

WebSphere MQ for z/OS Shared Queues

Paul Dennis
WebSphere MQ for z/OS Development



Agenda

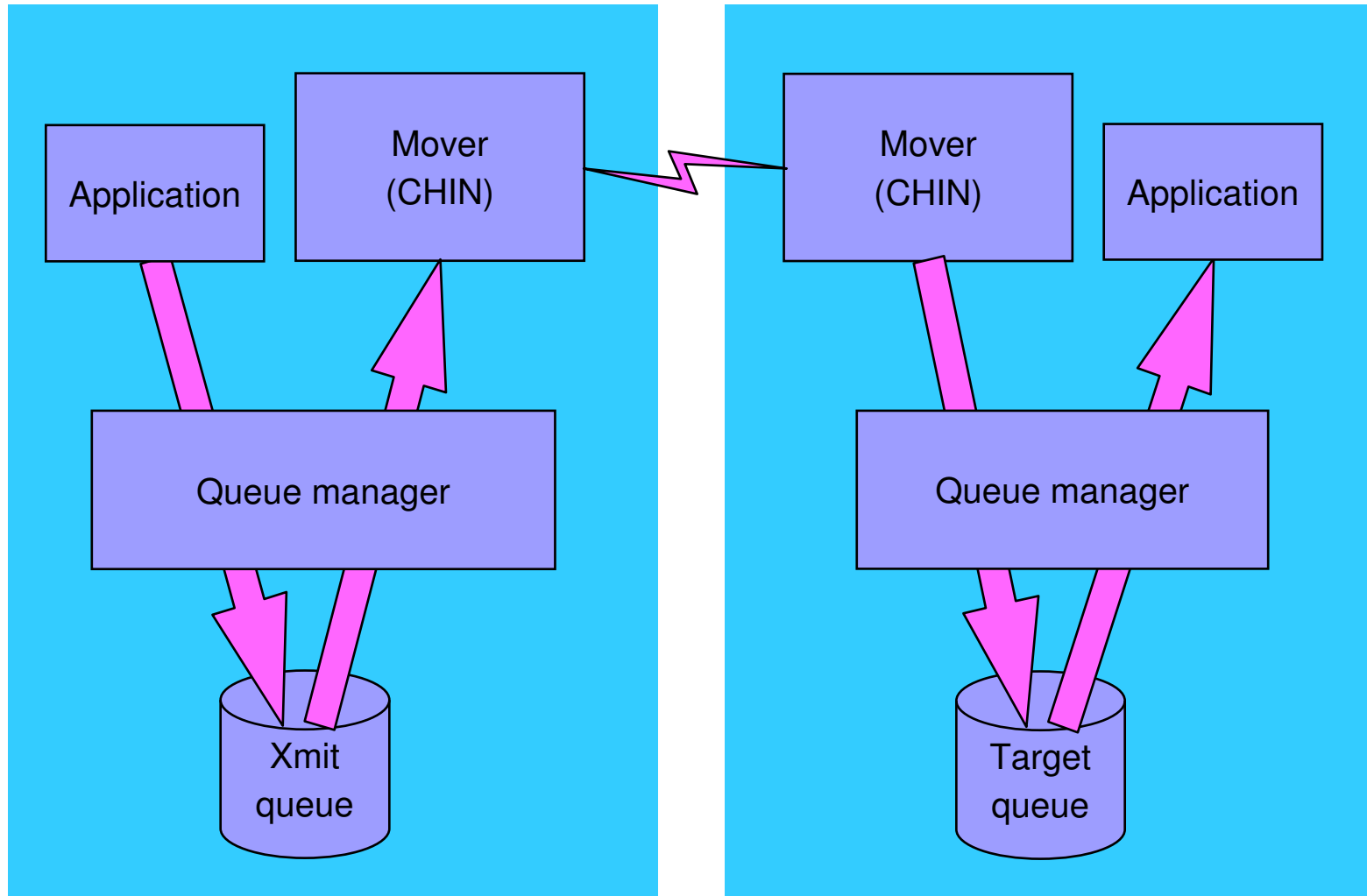
- What shared queues are
 - Shared queues
 - Queue-sharing groups
 - Coupling Facility (CF) structures
 - Persistence and transaction integrity
- Configuring channels with shared queues
 - Inbound channel configurations
 - Outbound channel configurations
- Exploiting shared queues
 - Availability benefits of queue sharing
 - Scalability

Shared Queues?

- Function:
 - ✓ Multiple Queue Managers can access the same shared queue messages
 - ✓ Multiple Queue Managers can access the same shared queue objects
- Benefits
 - ✓ Availability for new messages
 - ✓ Availability for old messages
 - ✓ Pull workload balancing
 - ✓ Scalable capacity
 - ✓ Low cost messaging within a sysplex

This page intentionally blank

Mover



Mover

N

O

T

E

S

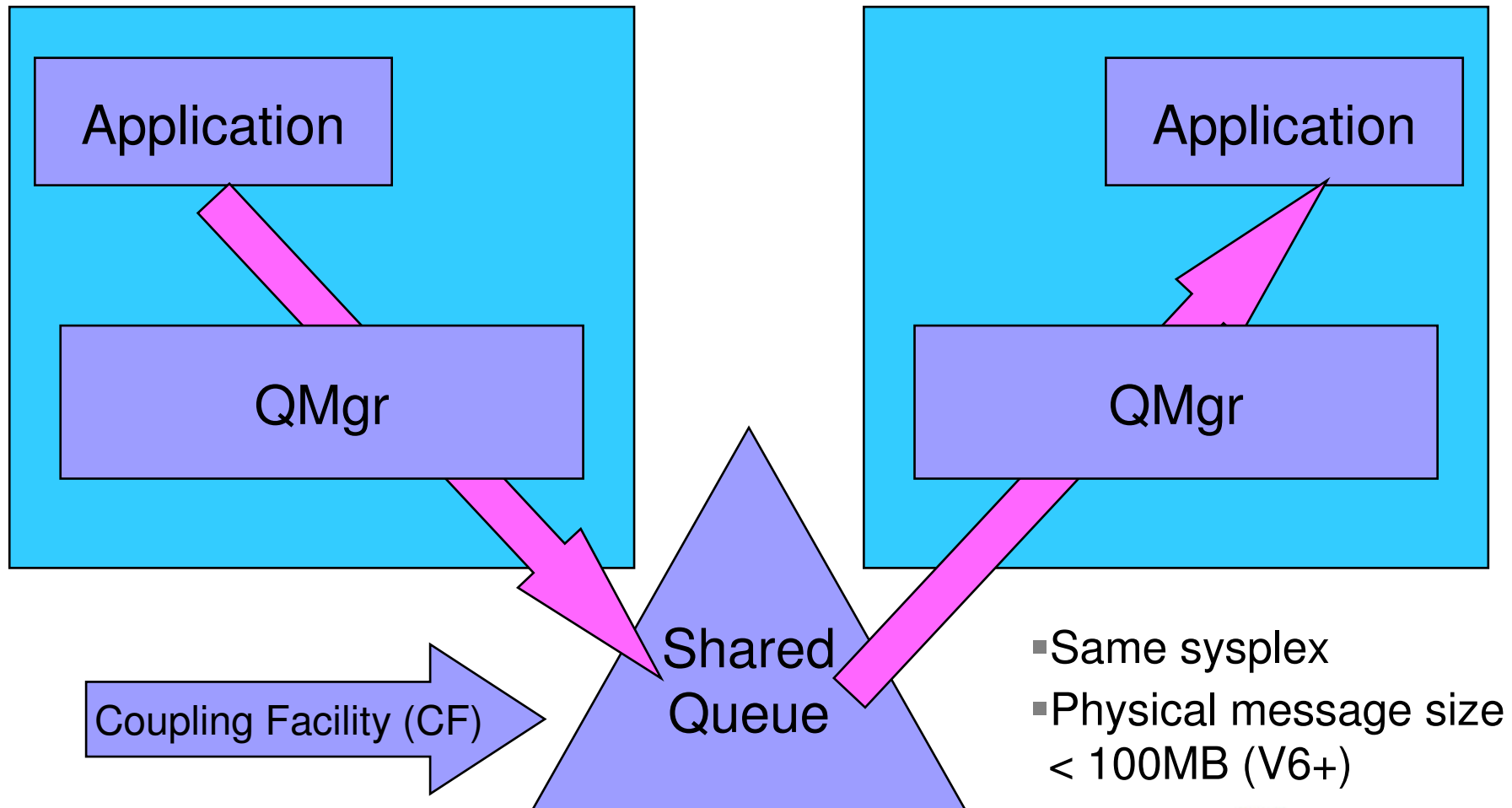
Chart shows an application put to a remote target queue -- that is, the target queue is local to another queue manager. This put uses the mover as follows:

1. Application puts to remote target queue
2. Queue manager puts message on local transmit queue
3. Local mover gets message and sends to remote mover
4. Remote mover puts message to target queue
5. Remote application can now get the message.

The remote application can put a message to the reply-to queue using the same method.

Note that only applications connected to the target queue manager can get the message.

Shared Queues



Shared Queues

N

Chart shows an application put to a shared target queue -- that is, the target queue is local to more than one queue manager. This put does not use the mover:

O

1. Application puts to shared target queue
2. Remote application can now get the message.

T

The remote application can put a message to the reply-to queue using the same method.

