Telcordia Technologies

Performance from Experience

MQSeries Introduction

Session E04

Prepared For:

IMS Technical Conference 2000

Telcordia Contact:

Gary J. Ward

Chief Technologist

Exchange Link Business Solutions

gward@telcordia.com

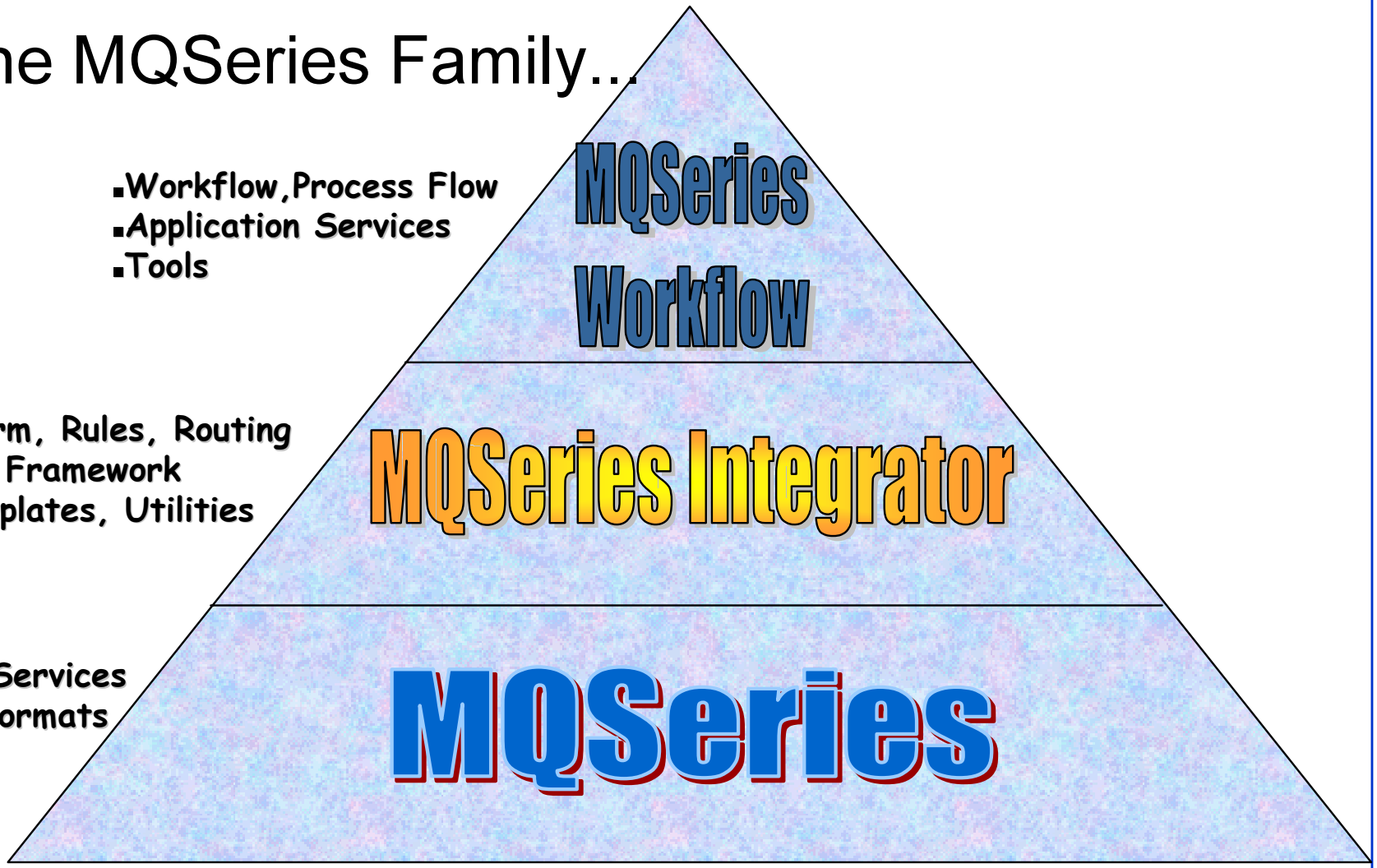
Presentation Outline

- What is MQSeries?
- MQSeries Queues
- MQSeries Channels
- MQSeries Clustering
- MQSeries Shared Queues
- Message Queuing Interface (MQI)
- “A day in the life of an MQ message”

Telcordia Technologies
Performance from Experience

What is MQSeries?

- The MQSeries Family...



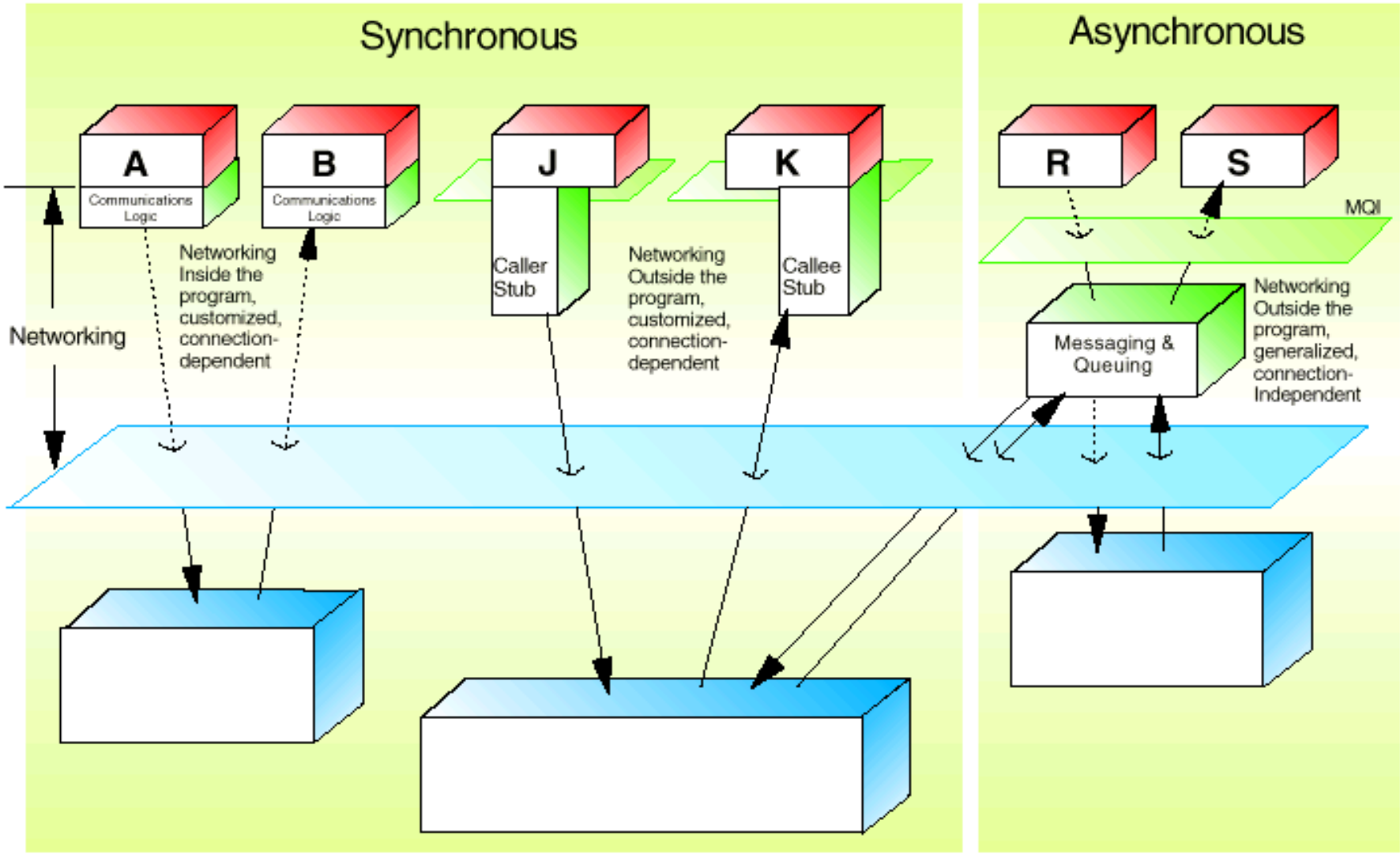
Telcordia Technologies
Performance from Experience

