IBM Software Group

IMS22

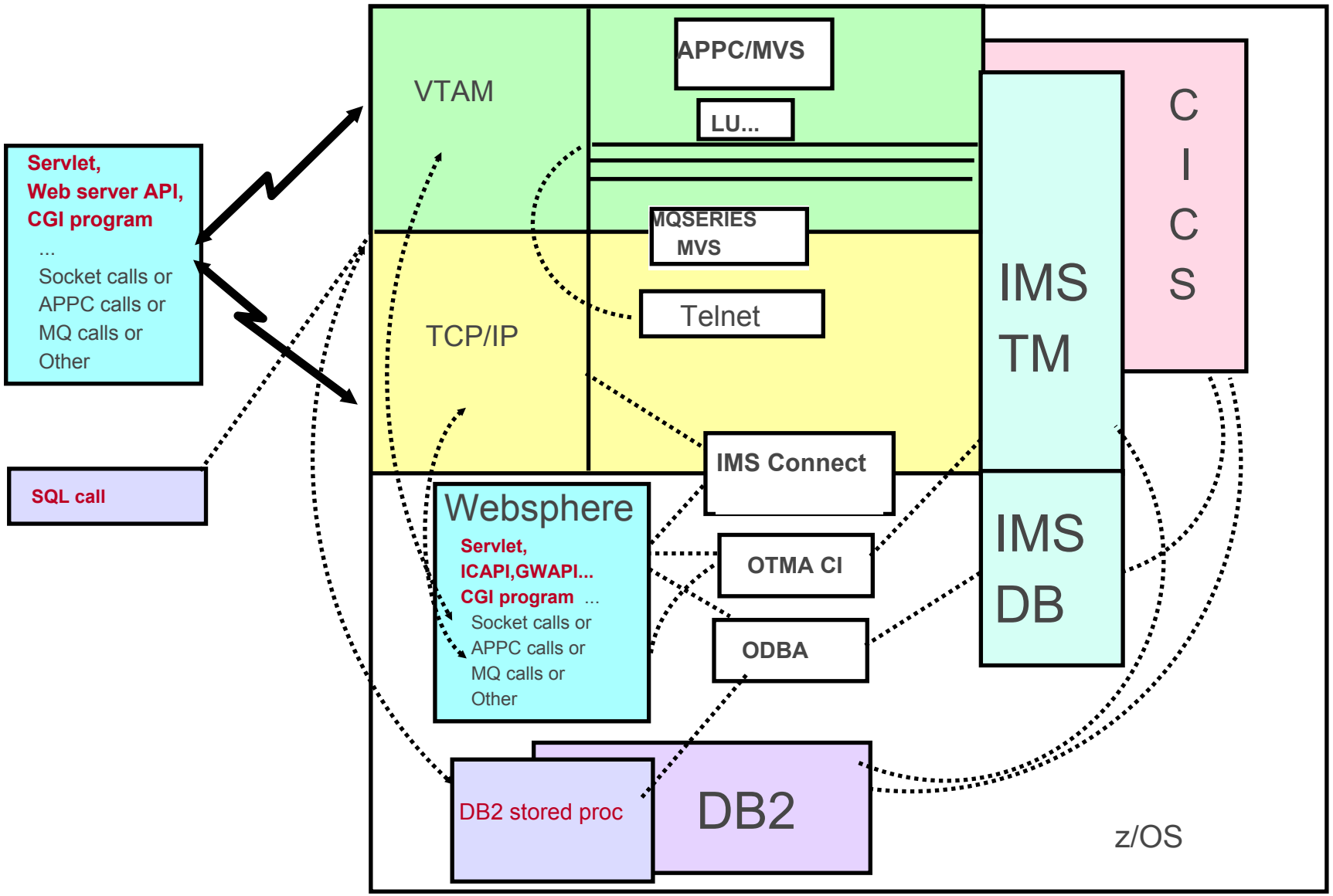
Designing Applications to Access IMS Transactions and Databases

Werner Müller

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

Choices


Servlet,
Web server API,
CGI program
...
Socket calls or
APPC calls or
MQ calls or
Other
SQL call
VTAM
APPC/MVS
LU...
MQSERIES
MVS
TCP/IP
Telnet
IMS
TM
C
I
C
S
IMS Connect
Websphere
Servlet,
ICAPI,GWAPI...
CGI program ...
Socket calls or
APPC calls or
MQ calls or
Other
OTMA CI
ODBA
IMS
DB
DB2 stored proc
DB2
z/OS

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

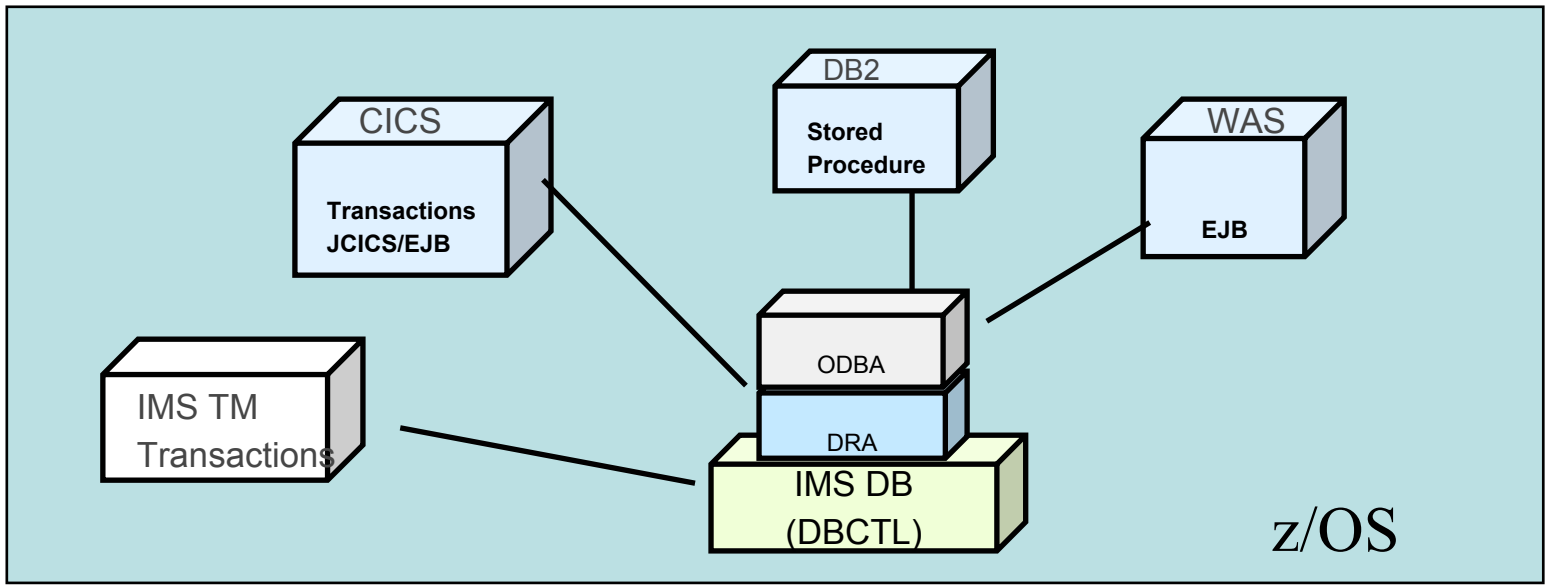
Comparisons

- **Direction Connection**
  - Natively synchronous (connection-oriented), supports asynchronous (connectionless)
  - Direct correlation between input and output
  - Potential issues with program-to-program switches when spawning multiple transactions
  - Easily supports IMS conversational transactions (relatively transaparent)
  - Designing for failure:
    - If connection can not be made, try later
    - Decide what to do when the connection breaks - understand IMS actions

- **Messaging and Queuing**
  - Natively asynchronous (connectionless), simulates synchronous (connection-oriented)
  - Need to consider how to correlate output to input
  - Easily supports program-to-program switches even when spawning multiple transactions
  - Requires keeping track of the conversation id to continue an IMS conversation
  - Designing for failure:
    - No knowledge of whether entire connection path is available
    - Handle Late reply messages and the dead letter queue