E

Note that applications connected to any queue manager with access to the shared queue can get the message. To access the same shared queues, queue managers must be:

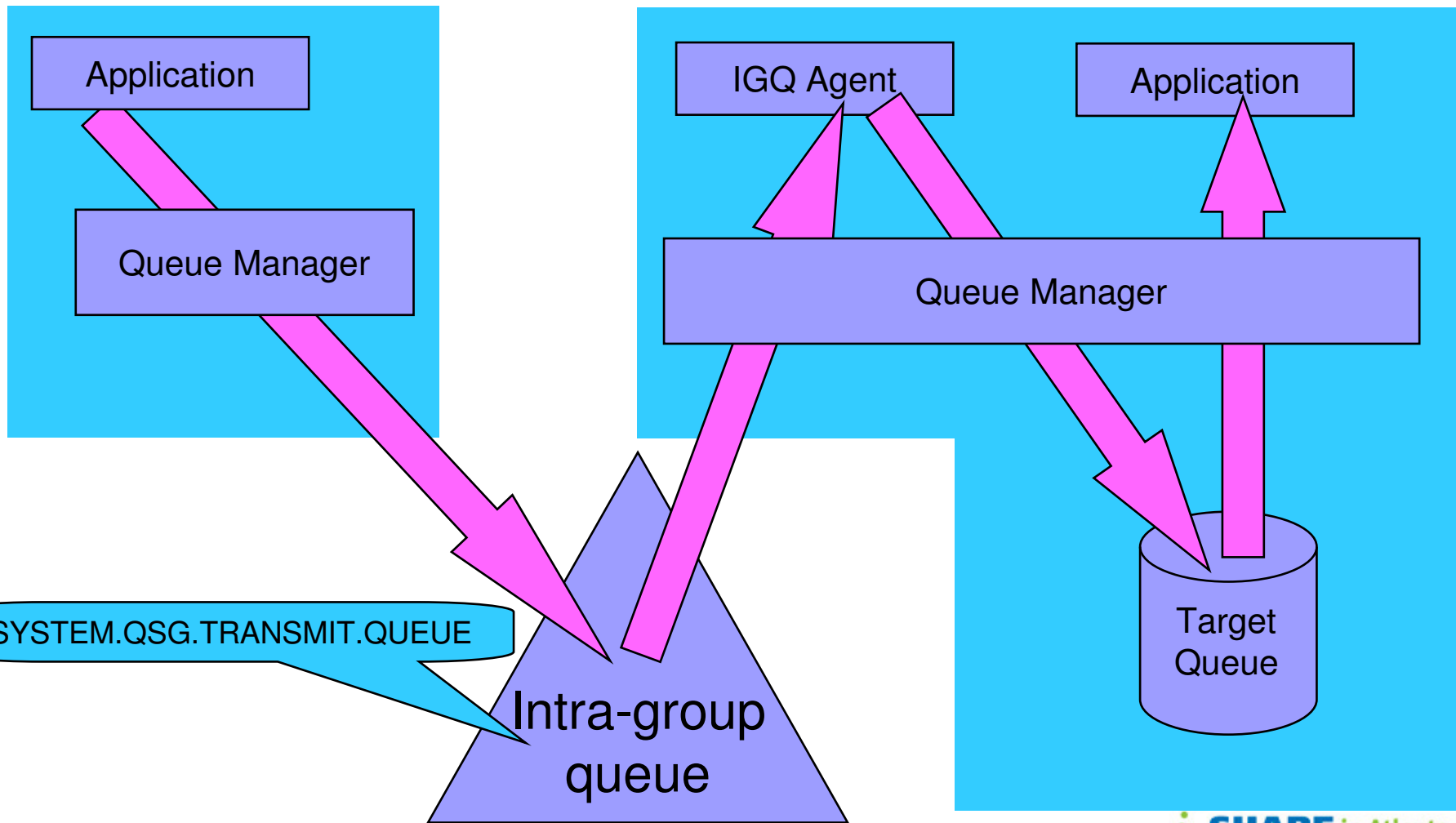
S

- In the same z/OS sysplex
- In the same queue-sharing group (QSG) -- we will explain QSGs later.

There are restrictions on shared queues, for example:

- CF capacity is limited (compared to DASD).

Intra-Group Queue



Intra-Group Queue

N

O

T

E

S

Chart shows an application put to a remote target queue -- but the target queue is local to another queue manager in the same queue-sharing group. This put uses the intra-group queuing (IGQ):

1. Application puts to remote target queue
2. Queue manager puts message on shared IGQ queue
3. Remote IGQ agent gets message from IGQ queue
4. Remote IGQ agent puts message to target queue
5. Remote application can now get the message

The IGQ method is similar to the mover method but more efficient -- that is, it uses much less processor power.

There is one IGQ queue for each queue-sharing group. It has the reserved name `SYSTEM.QSG.TRANSMIT.QUEUE`.

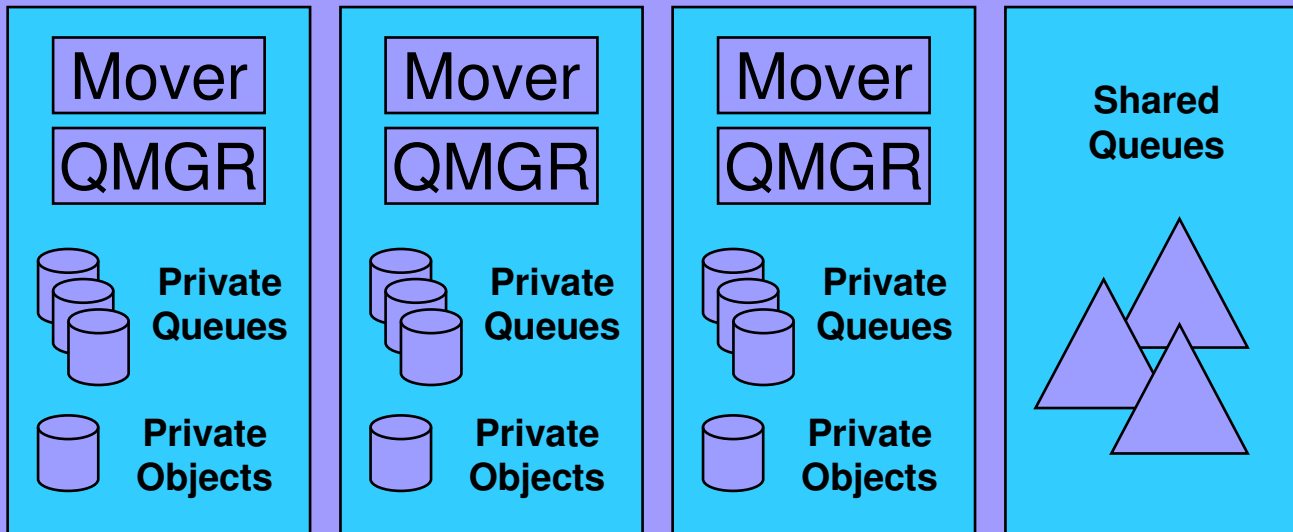
The IGQ queue (and the IGQ method) has the same restrictions as any shared queue.

If IGQ cannot be used (large message, no IGQ queue defined, or whatever) then the put will be handled by the mover (assuming there is a suitable channel).

Queue Sharing Groups (QSGs)

DB2 Data Sharing Group

WebSphere MQ Queue Sharing Group



Queue-Sharing Groups (QSGs)

N

Chart shows how queue managers are organized into queue-sharing groups (QSGs) and the relationship to DB2 data-sharing groups.

O

A queue-sharing group can contain one or more queue managers:

- Each queue manager has its own private (not shared) queues and object definitions.
- All queue managers in a QSG share the same set of shared queues and shared object definitions
- A queue manager cannot belong to more than one QSG.

T

Shared object definitions for a QSG are maintained for WebSphere MQ by DB2. Shared access to these definitions is by DB2 data sharing:

- You must have DB2
- You can have more than one data-sharing group, but all members of one QSG must be members of the same data-sharing group
- Shared object definitions are cached in the queue managers.
- A DB2 outage does not bring down the QSG (but you cannot add or change shared objects if DB2 is down).

