Telcordia Technologies

Performance from Experience

IMS Technical Conference

Accessing MQSeries Messages from IMS Applications at Telcordia

Session C04

Steve Nathan

stephen.nathan@telcordia.com

An SAIC Company

RETURN TO INDEX

Disclaimer

- The purpose of this presentation is to provide a technical perspective of Telcordia's experience using IMS and MQSeries.
- Although this document addresses certain IBM products, no endorsement of IBM or its products is expressed, and none should be inferred.
- Telcordia also makes no recommendation regarding the use or purchase of IMS or MQSeries products, any other IBM products, or any similar or comparable products.
- Telcordia does not endorse any products or suppliers. Telcordia does not recommend the purchase or use of any products by the participants.
- Each participant should evaluate this material and the products himself/herself.

Telcordia Technologies
Performance from Experience

Acknowledgements

- Special thanks to:
  - Jack Yuan and his team from IBM IMS OTMA development for all the enhancements in PQ32402 and PQ35615 (IMS 7.1 = PQ33996) and all of the other enhancements in OTMA
  - John Jones and Bob Millar and their team at IBM Hursley Labs for their work on the MQSeries IMS Bridge and sharing their knowledge of how it really works
  - Gary Ward for sharing his Introduction to MQSeries foils

Telcordia Technologies

Performance from Experience

Presentation Outline

- Introduction to MQSeries
  - What is MQSeries?
  - MQSeries Queues
  - MQSeries Channels
  - MQSeries Clustering
  - MQSeries Shared Queues
  - Message Queuing Interface (MQI)
- IMS Application Access to MQSeries Messages
- Connecting MQSeries and IMS via ESS
- Using MQSeries API with IMS Applications
- MQSeries IMS Trigger Monitor
- IMS BMP Monitor
- IMS OTMA Interface
- MQSeries IMS Bridge
- IMS OTMA Security
- Sources of Documentation

Telcordia Technologies

Performance from Experience

Introduction

- Telcordia has had a great deal of experience using MQSeries and IMS
- We have learned a lot that is not clearly documented
- This presentation will share those experiences and document what we have learned

Telcordia Technologies
Performance from Experience

What is MQSeries?

- IBM software suite for commercial messaging
  - Industrial strength, function rich, scaleable, store and forward messaging infrastructure
- Messaging and queuing is a paradigm for asynchronous program-to-program communication
  - Other paradigms: conversational and remote procedure call
- It is the rest of the computing community learning what IMS people have known for 30 years
  - Message Queues are the best way to go

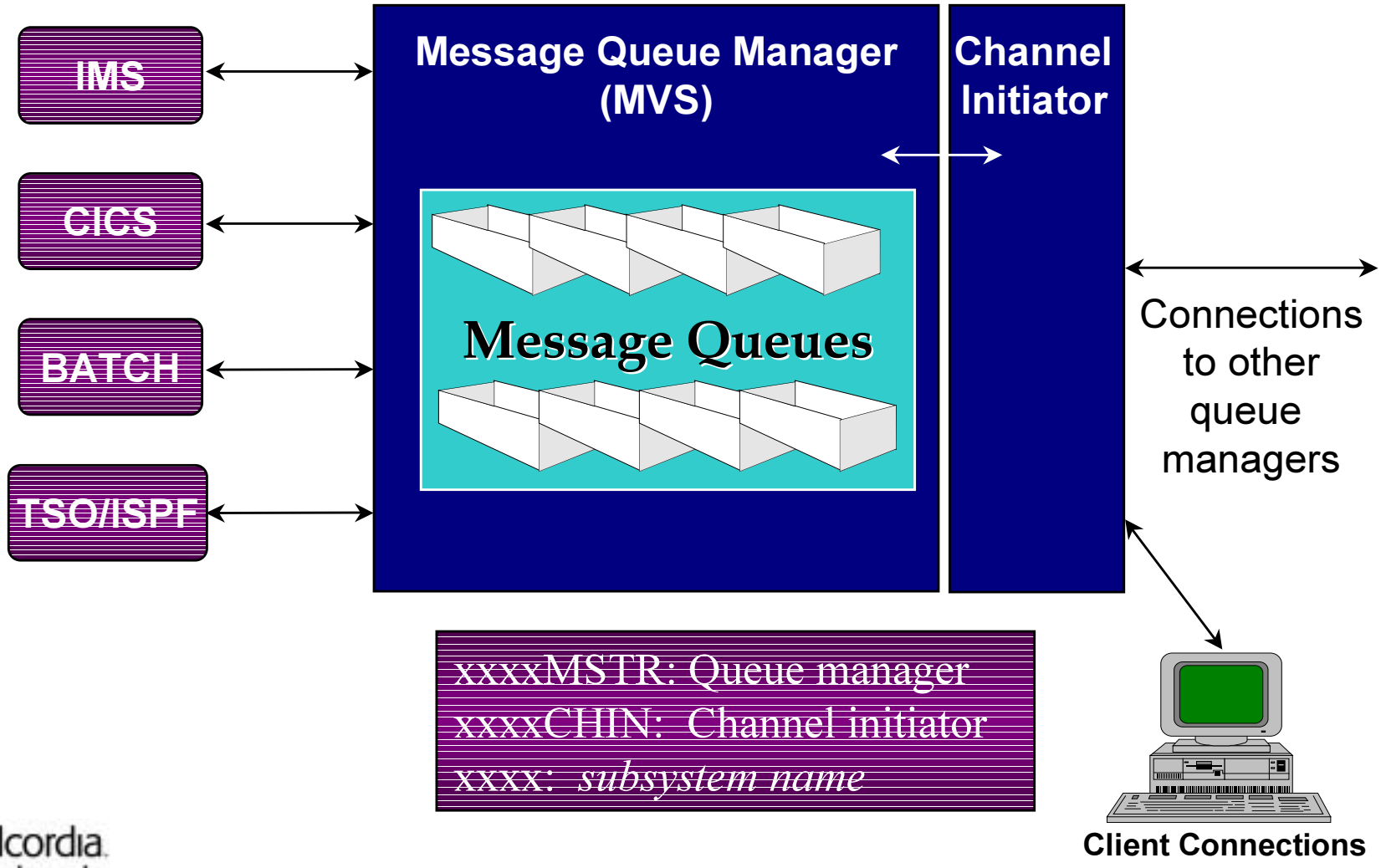
Telcordia Technologies
Performance from Experience

What is MQSeries?

- Provides key benefits
  - Once and only once, assured message delivery
  - Shields developer from network complexities
  - Enables portability of applications via consistent API
  - Allows for processing alternatives (time, location, parallelism)
- Currently supported on 35+ platforms
- Participates in coordinated sync points with other products (e.g., IMS, CICS, XA Compliant products)

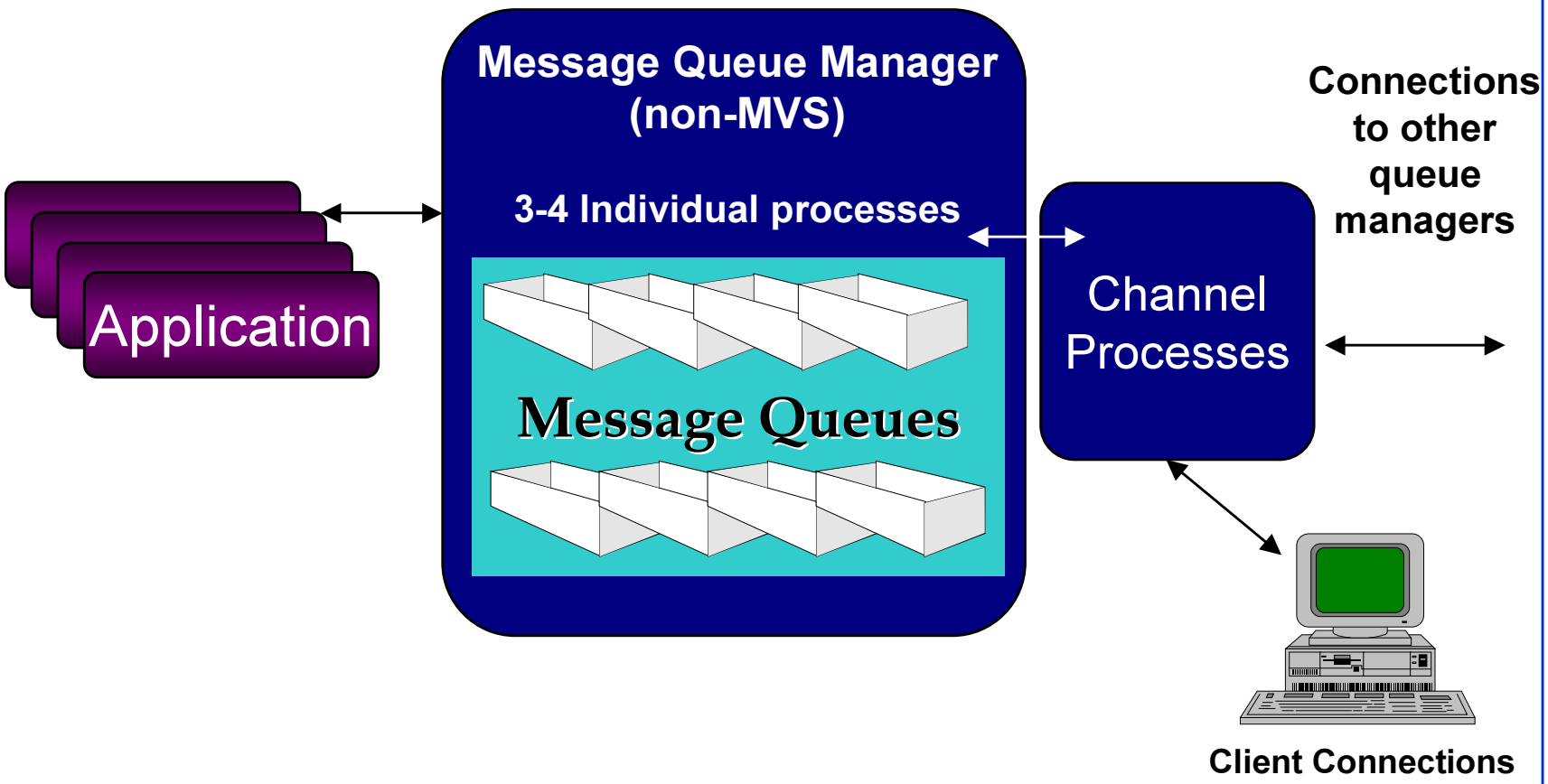
Telcordia Technologies
Performance from Experience

What is MQSeries?



Telcordia Technologies
Performance from Experience

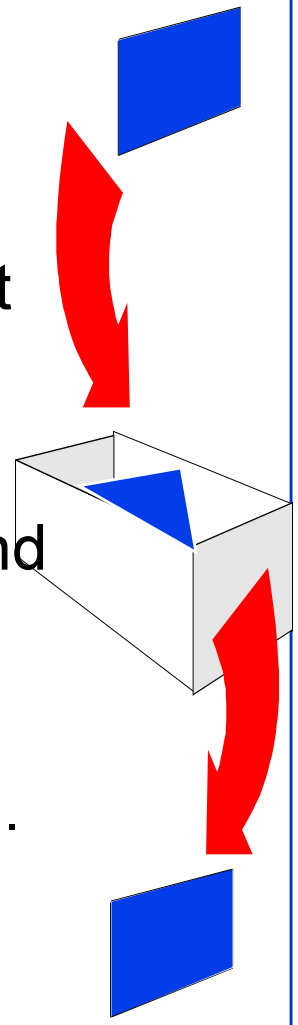
What is MQSeries?



Telcordia Technologies
Performance from Experience

MQSeries Queues

- A queue is the equivalent of an IMS or CICS transaction code (messages are enqueued to it and dequeued from it)
- Queues can be dynamically defined, altered and deleted (without a SYSGEN!)
- There are several types of MQSeries queues ...



Telcordia Technologies
Performance from Experience

MQSeries Queues

- Local Queues
  - Belong to the local MQSeries system
  - Messages can be enqueued to and dequeued from them
  - Default local queue attributes look like...

display queue(QUEUEA) all

DESCR(Test Queue)	MAXDEPTH(5000)	QDEPTHHI(80)
PROCESS( )	MAXMSGL(4194304)	QDEPTHLO(20)
BOQNAME( )	BOTHRESH(0)	QDPMAXEV(ENABLED)
INITQ( )	SHARE	QDPHIEV(DISABLED)
TRIGDATA( )	DEFSOPT(SHARED)	QDPLOEV(DISABLED)
QUEUE(QUEUEA)	NOHARDENBO	QSVCINT(999999999)
CRDATE(1998-09-09)	MSGDLVSQ(PRIORITY)	QSVCIEV(NONE)
CRTIME(10.35.45)	RETINTVL(999999999)	TYPE(QLOCAL)
GET(ENABLED)	USAGE(NORMAL)	DEFTYPE(PREDEFINED)
PUT(ENABLED)	**NOTRIGGER**	SCOPE(QMGR)
DEFPRTY(0)	**TRIGTYPE(FIRST)**	***IPPROCS(0)***
DEFPSIST(NO)	**TRIGDPTH(1)**	***OPPROCS(0)***
STGCLASS(DEFAULT)	**TRIGMPRI(0)**	***CURDEPTH(0)***

Telcordia Technologies

Performance from Experience

MQSeries Queues

- Notes on selected local queue attributes:
  - Default settings come from SYSTEM.DEFAULT.LOCAL.QUEUE
  - Persistence defines whether a message is recoverable across an MQSeries system restart
    - **After an IMS cold start (or lost logs), "IN-DOUBT" MQSeries messages must be resolved (see System Management Guide)**
    - **There is no COLD start facility for MQSeries**
  - If MAXDEPTH is exceeded, an appropriate return code is passed back to the application and the MQPUT fails
    - Messages go on the dead-letter queue if they arrive from another queue manager
  - Storage classes map queues to pagesets, but also designate the IMS Bridge interface