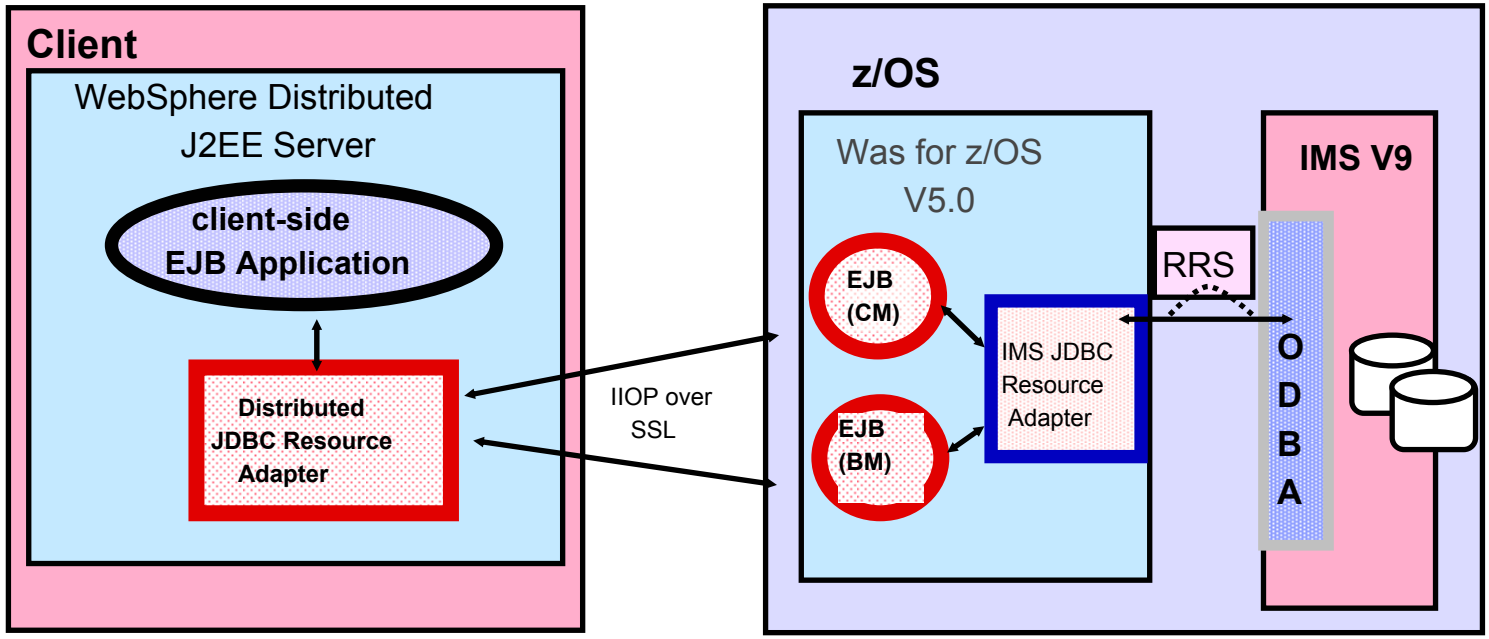
Application Requirements …

- **Direct Connection Model (database)**
  - Characteristics
    - Processing requires access only to IMS DB without TM
  - Support
    - ODBA interface (Open Data Base Access)
      - Programs that issue database calls must reside on the same MVS as IMS



Application Requirements …

- **Direct Connection Model (database) ...**
  - Announced with IMS V9
    - Ability for remote client EJBs to issue JDBC calls to access IMS DB
      - Network access is transparent and unknown to the client EJB



CM - container-managed
BM - bean-managed

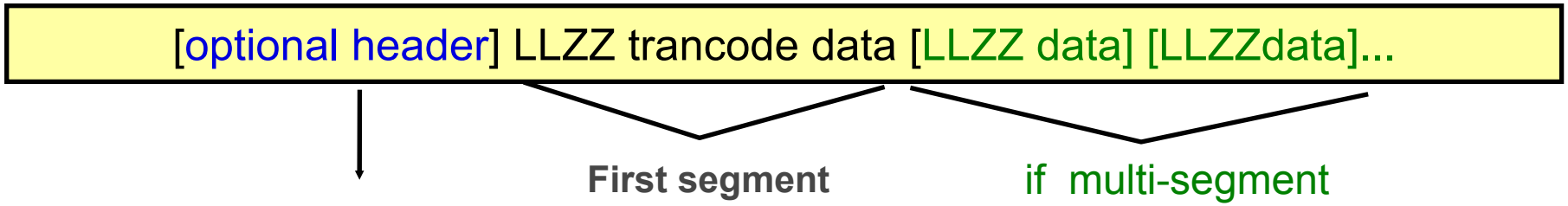
Application Design

- **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.

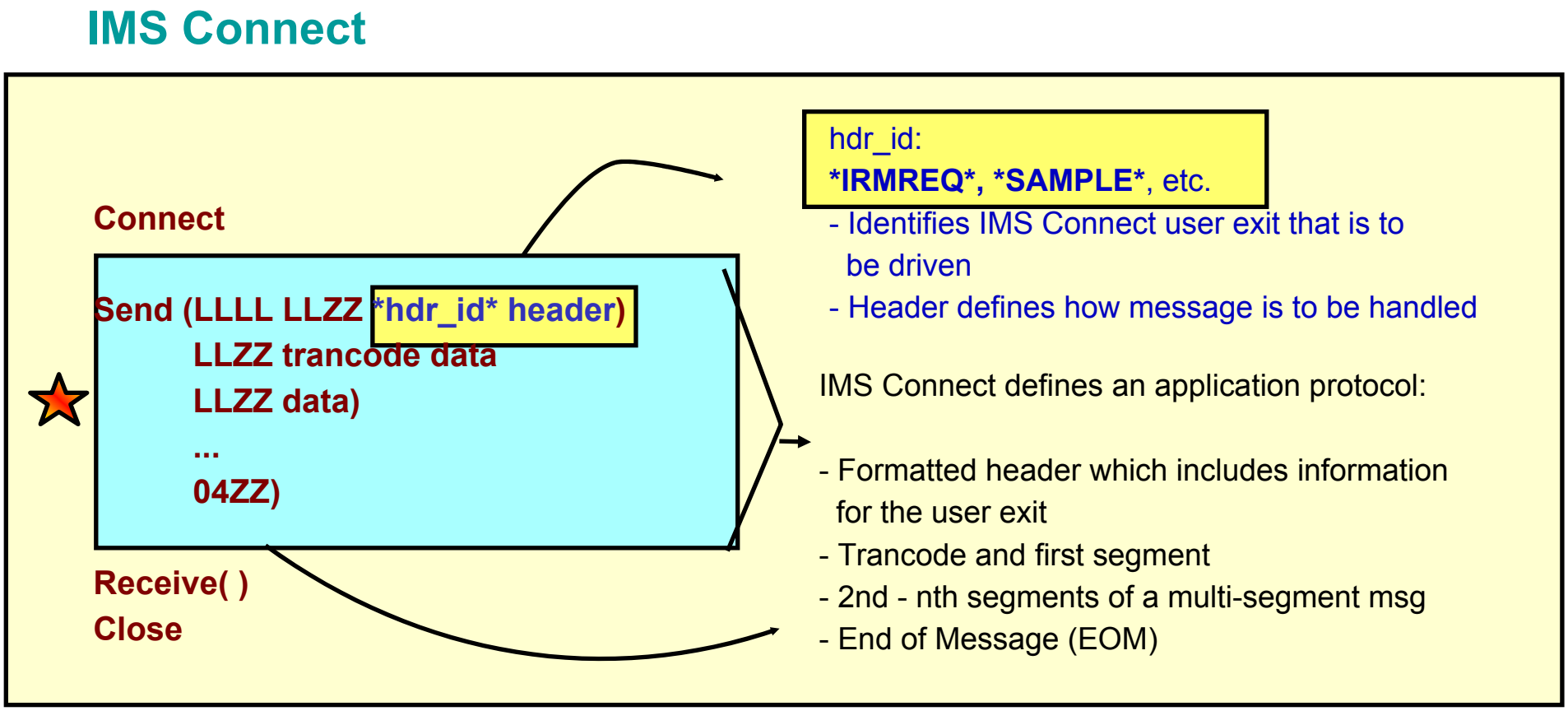
Messages

- **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 (Iterm, etc.), Timeouts...