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

Telcordia Technologies
Performance from Experience

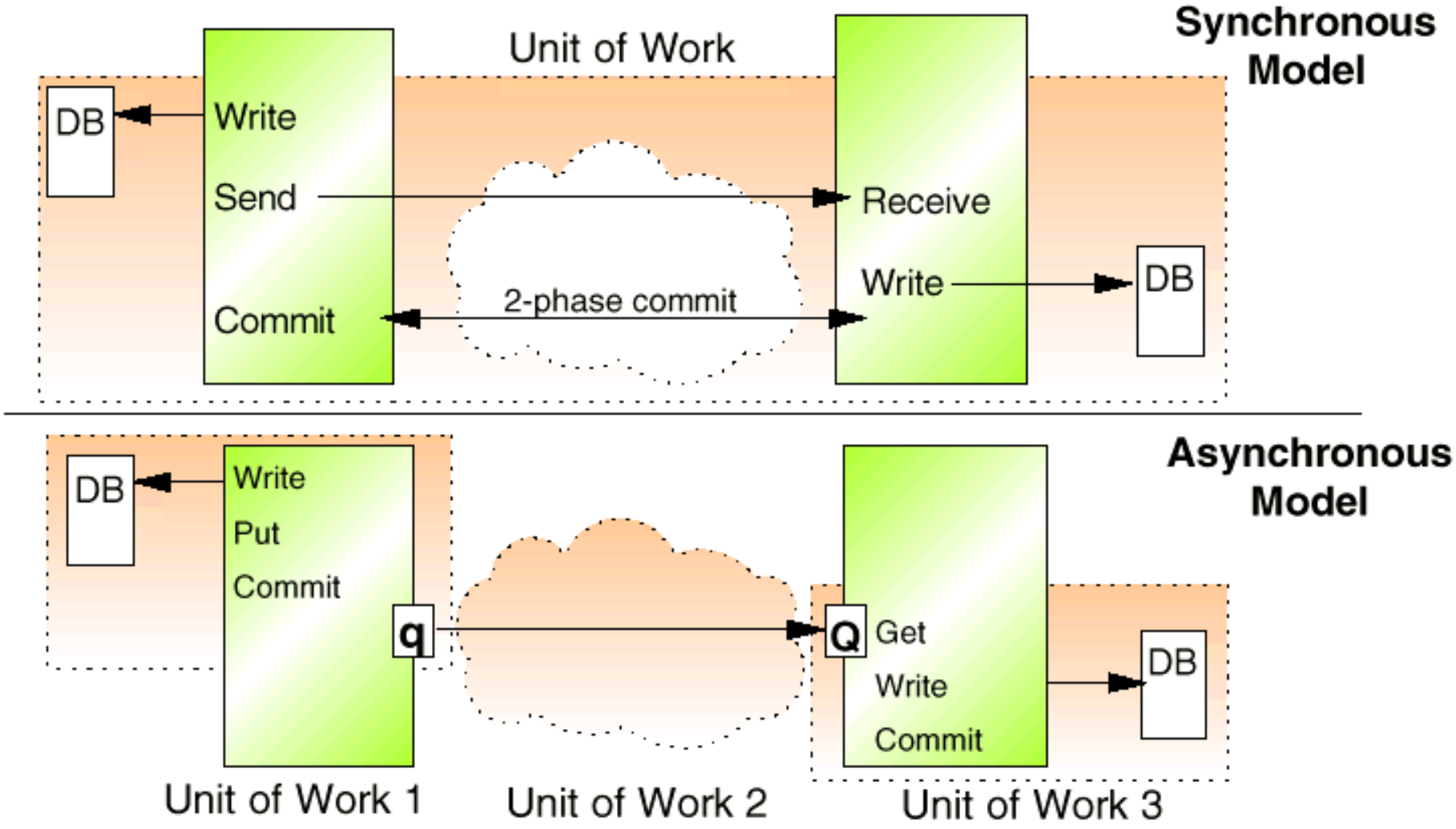
What Is MQSeries?



Telcordia Technologies

Performance from Experience

What is MQSeries?



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 sync points with other products (e.g., IMS, CICS, XA Compliant products)