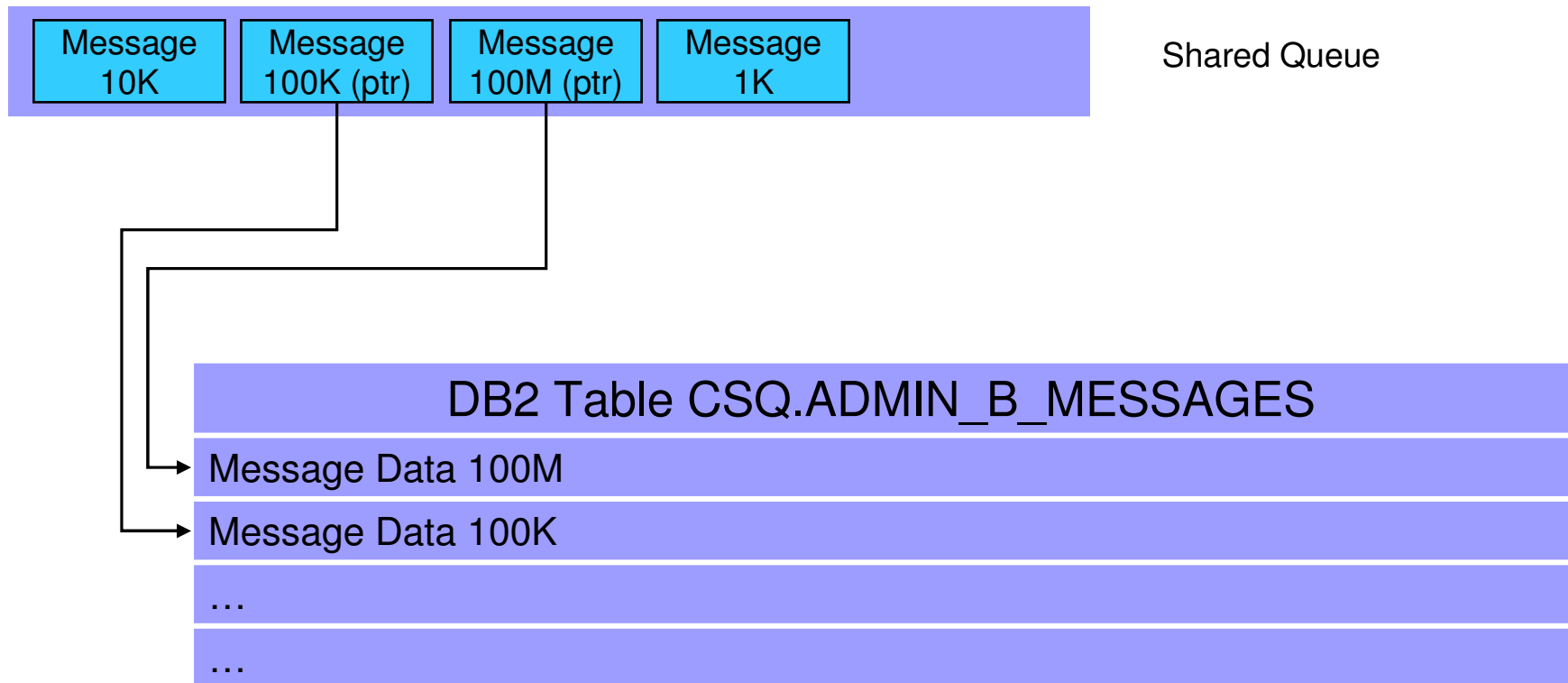
E

You do not have to define any queue-sharing groups if you do not run a sysplex (or if you just don't want to).

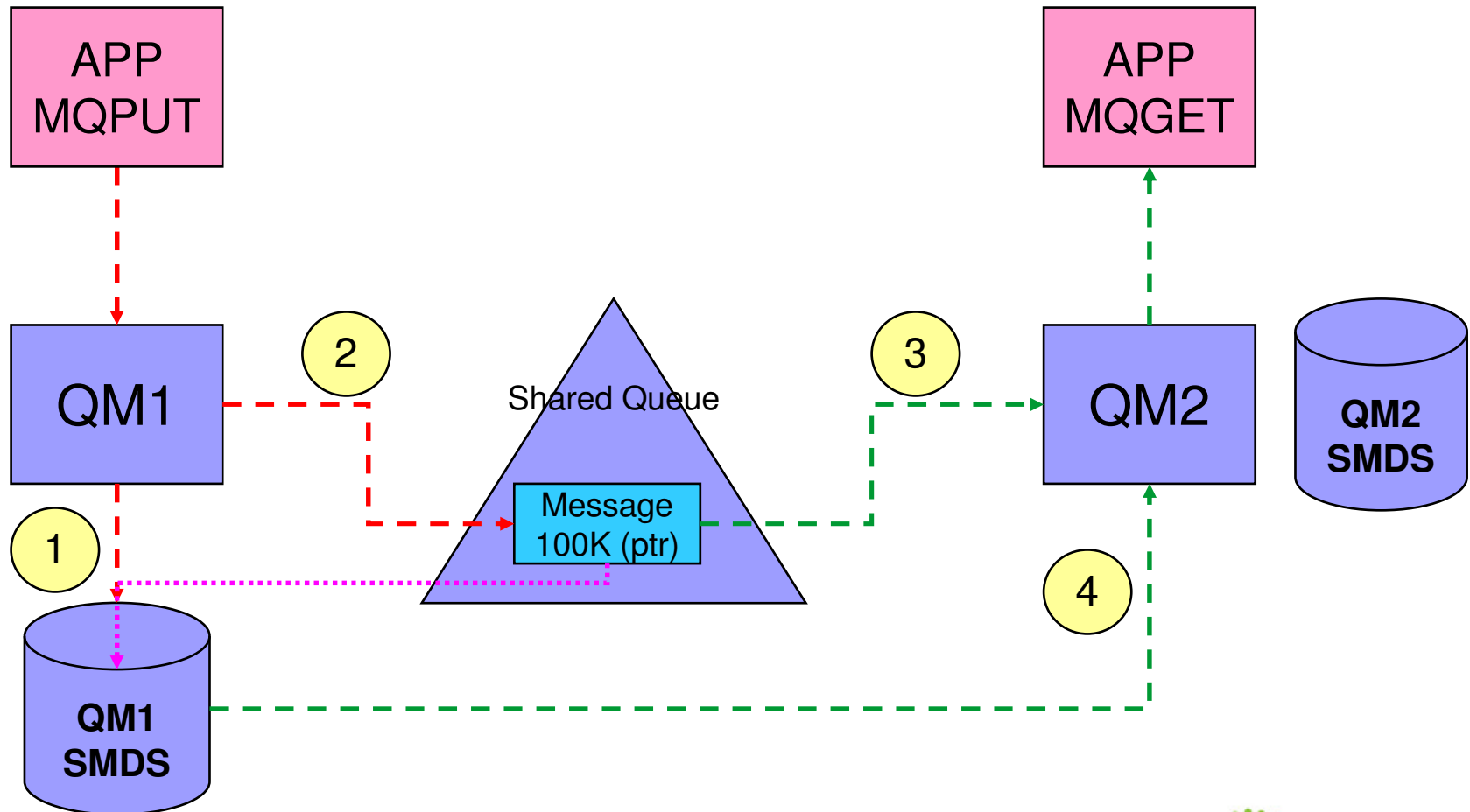
S

If using shared messages > 63KB then a small portion for the message is stored in the CF, and the rest is stored in DB2 or with V7.1 or higher there is the option of using Shared Message Data Sets (SMDS) for storing the rest of the message data.

Large Shared Queue Messages (using DB2)



Large Shared Queue Messages (using SMDS)



Shared message data set concepts

V7.1

N

Offloaded message data for shared messages is stored in data sets.

O

Each application structure has an associated group of shared message data sets, with one data set per queue manager.

Named using DSGROUP parameter on CFSTRUCT definition.

T

Each queue manager owns a data set for each structure, opened for read/write access, which it uses to write new large messages.

E

Each queue manager opens the data sets for the other queue managers for read-only access, so it can read their message data.

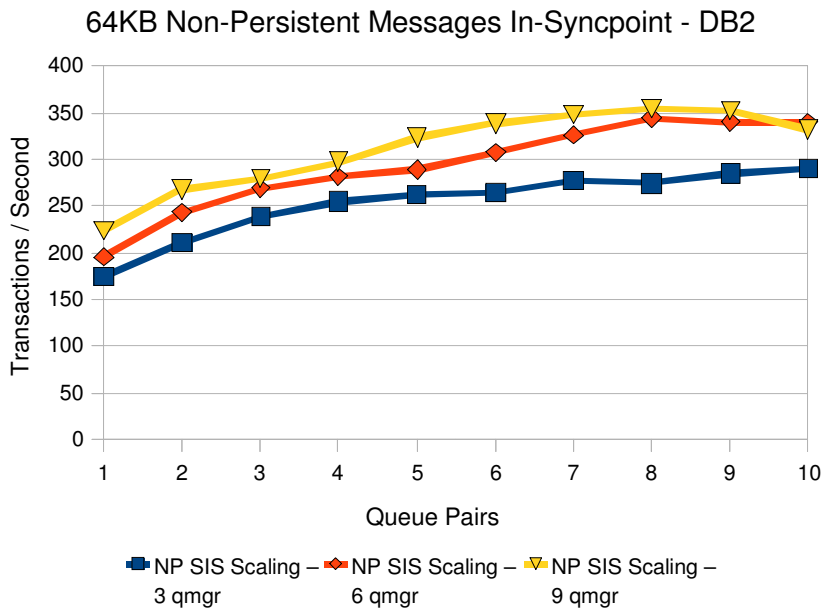
S

When a message with offloaded data needs to be deleted, it is passed back to the queue manager which originally wrote it, so that the queue manager can free the data set space when it deletes the message.

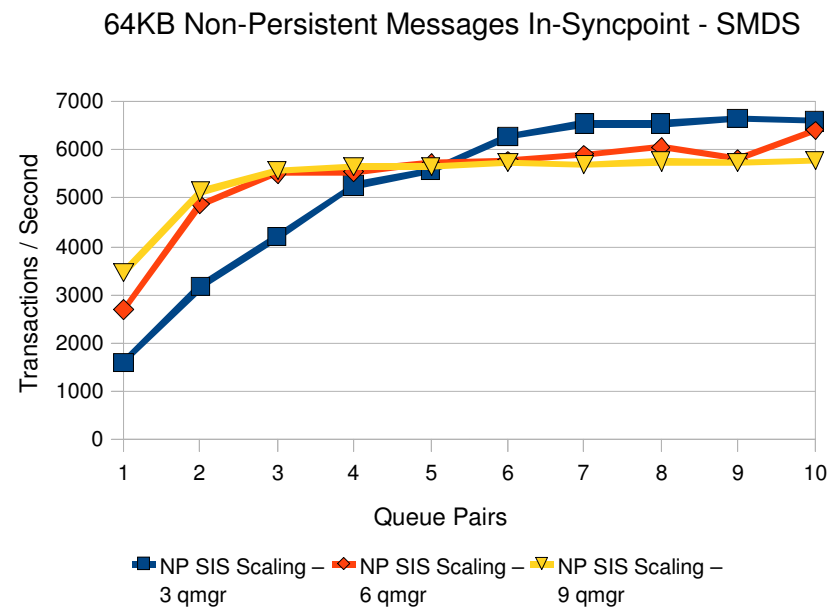
SMDS Performance Improvement

V7.1

•3 LPAR Test - DB2



•3 LPAR Test - SMDS



- Early Test Results on z196
- Tests show comparable CPU savings making SMDS a more usable feature for managing your CF storage
- SMDS per CF structure provides better scaling than DB2 BLOB storage

Selecting which messages to offload

V7.1

- Messages too large for CF entry ($> 63\text{K}$ bytes) are always offloaded.
- Other messages may be selectively offloaded using offload rules.
 - Each structure has three offload rules, specified on the CFSTRUCT definition.
 - Each rule specifies message size in Kbytes and structure usage threshold, using two parameters:
 - OFFLDnSZ(size) and OFFLDnTH(percentage), where $n = 1, 2, 3$.
 - Data for new messages exceeding the specified size is offloaded (as for a large message) when structure usage exceeds the specified threshold.
 - Default rules are provided which should be useful in most cases.
 - Rules can be set to dummy values if not required.
- Without offloading data, it is possible to store 1.25M messages of 63KB on a 100GB structure
- When offloading all messages, it is possible to store 45M messages on the same structure, irrespective of message size

Selecting which messages to offload



V7.1

N

As with previous releases of MQ, the amount of data that can be stored in the CF for a single message is limited to 63KB. This means that if the message is over 63KB in size then it must be “offloaded”. With V7.1 there are two offload methods, already seen, for offloading the data, using either DB2 or SMDS.