MQSeries Queues

- Notes on selected local queue attributes:
  - When debugging an application the following attributes should be your starting point:
    - GET - Can applications read from this queue?
    - PUT - Can applications write to this queue?
    - IPPROCS - Number of programs reading this queue
    - OPPROCS - Number of programs writing to this queue
    - CURDEPTH - The current number of messages on the queue
  - SHARE/NOSHARE specify if multiple applications can get messages from this queue
  - MSGDLVSQ can be set to FIFO or PRIORITY to change how messages are retrieved from the queue
  - Event switches (e.g., QDPHIEV) need to be enabled in order to allow certain system management products to be aware of critical conditions

MQSeries Queues

- Transmit queues
  - Used by channels to transport messages between queue managers
  - Same definition as a local queue except for USAGE(XMITQ)
- Initiation Queues
  - Hold trigger messages during triggering process
  - Just a plain local queue, but read by a trigger monitor program

Telcordia Technologies
Performance from Experience

MQSeries Queues

- Alias Queues
  - Redirect MQI calls to “real” queues
  - Useful for maintenance, migration, load-balancing, and security

```
display queue(ALIASQ) all
   SCOPE(QMGR)                  GET(ENABLED)
   DESCR(Alias for QUEUEA)      PUT(ENABLED)     DEFPRTY(0)
   TARGQ(QUEUEA)                TYPE(QALIAS)
   QUEUE(ALIASQ)                DEFPSIST(NO)
```

Telcordia Technologies
Performance from Experience

MQSeries Queues

- Model Queues
  - Templates of queue definitions used to create dynamic queues
  - Same definition as a local queue except
    - No SCOPE() parameter
    - Has DEFTYPE(PERMDYN/TEMPDYN)
- Dynamic Queues
  - Created by an application opening a Model Queue
  - Defined as permanent or temporary on MQOPEN call from DEFTYPE parameter of the model
    - Temporary (not recoverable, always deleted at MQCLOSE)
    - Permanent (recoverable, deleted only if MQCLOSE option for delete specified)
  - Useful for temporary work storage

Telcordia Technologies
Performance from Experience

MQSeries Queues

- Remote Queues
  - A logical queue definition for a local queue on another system
    - **Like an MSC remote transaction definition**
  - Messages can be written to but not read from this queue
  - The real message destination is the associated transmit queue

```
display queue(QUEUEB) all
   DESCR(Remote definition for QUEUEB)      PUT(ENABLED)
   RNAME(QUEUEB)                            DEFPRTY(0)
   RQMNAME(QMGRB)                           DEFPSIST(NO)
   XMITQ(QMGRA.TO.QMGRB)                    SCOPE(QMGR)
   QUEUE(QUEUEB)                            TYPE(QREMOTE)
```

Telcordia Technologies
Performance from Experience

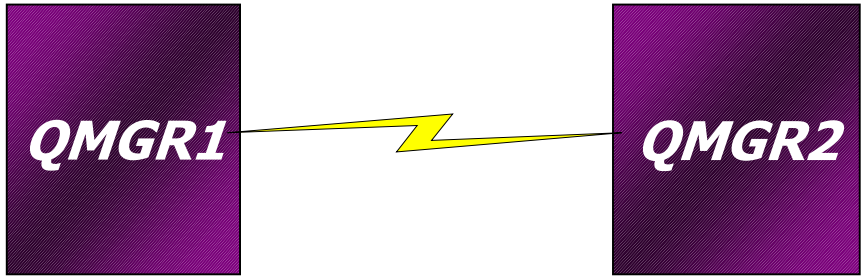
MQSeries Queues

- Dead-letter Queue
  - A local queue for holding messages which can not be delivered
    - Vendor tools are available to process this queue, or R.Y.O
- SYSTEM.COMMAND.INPUT Queue
  - A local queue for processing command strings
- SYSTEM.DEFAULT.* Queues
  - Queues which hold default settings for creating new queues (local, alias, remote, etc.)
  - **Be careful if you change the defaults!**
- SYSTEM.ADMIN.*.EVENT Queues
  - Contain event messages which reflect the status of MQSeries
    - Vendor tools are available to process these queues, or R.Y.O
- SYSTEM.CHANNEL.* Queues
  - Queues which are used for distributed queuing (number varies depending on platform)
- SYSTEM.CLUSTER.* Queues
  - Queues used for clustering operations

Telcordia Technologies
Performance from Experience

MQSeries Channels

- There are two types of MQSeries channels:
  - Message channels connect queue managers (our focus)
  - MQI channels connect MQSeries Clients to queue managers
- Message channels are used by processes called message channel agents (a.k.a., movers, MCAs)
- Provide uni-directional, point-to-point links between queue managers using a chosen protocol (TCP/IP, SNA, NetBIOS)
- Even though connections are point-to-point, multi-hop and hub configurations are also possible
- Class of service can also be set up using multiple channels between queue managers



Telcordia Technologies
Performance from Experience

MQSeries Channels

- There are four types of message channels:
  - Sender
    - Started by a system command
    - Starts a receiver
    - Re-starts a requester
    - Can be started by a requester
  - Receiver
    - Can only be started by a sender
  - Server
    - Started by a system command
    - Can start a requester
    - Can be started by a requester
  - Requester
    - Can be started by a system command
    - Can start a server or a sender

Telcordia Technologies
Performance from Experience

MQSeries Channels

display channel(QMGRA.TO.QMGRB) all

CHANNEL(QMGRA.TO.QMGRB)
CHLTYPE(SDR)
TRPTYPE(TCP)
DESCR(Sender channel to QMGRB)
XMITQ(QMGRA.TO.QMGRB)
MCANAME()
MODENAME()
TPNAME()
BATCHSZ(50)
DISCINT(6000)
SHORTRTY(10)
SHORTTMR(60)
LONGRTY(999999999)
LONGTMR(1200)
HBINT(300)
NPMSPEED(FAST)
SCYEXIT()
MSGEXIT()
SENDEXIT()
RCVEXIT()
SEQWRAP(999999999)
MAXMSGL(4194304)
CONVERT(NO)
SCYDATA()
MSGDATA()
SENDDATA()
RCVDATA()
USERID()
PASSWORD()
MCAUSER()
MCATYPE(PROCESS)
BATCHINT(0)
CONNAME(111.222.111.222(1414))

Telcordia Technologies
Performance from Experience

MQSeries Channels

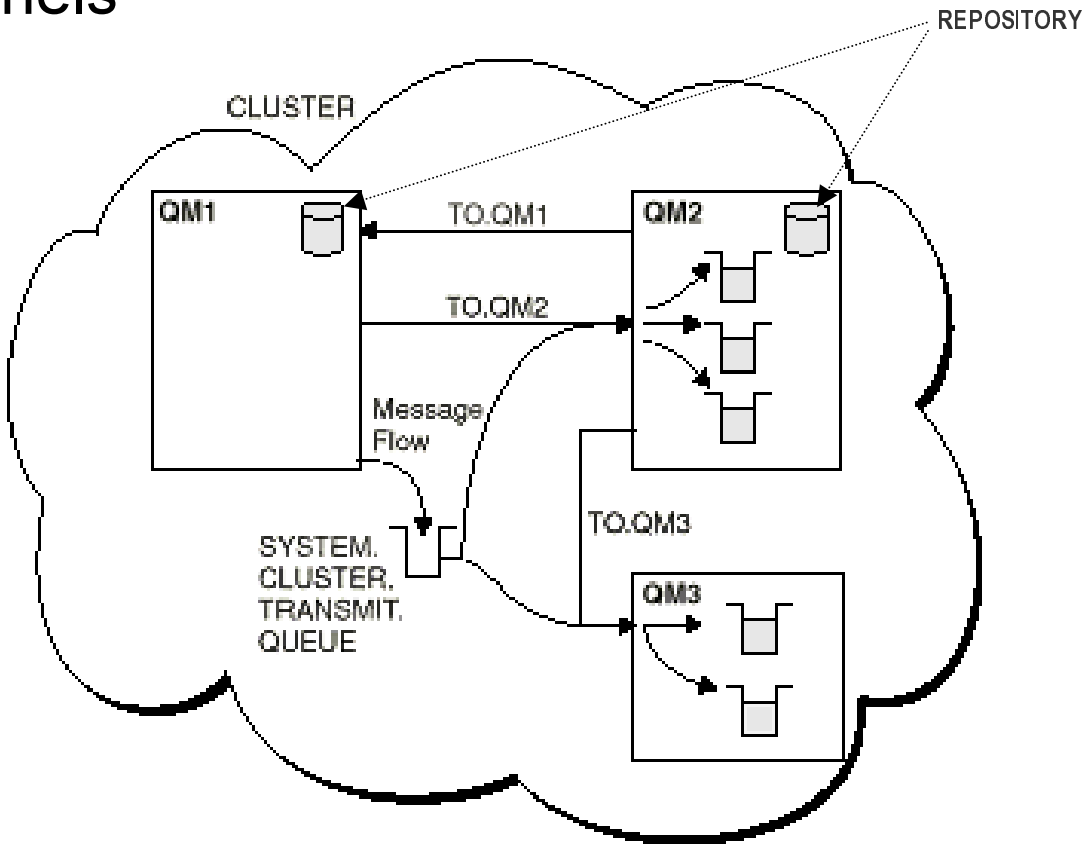
display channel(QMGRA.TO.QMGRB) all

CHANNEL(QMGRA.TO.QMGRB)
CHLTYPE(RCVR)
TRPTYPE(TCP)
DESCR(Receiver channel for QMGRA)
BATCHSZ(50)
SCYEXIT()
MSGEXIT()
SENDEXIT()
RCVEXIT()
SEQWRAP(999999999)
MAXMSGL(4194304)
PUTAUT(DEF)
SCYDATA()
MSGDATA()
SENDDATA()
RCVDATA()
MCAUSER()
MREXIT()
MRDATA()
MRRTY(10)
MRTMR(1000)
NPMSPEED(FAST)
HBINT(300)

Telcordia Technologies
Performance from Experience

MQSeries Clustering

- A new architecture which allows queue managers to dynamically share workload through cluster queues and channels



Telcordia Technologies
Performance from Experience

MQSeries Clustering

- Reduced administration requirements
  - Queue managers create connections to other queue managers in the cluster dynamically using a directory service called a repository
    - Removes the need to pre-define remote queues, transmit queues, channels for the queue managers within the cluster
- Higher availability and workload balancing
  - Sending queue managers are informed of the available destinations (failure isolation and graceful degradation)
  - Round-robin algorithm (or your own) spreads the message traffic
- Application simplicity
  - No “master” queue manager
  - Multiple queues with a single image
  - MQGET is always local
- Available in MQSeries 5.1 (distributed systems) and MQSeries 2.1 (OS/390)

Telcordia Technologies

Performance from Experience

MQSeries Shared Queues

- Feature available in MQSeries for OS/390 5.2
- Function
  - Multiple queue managers can access the same shared queues
  - Local (non-shared) queues work as in version 2.1
- Benefits
  - Availability for new messages
  - Availability for old messages
  - Pull workload balancing
  - Scaleable capacity
  - Low cost messaging within a sysplex
  - Easier administration
    - Group Objects, Command Scope, New Security Profiles

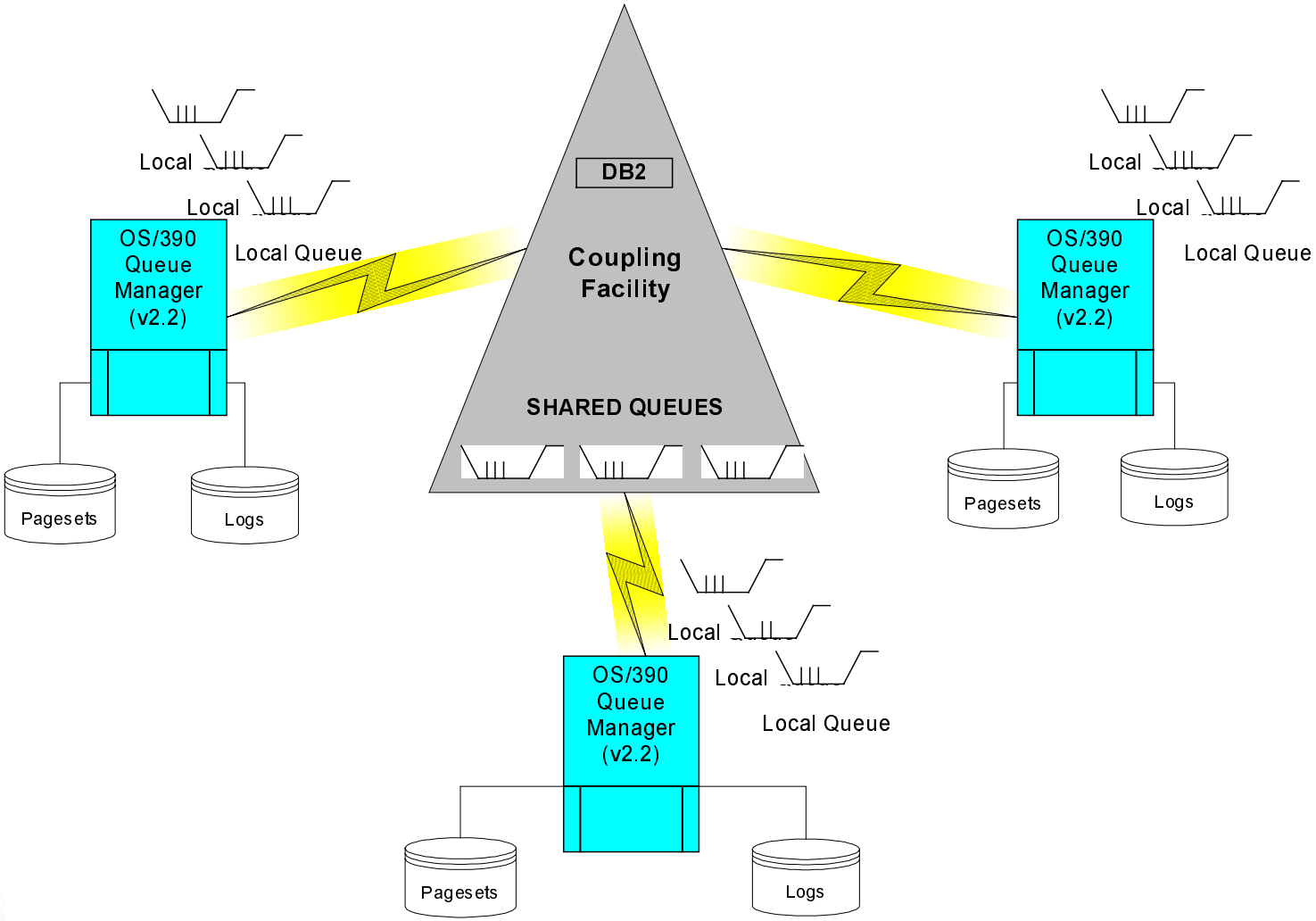
Telcordia Technologies
Performance from Experience