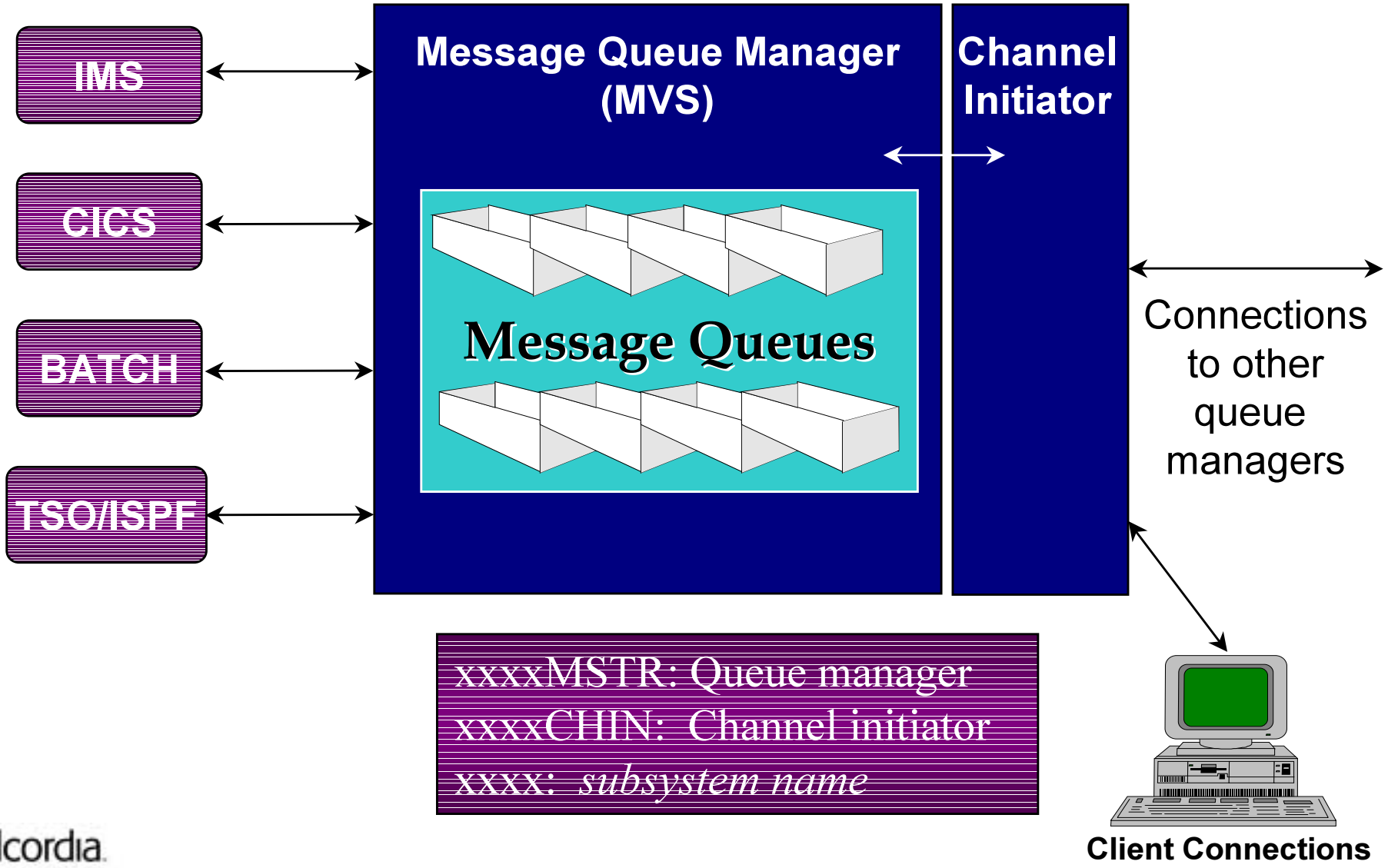
Telcordia Technologies
Performance from Experience

What is MQSeries?

- MQSeries software includes:
  - Queue manager code
  - Basic administration tools
  - Some advanced administration tools (Candle, IBM tools in 5.1 on NT)
  - Message Queuing Interface (MQI) libraries
  - Sample programs (source and executable)
  - Softcopy documentation (ps files, ‘man’ pages, HTML)
  - Client interfaces
- The exact implementation of the queue manager and low-level architecture vary from platform to platform
- However, the objects, MQI, and general usage rules are the same on all platforms

Telcordia Technologies
Performance from Experience

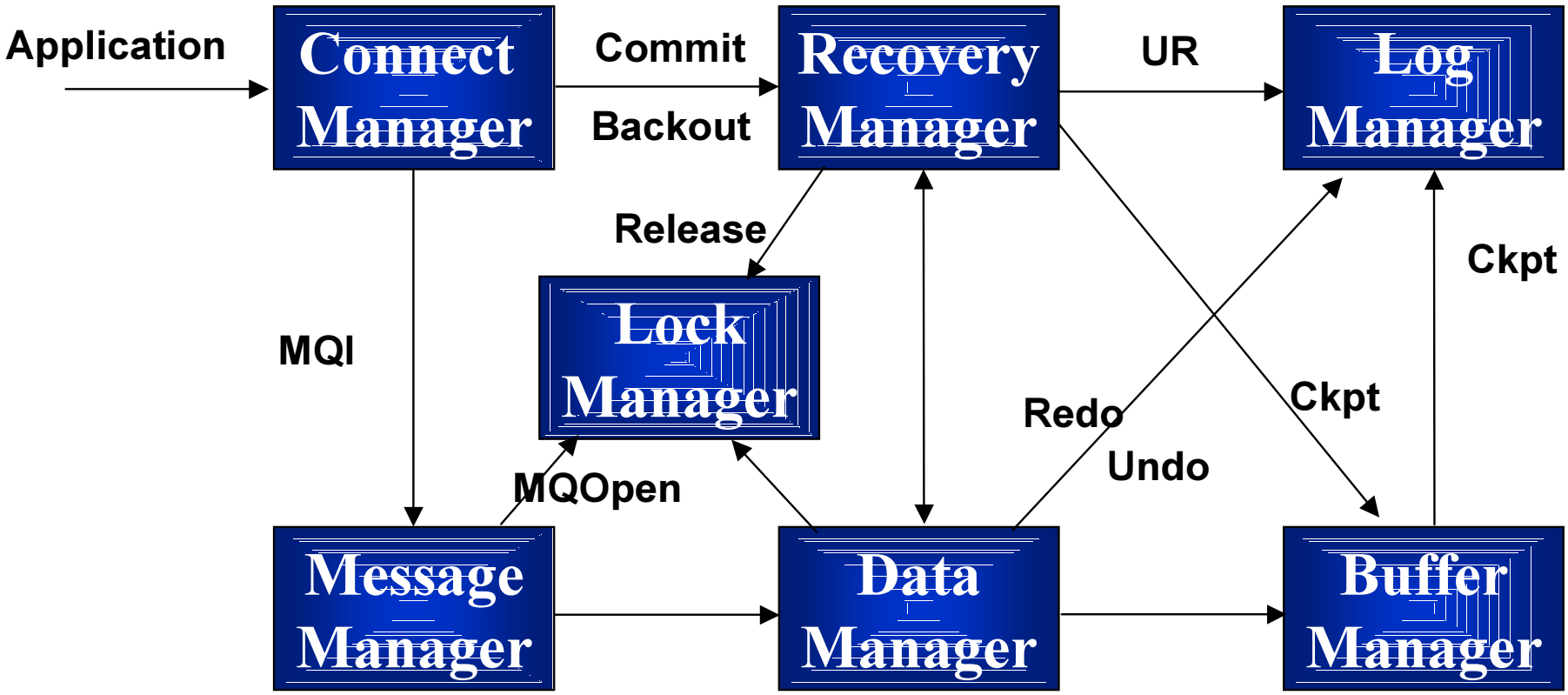
What is MQSeries?



Telcordia Technologies
Performance from Experience

What is MQSeries?

- Some of the internals of the OS/390 Queue Manager



Telcordia Technologies
Performance from Experience

What is MQSeries?