O

In addition to offloading all messages over 63KB, it is possible to specify that messages smaller than this size should also be offloaded. There are three sets of rules that are used to control this, and each set is made up of two parameters, the size of the message and how full the CF structure is when the message is put. These offload rules enable a balancing to be performed between performance and CF capacity. For example, you might use a rule that says that when the CF structure is 70% full then all messages over 32KB will be offloaded, and then another rule that says that when the CF structure is over 80% full, all messages over 4KB will be offloaded. When migrating a structure from CFLEVEL 4 to CFLEVEL 5 (required to use these rules), the defaults will be set to mimic the CFLEVEL 4 behavior. When defining a new structure at CFLEVEL 5, the default rules will be set as follows:

T

E

S

OFFLD1SZ = 32K OFFLD1TH = 70
OFFLD2SZ = 4K OFFLD2TH = 80
OFFLD3SZ = 0K OFFLD3TH = 90

Creating a queue-sharing group

Use the CSQ5PQSG utility to create a QSG:

1 Add the QSG into the DB2 tables:

```
//stepname EXEC PGM=CSQ5PQSG,  
//      PARM='ADD QSG, qsg-name, dsg-name, DB2-ssid'
```

2 Add the queue managers into the DB2 tables as members of the QSG:

```
//stepname EXEC PGM=CSQ5PQSG,  
//      PARM='ADD QMGR, qmgr-name, qsg-name, dsg-name, DB2-ssid'
```

qsg-name	Name for the queue-sharing group
qmgr-name	Name of the queue manager
dsg-name	Name of the DB2 data-sharing group
DB2-ssid	DB2 subsystem ID

Creating a queue-sharing group

N

Chart shows JCL fragments for defining a QSG and adding a queue manager into a QSG (for more complete information, see *System Administration Guide*).

O

You can also use CSQ5PQSG to remove a queue manager from a QSG. To do this, you must ensure that the queue manager cannot have any recovery to do -- either:

- The queue manager was stopped normally, or:
- The queue manager has never been started.

T

You can also use CSQ5PQSG to delete a QSG which has no queue managers in it -- that is:

E

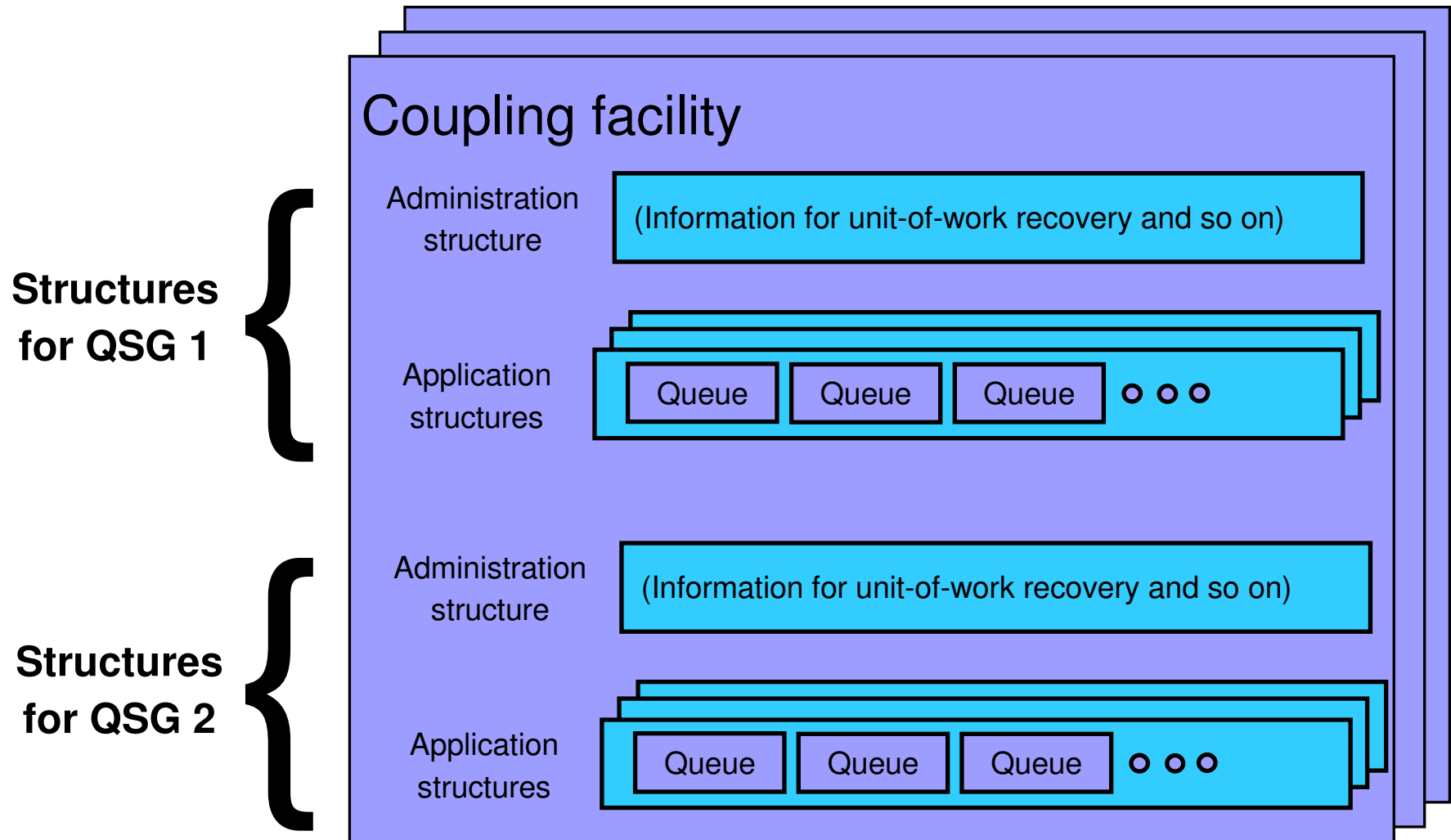
- You removed all the queue managers, or:
- You never added any queue managers in the first place.

S

To start a queue manager as a member of a QSG, you provide the QSG information in CSQZPARM.

We advise you do **not** update the Websphere MQ DB2 tables directly -- use the utilities we provide.

CF Structures for shared-queues



CF Structures for shared-queues

N

Chart shows organization of WebSphere MQ data in coupling facility (CF) structures (actually *list* structures).

O

For clarity the chart shows:

- All structures in one CF -- actually they can be spread arbitrarily over many CFs
- Only WebSphere MQ structures -- actually other subsystems and applications can have structures in the same CF as Websphere MQ.

T

Each queue-sharing group needs:

- One administration structure -- this is used for information that WebSphere MQ itself needs, for example to manage unit-of-work recovery
- One or more (up to a maximum of 63) application structures -- these are used to hold the shared queues.

E

S

Each application structure can hold up to 512 shared queues.

Creating CF structures and shared queues

- Define a structure to z/OS (not to WebSphere MQ) by updating the CFRM policy (see *System Setup Guide*):
 - Structure is known to WebSphere MQ by its 12-character *str-name*.
 - Structure is known to z/OS by the 16-character name formed by:
 - *qsg-name* || *str-name* (Application structures)
 - *qsg-name* || CSQ_ADMIN (Administration structure)
- Define a shared queue using the DEFINE QLOCAL command on any queue manager in the QSG:
 - DEFINE QLOCAL(*queue-name*) QSGDISP(SHARED) CFSTRUCT(*str-name*)
- z/OS creates the structure when required (first use).
- WebSphere MQ creates the queue when required (first use).

Creating CF structures and shared queues

N

Chart shows the processes for creating CF list structures for use by WebSphere MQ QSGs and for creating shared queues in these structures.

O

The z/OS CFRM policy for the sysplex specifies how z/OS should allocate resources for each structure:

- What type of CF (for example, CF must have battery back-up)
- How big to make the structure.

T

z/OS does not actually allocate any resource for the structure until first use -
- in our case, the first time a queue manager connects to the structure:

- At startup for the administration structure
- At first queue open for application structures.