Messages



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 Websphere Host Integration Solution

Data Sensitivity …

- **Sensitivity to MOD/LTERM name in the IOPCB**
  - Issue:
    - IMS applications look in the IOPCB mask to
      - Branch to other parts of the program
      - Use the value to determine an ALTPCB destination
      - ...
  - Default LTERM name
    - APPC - partner_lu name
    - OTMA - Tpipe name
  - Default MOD name
    - blanks
    - Or provided by MQ for MQ clients
  - To override defaults
    - APPC - implement DFSLUEE0 - LU 6.2 Edit Exit Routine
    - OTMA - differs per OTMA client
      - MQ: in the MQIIH header
      - IMS Connect: in the message exits or in the header

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

- **Synchronous (APPC) - Send-then-Commit (OTMA) CM1**

  - Remote program waits for a reply
    - Direct correlation between input and output messages

  - 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

Commit Scopes and Implications …

- **Asynchronous (APPC) - Commit-then-Send (OTMA) CM0**

  - Remote program does not have to wait for a reply
    - Application design needs to consider correlation between input and output messages

  - 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:

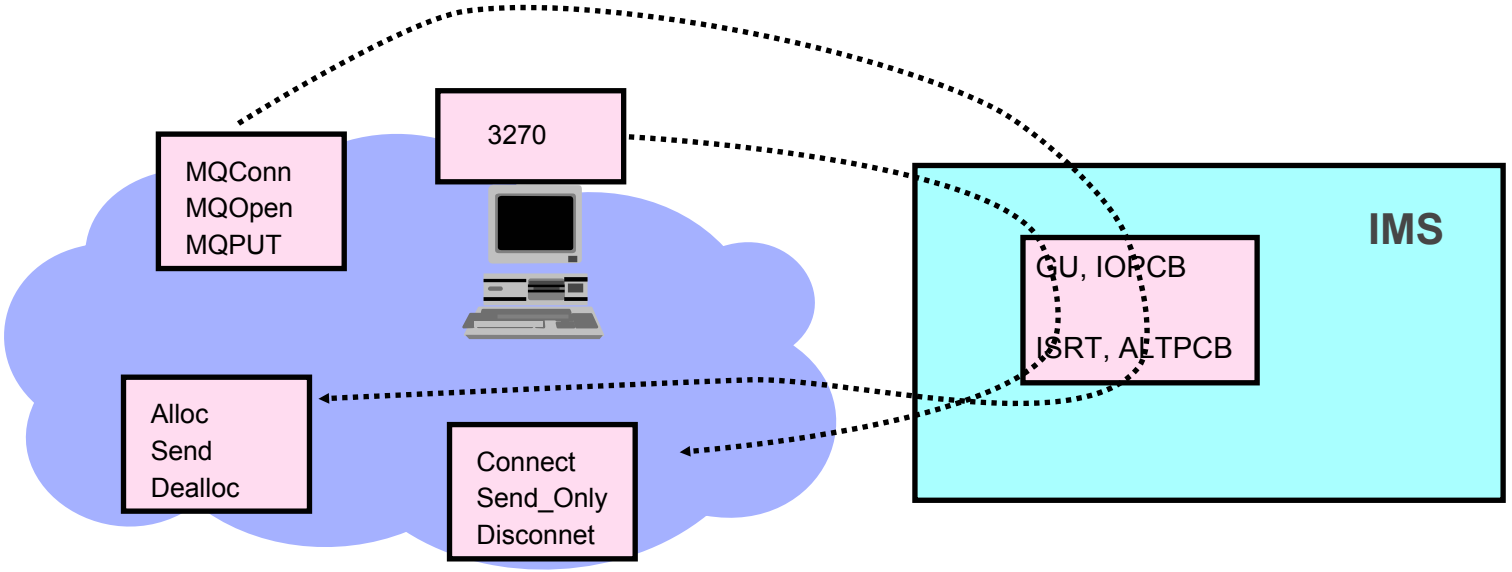
      - APPC requests cannot access:
        - Response mode, Conversational, or IFP transactions
      - OTMA requests cannot access:
        - Conversational or IFP transactions

Asynchronous Input

- **Messages can be sent into IMS with no requirement for output replies**
  - APPC: Allocate, Send, Deallocate
    - Each request is on a conversation that can reuse an existing session
  - MQSeries: MQConn, MQOpen, MQPUT
    - Each request is sent using the MQ connections/sessions
  - IMS Connect: Connect, Send_Only, Disconnect
    - Base support uses a transaction socket (only one send)
      - Requires: Connect, Send_Only, Disconnect
    - Support for a persistent socket was delivered via APAR PQ80468 and is part of base IMS Connect V2.2
      - Allows: Connect, Send_Only, Send_Only, Send_Only ....

Asynchronous Output

- **Asynchronous output support**
  - Alternate PCB (ALTPCB) messages
  - Queued commit-then-send reply messages (IOPCB) that could not be sent back on the original connection



Asynchronous Output

- **APPC**
  - Destination is specified in CHNG call or through LU6.2 descriptors
    - Default TPName is DFSASYNC
  - Remote program is written to:
    - ACCEPT, RECEIVE, ACK, DEALLOCATE
- **MQSeries**
  - Destination is specified in the header ReplyToQ
  - Requires OTMA exits
    - Pre-Routing Exits (DFSYPRX0)
    - Destination Resolution Exit (DFSYDRU0)
      - Specifies destination and MQ header
  - Remote program is written to:
    - MQConn, MQOpen, MQGet, ...

Asynchronous Output

- **IMS Connect**
  - IMS environment
    - IMS application ALTPCB destinations
      - Specify a destination = tpipe name = client id
        - Assumes remote client will retrieve messages using this name
    - IMS OTMA Exits needed for ALTPCB output
      - Prerouting Exit Routine (DFSYPRX0)
      - Destination Resolution Exit Routine (HWSYDRU0)

Asynchronous Output

- **IMS Connect ...**
  - Retrieval of messages is a remote client responsibility
    - Messages stay on the IMS message queue until requested
  - Remote client program must be written to

```
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 ...
```

  - Specify Asynchronous request type

```
SINGLE   - receive one msg and disconnect the socket
SINGLE with WAIT – wait for one message if one is not already there
NOAUTO - receive all available msgs, wait a specified time and disconnect if no more messages
AUTO     - receive all available msgs, wait for next message with no timeout
```

  - Optional wait values: .01-.95 sec, 1-60 sec, 1-60 min, no wait, infinite

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 APPC actions
      - 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

IMS Connect - Timers

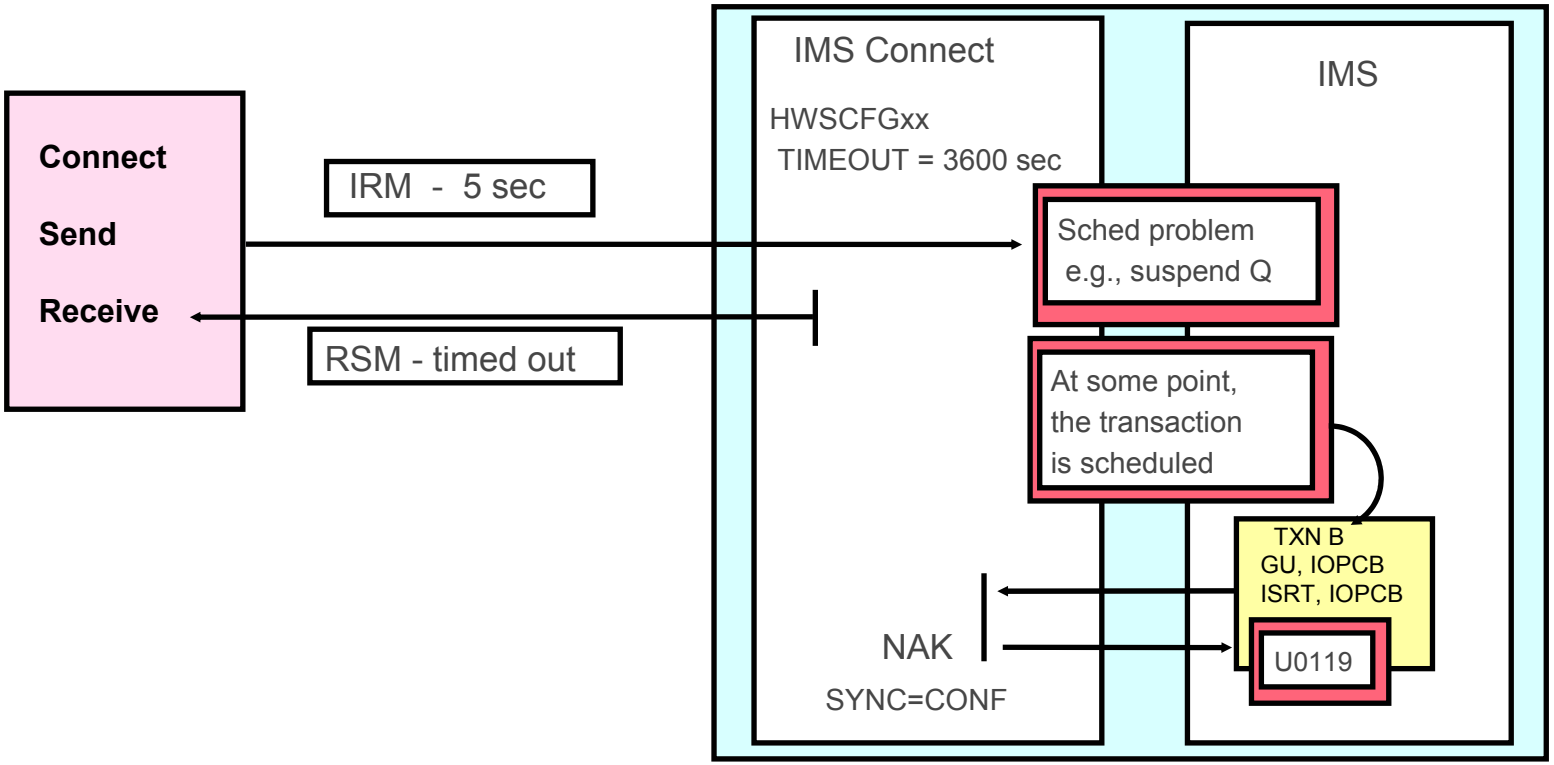
- **HWSCFGxx TIMEOUT value (default)**
  - Defines time to wait for a response from IMS that is being sent to the client
    - Time Values: hundredths of seconds