OS/390 QUEUE MANAGER COMPONENTS:

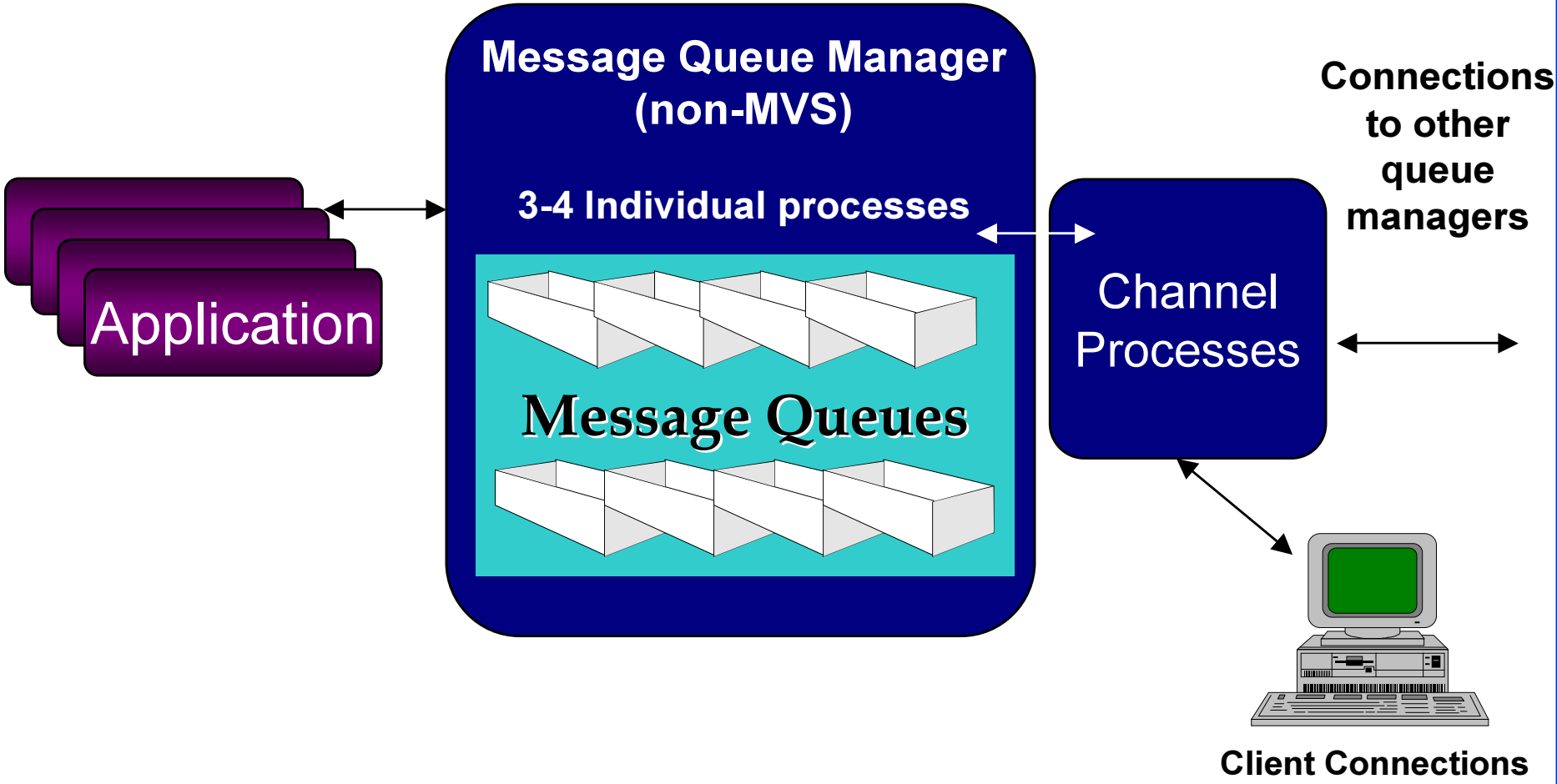
- Connection Manager
  - Entry point from adapters
  - Understands CICS tasking scheme
  - Commit and Backout requests sent to
  - Recovery Manager
  - Remainder of MQI to Message Manager
  - Handles task termination
- Message Manager
  - Implements MQOPEN, MQCLOSE, MQGET,MQPUT, MQINQ, MQSET
  - Object (Queue, process, namelist) definition
  - Triggering
  - Alias & remote name resolution
  - Directly implements most MQI semantics
- Data Manager
  - Manages message lists in buffers
  - Space management
  - Unit- of- work participation
  - REDO and UNDO logging
  - Acquire commit duration locks on queues, pages, messages
  - Pageset restart recovery
- Buffer Manager
  - Performs DASD pageset I/ O
  - Caches most recently referenced pages
  - “Steal” and “no- force” buffer policies
  - Writes “dirty” buffers asynchronously
  - Enforces “log write- ahead rule”
  - Same buffers for persistent &
  - non- persistent messages

What is MQSeries?

OS/390 QUEUE MANAGER COMPONENTS (CON'T):

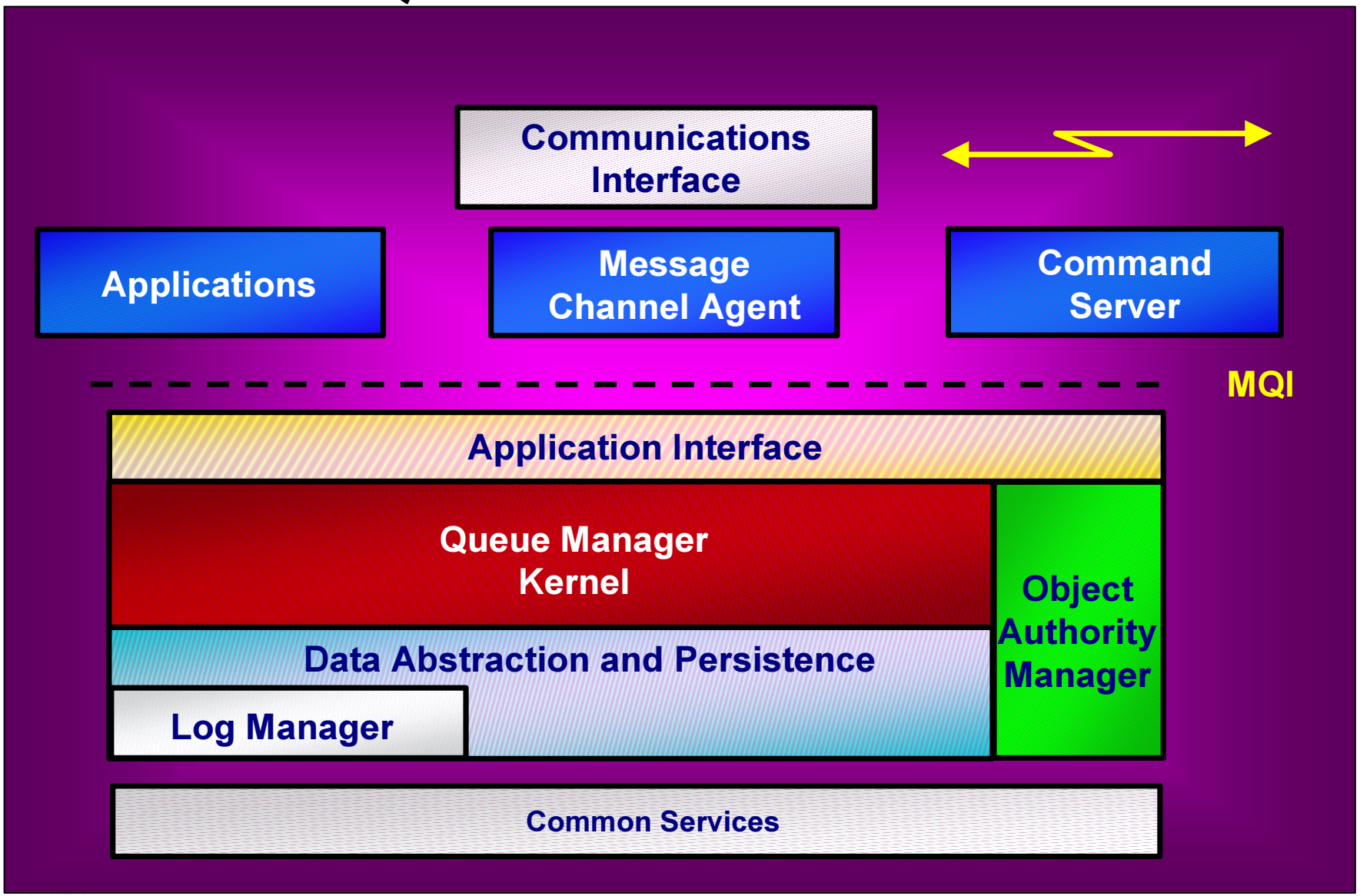
- Recovery Manager
  - Coordinates transaction participation
  - Manages transaction state transitions
  - Coordinates checkpoint
  - Coordinates restart/ recovery
- Log Manager
  - Log read and write functions
  - Multiple active log datasets and archive
  - Archive inventory management
  - Duplexed for reliability
  - "Bootstrap" file used to locate end of log and for archive inventory
- Lock Manager
  - Enforces MQOPEN sharing rules
  - Arbitrates between competing requests for "next message"
  - Prevents deletion of in-use objects
  - No lock waits
  - Deadlock prevented
  - Message (record) level locking
  - Hierarchical locking (queue, page, message)
  - X- lock implicitly locks all lower levels
  - Intention (IX) lock acquired top down
  - Locks released bottom- up