E

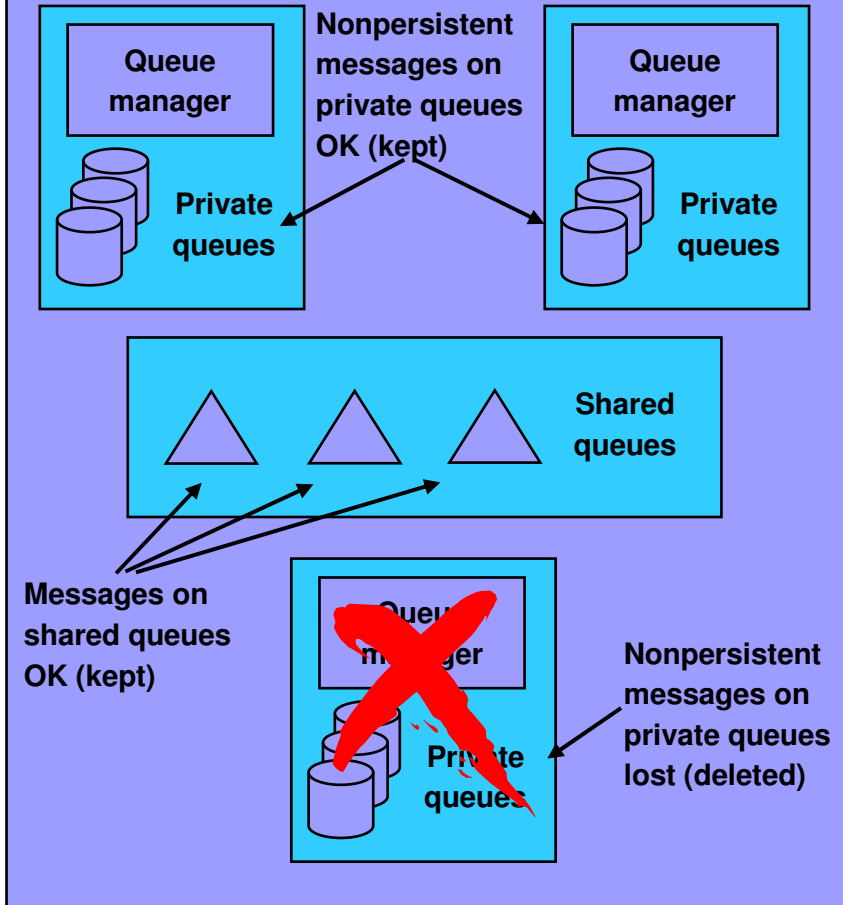
As with private queues, defining the queue to WebSphere MQ does not create the queue. The queue is created when it is first used.

S

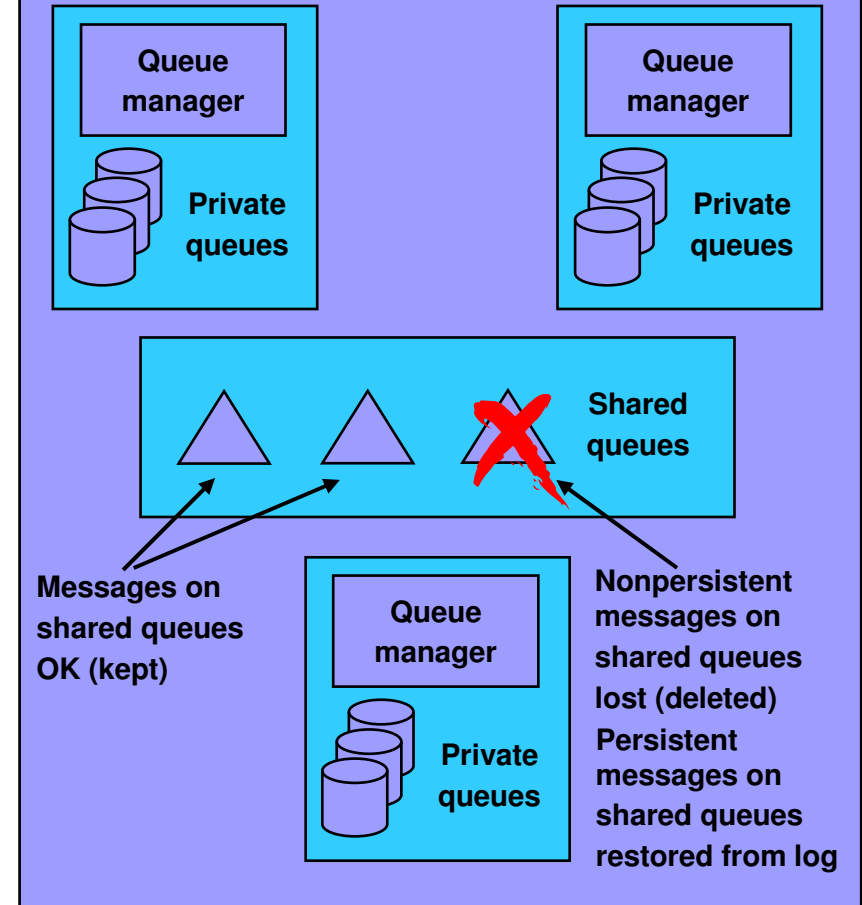
It is best to allocate queues so that (as far as possible) all the queues accessed by any one unit-of-work are in the same structure.

Failure and persistence

Queue manager failure



Coupling facility failure



Failure and persistence

N

O

T

E

S

Chart shows implications of failures in a queue-sharing group.

Left side of chart shows queue manager failure. If one or more queue managers in a queue-sharing group fail, or are stopped normally:

- Nonpersistent messages on queues private to the failing queue manager or managers are lost -- in fact they are deleted when a queue manager restarts
- Messages on shared queues are not lost, they are kept -- even if *all* queue managers in the queue-sharing group fail.

Right side of chart shows coupling facility structure failure (for simplicity the chart shows an entire CF failing). If one or more CF structures fail:

- Messages on queues in other CF structures are not lost
- Nonpersistent messages on queues in failing CF structures are lost
- Persistent messages on queues in failing CF structures must be restored from backup and log information on the logs
- Restoring queue manager accesses logs of all queue managers in the QSG.

If the administration structure fails, all the queue managers in the QSG fail.

Admin Structure Recovery



- Prior to V7.0.1 each queue manager would rebuild own admin structure entries
 - Particularly an issue in a DR situation.
 - Need to start all queue managers to rebuild admin structure
 - Once recovered, application structures could be recovered
- At V7.0.1 active queue managers notice if other queue managers don't have entries, and initiate rebuild on their behalf

Admin Structure Recovery



N

O

T

E

S

If the Admin Structure was lost for some reason (DR situation, loss of power to the CF etc), then prior to V7.0.1 each queue manager had to rebuild its own Admin Structure entries. As the admin structure needs to be complete for application structure recovery to take place, it was necessary in a DR situation to start up all the queue managers in a QSG before application structure recover could take place.

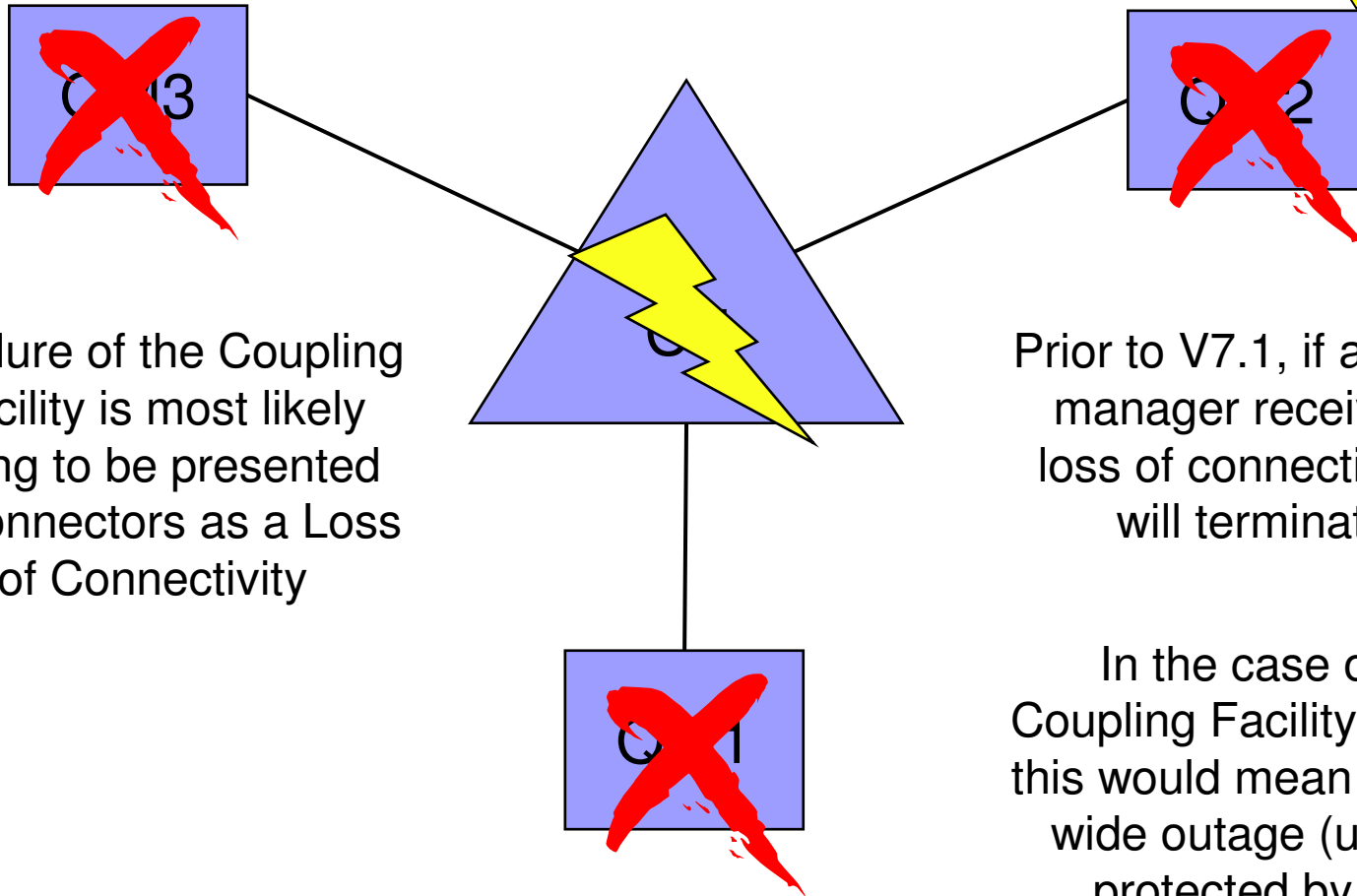
In V7.0.1 an enhancement has been made to admin structure recovery so that a single queue manager is able to recover the admin structure entries for all the other queue managers in the QSG. If a V7.0.1 (or higher) queue manager notices that the admin structure entries are missing for another queue manager then it will attempt to recover them on behalf of the other queue manager. It can only do this if the other queue manager is not running at the time. In a DR situation this means that it is only necessary to start a single queue manager at V7.0.1 (or higher) before being able to recover the application structures.