- **IRM_TIMER value in IRM header**
  - Provides a greater level of granularity for timeout settings
    - 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

If specifying long wait values check out the TCP values for: KEEPALIVEOPTIONS or INTERVAL

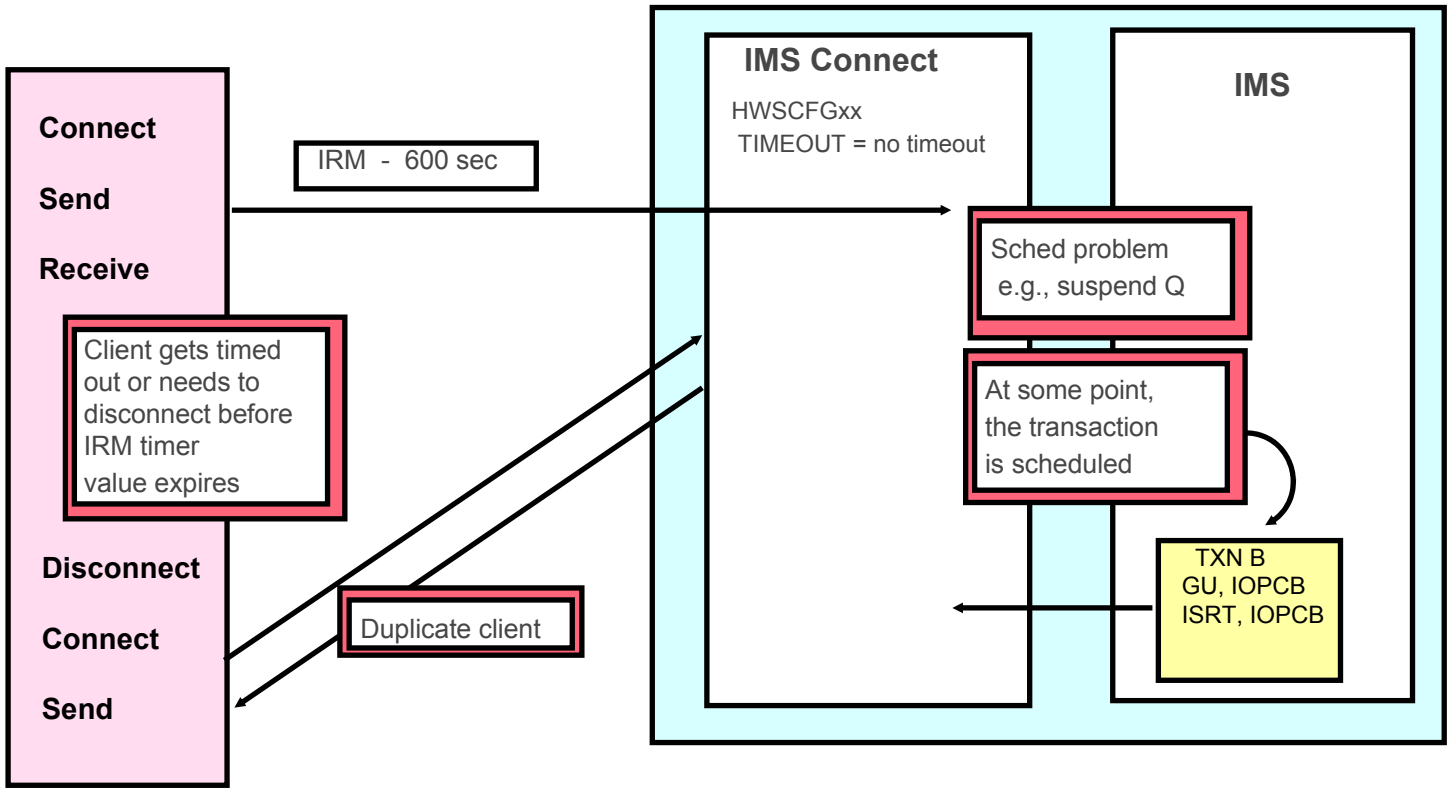
IMS Connect – Timers …

- **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



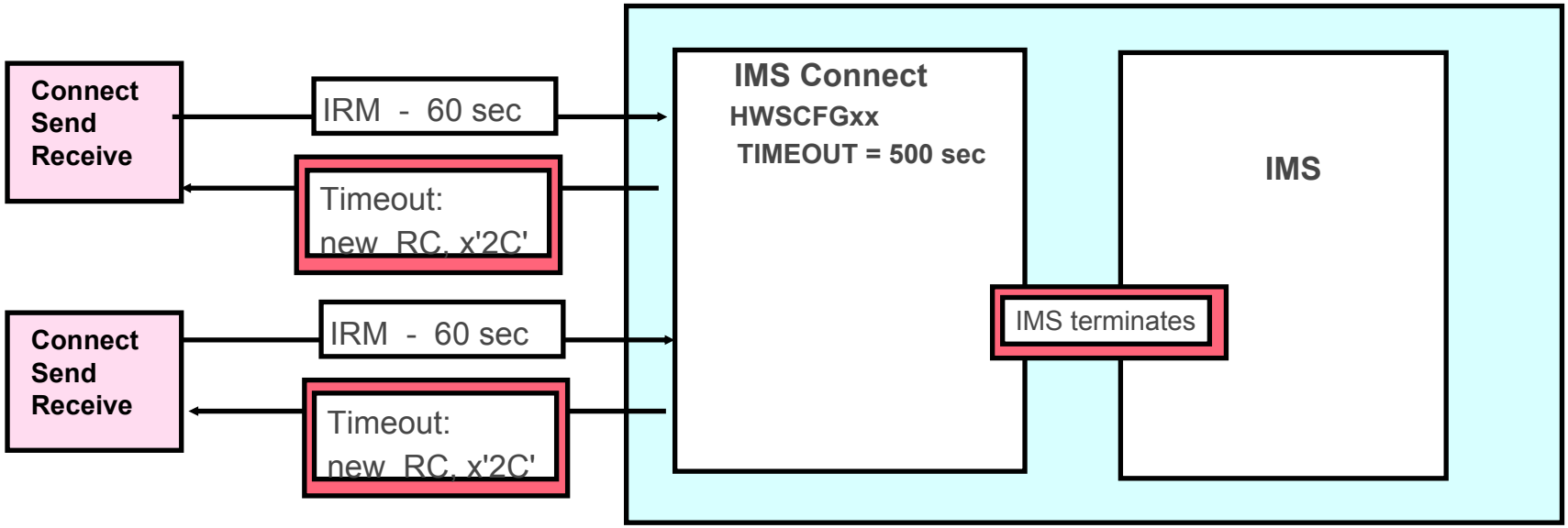
IMS Connect – Timers …

- **Considerations**
  - Attempts to disconnect while timeout is in effect (ignored)
    - Duplicate client
      - Remote client must wait until IRM timer value expires
      - Or, STOPCLNT is issued (PQ74146 or IMS Connect V2.2)