What is MQSeries?



Telcordia Technologies
Performance from Experience

What is MQSeries?



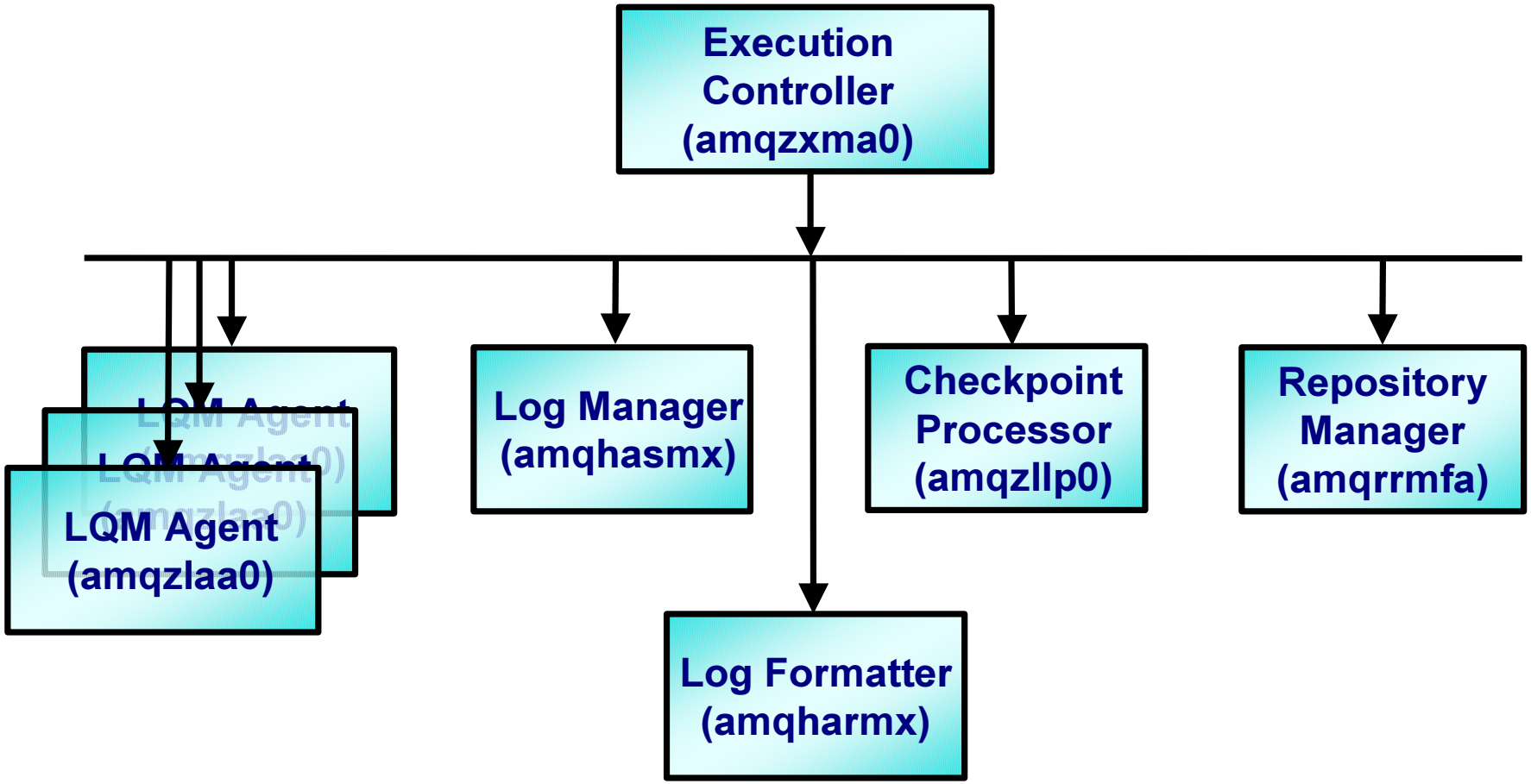
Telcordia Technologies
Performance from Experience

What is MQSeries?

DISTRIBUTED QUEUE MANAGER MAJOR COMPONENTS:

- Application Interface
  - It provides the environment and mechanism for execution of MQI calls.
- Queue Manager Kernel
  - It provides most of the function of the MQI. For example, triggering is implemented here.
- Object Authority Manager (OAM)
  - It provides access control for the queue manager and its resources. It allows specification of which users and groups are permitted to perform which operations against which resources.
- Data Abstraction and Persistence (DAP)
  - It provides storage and recovery of the data held by the queue manager. DAP holds the messages.
- Log Manager
  - This maintains a sequential record of all persistent changes made to the queue manager. This record is used for recovery after a machine crash and for remembering which operations were in which transaction.
- Message Channel Agents
  - These are special applications using the MQI for the majority of their operations. They are concerned with reliable transmission of messages between queue managers. This code is portable too. To add a new protocol, you only have to write a new implementation of the communications interface.
- Command Server
  - Again, this is a special application. It is concerned with processing messages containing commands to manage the queue manager.
- Common Services
  - This encapsulates the rest of the queue manager from the operating system. It provides a set of operating system-like services such as storage management, serialization and process management.