A V7.0.1 queue manager can recover the entries on behalf of any version of queue manager, so you don't need to have all queue managers in the QSG to be running at V7.0.1 before this functionality will take place.

CF Loss of Connectivity Tolerance

Pre V7.1 Queue Managers

V7.1



A failure of the Coupling Facility is most likely going to be presented to connectors as a Loss of Connectivity

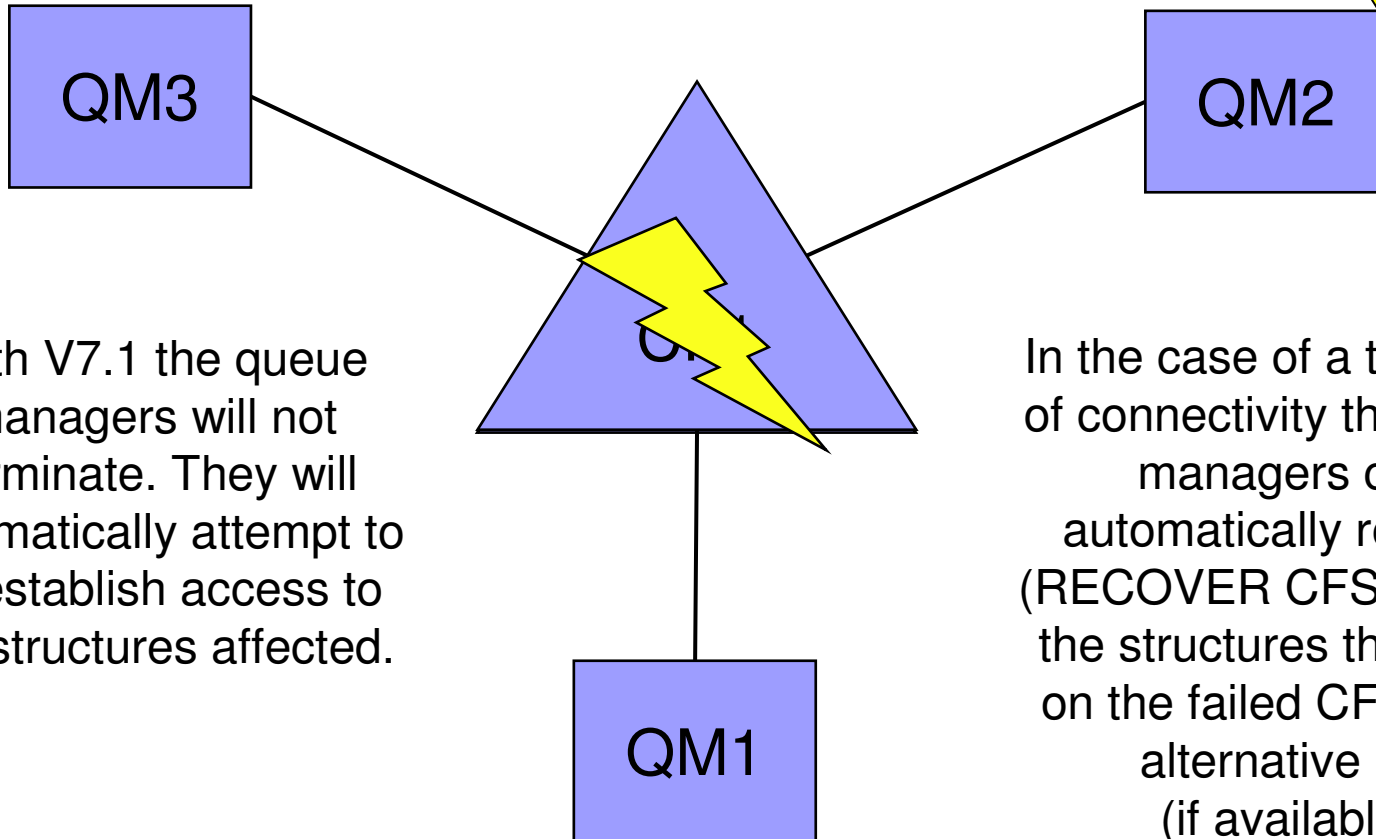
Prior to V7.1, if a queue manager receives a loss of connectivity, it will terminate.

In the case of a Coupling Facility failure, this would mean a QSG wide outage (unless protected by CF Duplexing)

CF Loss of Connectivity Tolerance

V7.1+ Queue Managers

V7.1



With V7.1 the queue managers will not terminate. They will automatically attempt to re-establish access to the structures affected.

In the case of a total loss of connectivity the queue managers can automatically recover (RECOVER CFSTRUCT) the structures that were on the failed CF into an alternative CF (if available)

Admin structure loss of connectivity

V7.1

N

Queue managers will tolerate loss of connectivity to the admin structure without terminating if:

- the QMGR CFCONLOS attribute is set to TOLERATE
- all the queue managers in the QSG are at V7.1

O

All queue managers in the QSG will disconnect from the admin structure, then attempt to reconnect and rebuild their own admin structure data.

T

If a queue manager cannot reconnect to the admin structure, for example because there is no CF available with better connectivity, some shared queue operations will remain unavailable until the queue manager can successfully reconnect to the admin structure and rebuild its admin structure data.

E

The queue manager will automatically reconnect to the admin structure when a suitable CF becomes available on the system.

S

Failure to connect to the admin structure during queue manager startup is not tolerated, regardless of the value of CFCONLOS.

Application structure loss of connectivity



N

Queue managers will tolerate loss of connectivity to application structures if:
they are at CFLEVEL(5)
the CFCONLOS attribute is set to TOLERATE

O

All queue managers that lose connectivity to an application structure will disconnect from the structure.

T

The next action depends on whether it is a partial or total loss of connectivity (according to MQ's definition).

E

- loss of connectivity is partial if there is at least one system in the sysplex that still has connectivity to the CF that the structure is allocated in.
- loss of connectivity is total if all systems in the sysplex have lost connectivity to the CF that the structure is allocated in.