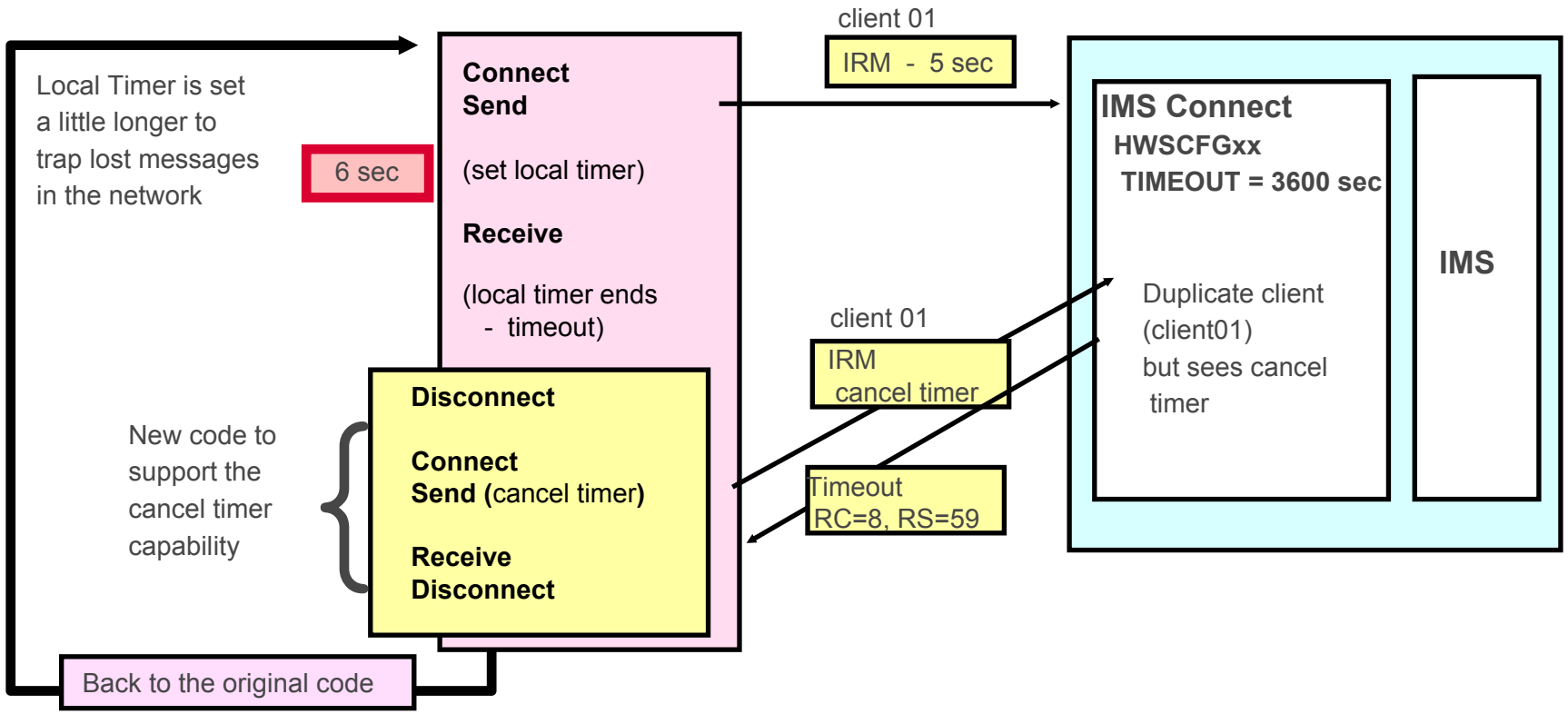
IMS Connect – Timers …

- **PQ74039 or IMS Connect V2.2**
  - Notifies the remote client that OTMA has terminated
    - Supported by HWSSMPL0, HWSSMPL1, HWSIMSO0, HWSIMSO1



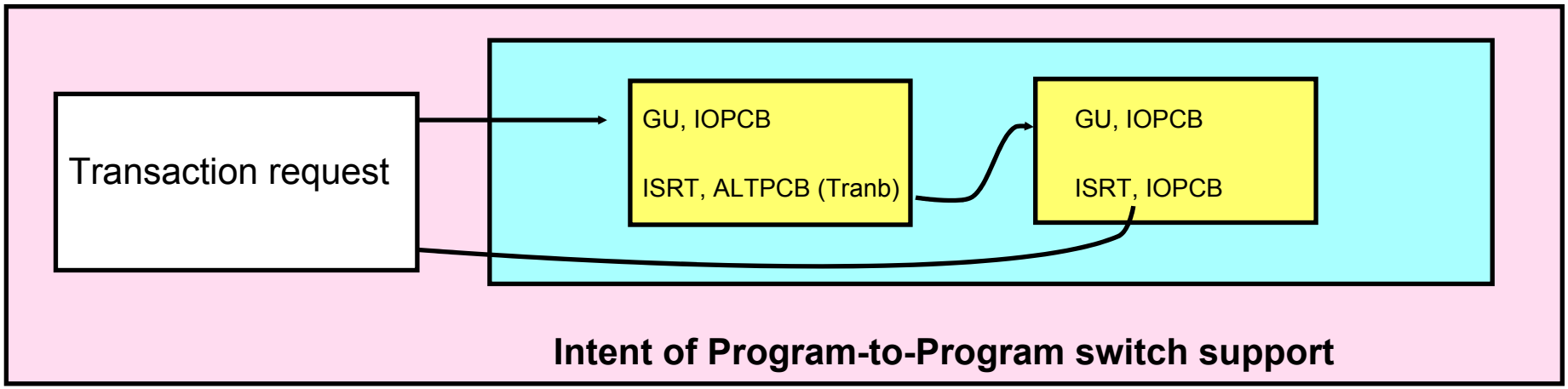
IMS Connect – Timers …

- **PQ74039 or IMS Connect V2.2 …**
  - Allows a "cancel timer" request from the remote client program that is waiting for a response from IMS
    - Supported by HWSSMPL0, HWSSMPL1



Program-to-Program Switches

- **Transfer responsibility of replying to the IOPCB**



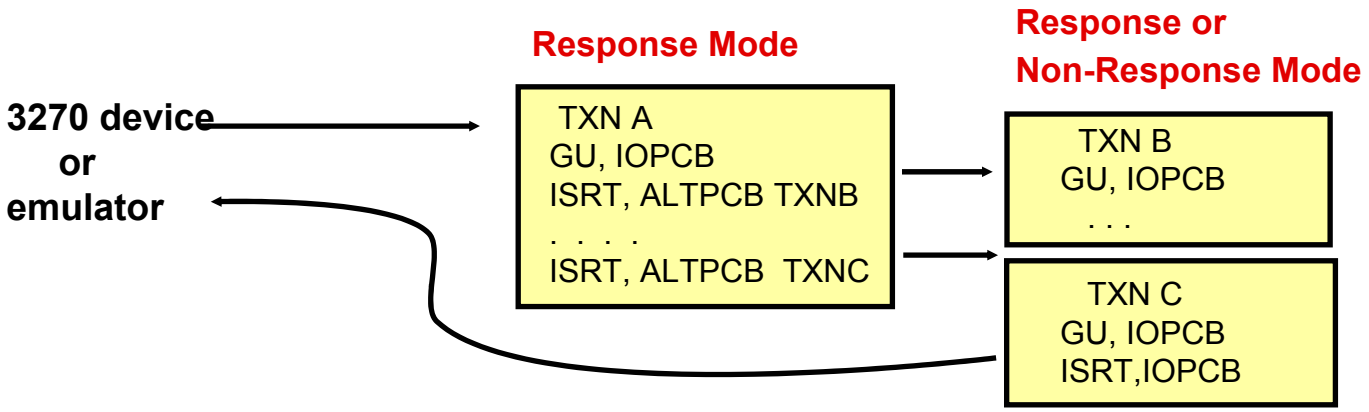
  - IMS transaction chain
    - Can involve multiple transactions e.g., A -> B -> C ....
    - Can be sent across an MSC link
  - The Remote transaction request can be
    - Asynchronous / Commit-then-send
    - Synchronous / Send-then-commit

Program-to-Program Switches …

- **Asynchronous conversations (commit-then-send)**
  - Remote program design accounts for a reply to be sent asynchronously or expects no reply
- **Synchronous conversations (send-then-commit)**
  - 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