What is MQSeries?



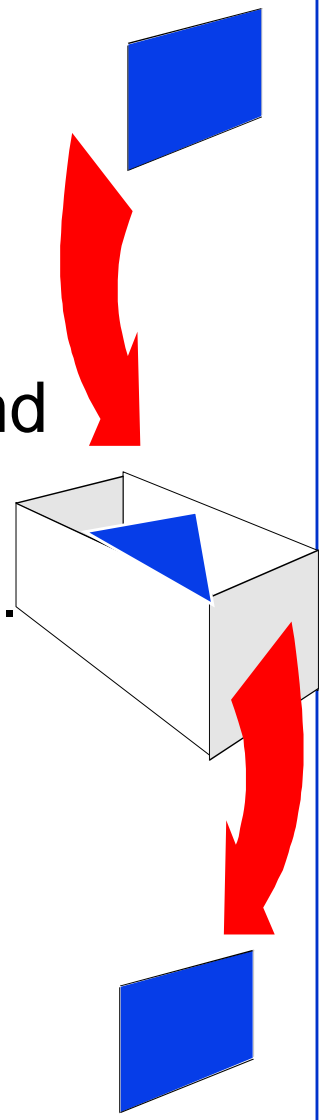
Telcordia Technologies
Performance from Experience

What is MQSeries?

- These are the processes you see when a distributed queue manager is running. The example is taken from AIX although OS/2, NT and other Unix ports are similar. The OS/400 process structure is different, but still contains much the same blocks of code.
- The Execution Controller is program **amqzxma0**. This is the root process of the queue manager and the parent of all of the other processes. It can be thought of as the owner of all of the queue manager's shared resources. It is concerned with managing and monitoring the other queue manager processes and the MQI applications.
- The LQM agents are program **amqzlaa0**. Agents perform the operations required to process MQI calls on behalf of applications. Nearly all of the code beneath the MQI is actually executed by the agents.
  - The separation of application programs from the queue manager's critical resources protects the queue manager from rogue or malicious applications.
  - The number of agent processes depends on the workload and whether the queue manager is threaded on a particular platform. On single threaded platforms there will be one agent for each connection to the queue manager. On threaded platforms agents can each handle about 60 concurrent connections. On Unix platforms, (before CSD3 for V5) the agents are always single threaded.
- The log manager is program **amqhasmx**. All log reading and writing requests go through this process. Communication with the agents is achieved through a set of shared memory buffers.
- If a queue manager is running with linear logging enabled, in support of media recovery, there will be a log formatter running, This program is **amqharmx**. Its task is to pre-format log files in time for the log manager to use them. This process is not needed when a queue manager is running a circular log as all the log files are created and initialized when the queue manager is created.
- The checkpoint processor is program **amqzllp0**. It is concerned with minimizing the amount of recovery processing when the queue manager is started.
- New to V5.1 is the repository manager process **amqrrmfa** for handling cluster queues. The channel initiator runmqchi is also now automatically started.

MQSeries Queues

- A queue is the equivalent of an IMS or CICS transaction code (messages are enqueued to 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
  - 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

- 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

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
- 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

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

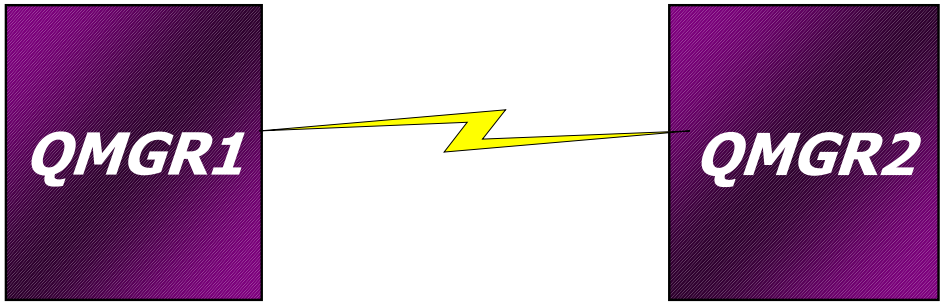
MQSeries Queues

- 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

- Which combination is best for you?
  - Simplest: Sender-Receiver
  - If you need to announce availability: Requester-Server
  - If you want the sender to “pay for the call”: Requester-Sender
- Channels also have exit points available for manipulating messages, enforcing security, etc.

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

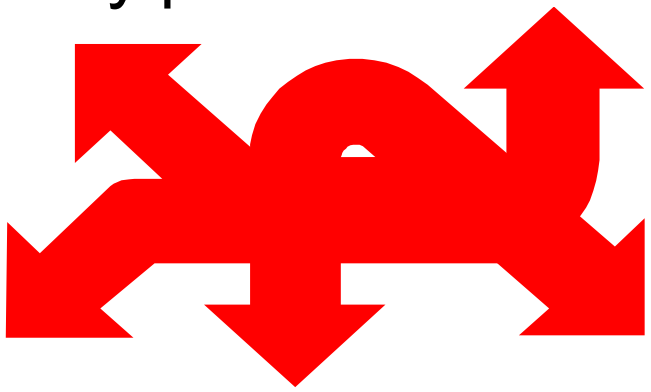
display channel(QMGRA.TO.QMGRB) all

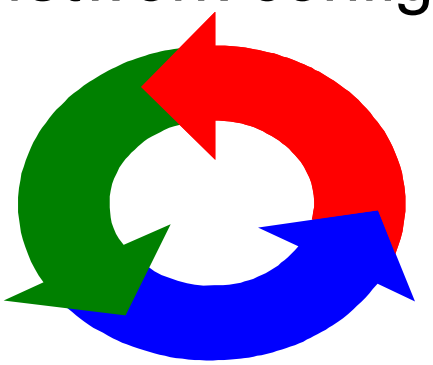
CHANNEL(QMGRA.TO.QMGRB)
CHLTYPE(RCVR)
TRPTYPE(TCP)
DESCR(Receiver channel for QMGRA)
BATCHSZ(50)
SCYEXIT()
MSGEXIT()
SENDEXIT()
RCVEXIT()
SEQWRAP(999999999)
MAXMSGL(4194304)
PUTAUT(DEF)
SCYDATA()
MSGDATA()
SENDDATA()
RCVDATA()
MCAUSER()
MREXIT()
MRDATA()
MRRTY(10)
MRTMR(1000)
NPMSPEED(FAST)
HBINT(300)