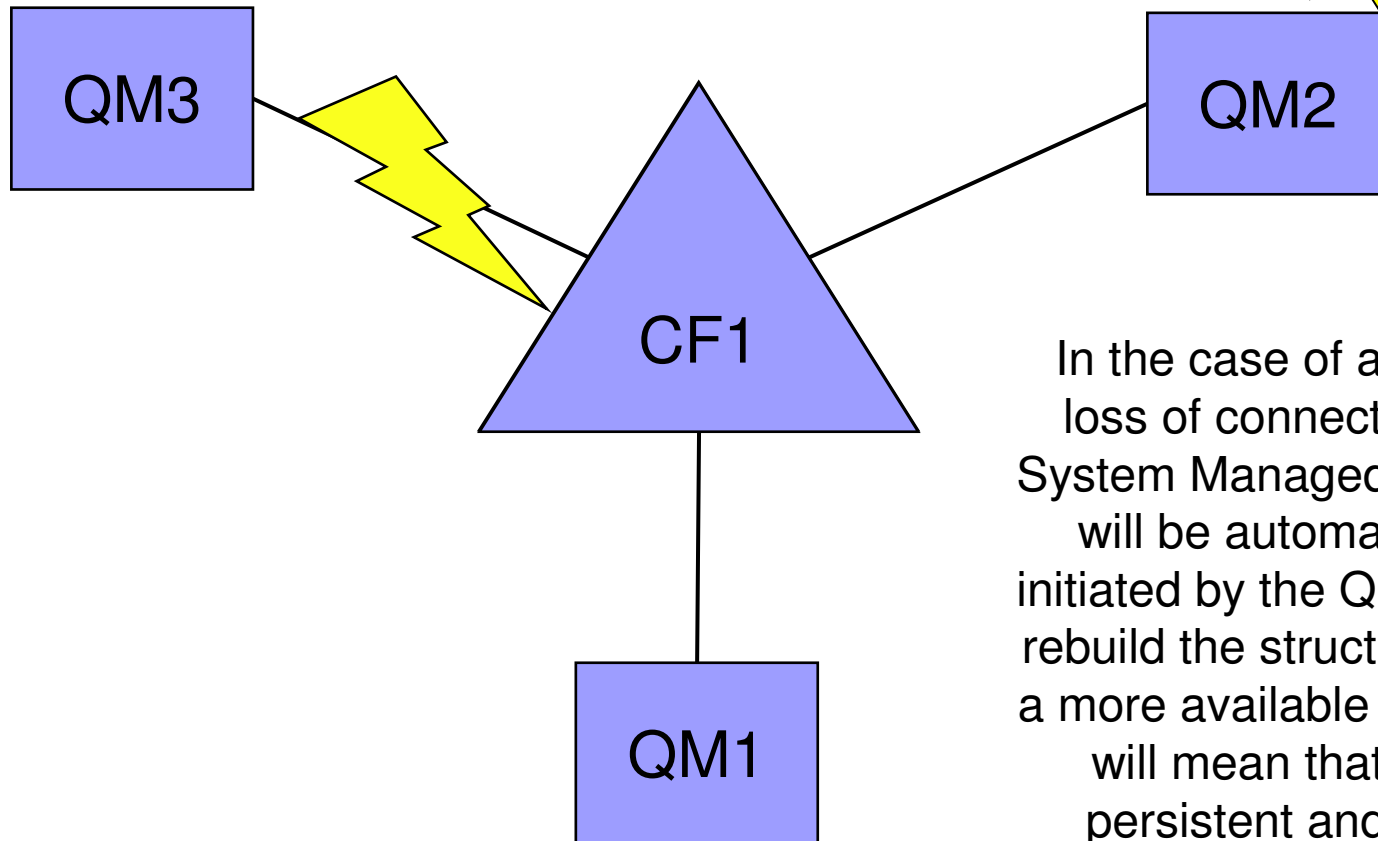
S

In the case of total loss of connectivity

- the structure will (probably) need to be recovered using the RECOVER CFSTRUCT command.
- non-persistent messages will be lost.

CF Loss of Connectivity Tolerance

V7.1+ Queue Managers



V7.1

In the case of a partial loss of connectivity, a System Managed Rebuild will be automatically initiated by the QMGRs to rebuild the structures into a more available CF. This will mean that both persistent and non-persistent messages will be retained.

Application structure loss of connectivity



N

In the case of partial loss of connectivity

O

- queue managers that lost connectivity to the structure will attempt to initiate a system-managed rebuild in order to move the structure to another CF with better connectivity.

T

- if the rebuild is successful, both persistent and non-persistent messages will be copied to the other CF.

E

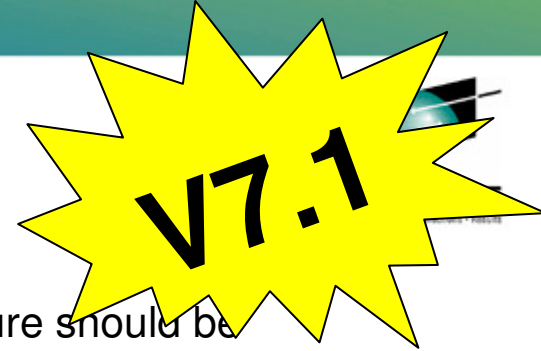
- queue managers that didn't lose connectivity to the structure may experience a slight delay during system-managed rebuild processing, but shared queues will remain available.

S

If an application structure cannot be reallocated in another CF with better connectivity, queues on the structure will remain unavailable until connectivity is restored to the CF that the structure is currently allocated in.

Queue managers will automatically reconnect to the structure when it becomes available.

CF Loss of Connectivity Tolerance



- QMGR CFCONLOS(TERMINATE|TOLERATE)
 - Specifies whether loss of connectivity to the admin structure should be tolerated
 - Default is TERMINATE
 - Can only be altered to TOLERATE when all QSG members are at 7.1
- CFSTRUCT CFCONLOS(TERMINATE|TOLERATE|ASQMGR)
 - Specifies whether loss of connectivity to application structures should be tolerated
 - Only available at CFLEVEL(5)
 - Default is ASQMGR for new CFLEVEL(5) structures, and TERMINATE for structures altered to CFLEVEL(5)
- CFSTRUCT RECAUTO(YES|NO)
 - Specifies whether application structures should be automatically recovered
 - Only available at CFLEVEL(5)
 - Default is YES for new CFLEVEL(5) structure, and NO for structures altered to CFLEVEL(5)

Safeguarding against CF failure

- **Administration structure updates are logged so that this structure can be restored.**
- **Coupling Facilities are very rugged (zSeries processor technology).**
- **CF can have its own separate power supply.**
- **CF can have nonvolatile memory (battery power backup).**
- **Lost application structures can be restored from backups and logs. (can use BACKUP CFSTRUCT(*) at V7.0.1)**

Safeguarding against CF failure

N

Losing a Coupling Facility has a severe impact on a queue sharing group. In this respect a CF is a critical resource, similar to the log for private queues and private objects.

O

Chart summarizes safeguards against CF failures.

T

CFs are inherently very rugged -- especially with separate power supplies and battery backup.

E

WebSphere MQ does not provide its own CF structure duplexing because this facility will be provided by System-Managed Structure Duplexing as a part of z/OS.

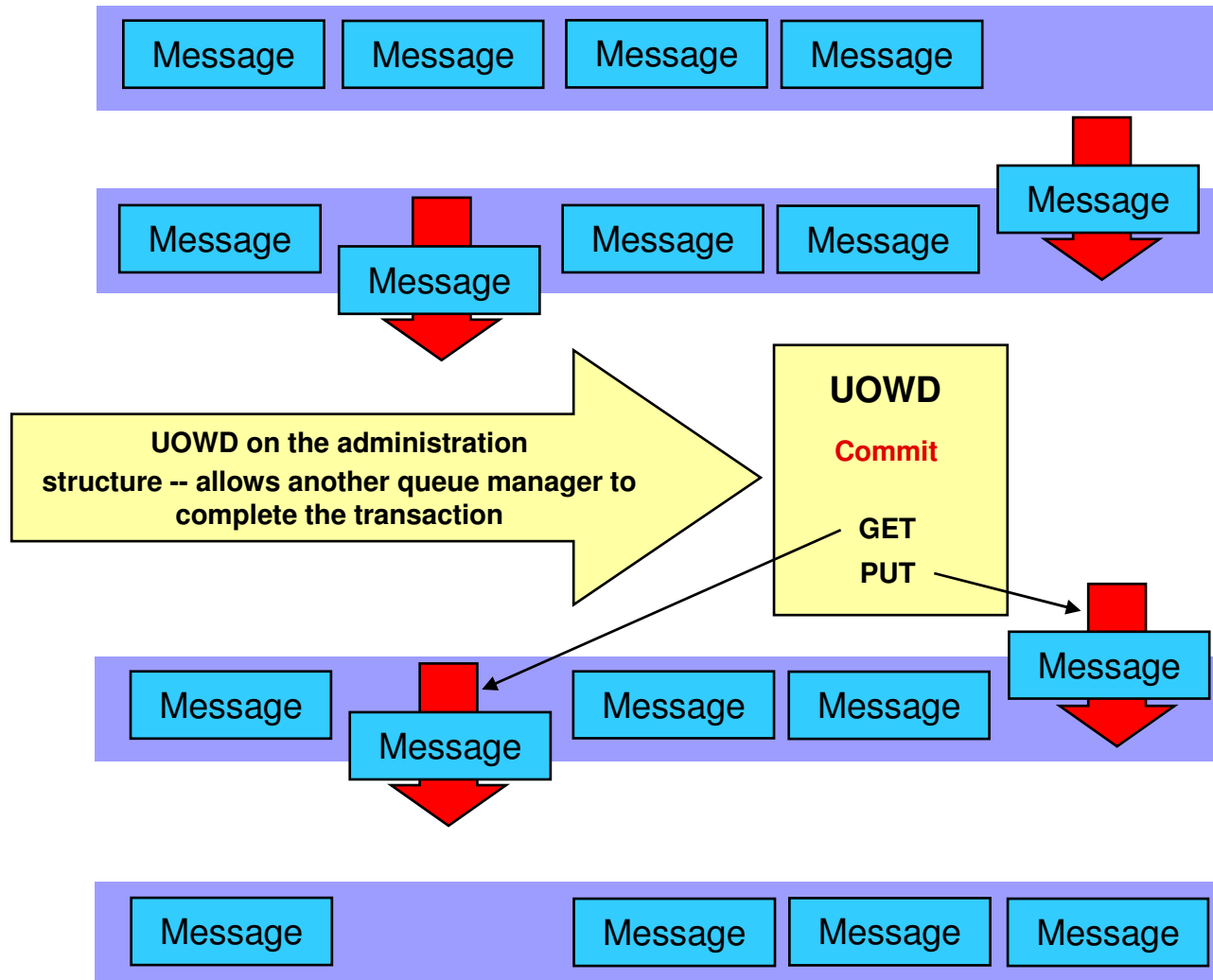
S

Transaction state information recorded on the administration structure is logged so that a failed administration structure can be restored.

Application structures can be backed up and persistent messages written to application structures are logged so that persistent messages in a failed application structure can be restored.

This page intentionally blank

Transaction integrity



Original state of shared queue

Transaction in flight

Queue manager writes the UOWD to the administration structure

Queue manager commits the transaction

Transaction complete -- UOWD deleted

Transaction integrity

N

O

T

E

S

Chart shows how WebSphere MQ maintains transaction integrity for shared queues.

For clarity the chart shows a transaction acting on one shared queue. WebSphere MQ also maintains the integrity of transactions that act on multiple shared queues or that act on both shared and private queues.

Chart shows the following states for a shared queue on an application structure (in a CF):

1 Original state -- transaction has not started:

- Queue has four messages on it (all committed).