MQSeries Shared Queues

- Prerequisites
  - OS/ 390 2.9 (2.6 if no shared queues)
  - Coupling Facility level 9
  - DB2 5.1 or later
- Constraints
  - Non-persistent messages only in 5.2
  - Maximum message size 63 KB (including headers)
  - Maximum of 8 million messages per CF structure
  - Maximum of 512 shared queues per CF structure
  - Maximum of 512 CF structures per sysplex
  - Maximum of 32 active QMgrs per QSG

Telcordia Technologies
Performance from Experience

MQSeries Shared Queues



Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

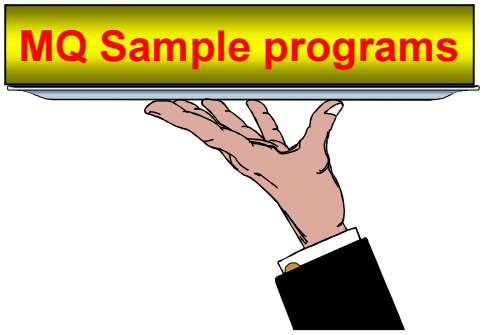
- An MQSeries message is made up of two components:

- On most platforms the current maximum size for a message is 100MB
- Part of the message header is the message descriptor which contains information such as:
  - Message priority, persistence
  - MsgId, CorrelId (can be used to pick specific messages out of a queue)
  - Reply-to information
  - Format
  - Report requests (confirm arrival/delivery, message expired, exception)
- There are also other headers:
  - Transmit queue header
  - Dead-letter queue header

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- Coding in an MQSeries environment is very easy
- The MQSeries Application Programming Guide and the MQSeries Application Programming Reference are the only manuals you will need
- Application programming can be done in C/C++, Java, COBOL, Assembler, REXX, PL/I, Smalltalk among others…
- Currently there are only 12 MQSeries verbs in the API (and fewer in the OO languages)
- All calls return a completion code and a reason code
- Sample programs make this a breeze!



Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQCONN
  - Used to connect to an MQSeries subsystem
  - Can use a known queue manager name or blanks for the system default queue manager (IMS only)
  - A program can only be connected to one queue manager at a time (unless it is an IMS program)

```
strncpy(QMgr,"QM1",MQ_Q_MGR_NAME_LENGTH);
MQCONN(QMgr,&Hconn,&CompCode,&Reason);
printf("\n MQCONN -  CC: %d , Reason: %d", CompCode, Reason);
```

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQOPEN
  - Used to access MQSeries objects for subsequent processing

```
strncpy(ObjDesc.ObjectName,"TESTQ",MQ_Q_NAME_LENGTH);
strncpy(ObjDesc.ObjectQMgrName,QMgr,MQ_Q_MGR_NAME_LENGTH);
OpenOptions=  MQOO_OUTPUT +              /* For Writing */
              MQOO_INPUT_SHARED +        /* For Reading */
              MQOO_INQUIRE +             /* For MQINQ */
              MQOO_FAIL_IF_QUIESCING;    /* Qmgr shutting down? */
MQOPEN(Hconn,&ObjDesc,OpenOptions,&Hobj,&CompCode,&Reason);
printf("\n MQOPEN -  CC: %d , Reason: %d", CompCode, Reason);
```

  - Other open options include:
    - MQOO_INPUT_AS_Q_DEF (as the queue default specifies)
    - MQOO_BROWSE
    - MQOO_SET

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQGET
  - Retrieves a message from a queue based on specified options

```
GetMsgOpts.Options = MQGMO_NO_WAIT     +
                                MQGMO_SYNCPOINT;
MQGET(Hconn, Hobj, &MsgDesc, &GetMsgOpts, BufferLength, Buffer,
            &DataLength, &CompCode, &Reason);
printf("\n MQGET from queue %s -  CC: %d , Reason: %d",
            ObjDesc.ObjectName,CompCode,Reason);
printf("\nInput Message: %s",Buffer);
strncpy(replyq,MsgDesc.ReplyToQ,6);
strncpy(replyqmn,MsgDesc.ReplyToQMgr,6);
memcpy(MsgDesc.MsgId,MQMI_NONE,sizeof(MQBYTE24));
memcpy(MsgDesc.CorrelId,MQCI_NONE,sizeof(MQBYTE24));
```

  - MQGMO_NO_WAIT (do not wait if no message available)
  - MQGMO_ACCEPT_TRUNCATED MESSAGE gives warning completion code if message does not fit in Buffer and removes the message (be careful!)
    - Unlike IMS which does not ask and just overlays anything past your I/O area

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQGET
  - Other MQGET options include:
    - MQGMO_WAIT (used with MQGMO.WaitInterval) which will wait for a message to arrive if none are available
    - MQGMO_SET_SIGNAL which allows a program to wait on more than one queue
      - MVS only... posts an ECB when a message is available
      - Both of these are like IMS WFI
    - MQGMO_SYNCPOINT returns an input message to the queue if an abend or backout occurs
      - Unlike IMS which always discards the message if the abend is after the GU to the IOPCB
    - MQGMO_BROWSE_FIRST (NEXT, MSG_UNDER_CURSOR)
    - MQGMO_CONVERT will perform codepage translation for the message data (if the MQMD.Format=MQFMT_STRING)

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQPUT
  - Puts a message onto a queue using specified options

```
pBuff = (void *) malloc(BUFFERLENGTH * sizeof(char));
  sprintf(pBuff,"Hello World");
  Buff_len = strlen(pBuff);
  Buffer[BUFFERLENGTH]= '\0';
  PutMsgOpts.Options =          MQPMO_SYNCPOINT +
                                MQPMO_FAIL_IF_QUIESCING;
  MQPUT(Hconn,Hobj,&MsgDesc,&PutMsgOpts,
                    Buff_len,pBuff,&CompCode,&Reason);
  printf("\n MQPUT -  CC: %d , Reason: %d", CompCode, Reason);
```

  - MQPMO_SYNCPOINT specifies that the message(s) are within the current unit of work (UOW)
  - MQPMO_NO_SYNCPOINT puts the message outside the current UOW which makes it instantly available
    - Like messages to an IMS EXPRESS ALTPCB

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQCLOSE
  - Closes the “connection” to a specified object

```
CloseOptions= MQCO_NONE;
MQCLOSE(Hconn,&Hobj,CloseOptions,&CompCode,&Reason);
```

  - MQCO_DELETE and DELETE_PURGE are used for dynamic queues

- MQDISC
  - Relinquishes the connection to a queue manager

```
MQDISC(&Hconn,&CompCode,&Reason);
```

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQPUT1
  - Puts a single message onto a queue using specified options (no need to use MQOPEN and MQCLOSE)
- MQINQ
  - Retrieves characteristics of an MQSeries object or the queue manager itself (but not channels)
- MQSET
  - Sets the characteristics of an MQSeries queue

Telcordia Technologies
Performance from Experience

Message Queuing Interface (MQI)

- MQCMIT
  - Commits the current unit of work (not under IMS MPP/BMP)
- MQBACK
  - Backs out the current unit of work (not under IMS MPP/BMP)
- MQBEGIN
  - Begins a unit of work
  - This verb is used on non-MVS platforms where MQSeries is the XA-compliant resource coordinator
    - Example: MQSeries coordinates 2-phase commit between MQSeries messages (in syncpoint) and database updates

Telcordia Technologies
Performance from Experience

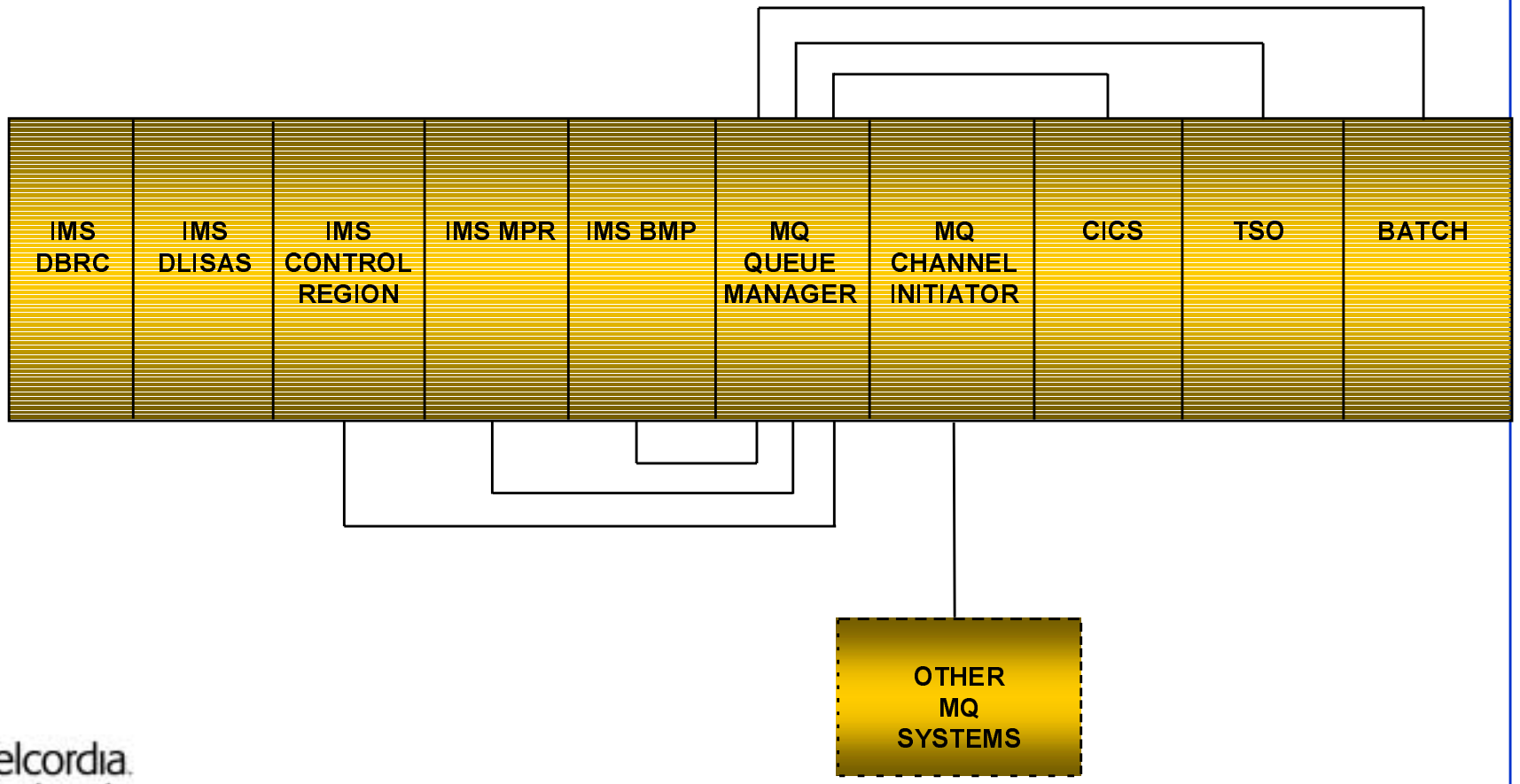
IMS Application Access to MQSeries Messages

- IMS applications can access MQSeries messages in two ways:
  1. The IMS application uses the MQSeries API to Get and Put messages with syncpoint coordination with IMS
     - Requires connecting MQSeries to IMS via ESS
     - IMS BMP
     - IMS MPP (IFP too)
       - Standard IMS transaction can issue MQSeries API calls
       - MQSeries message can be inserted to the IMS Message Queue by an application program (BMP/MPP)
         - Could be a Trigger message
         - Could be the real Message
     - IMS Batch
       - No syncpoint coordination or ESS interface
  2. The MQSeries IMS Bridge puts the message on the IMS Message Queue via OTMA
     - Does not require connecting MQSeries to IMS via ESS
       - But the connection could exist for programs using the MQSeries API
     - Requires OTMA configuration in MQSeries CSQZPARM

Telcordia Technologies
Performance from Experience

Connecting MQSeries and IMS via ESS

- MQSeries for OS/390 attaches to IMS just like DB2 using the external subsystem (ESS) interface



Telcordia Technologies
Performance from Experience

Connecting MQSeries and IMS via ESS

- Define MQSeries to IMS by adding ESS information to the IMS PROCLIB SSM member

  FORMAT: SST=,SSN=,LIT=,ESMT=,RTT=,REO=,CRC=
  - SST: Subsystem Type - “MQS”
  - SSN: Subsystem Name - MQSeries for MVS/ESA subsystem
  - LIT: Language Interface Token - See CSQQDEFV
  - ESMT: External Subsystem Module Table - “CSQQESMT”
  - RTT: Resource Translation Table -Not Used by MQSeries
  - REO: Region Error Option - “R”, “Q”, or “A”
  - CRC: Subsystem Recognition Character - Not Used by MQSeries
    - The /SSR command is not supported

Telcordia Technologies
Performance from Experience

Connecting MQSeries and IMS via ESS

- The IMS Control Region SSM member name in the IMS PROCLIB is xxxxyyyy
  - xxxx = IMS subsystem name
  - yyyy = suffix
- The suffix is defined in the SSM= parameter on the DFSPBxxx member in the IMS PROCLIB
- An SSM parameter can also be in the dependent region PARM
  - If it is not specified the dependent region has access to all external subsystems in the IMS Control Region SSM member
  - A subset of external subsystems could be specified in a different SSM member used for the dependent region
    - Then it only has access to the subset