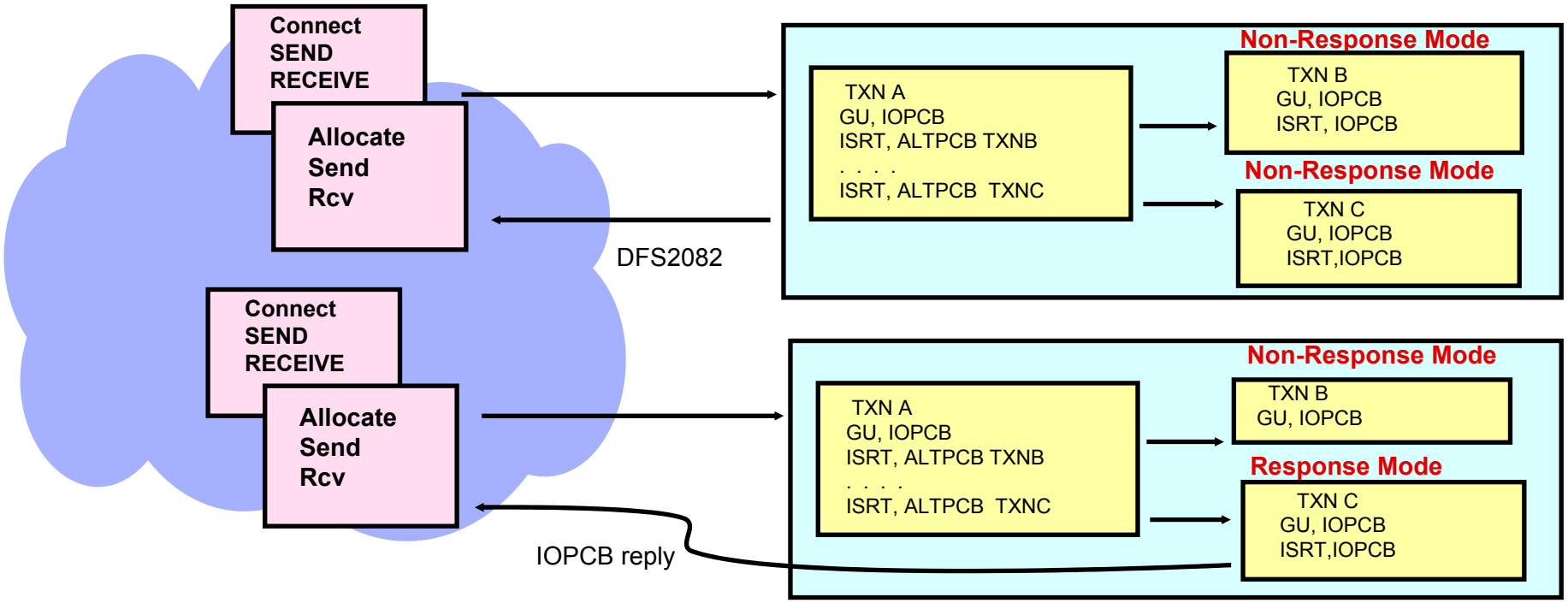
B. HANGS, if no transaction responds to the IOPCB
   - DFS2802 sent only if TXN A does not respond AND does not spawn another transaction

Program-to-Program Switches …

- **Pgm-to-pgm switch and synchronous requests**
  - APPC - based on the APPCASY=Y | N
  - OTMA - based on the OTMAASY=Y | N
    - 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 …

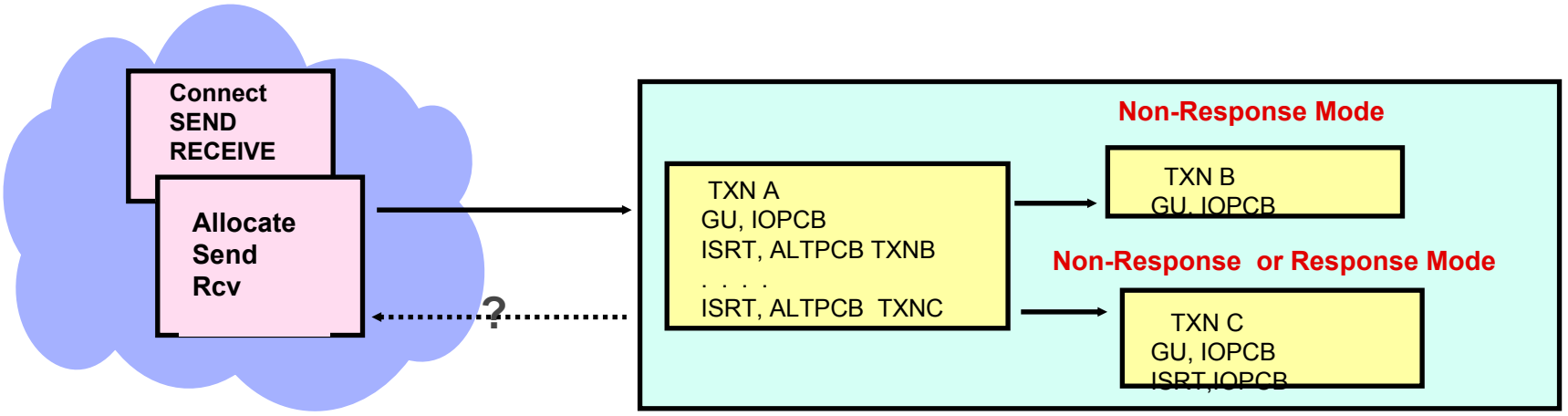
- **Pgm-to-pgm switch and synchronous requests...**
  - APPC: APPCASY=Y (default)
  - OTMA: OTMAASY=Y (needs to be set if desired)
    - Responsibility for synchronous reply is given to response-mode transaction, else DFS2082



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)

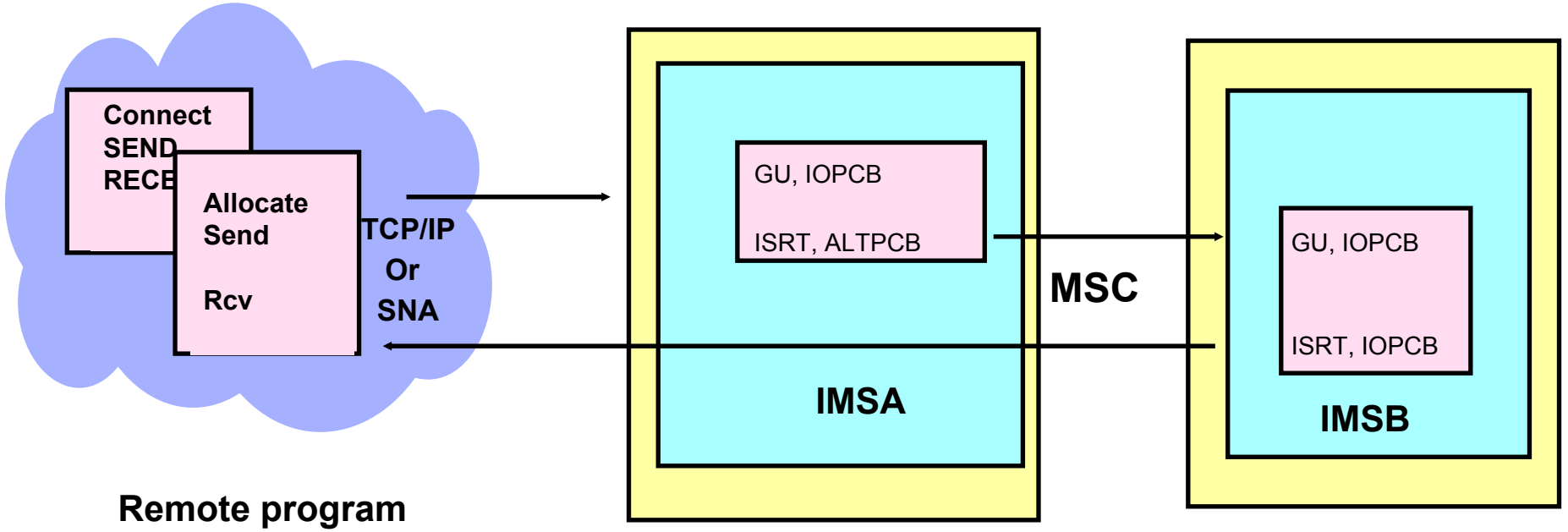
    - Responsibility for the synchronous reply is given to the first spawned transaction that goes through commit



- If TXNB completes first, then a DFS2082 is sent
- If TXNC completes first, the IOPCB reply is sent