Telcordia Technologies
Performance from Experience

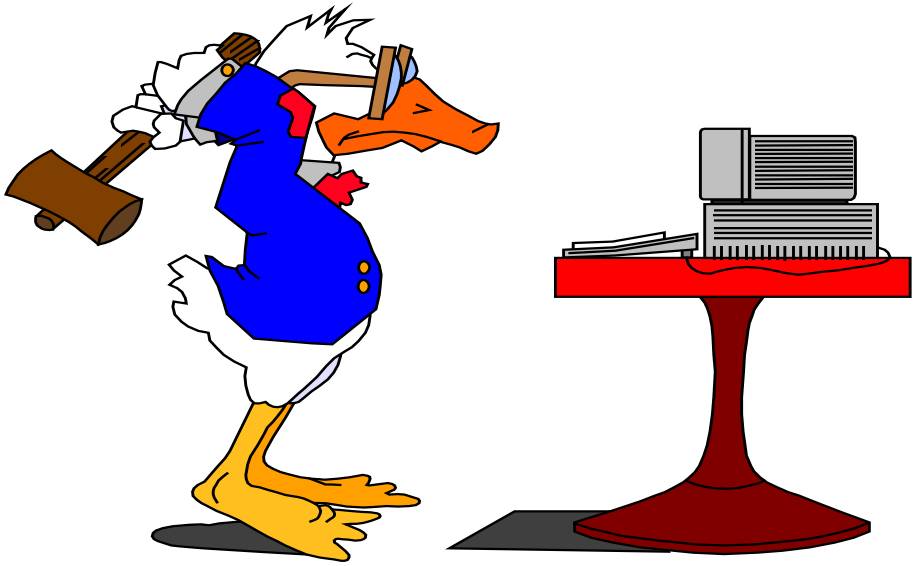
MQSeries Channels

- AVOID the scenarios below by taking the time to properly plan and test out your network configuration!





- Otherwise you may do this...

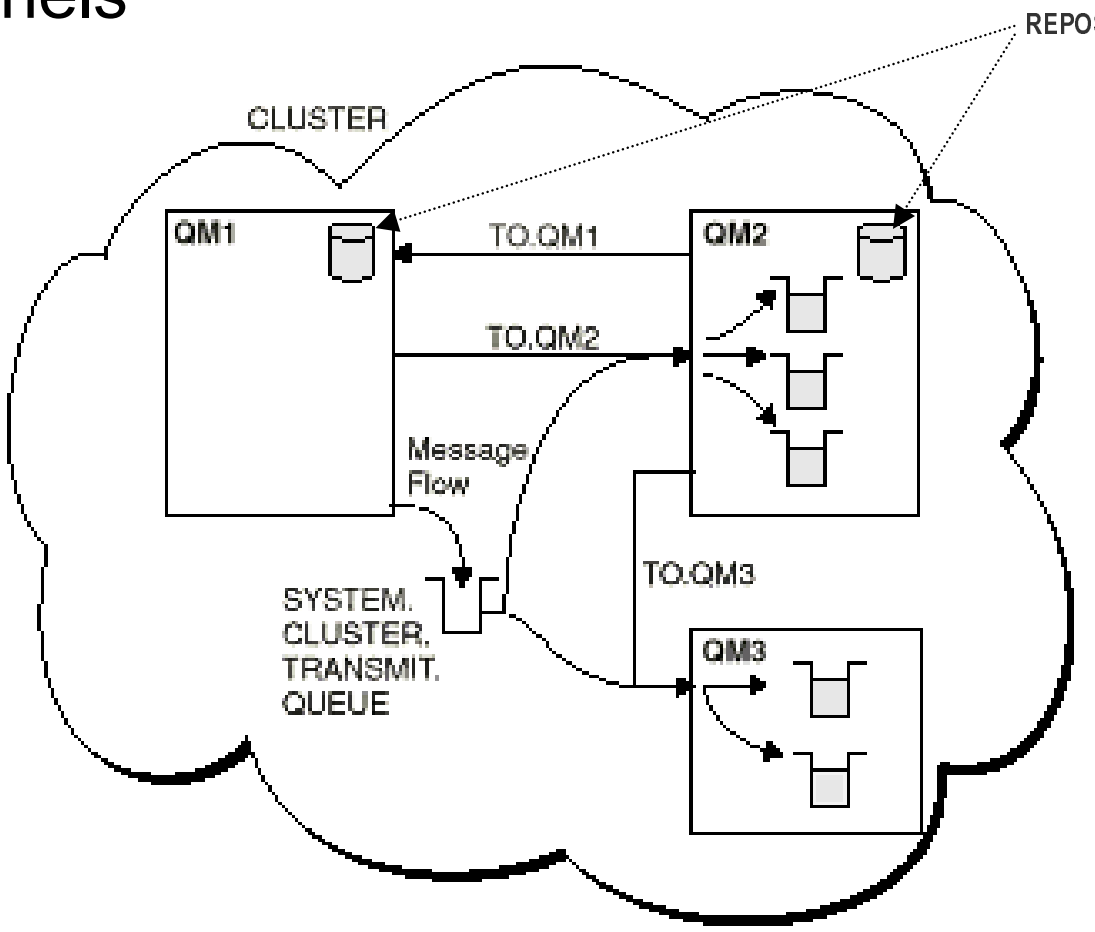


Telcordia Technologies

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 dynamically using a directory service called a repository (removes the need to pre-define remote queues, transmit queues, channels)
  - Windows NT comes with a cluster wizard
- 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
  - Scalability is easily addressed
- 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)

MQSeries Shared Queues

- Feature coming in MQSeries for OS/390 2.2…. When?
- 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