Telcordia Technologies
Performance from Experience

Connecting MQSeries and IMS via ESS

- Place the MQSeries authorized library (HLQ.SCSQAUTH) in the IMS control region and dependent region STEPLIB and DFSESL concatenations
- Copy module CSQQDEFV from HLQ.SCSQASMS to be customized, assembled, and linked into an authorized library in the IMS control region STEPLIB concatenation
  - The LIT= values must equal the values in the IMS SSM member

```
CSQQDEFV CSECT
         CSQQDEFX NAME=CSQ1,LIT=LIT1,TYPE=DEFAULT
         CSQQDEFX NAME=CSQ3,LIT=LIT2
         CSQQDEFX TYPE=END
```

Telcordia Technologies
Performance from Experience

Connecting MQSeries and IMS via ESS

- Subsystem Connection

```
/DIS SUBSYS ALL

SUBSYS   CRC  REGID  PROGRAM  LTERM     STATUS
CSQ3     !                              CONN
CSQ1     <                              CONN
DB2R     =                              CONN
                  1                     CONN
                  5                     CONN
```

Telcordia Technologies
Performance from Experience

Connecting MQSeries and IMS via ESS

- The IMS application must be linked with the proper MQSeries application interface stub
- For MPP and BMP programs it is CSQQSTUB
- For IMS batch programs it is CSQBSTUB
- This means that if the program is going to run as a BMP sometimes and as IMS batch sometimes that there must be two versions of the load module

Telcordia Technologies
Performance from Experience

Using MQSeries API with IMS Applications

- Calls to MQSeries, IMS and DB2 can be made within the same unit of work (UOW)
  - MQSeries API calls
  - IMS IOPCB calls
  - IMS alternate output PCB calls
  - IMS database calls
  - DB2 calls
- An IMS commit is also an MQSeries and DB2 commit
  - SYNC, CHKP, GU to IOPCB (MODE=SNGL), normal program termination
- An IMS backout (ROLB) is also an MQSeries and DB2 backout
- Any IMS abend is also an MQSeries and DB2 backout
  - ROLL, miscellaneous abends

Telcordia Technologies
Performance from Experience

Using MQSeries API with IMS Applications

- At normal syncpoint....
  - IMS input message is dequeued
  - MQSeries input messages marked with SYNCPOINT, and MARK_SKIP BACKOUT are dequeued
  - MQSeries input messages marked with NO_SYNCPOINT have already been dequeued
  - IMS NON-EXPRESS output messages are sent
  - IMS EXPRESS output messages have already been sent
  - MQSeries output messages marked with SYNCPOINT are sent
  - MQSeries output messages marked with NO_SYNCPOINT have already been sent
  - IMS database updates are committed
  - DB2 updates are committed

Telcordia Technologies
Performance from Experience

Using MQSeries API with IMS Applications

- At normal syncpoint....
  - If the IMS application is message driven (BMP or MPP) the MQSeries queues are closed and the connection handle is closed
    - For security reasons
      - Connection security is by userid
      - Each message can be from a different userid
  - This means that you must issue MQCONN and MQOPEN's for every IMS message
    - This can cause a great deal of overhead
    - 99% of this overhead can be eliminated by IMS preloading all of the CSQQxxxx modules in the HLQ.SCSQAUTH library into the IMS dependent region
      - All of the modules are reentrant
      - 50K above the line
      - 11K below the line
  - IBM recommends putting the HLQ.SCSQAUTH into LLA to get a similar performance improvement for IMS Batch and non-IMS MQSeries programs

Telcordia Technologies
Performance from Experience

Using MQSeries API with IMS Applications

- At abnormal termination or ROLx....
  - IMS input message is dequeued
    - IMS 6.1 has Non-Discardable Message Exit
  - MQSeries input messages marked with SYNCPOINT are re-queued
  - MQSeries input messages marked with NO_SYNCPOINT have already been dequeued
  - MQSeries input messages marked with MARK_SKIP_BACKOUT are
    - Passed to the next UOW if this was the first UOW if ROLB
    - Re-queued if this was the second UOW or if ABEND
  - IMS NON-EXPRESS messages are discarded
  - IMS EXPRESS output messages have already been sent
  - MQSeries output messages marked with SYNCPOINT are discarded
  - MQSeries NO_SYNCPOINT output messages have already been sent
  - IMS database updates are backed out
  - DB2 updates are backed out

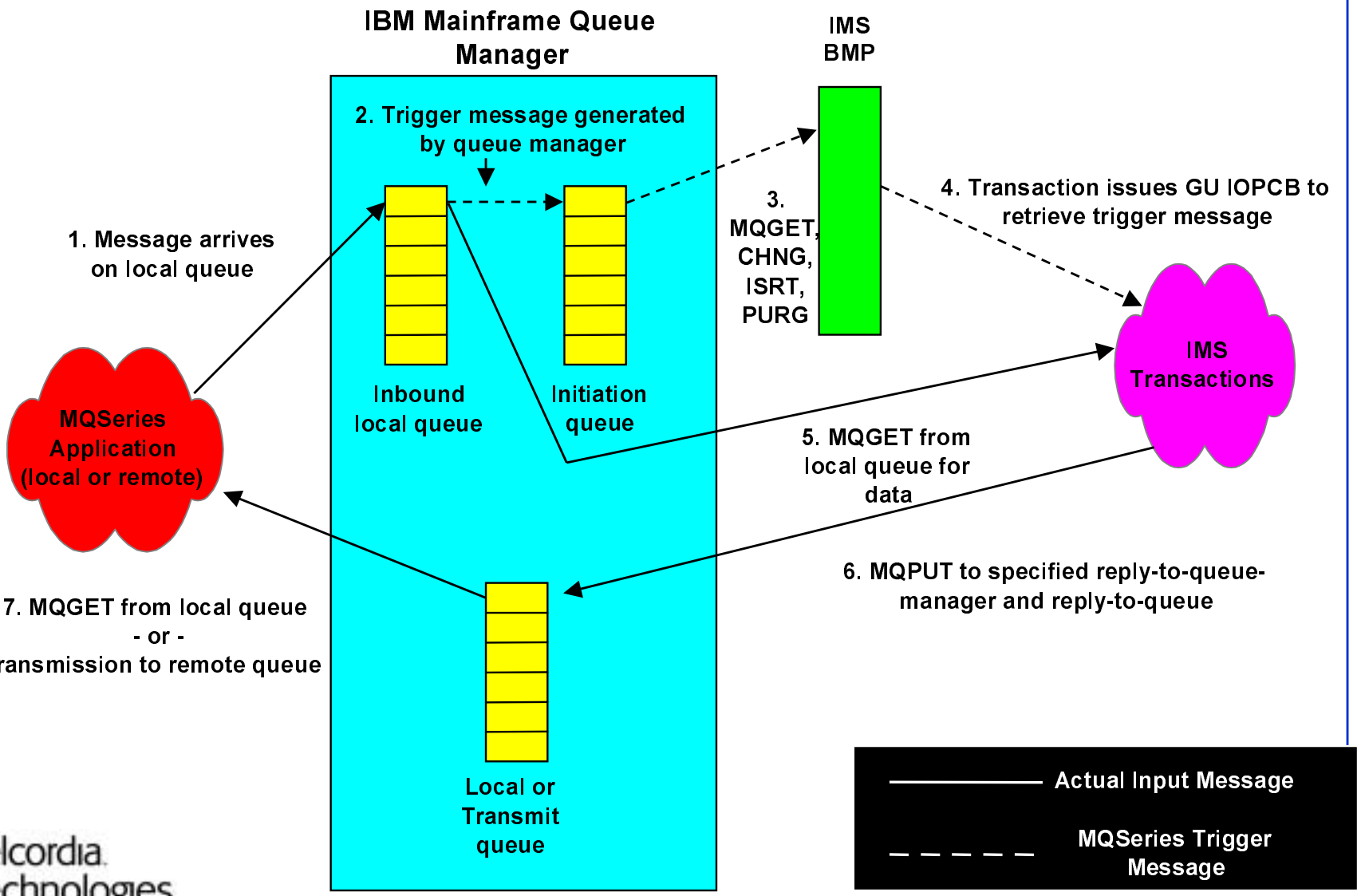
Telcordia Technologies
Performance from Experience

Using MQSeries API with IMS Applications

- Getting the default queue manager name is not straightforward...
  - MQCONN using default name (blank)
  - MQOPEN the Queue Manager
    - MQOD Objectype = MQOD_Q_MGR
    - MQOD Objectname = blanks
    - MQOO_INQUIRE
  - MQINQ for object name

Telcordia Technologies
Performance from Experience

MQSeries IMS Trigger Monitor



Telcordia Technologies
Performance from Experience

MQSeries IMS Trigger Monitor

- IMS triggering flow - MQSeries

❶A message arrives on an MQSeries local queue

❷A “trigger message” is sent to an MQSeries initiation queue

❸A long-running non-message driven BMP waiting on the initiation queue reads the trigger message and inserts it to the IMS trancode named in the MQSeries process definition

  - This is an IBM supplied trigger monitor application (CSQQTRMN)

Telcordia Technologies

Performance from Experience

MQSeries IMS Trigger Monitor

- IMS triggering flow - IMS

❹The IMS transaction is scheduled and does a GU to the IOPCB, retrieving the trigger message

- The trigger message identifies the MQSeries queue manager and local queue which caused the trigger to be pulled

❺The IMS transaction connects to the queue manager and reads the message from the MQSeries local queue via MQGET

❻The IMS transaction responds by inserting a message to an MQSeries queue via MQPUT

- The MQSeries message descriptor identifies a reply-to queue manager and queue

Telcordia Technologies
Performance from Experience

MQSeries IMS Trigger Monitor

- IMS triggering definitions
  - Define the MQSeries local queue
    - Include the process name attribute
    - Include the initiation queue attribute
    - Include the trigger attributes
  - Define the process definition
    - APPLTYPE is IMS
    - APPLID is the IMS transaction code
  - Define the MQSeries local queue to be used as the initiation queue

Telcordia Technologies
Performance from Experience

MQSeries IMS Trigger Monitor

- More IMS triggering definitions...
  - Define the IMS non-message driven BMP CSQQTRMN in the IMS sysgen
    - See member CSQQTAPL in HLQ.SCSQPROC
    - The GPSB option can be used
  - If not GPSB build the CSQQTRMN PSB
    - See member CSQQTPSB in HLQ.SCSQPROC
    - Run PSBGEN
    - Run ACBGEN
  - Create one BMP for each initiation queue to be monitored
    - See member CSQ4ICBR in HLQ.SCSQPROC

```
//CSQQUT1  DD  *
QMGRNAME=queue manager name
INITQUEUENAME=initiation queue name
LTERM=IMS LTERM for error messages
/*
```

Telcordia Technologies
Performance from Experience

MQSeries IMS Trigger Monitor

- Last two IMS triggering definitions...
  - Define the IMS application in the IMS sysgen
    - NONRESPONSE
  - Build the IMS application transaction
    - MQSeries Calls
    - IMS Calls

Telcordia Technologies
Performance from Experience

IMS BMP Monitor

- User written IMS BMP monitor program
  - It is easy to write your own IMS BMP monitor
  - Telcordia developed one before the MQSeries IMS Trigger Monitor program was available
  - One enhancement could be to have one BMP wait on multiple queues using the MQGMO_SET_SIGNAL option (ECB)
  - Here's how to do it:
    - Write IMS BMP with MQI calls and an alternate output modifiable PCB (ALTPCB)
    - Wait on MQSeries initiation queue(s) or local queue(s)
      - Initiation queues hold trigger messages; local queues hold "real" messages
    - Insert messages to IMS ALTPCB

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- IMS/ESA 5.1 introduced the OTMA (Open Transaction Manager Access) feature
- This feature uses the MVS cross-coupling facility (XCF) to send data to IMS from other MVS applications (OTMA clients)
  - No VTAM or TCP/IP is involved

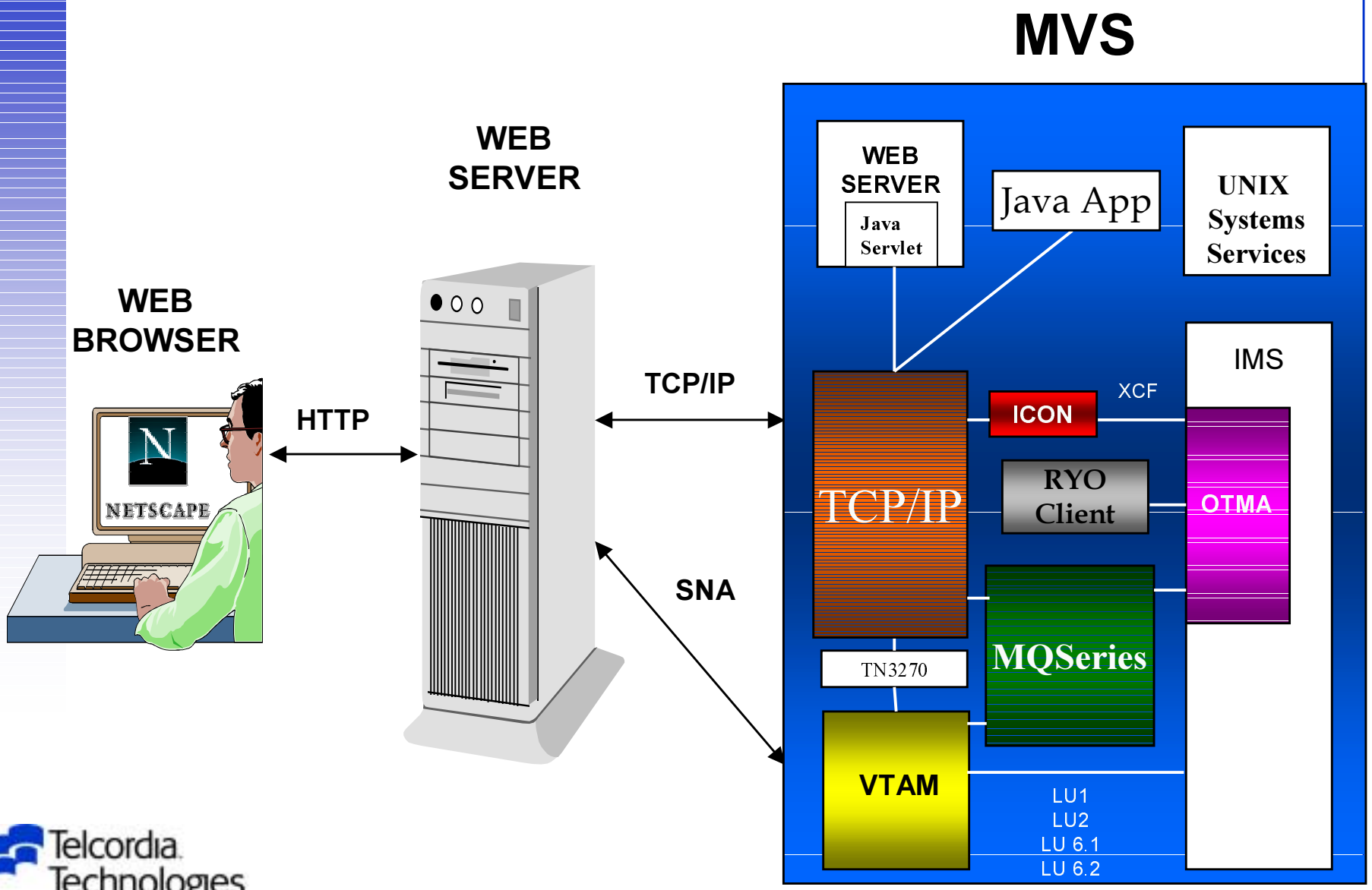
Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- IMS Connect is an IBM provided OTMA client for TCP/IP
  - This was the IMS TCP/IP OTMA Connection (ITOC)
- MQSeries includes an IMS OTMA client
  - "MQSeries-IMS Bridge"
- You can write your own OTMA client using the OTMA Callable Interface

