change procedures

IMS Conversational Transactions ...

APPC

- Straightforward support - IMS keeps track of both the IMS and the APPC conversation
- Remote program must maintain a synchronous conversation
  - Each send/receive invokes an iteration of the IMS conversation

OTMA

- IMS provides information to the OTMA client (MQ, IMS Connect)
  - Conversation state and conversation id
- Relies on the OTMA client/remote pgm to manage the conversation
  - OTMA client can choose to externalize this to remote programs
- Remote programs
  - Must use Send-then-Commit requests
  - Check for conversation state and save/pass the conversation id

Design for Failure

Direct Connection Model (transactions)

- Designing for failure (when transactions in IMS do updates)
  - Did the commit occur in IMS?
    - Depends on
      - Commit mode (commit-then-send vs send-then-commit)
      - Synchronization level (none vs confirm vs syncpoint)
      - Whether any output messages have been received
    - Possible actions for the web/client program
      - Act on error indicators
      - Provide the ability to send an inquiry
  - On the IMS side - review exit routines
    - DFSCMUX0 - can choose to retain the output reply on a connection failure and send it later/elsewhere
    - DFSNDMX0 - can choose to preserve the input message when the IMS application abends

Designing for Failure ...

Messaging and Queuing Model

- Designing for failure
  - Determine what messages could go to the dead-letter queue
  - Decide on specifying timeout values
    - What happens to delayed messages?
  - If a web server program times out waiting for a message reply
    - Did the commit occur in IMS?
    - Is there a forthcoming output reply that needs to be handled,
      - Application output or DFS error message
    - Possible actions
      - Create a process to respond to the reply - save/ print/ route, ...
      - Provide the ability to send an inquiry
  - IMS Exit Routines (DFSCMUX0, DFSNDMX0)
    - ...

Application Design - Summary

Remote program - IMS program interaction sequences

- Understand the existing process / requirement of navigating from one transaction to the next
  - Save output from one transaction as input to next
  - Buttons instead of PFKeys
  - Determine if the browser back button should be disabled
  - ...
- Determine the extent to which the IMS transaction is coded to 3270 screen behavior
  - Define what needs to be added to the remote program
- Understand the failure scenarios

RETURN TO INDEX