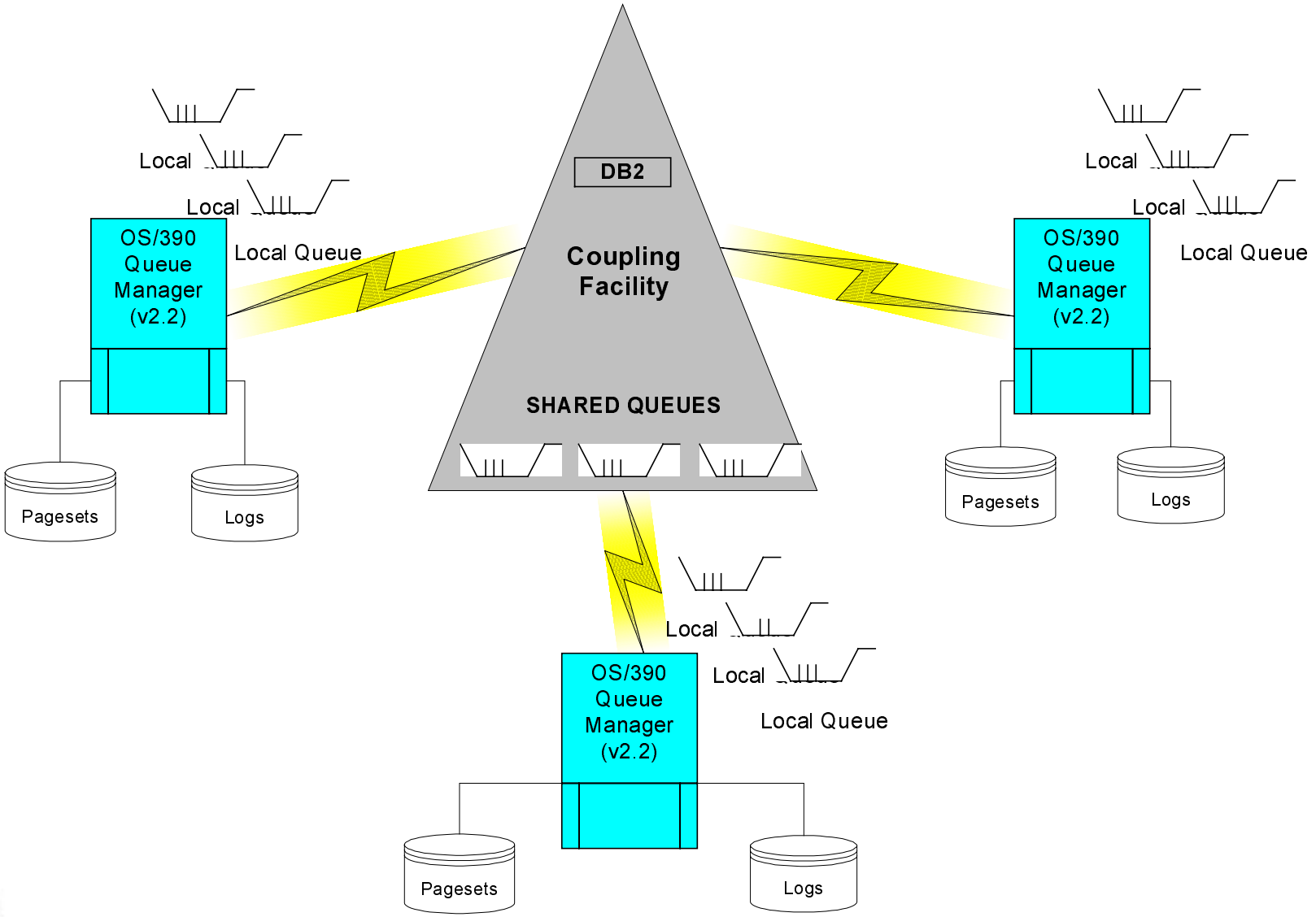
Telcordia Technologies
Performance from Experience

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 2.2
  - 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

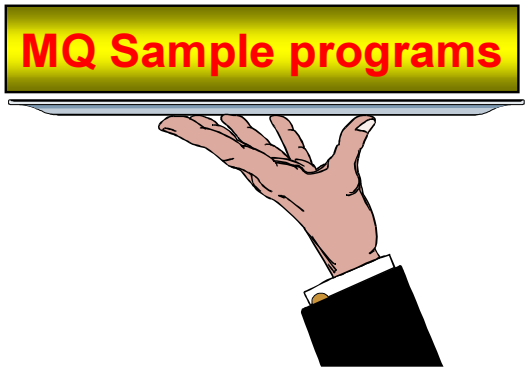
Message Queuing Interface (MQI)

- An MQSeries message is made up of two components:

- On most platforms the current maximum size for a message is 100MB (MVS is still 4MB)
- 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

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
  - 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!)

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);
```

- 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)
- MQBACK
  - Backs out the current unit of work (not under IMS)
- MQBEGIN
  - Begins a unit of work (not under IMS)
  - 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

A day in the life of an MQ Message

- An MQSeries message is “born” in an application program (either one you’ve written, bought, etc.)
  - Data is placed in a buffer
  - If using an MQSeries “bridge” (e.g., IMS Bridge) a special header may be placed at the front of the application data
  - MQSeries Message Descriptor (MQMD) fields are set
  - MQOPEN, MQPUT options are set
  - Message is inserted/PUT to a “queue”
  - Security checking is performed throughout this process (from MQCONN to MQI options and other MQI verbs)
- The queue manager performs “queue name resolution”
  - Alias, remote, or local queue names resolved
  - If no destination can be resolved, a failure return code goes back to the application
- Message in placed in a queue and logged (if persistent)
  - Message may later be written to disk if not accessed

A day in the life of an MQ Message

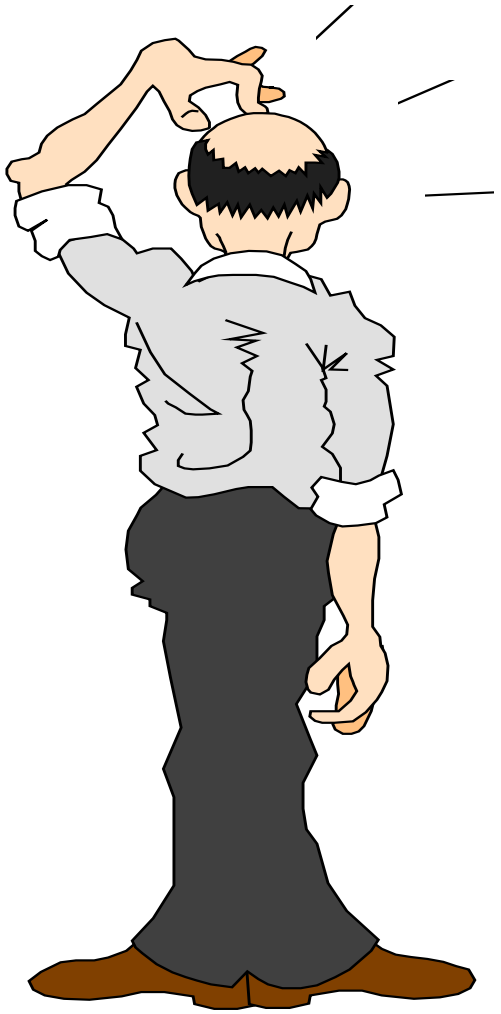
- If the destination queue is remote, the message will end up on a transmission queue
  - An MQSeries channel will transport the message in its next scheduled batch (which is configuration dependent)
  - Sending and receiving channels exchange data and commit
  - Receiving channel issues MQI calls to place the message on its next destination
    - Remote queue manager again performs queue name resolution (which could send it on to yet another queue manager)
    - If destination cannot be resolved, the message goes on the dead-letter queue
    - If no dead-letter queue exists, the message is returned to the sending transmit queue and the channel shuts down (this maintains integrity)

A day in the life of an MQ Message

- If the destination queue is local, the message is placed on that queue directly
  - The queue manager checks for application triggering and starts a process if needed
  - Report messages may be sent by the local queue manager
  - Message could be sent to another application if this is a “bridge queue”
    - CICS, IMS, SAP R/3 Link, Lotus Notes are some types of “bridges”
- The message remains on the local queue until it is retrieved (MQGET), expired, or deleted at queue manager restart (non-persistent)
- That’s it! Very simple… right?

Telcordia Technologies
Performance from Experience

Questions?



Telcordia Technologies

Performance from Experience

RETURN TO INDEX

Telcordia Technologies

Performance from Experience

About Science Applications International Corporation

SAIC is one of the world's leading providers of systems integration, information management, data security, and network solutions. SAIC and its subsidiary, Telcordia Technologies, have an unsurpassed record in helping clients succeed with end-to-end information technology and networking solutions. For more information about SAIC, please call +1-619-546-6000 or visit the SAIC home page at www.saic.com.