Telcordia Technologies
Performance from Experience

IMS OTMA Interface



Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- Define IMS OTMA parameters in the IMS Control Region JCL or DFSPBxxx
  - OTMA=Y/N (autostart OTMA)
  - GRNAME=XCF group of IMS Control Region
  - APPLID1=XCF member name of IMS Control Region
  - OTMANM=XCF member name of IMS - APAR PQ12917
  - USERVAR=XCF member name of IMS - only if RSR or XRF
  - APPCASY=Y/N - APARS PQ17309 & PQ19330
    - This now has nothing to do with OTMA - it is APPC only
  - OTMAMD=Y/N - APAR PQ32402
    - Allow DFSYPRX0 to override OTMA output client if OTMA input
  - OTMASP=Y/N - APAR PQ32402
    - Set the default output TPIPE to SYNChronized if no DRU exit
  - OTMADB= (Never Code!!!!! – more later)
  - OTMASE=C,F,N,P - APAR PQ30626 & PQ35615
    - Provide default OTMA security level

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- Define OTMA Destination Resolution Exit name for each client
  - Member DFSYDTx in IMS.PROCLIB
  - Can be overridden by OTMA client
- Set OTMA security via command
  - /SEC OTMA xxxx (NONE, PROFILE, CHECK, FULL)
- Start (and stop) OTMA via command
  - /STA OTMA (like /STA DC)
  - /STO OTMA (like /STO DC)
- There are no SYSGEN requirements for OTMA

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- TPIPEs (Transaction Pipes)
  - OTMA equivalent of LTERMs
  - Input TPIPE names are specified by the client
  - Asynchronous output (ALTPCB) TPIPE names
    - The destination in the CHNG call for modifiable ALTPCB's
    - The destination name in static ALTPCB's
    - Can be overridden by the DFSYDRU0 exit
  - Control blocks are dynamically created by IMS
  - TPIPE name must be unique only within a client (multiple clients can use the same TPIPE name)
    - IMS recognizes TPIPEs as **XCFmember**.**TPIPEname**
  - A TPIPE name cannot be the same name as an IMS transaction
  - Client manages the use of TPIPEs (number, routing, etc.)

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- TPIPEs
  - There are two types of TPIPEs: SYNChronized and Non-SYNChronized
    - SYNChronized TPIPEs exchange sequence numbers and can be resynchronized across failures
    - SYNChronized TPIPEs will survive an IMS warm start but will be deleted during an IMS cold start
    - Messages on SYNChronized TPIPEs are considered "recoverable" by MQSeries
    - Non-SYNChronized TPIPEs will be deleted during an IMS warm start
  - Client specifies the type for each TPIPE
  - For ALTPCB output from IMS creating a new TPIPE
    - Default is Non-SYNChronized
    - Default can be changed with OTMASP parameter in DFSPBxxx
    - Can be overridden by DFSYDRU0 exit

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- OTMA Commit Mode
  - Specified by the OTMA client for each message
  - Determines how output messages are sent
  - Similar to APPC-IMS Commit Mode
  - Commit Mode 0 - Commit-then-send
    - Output is queued
    - IMS sends output after syncpoint is complete
    - OTMA requires ACK to dequeue message
    - Queued on a control block called a QAB
  - Commit Mode 1 - Send-then-commit
    - IMS sends output and may wait for ACK before syncpoint is complete
      - Synch level 0 (None) – no ACK
      - Synch level 1 (Confirm) – ACK
      - Set in input message OTMA prefix
    - Increases region occupancy
    - Messages are not queued
    - Transaction abends with U119 and backs out if message can not be sent or is NAKed
    - “Queued” on a control block called a TIB

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- OTMA Commit Mode
  - ALTPCB output is always Commit Mode 0
  - Message switching a CM1 input message to a NONRESPONSE transaction will result in CM0 output
  - Message switching a CM1 input message to a RESPONSE transaction may also result in CM0 output
    - If the ALTPCB is EXPRESS the ISRT takes place immediately
    - The second transaction may be scheduled before the first one is complete
    - If that happens the CM1 TPIPE control block (TIB) is still owned by the first transaction
    - This causes the output messages from the second transaction to be converted to CM0
    - One way to avoid this is to not make the ALTPCB EXPRESS
  - CM1 input message which does ISRT - PURG - ISRT - PURG - ISRT - PURG to IOPCB will generate one multi-segment message instead of 3 single segment messages

IMS OTMA Interface

- Commit mode has an effect on IMS Response Mode
  - If the message is sent with Commit Mode 1 (send-then-commit) then IMS will treat the transaction as RESPONSE mode even if it is defined as NONRESPONSE
  - If the application does not respond to the IOPCB IMS will respond with a DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY message
  - This applies even if there are message switches before ending
  - You must apply PQ23213 or client can hang

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

```
/DIS OTMA
GROUP/MEMBER  XCF-STATUS  USER-STATUS          SECURITY
   OTMAZZS
   -ZZS           ACTIVE       SERVER               NONE
   -CSQ2M         ACTIVE       ACCEPT TRAFFIC
   -ITOCA         ACTIVE       ACCEPT TRAFFIC
```

```
/DIS TMEMBER CSQ2M TPIPE ALL
MEMBER/TPIPE   ENQCT  DEQCT    QCT          STATUS
    CSQ2M
    CSQ00035      1       1       0          SYN
    CSQSTUFF      3       3       0          SYN
    CM1TPIPE      0       0       0
```

- CM1 TPIPE counts are always 0

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- Use IMS TPIPE trace for debugging
  - /TRA SET ON TMEMBER xxxx TPIPE yyyy
  - Produces x'6701' IMS log records
  - Format with DFSERA10 and DFSERA30
- Diagnosis and Reference Guide needs updating
- OTMA headers start at offset x'50' in the MSG PREF section
  - They are truncated - IBM will fix - sometime
- ACK's and NAK's are not traced - in or out

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- Message text is in the I/O BUFF section
  - Only the first segment of a multi-segment message is traced
  - APAR PQ25881 identifies input and output message by the ID= in the trace record
    - TIB0 - Input message before DFSYIOE0
    - TIB2 - Input message after DFSYIOE0
    - QAB0 - CM0 output message before DFSYIOE0
    - QAB2 - CM0 output message after DFSYIOE0
    - SLM0 - CM1 output message before DFSYIOE0
    - SLM2 - CM1 output message after DFSYIOE0

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- /TRA SET ON TABLE OTMT (not a typo) is an internal trace for IBM use only
  - It does not produce enough information for IBM to debug
- You can specify OTMADB=Y in your DFSPBxxx to get more trace data for IBM
  - **DO NOT!!!**
  - It will flood your MVS console with WTO's
  - It can only be removed by recycling IMS

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- The OTMA client communicates information to IMS and gets information from IMS in a prefix passed in front of the message
  - The prefix has 4 sections mapped by macro DFSYMSG:
    - **Control**: TPIPE name and type, message type, chaining, etc.
    - **State Data**: Commit mode, IOPCB LTERM and MODNAME override, etc.
    - **Security**: Security scope, userid, RACF group, Utoken
    - **User**: Client specific
      - Saved on input (in the IMS Message Prefix)
      - Passed back on output
      - ICON shares with user
      - MQSeries passes MQMD (message descriptor)
      - OTMA/CI client can use for its own purposes - APAR PQ32398
      - Accessible by OTMA user exits

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- IMS Output Messages - WARNING!!!
  - All messages from OTMA originated transactions inserted to an alternate output PCB return down the incoming TPIPE by default (except for program-to-program message switches)
  - Alternate destinations can be reached by using the OTMA routing exits DFSYPRX0 and DFSYDRU0
  - Asynchronous output from non-OTMA originated transactions is also possible using the OTMA routing exits
    - Must be supported by the OTMA client
      - **ITOC does not support**
      - **IMS Connect will support**
      - **MQSeries does support**
      - **OTMA Callable Interface does support**

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- In an IMS 6.1 Shared Queue environment all OTMA messages must run on the IMS copy that received them
- In IMS 7.1 Commit Mode 0 messages can run on any IMS copy in the shared queue group
  - IOPCB output is routed back through the original IMS
  - ALTPCB output is sent from the processing IMS
- Commit mode 1 messages must still run on the IMS copy that received them
  - IBM would like to remove this restriction in a future release

Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- The maximum OTMA input segment size is 32767
  - LLZZ + 32763 bytes of user data
  - IMS will break up the segment into pieces that fit into the LGMSG and chain them together
- The maximum OTMA total message length to input to MQSeries is the MQ Max Message Length
  - This is usually 4MB
  - MQSeries long message support increases this to 100MB
    - This has not been tested with OTMA
- Thanks to Bob Millar of IBM Hursley for this information

Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- There are two IMS OTMA output routing exits
  - DFSYPRX0: Pre-routing exit
  - DFSYDRU0: Destination resolution exit
- Invoked for CHNG call to modifiable ALTPCB
- Invoked for ISRT call to static ALTPCB
  - Invoked in XM mode so no SVCs or IMS services
- Invoked even if input message was not from OTMA
  - Allows asynchronous output to OTMA
- Not invoked for ISRT to IOPCB
- Not invoked for ISRT to transaction

Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- DFSYPRX0: Pre-routing exit
  - Actions:
    - RC=0: Input message came from OTMA, destination is same OTMA client or input message did not come from OTMA, output is not OTMA
      - If input is from OTMA and you want to send the output to a different OTMA client you need PQ32402 (PQ33996 for IMS 7.1) and OTMAMD=Y in DFSPBxxx
    - RC=4: message originally did not come from OTMA, but destination is OTMA
      - Need to set XCF member name of OTMA client
    - RC=8: message came from OTMA, but destination is not OTMA
  - IMS sample exit is not useful
    - MQSeries sample exit in Appendix B of the *MQSeries for OS/390 System Management Guide* is very useful for MQSeries client
  - Needed because there can be a different DFSYDRU0 exit for each client
    - This exit determines the client and thus the DFSYDRU0 exit to invoke

Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- DFSYDRU0: Destination Resolution exit (default name)
  - Each OTMA client can have their own exit
  - Name for the exit for a client can be specified
    - In DFSYDTx
    - By the client when it connects
      - In CSQFSYSP in CSQZPARM for MQSeries
        - Recommendation is DRU0xxxx (xxxx=QMGR name), then exit can know XCF member name for asynchronous output
      - On DATASTORE card for IMS Connect
      - Via otma_openx function for OTMA Callable Interface (PQ32398)

Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- DFSYDRU0: Destination Resolution exit
  - Actions:
    - RC=0: destination is the original OTMA TPIPE
    - RC=4: destination is non-OTMA LTERM
    - RC=8: destination is new OTMA Client (need to specify)
      - The new client DRU0 exit will then be invoked
      - No need for this after PQ32402
    - RC=12: destination is invalid
      - A1 status on CHNG call
  - Can override the output TPIPE name - PQ27207
    - Client is waiting on a specific TPIPE
  - Can create a SYNChronized TPIPE
    - Needed for MQSeries (more later)
    - Default for a new TPIPE is non-SYNChronized
    - PQ32402 (PQ33996) allows default to be SYNChronized
      - OTMASP=Y in DFSPBxxx

Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- DFSYDRU0 (continued)
  - Passed address of OTMA user data
    - Early IMS documentation said maximum was 256 bytes
    - Real maximum is 1024 bytes
    - For asynchronous output you must value this as expected by the OTMA client
      - MQSeries requirements are discussed later
      - IMS Connect also has a required format for this data
  - NOT passed address of first segment of output message
    - There is none on a CHNG call
    - IBM is looking at alternatives