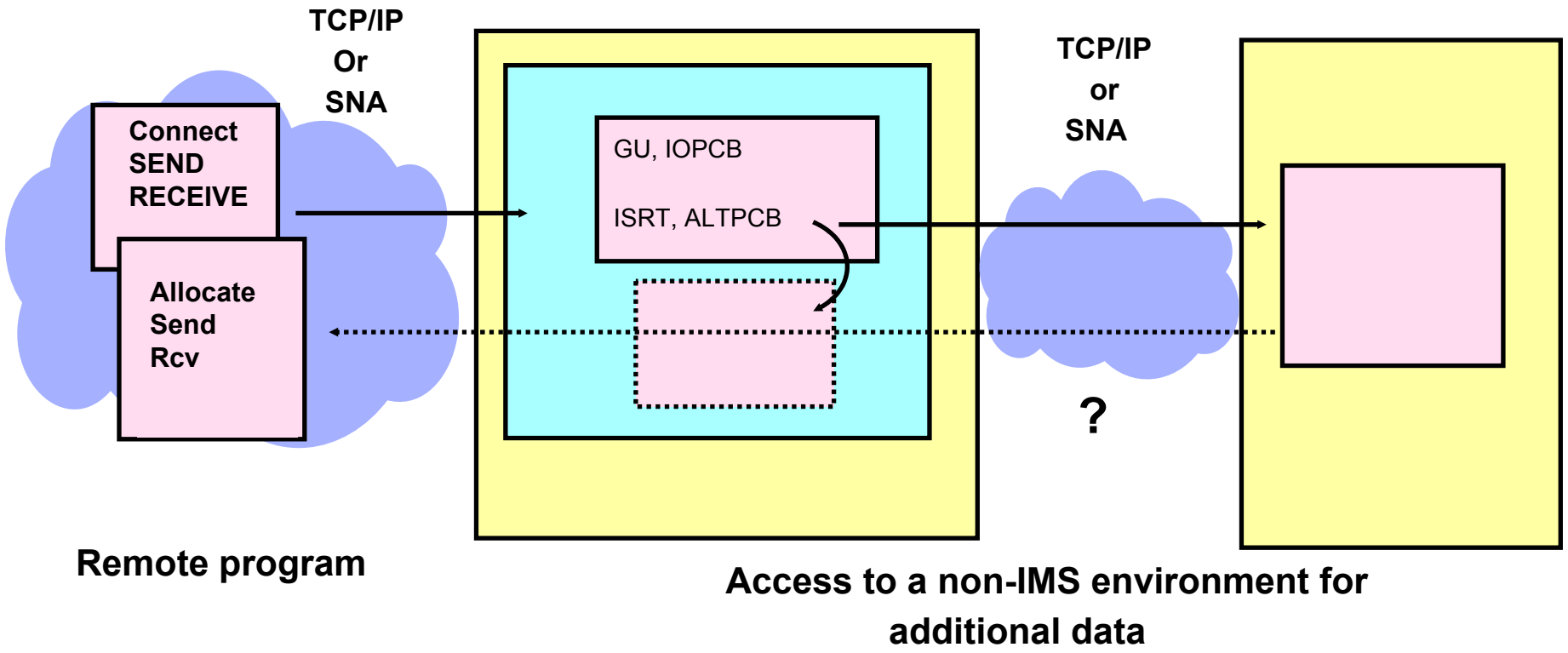
Sending Messages Across an MSC Link

- **Access to another transaction in a different IMS**
  - Supported using MSC
    - APPC/OTMA headers are preserved



Accessing Other Environments

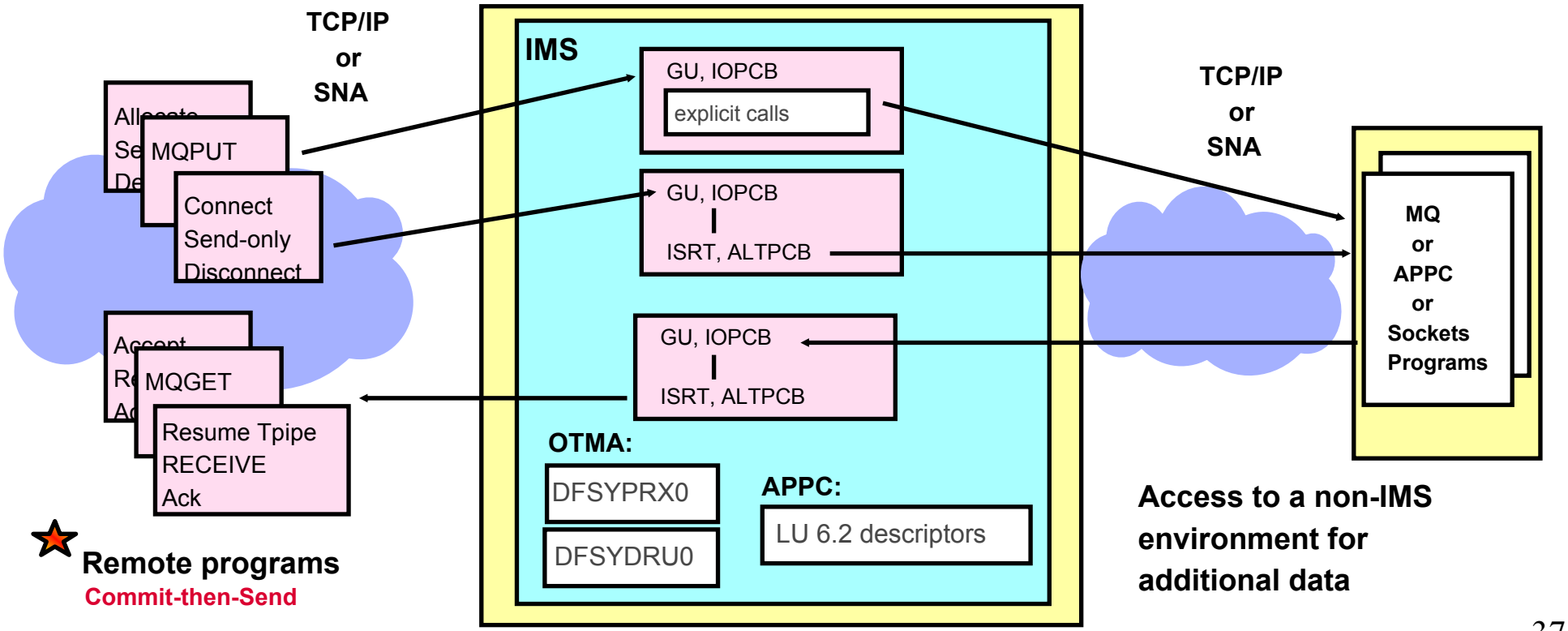
- **Invoking another environment before replying**



- **Does the remote program wait for a reply?**

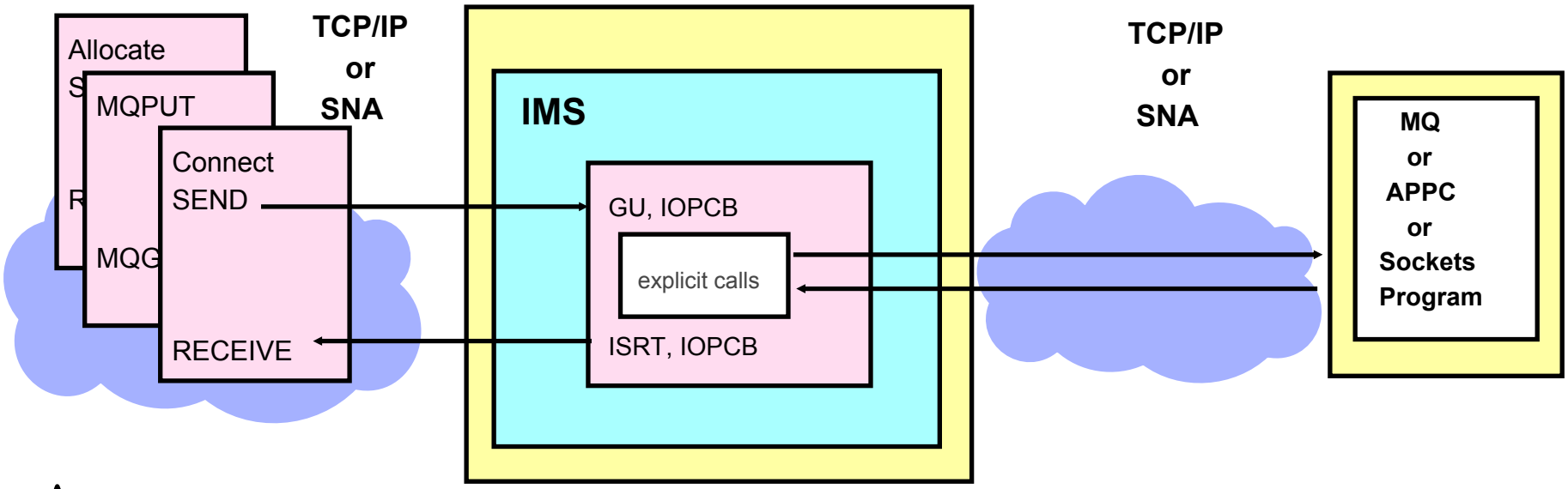
Accessing Other Environments …

- **If the remote program does not wait for a reply** ★
  - Considerations include
    - OTMA Exits, APPC descriptors, MQ queue names
    - Sending programs versus receiving programs



Accessing Other Environments …

- **If the remote program waits for the reply** ★
  - An explicit call can be made from the IMS application
    - Outside IMS control and knowledge
  - Synchronous communication where the IMS application is the client
    - MQ, APPC, Sockets



★ **Remote program**
Send-then-Commit

Access to a non-IMS environment for additional data

Accessing Other Environments …

- **Explicit calls - based on protocol chosen**
  - MQ
    - MQPUT, MQGET
  - APPC
    - ALLOCATE, SEND, RECEIVE
  - TCP/IP Sockets
    - OS/390 and z/OS Sockets support
      - Standard sockets api - C, Java
      - Extended sockets api - Assembler, Cobol, PL/I
        - Callable sockets api

Accessing Other Environments …

- **Socket calls**

Extended Socket functions:
Initapi() - establishes the extended sockets environment if Cobol, Assembler, or PL/I
Socket() - allocates a socket on which communication will flow
Connect() - defines and connects to a server
Write() ┐ transfers data
Read() ┘
Close() - closes the connection

Cobol:
CALL 'EZASOKET' USING SOC-FUNCTION parm1, parm2, .. ERRNO RETCODE.