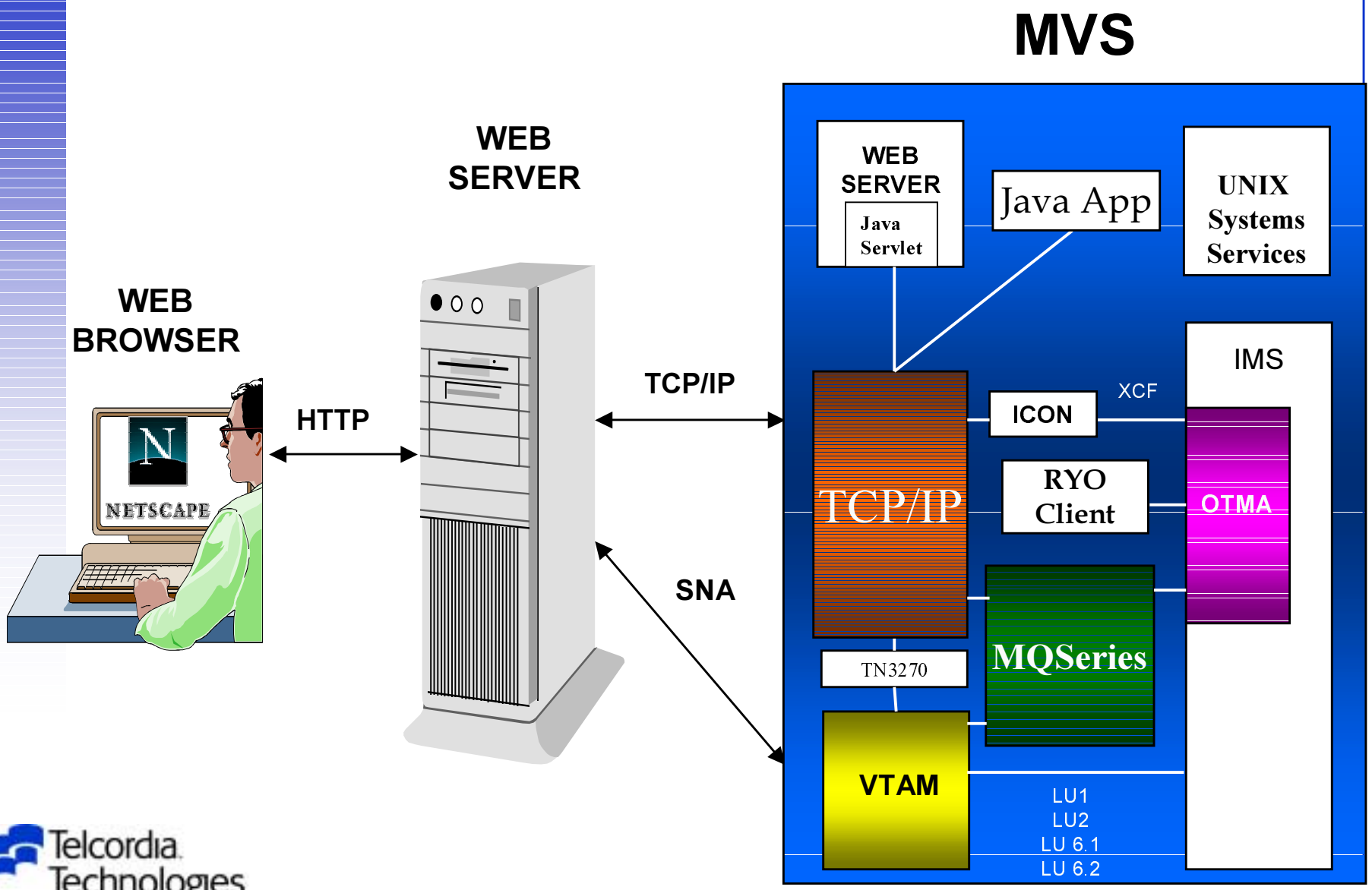
Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- DFSYDRU0 (continued)
  - IMS sample exit is not useful
    - MQSeries sample exit in Appendix B of the *MQSeries for OS/390 System Management Guide* is very useful
      - Follow this code VERY carefully
      - More later
  - Fifth input parameter is DFSINTX0 table address
    - **Not documented**

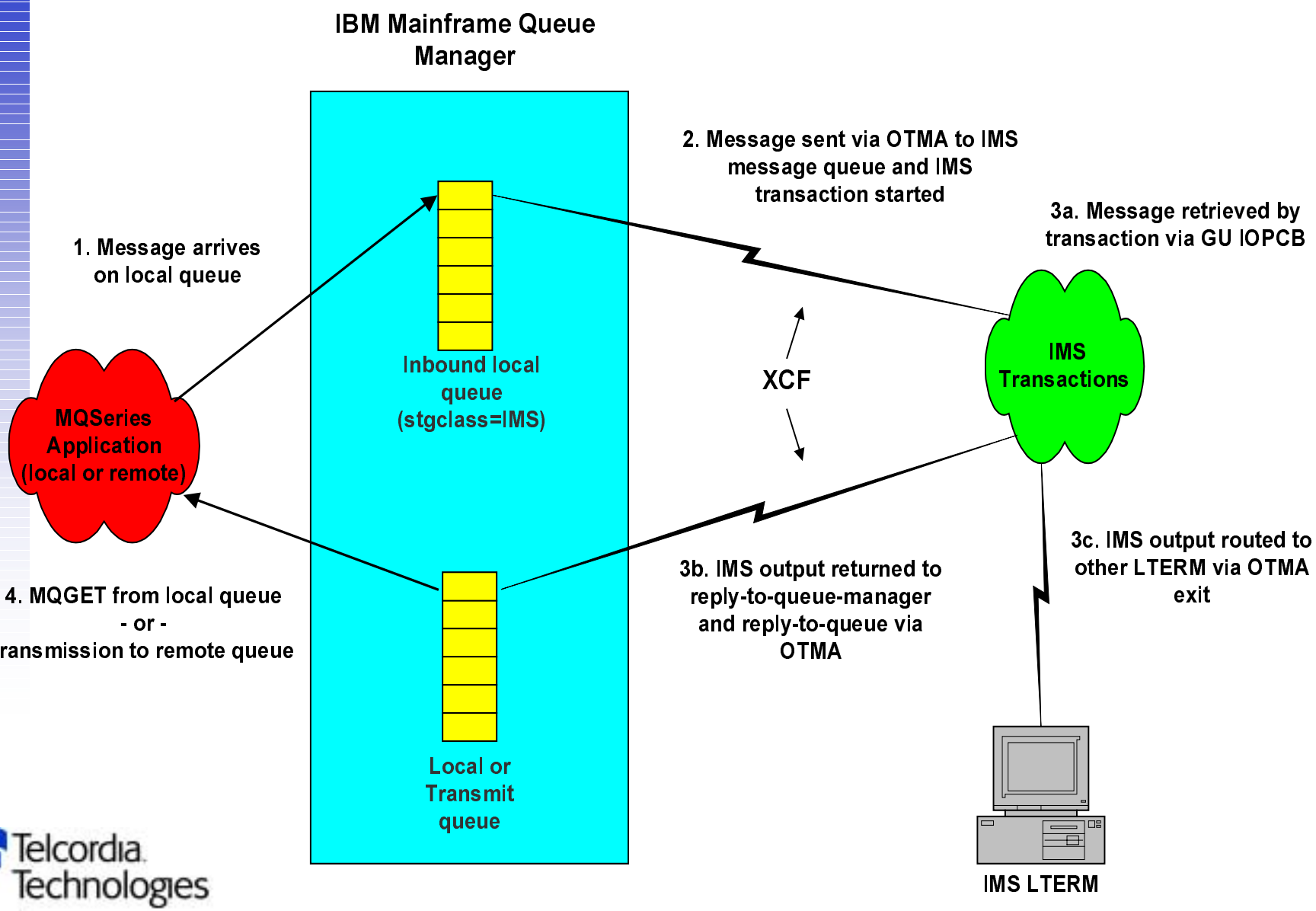
Telcordia Technologies
Performance from Experience

IMS OTMA Interface



Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge



Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- One MQSeries queue manager can connect to multiple IMS control regions
- One IMS control region can connect to multiple MQSeries queue managers
- MQSeries and all IMS control regions must be in the same XCF group
  - I am hoping to get this fixed
- MQSeries and IMS can be on different MVS copies in the same Sysplex
- MQSeries IMS Bridge start and stop events are sent to the SYSTEM.ADMIN.CHANNEL.EVENT.QUEUE

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- When the message arrives in MQSeries it will be sent via XCF to the IMS OTMA interface.
- Message may be:
  - an IMS transaction
  - an IMS command (only a subset of commands are allowed)
    - IMS does not support the /EXIT command via OTMA (today)
    - MQSeries does support this by translating it into OTMA control commands before sending to IMS
  - NOT a message to an IMS LTERM
- IMS will put it on the IMS message queue
- The application will do a GU to the IOPCB to retrieve the message
- This is very similar to the implicit LU6.2 process
- A remote queue manager can send a message to a local queue destined for IMS via OTMA

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Define MQSeries to XCF in CSQZPARM
  - OTMACON keyword on CSQ6SYSP macro
    - OTMACON(Group,Member,Druexit,Age,TPIPEPrefix)
      - Group=XCF group
      - Member=MQSeries XCF member
      - Druexit=IMS exit to route output (overrides DFSYDTx)
      - Age=how long a userid from MQSeries is considered previously verified in IMS
      - TPIPEPrefix=three character prefix for TPIPE name
        - To avoid collision with IMS transaction code names
        - Two characters for MQSeries shared queues
  - Member CSQ4ZPRM in data set hlq.SCSQPROC has default CSQZPARM members you can use to build your members

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Define one or more storage classes with the XCFGNAME and XCFMNAME parameters of the IMS systems to which you will connect
  - DEFINE STGCLASS(IMSA) -
    PSID(02) –
    XCFGNAME(XCFGROUP)
    XCFNAME(XCFIMSA)
- Define local queue(s) referencing the storage classes
  - DEFINE QLOCAL(MQID_TO_IMSA) –
    STGCLASS(IMSA)
- Define reply-to-queue(s)
  - DEFINE QLOCAL(MQID_FROM_IMSA)

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Confirm On Arrival (COA) is provided when the message reaches the IMS queue
- Confirm On Delivery (COD) is not available
  - The message is rejected if this option is specified
- Expiry
  - A message can expire before reaching the IMS Bridge
  - It can not expire once it has reached the IMS Message Queue
    - There are user requirements to change this

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- The sending application can optionally provide an IMS sub-header (MQIIH)
  - Several reserved fields...
  - IMS LTERM name to put in IOPCB
  - MFS Format name to put in IOPCB (no MFS formatting is actually done)
  - Reply To Format (MQSeries format name)
  - Authenticator (RACF password or PassTicket)
  - Transaction Instance ID (if IMS conversational)
  - Transaction State (conversational or not, architected command)
  - Commit Mode (commit-then-send or send-then-commit)
  - Security Scope (ACEE in control region or in control region and dependent region)
    - Only honored if /SEC OTMA PROFILE used

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Specify the presence of the MQIIH sub-header by using the MQMD.Format=MQFMT_IMS (“MQIMS”)
  - WARNING: MQIIH must be fullword aligned
- If no MQIIH is presented to the IMS Bridge (MQMD.Format=MQFMT_IMS_VAR_STRING) (“MQIMSVS”) default values are assumed:
  - LTERM=TPIPEName
  - MODNAME=MQMD.Format
    - Default is “MQIMSVS”
  - MQMD.Format is used as the MFSMapName
  - Non-conversational
  - Commit-then-send (commit mode 0)
  - Security mode is MQISS_CHECK (ACEE only in CR)
- Data is passed to IMS in the OTMA prefix

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- MQSeries creates two TPIPE control blocks per local queue defined as using the IMS Bridge
  - One for “asynchronous” messages
    - Commit mode 0 - commit-then-send
    - Output is “asynchronous” to transaction completion
    - Default if no IIH
    - This is a SYNChronized TPIPE
  - One for “synchronous” messages
    - Commit mode 1 - send-then-commit
    - Output is “synchronous” with transaction completion
    - This is a non-SYNChronized TPIPE
- The TPIPE control blocks in IMS are created when the first message for the type (sync, async) arrives on the IMS Message queue

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- There is a limit to the capacity of an IMS TPIPE
- There are several factors involved
  - SYNChronized versus non-SYNChronized
  - MQSeries Persistent versus non-Persistent
  - IMS RECOVER versus NONRECOVER
  - Message size
- More capacity will require more TPIPEs which will require more MQSeries queues
  - There are only 2 TPIPES per queue
  - The application has to round-robin the messages to the queues
  - I am hoping to get relief in this area
- Support Pack MP16, *Capacity Planning for MQSeries for OS/390*, documents IMS Bridge performance
  - http://www-4.ibm.com/software/ts/mqseries/txppacs/mp16.html

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- There are special requirements for Commit Mode 0 input messages to IMS
  - If the IMS transaction is defined as RECOVER then the MQSeries message must be persistent
  - If the IMS transaction is NORECOV then the MQSeries message must be non-persistent
  - If this is not the case IMS will reject the message with sense code 00230000

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Commit Mode 0 input messages (continued)
  - How does IMS know whether a message is persistent or nonpersistent?
    - All CM0 messages arrive on SYNChronized TPIPES
    - There are no flags in the OTMA headers for this
  - The answer is that IMS and MQSeries use a little known but documented OTMA interface
    - If the message is persistent it is sent on the SYNChronized TPIPE with a valid sequence number
    - If the message is non-persistent it is sent on the SYNChronized TPIPE with the sequence number set to zero

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- The data stream passed to MQSeries by the application MQPUT is in LLZZTrancodeDataLLZZData... format
  - This allows for multi-segment input messages via OTMA to IMS
- The sending message is always treated by MQSeries as “in syncpoint”
  - If the message is “not in syncpoint” it is treated as “in syncpoint” followed by an immediate commit

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- If the input message cannot be put to the IMS queue because:
  - Invalid message - input message goes on DLQ and warning sent to system console
  - IMS rejected message (sense 001A) - DFS message from IMS is put into MQSeries reply message and put on reply-to queue
    - If reply message cannot be PUT, it is placed on the DLQ
    - Input message now goes to DLQ (PQ05963)
  - IMS rejected message (message error) - input message goes on DLQ and warning sent to system console
  - IMS rejected message (other) - messages go back to their original queue, the IMS Bridge is stopped, and warning sent to system console
- If any return messages cannot be PUT to the DLQ, messages go back to their original queue
  - If it was a queue problem the queue is stopped
  - If it was an IMS Bridge problem the IMS Bridge is stopped

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- IMS Output Messages
  - All IMS message segments are assembled into a single MQSeries message which must not exceed the MAXMSGLEN parameter of the queue manager
  - If PUT to the reply-to queue fails, the message goes on the DLQ
  - If PUT to the DLQ fails, a NAK is sent to IMS and the message remains in IMS

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- IOPCB Output Reply messages are placed on the MQSeries reply-to-queue specified in the original message header
  - This could be a local queue or a remote queue
- If you do nothing ALTPCB output from an IMS Bridge initiated transaction will also be placed on the MQSeries reply-to-queue specified in the original message header
  - This can be overridden by the OTMA routing exits

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- In order to send ALTPCB output to MQSeries over the IMS Bridge you must build (or override) the OTMA user data in the DFSYDRU0 exit
- If the message did not originate from the MQSeries IMS Bridge you must build the entire OTMA user data
- If the message did originate from the MQSeries IMS Bridge the OTMA user data is set to return the message to the MQSeries reply-to-queue specified in the original message header when the exit is invoked
  - This can be changed by updating the OTMA user data

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- The format is of the OTMA user data going to the MQSeries IMS Bridge is:
  LL | | MQMD | | Reply-to-Format
- The LL must be exactly 2 (LL) + L'MQMD + 8 (Reply-to-Format) or MQSeries will ignore the entire MQMD and use default values
- The MQSeries sample exit in Appendix B of the *MQSeries for OS/390 System Management Guide* is very useful
  - Follow this code VERY carefully

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- What you are building is actually an INPUT MQMD as if the message had come from MQSeries originally
  - You must get this mindset
- The Reply-to-Queue and Reply-to-Queue Manager in the MQMD become the message destination

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- The MSGID and CORRELID are the INPUT MSGID and CORRELID - not necessarily what will be presented to the MQSeries application
  - You must use the MQMD_REPORT field to specify that the input MSGID and CORRELID be moved to the corresponding output MQMD
  - The default is that the input MSGID is moved to the output CORRELID and a new MSGID is created

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Setting the format name for MQSeries to use for asynchronous OTMA output messages is not straightforward
- If you are sending back LLZZdataLLZZData as is probably the case you need to set the format name to MQFMT_IMS_VAR_STRING (“MQIMSVS”)
- The “Reply-to-Format” name in the OTMA User Data is the obvious choice
- But - the “Reply-to-Format” in the OTMA User Data is ignored unless the original message was MQPUT with an MQIIH IMS prefix and the Reply-to-Format in the MQIIH was valued

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- But - there was no input message – how do you tell MQSeries there was an MQIIH
  - Remember you are building an INPUT MQMD in the OTMA User Data
    - Lie – set the MQMD_FORMAT to MQFMT_IMS (“MQIMS”)
- This tells MQSeries the “input message” had an MQIIH
- Then set the Reply-to-Format name in the OTMA User Data to “MQIMSVS”
- AND THIS DOES NOT WORK EITHER!!!!!
  - MQSeries will create an MQIIH in front of the output message
    - Your application is probably not expecting this
    - The MQMD_FORMAT field will remain MQ_IMS (“MQIMS”)
    - The Reply-to-Format (“MQIMSVS”) will be in the MQIIH

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- If the Reply-to-Format is not valid as previously described or is blank then field TMAMHMAP in the OTMA State Prefix is used as the MQSeries Format Name
- This is set by the MODNAME parameter in the IMS ISRT call, if present
  - CALL “CBLTDLI” USING ISRT ALTPCB MSGIO MODNAME
    - Do you trust your application to do this properly?

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- The TMAMHMAP field can be overridden in the OTMA Input/Output Edit Routine (DFSYIOE0)
- This lets your application be output independent
  - It can set a valid MODNAME in case the message is going to a terminal
  - The DFSYIOE0 exit can override it to “MQIMSVS”
- If TMAMHMAP is blanks or nulls the MQSeries Format name is set to MQFMT_NONE (blanks)
  - If output is LLZZdata this will cause problems

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- The MQSeries application that MQGET's the reply message must be able to process the LLZZ's
  - If the application is written in C you must "cast" the buffer as MQBYTE or very bad things will happen
    - #include <cmqc.h>
    - MQBYTE *pbuff;
    - pbuffer=(MQBYTE *)malloc(bufsize);
- You can also consider using MQSeries channel exits to remove the LLZZ's and set the Format Name to MQFMT_NONE

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Determining the MQSeries Persistence for output messages from OTMA to MQSeries is complicated and is affected by commit mode
- The determination takes place in two phases
- It is also different if the ALTPCB output is going to a TPIPE originally created by MQSeries
  - The CHNG call destination or the DFSYDRU0 exit set the TPIPE name to an MQSeries created TPIPE

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Persistence Determination – Phase 1
  - If the output is on a TPIPE created by MQSeries
    - If the commit mode is 1 (send-then-commit) then the output message is persistent
    - If the commit mode is 0 (commit-then-send) then persistence is determined by message recoverability
      - If the message is “recoverable” then the output message is persistent
      - If the message is not “recoverable” then the output message is nonpersistent
    - A message is “recoverable” to MQSeries if it is sent on a SYNChronized TPIPE
      - Not IMS RECOVERable

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Persistence Determination – Phase 1
  - If the output is on a TPIPE not created by MQSeries
    - If the commit mode is 1 and flag TMAMCASY in not set in the OTMA State Header then the output message is persistent
    - If the commit mode is 0 or the TMAMCASY flag is set then persistence is determined by message recoverability
      - If the message is “recoverable” then the output message is persistent
      - If the message is not “recoverable” then the output message is nonpersistent
    - TMAMCASY indicates “asynchronous/unsolicited queued messages”
      - It may be set on for some IMS DFS messages
      - These messages must be treated as CM0 even if the input was CM1

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Persistence Determination – Phase 1
  - There are two ways to make an ALTPCB TPIPE SYNChronized when it is created
  - PQ32402 (PQ33996) added the OTMASP parameter in DFSPBxxx which can be used to default newly created TPIPEs to SYNChronized
    - This parameter was created for MQSeries users who wanted persistent output messages but did not want to write DFSYDRU0 exits
  - The DFSYDRU0 exit can specify that a newly created TPIPE is SYNChronized
    - Turn on the high-order bit in the first flag in the member override area

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- Persistence Determination – Phase 2
  - If there is a valid OTMA User Data Prefix
    - If the output is on a TPIPE created by MQSeries
      - If the message is Commit Mode 1
        - Then the output message persistence is the persistence of the input message as reflected in the MQMD_PERSISTENCE field in the OTMA User Data

Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- MQMD as seen by the MQGET
  - MQMD_REPORT = MQRO_NONE (x'00')
  - MQMD_MSGTYPE = MQMT_REPLY (x'02')
  - MQMD_FORMAT = (see previous discussion)
  - MQMD_PERSISTENCE = (see previous)
  - MQMD_MSGID = (see previous)
  - MQMD_CORRELID = (see previous)
  - MQMD_REPLYTOQ = blanks
  - MQMD_REPLYTOQMGR = MVS Queue Manager

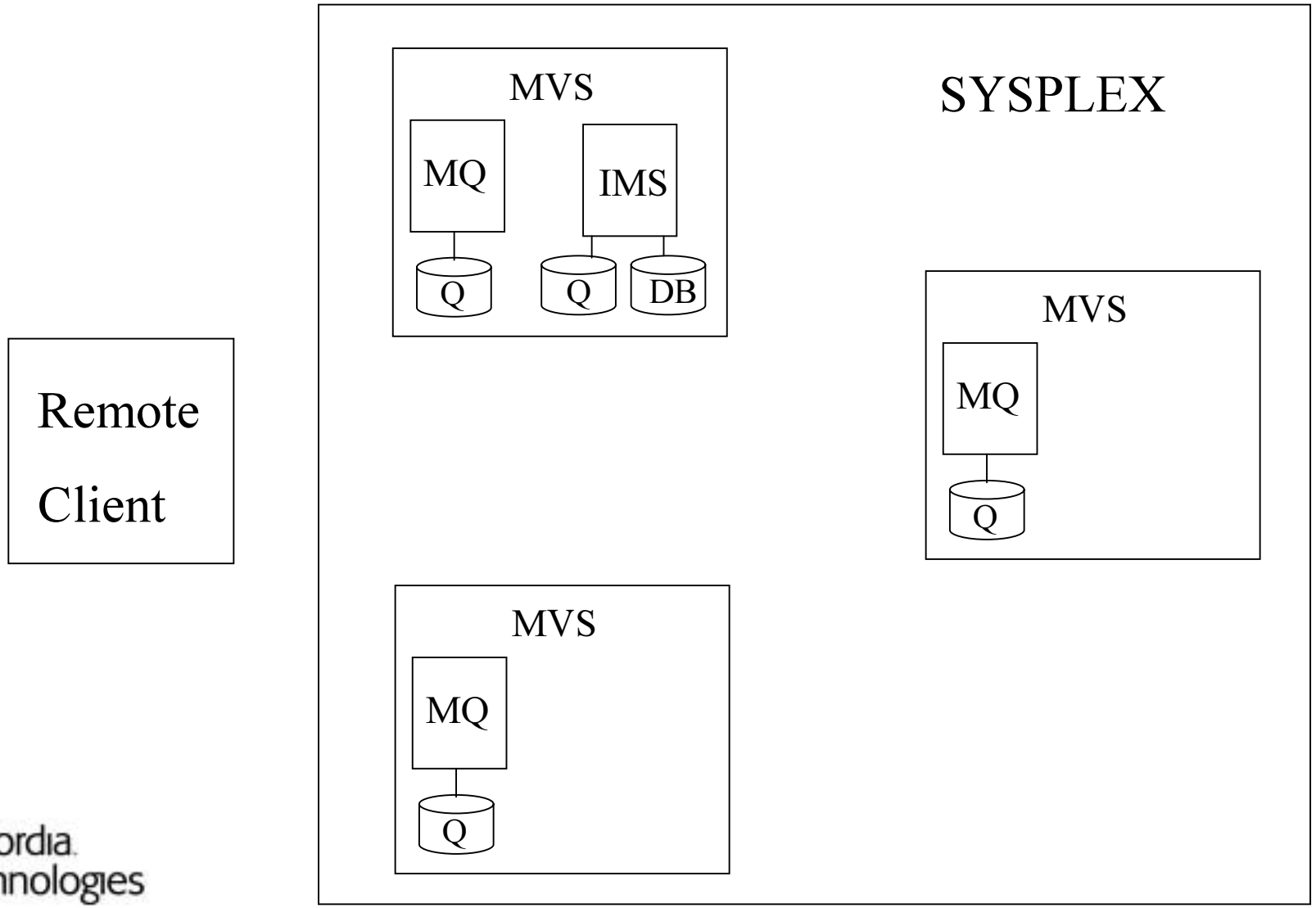
Telcordia Technologies
Performance from Experience

MQSeries IMS Bridge

- MQMD as seen by the MQGET (continued)
  - MQMD_USERIDENTIFIER = blanks
    - Unless you explicitly value it in the DFSYDRU0 exit
      - or the input message came from the IMS Bridge
    - You can copy the USERID passed in the DRU0 PARM
  - MQMD_ACCOUNTINGTOKEN = Garbage
  - MQMD_APPLIDENTITYDATA = blanks
    - Unless you explicitly value it in the DFSYDRU0 exit
      - or the input message came from the IMS Bridge and it was valued by the application that did the MQPUT
    - Then MQSeries will pass whatever you put in there
  - MQMD_PUTAPPLTYPE = MQAT_XCF (x'14')
  - MQMD_PUTAPPLNAME =
    - 8 Byte XCF Group Name
    - 16 Byte XCF Member Name of the sending IMS

Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 1



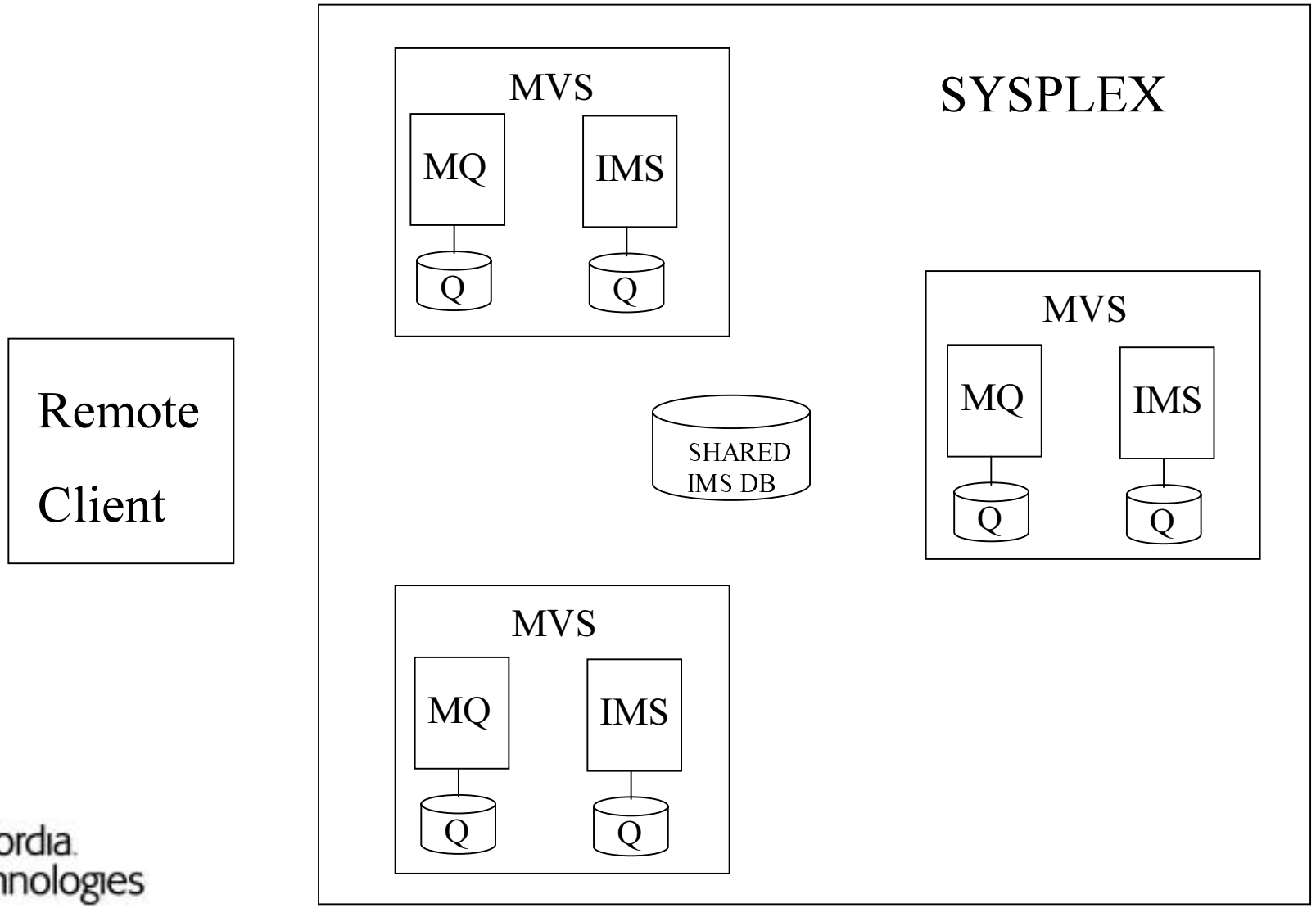
Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 1