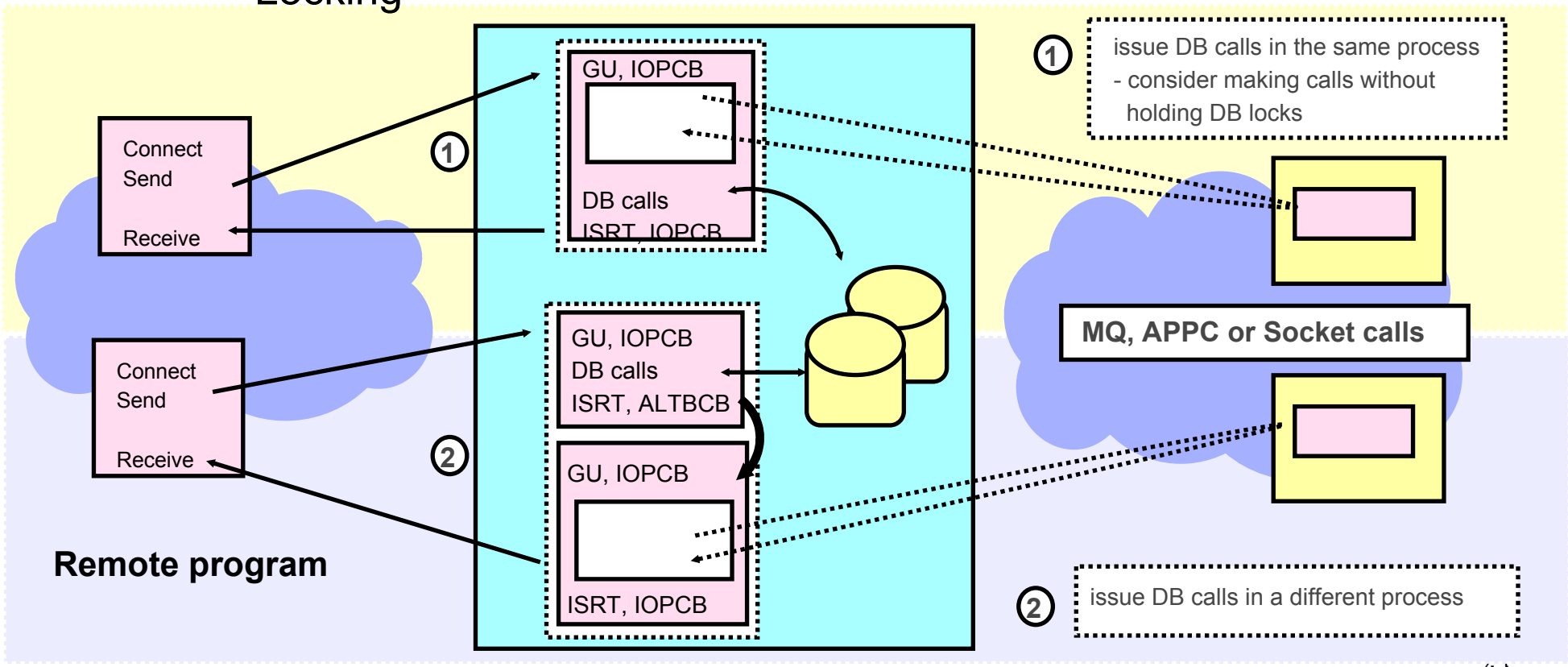
Assembler:
CALL EZASOKET,(SOC-FUNCTION,__parm1, parm2, ...__ERRNO RETCODE),VL

PL/I:
CALL EZASOKET (SOC-FUNCTION__parm1, parm2, ...__ERRNO RETCODE);

EZASOKET interface delivered in PDS "hlq.SEZATCP"

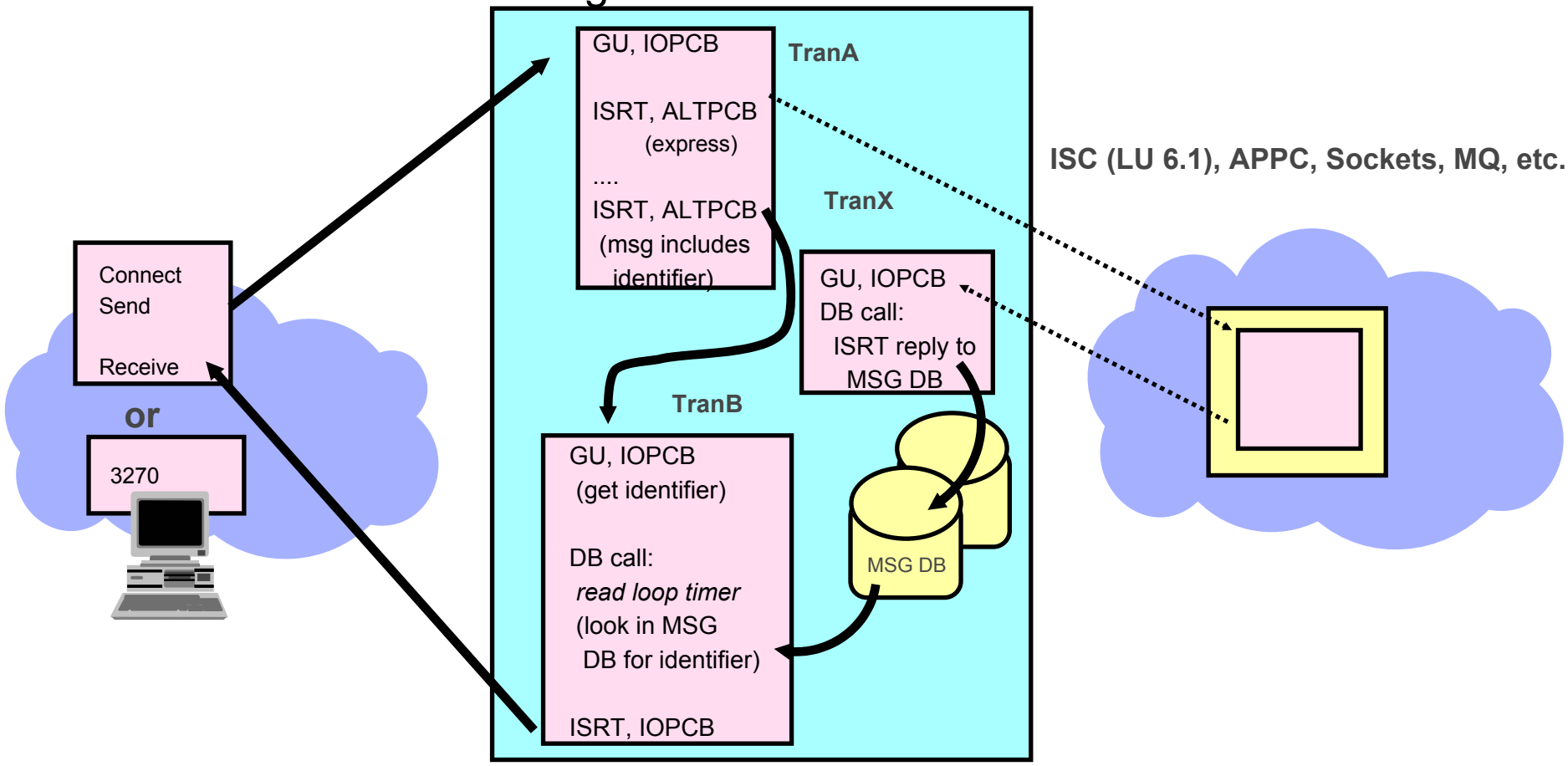
Accessing Other Environments …

- **Considerations when the remote program waits for a reply**
  - Decide on which protocol to use
  - Understand implications on
    - IMS dependent region occupancy
    - Locking



Accessing Other Environments …

- **Creativity**
  - One user's alternative solution
  - Consideration - timing



In Summary ...

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

Design 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