- There are multiple MQSeries Queue Managers in one MVS SYSPLEX
- Each Queue Manager has an IMS Bridge connection to the same IMS
- Messages can be sent to MQSeries as long as any one Queue Manager is available
- If a Queue Manager is unavailable any messages in its queues or in IMS queues waiting for that Queue Manager are unavailable
- If IMS in unavailable no messages can process

Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 2



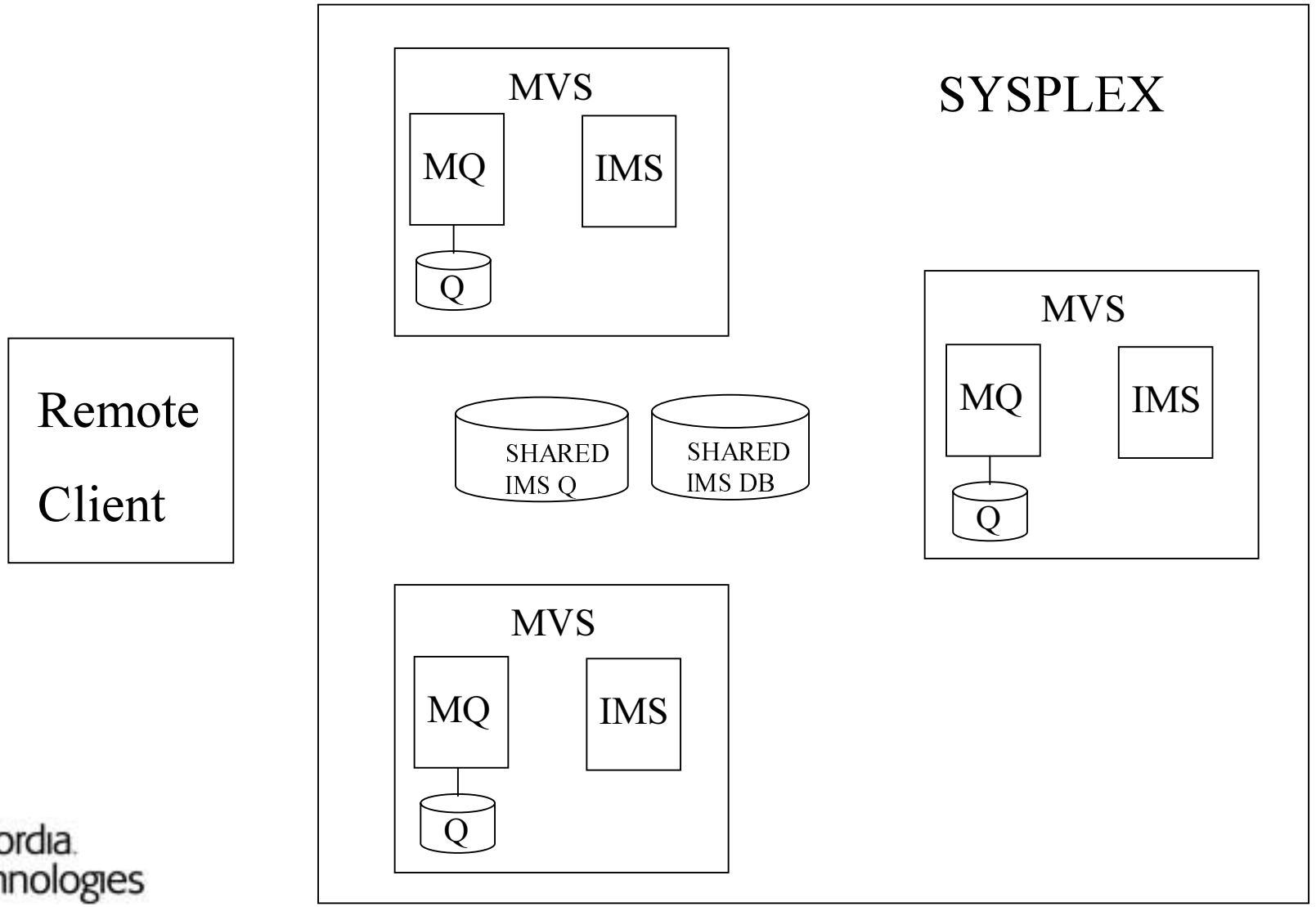
Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 2

- There are multiple MQSeries Queue Managers in one MVS SYSPLEX
- Each Queue Manager has an IMS Bridge connection a different IMS
  - The IMS copies share databases
  - Any transaction can run on any IMS copy
- Messages can be sent to MQSeries as long as any one Queue Manager is available
- If a Queue Manager and/or IMS is unavailable any messages in its queues or in IMS queues waiting for that Queue Manager are unavailable

Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 3



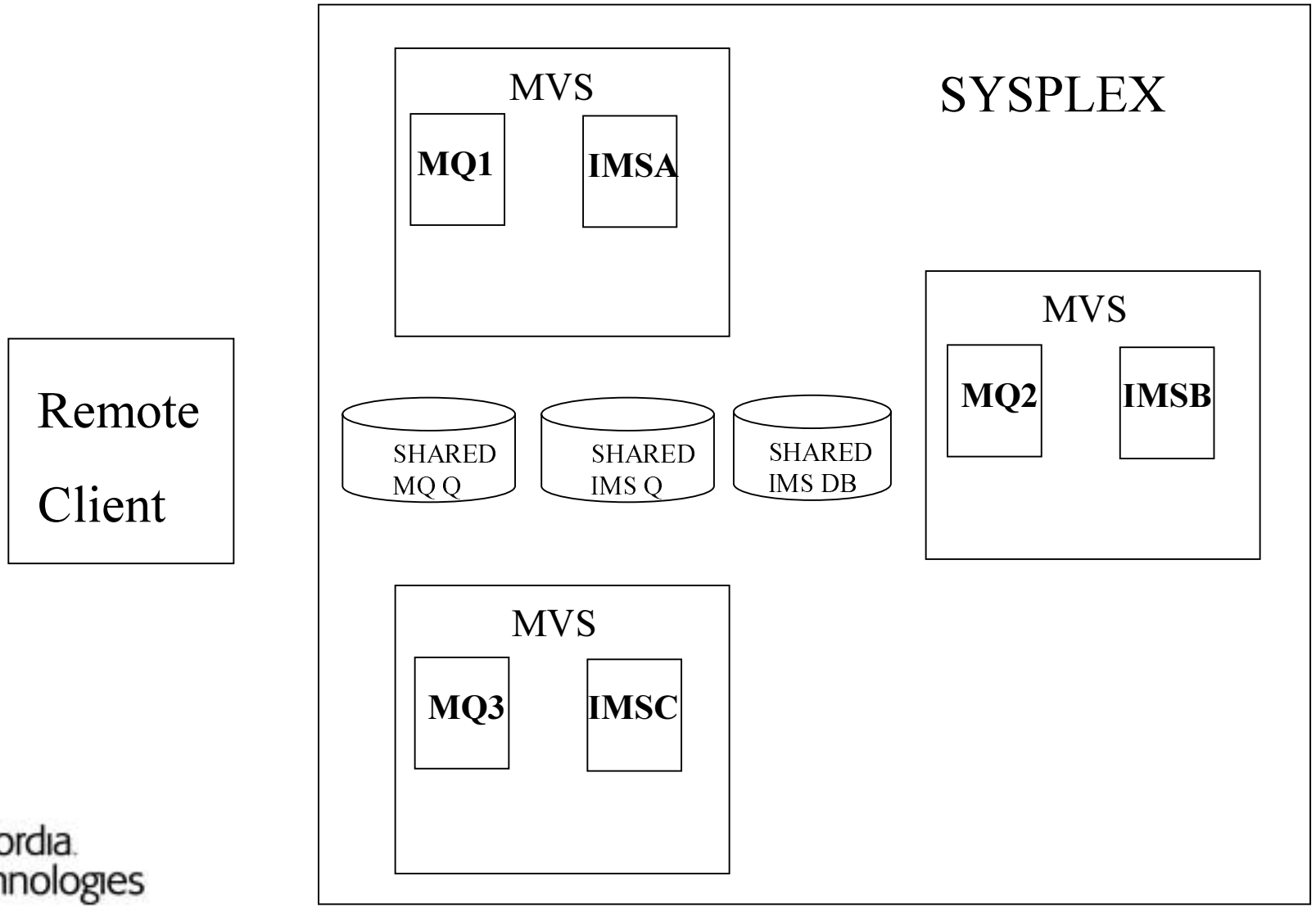
Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 3

- The IMS copies share databases and message queues
- If the messages are Commit Mode 1 there is no difference in availability with Case 2
  - CM1 messages must run on the IMS which receives them
- Commit Mode 0 messages can run on any IMS copy (requires IMS 7.1)
  - IOPCB output will be sent by the receiving IMS
  - ALTPCB output will be sent by the processing IMS
- There are still places where messages are unavailable if MQSeries or IMS is unavailable

Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 4



Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 4

- MQSeries has shared queues
- The IMS copies share databases and message queues
- The availability depends on the MQSeries Queue and STGCLASS definitions and Commit Mode and IMS Version

Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 4

- There would be one definition for the Shared IMS Bridge Queue
  - DEFINE QLOCAL(SBQ1) STGCLASS(IMSX)
- But each Queue Manager has its own definition for STGCLASS IMSX
- One option is to have them all the same
  - DEF STGCLASS(IMSX) XCFNAME(IMSA)
  - The shared Q has the NOSHARE option
  - This results in the same availability as Case 3
  - The first queue manager available gets the connection to the shared queue
    - If that queue manager becomes unavailable another queue manager will get the connection

Telcordia Technologies
Performance from Experience

MQSeries Clustering with the IMS Bridge – Case 4

- Another option is to have them each go to a different IMS and use the SHARED option
  - MQ1: DEF STGCLASS(IMSX) XCFNAME(IMSA)
  - MQ2: DEF STGCLASS(IMSX) XCFNAME(IMSB)
  - MQ3: DEF STGCLASS(IMSX) XCFNAME(IMSC)
- The message will be put on the MQSeries shared queue by any receiving queue manager
- The message can be retrieved from the MQSeries shared queue and sent to IMS by any queue manager which currently has a connection to IMS
- Once on the IMS shared queue the message can be processed as documented before based on Commit Mode and IMS Version

Telcordia Technologies
Performance from Experience

IMS OTMA Security

- See Session A50 for a complete discussion of OTMA security
- Requires RACF 1.9.2 (or equivalent)
  - ACF2 has many OTMA fixes and considerations
- Three levels of security
  1. Connections from MQSeries to IMS (during client bid)
     - Uses the RACF Facility class
       - Not used if OTMA security is set to none
     - IMS checks ***IMSXCF.XCFgroupname.MQ-member-name***
       - MQSeries must have read access to this class
     - MQSeries checks ***IMSXCF.XCFgroupname.IMS-member-name***
       - The MQSeries subsystem userid access to this class determines userid validation for each message that crosses the IMS Bridge

Telcordia Technologies
Performance from Experience

IMS OTMA Security

- Three levels of security (continued)
  2. Userid Validation in MQSeries
     - MQMD.UserIdentifier field is used
     - Based on MQSeries Subsystem userid access to ***IMSXCF.XCFgroupname.IMS-member-name***
       - CONTROL/ALTER: Userids trusted, no checks
       - UPDATE: Userid validated by RACF prior to passing to IMS
       - READ: Userid/password validated by RACF prior to passing to IMS. Result of this check is cached and used on subsequent calls.
       - NONE (or no profile): Userid/password validated by RACF prior to passing to IMS. No cache is performed.
     - Once validated, the userid is passed to IMS and can be used for normal IMS security

Telcordia Technologies
Performance from Experience

IMS OTMA Security

- Three levels of security (continued)
  3. OTMA Security (set by /SECURE OTMA)
     - CHECK
       - Existing RACF calls are made
       - IMS commands are checked against the CIMS class
       - IMS transactions are checked against the TIMS class
     - FULL
       - Same as CHECK, but the ACEEs are built in the dependent regions as well as the control region
         - Default at IMS cold start
         - PQ32402 (PQ33996) introduces parameter OTMASE in DFSPBxxx which allows the definition of a default cold start security level
     - NONE
       - No calls to RACF are made
     - PROFILE
       - Each message defines the level of security checking to be done
       - MQIIH.SecurityScope field allows for FULL or CHECK to be specified

Telcordia Technologies
Performance from Experience

IMS OTMA Security

- MQSeries security profiles in the MQADMIN class
  - If ***subsysid.NO.SUBSYS.SECURITY*** is defined...
    - MQSeries does not pass userid to IMS
    - Client bid only succeeds when /SEC OTMA NONE is in effect (no userid is passed to the IMS transaction)
    - Good for early testing
  - If ***subsysid.NO.SUBSYS.SECURITY*** is not defined...
    - Define the ***subsysid.NO.xxxx.CHECKS*** profiles
    - Give MQSeries Subsystem userid at least UPDATE access in the ***IMSXCF.XCFgroupname.IMS-member-name*** profile
    - Client bid will succeed and userids will be passed to the IMS transactions

Telcordia Technologies
Performance from Experience

Information on the Web

- http://www.software.ibm.com/ts/mqseries/
- http://www.software.ibm.com/data/ims
- http://www.redbooks.ibm.com/

Telcordia Technologies
Performance from Experience

Questions?

Telcordia Technologies

Performance from Experience

RETURN TO INDEX