2 Transaction in flight:

- Transaction has done one get -- message is marked in-flight-get
- Transaction has done one put -- message is marked in-flight-put
- Messages marked in-flight are "invisible" to other transactions
- If queue manager fails, any other queue manager can back-out.

3 Transaction in commit:

- Queue manager has written unit of work descriptor (UOWD)
- If queue manager fails, any other queue manager can complete.

4 Transaction complete:

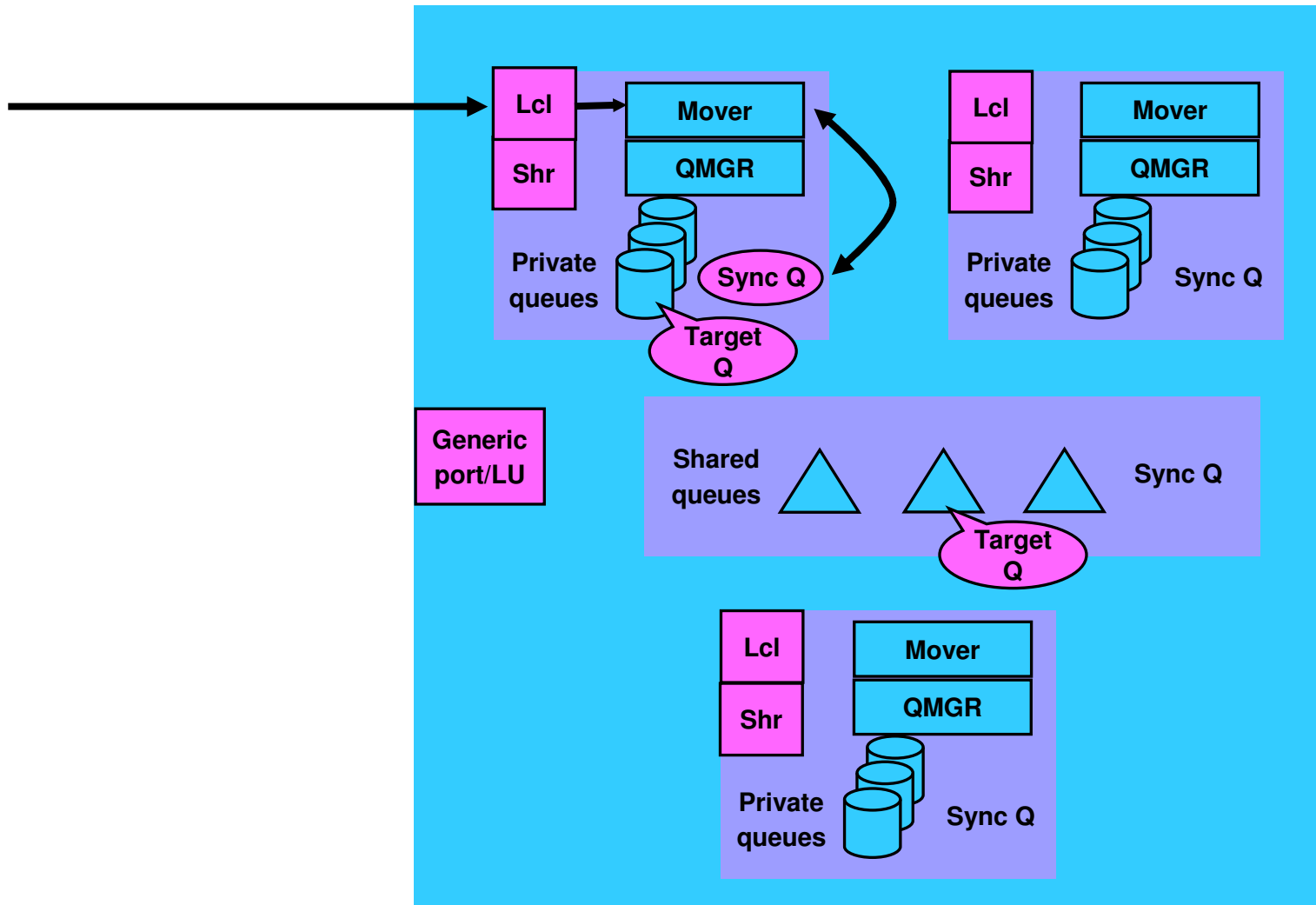
- In-flight-put message unmarked -- becomes "visible"
- Queue manager deletes the UOWD.

Agenda

- What shared queues are
 - Shared queues
 - Queue-sharing groups
 - Coupling Facility (CF) structures
 - Persistence and transaction integrity
- **Configuring channels with shared queues**
 - Inbound channel configurations
 - Outbound channel configurations
- Exploiting shared queues
 - Availability benefits of queue sharing
 - Scalability

This page intentionally blank

Inbound channel configuration 1



Inbound channel configuration 1

N

This chart is the first of three which show different ways to configure inbound channels.

O

This configuration uses the local port (TCP/IP) or logical unit (LU6.2) to connect to a specified mover (queue manager). It works almost exactly the way existing channel configurations work, including using the local sync queue for the channel -- but:

T

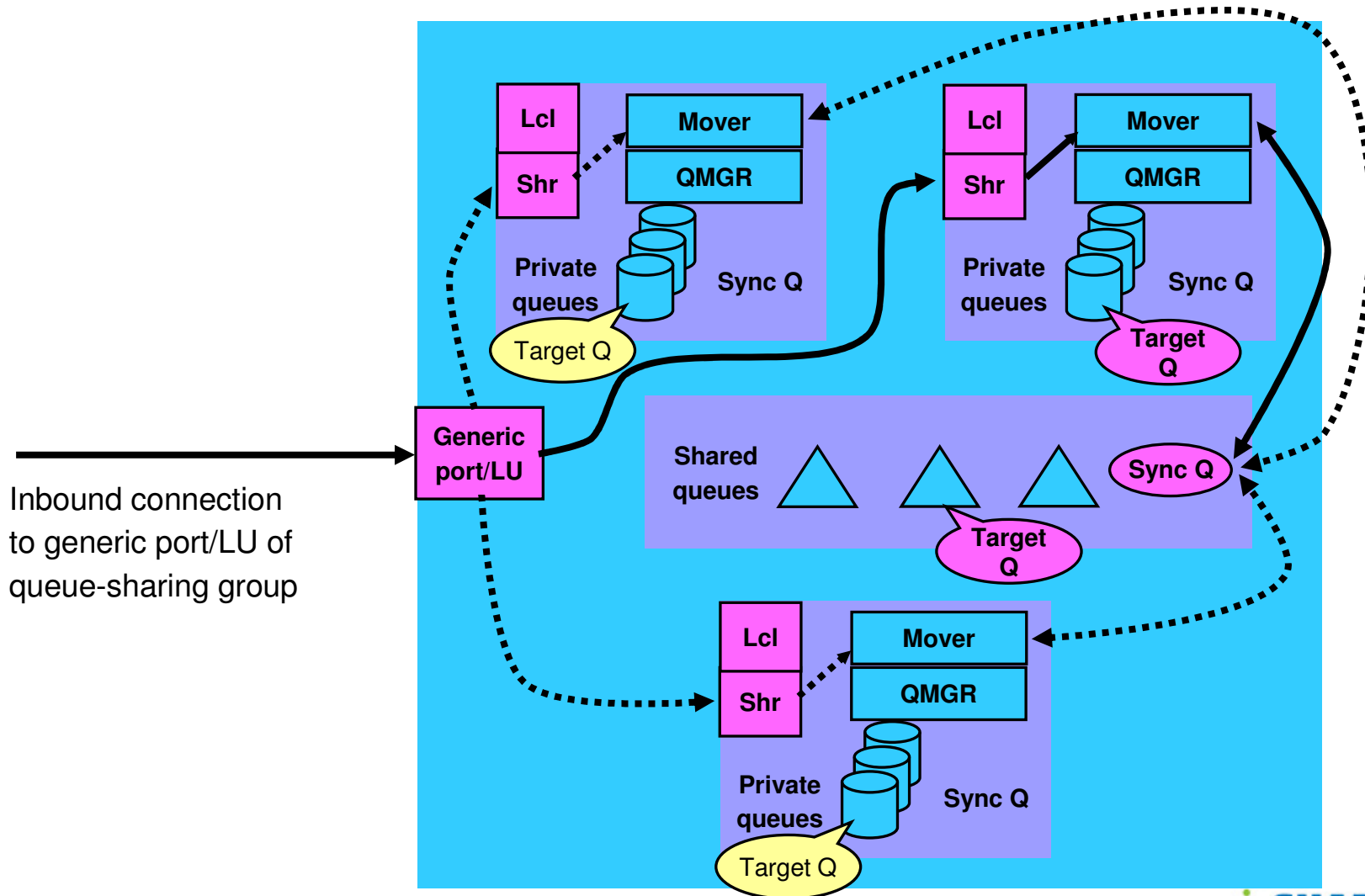
Because the queue manager is part of a queue-sharing group, the channel can put messages directly onto a shared queue. That is, the target *application* can be on any of the queue managers in the QSG.

E

The chart shows other ports/LUs not used by this configuration.

S

Inbound channel configuration 2



Inbound channel configuration 2

N

This configuration uses VTAM generic resources (LU6.2) or dynamic DNS (TCP/IP) to connect to *any* mover in the queue-sharing group.

O

The chart shows that the connection has "selected" the mover shown at top right, but a subsequent connection could select another mover in the QSG.

T

Notice that the mover uses the shared sync queue for this channel (because access was through the shared LU or port). The shared sync queue is: `SYSTEM.QSG.CHANNEL.SYNQCQ`.

E

If the channel loses its connection (for example, because this queue manager fails), it can connect to a different mover. But this different mover can resynchronize the channel using the shared sync queue.

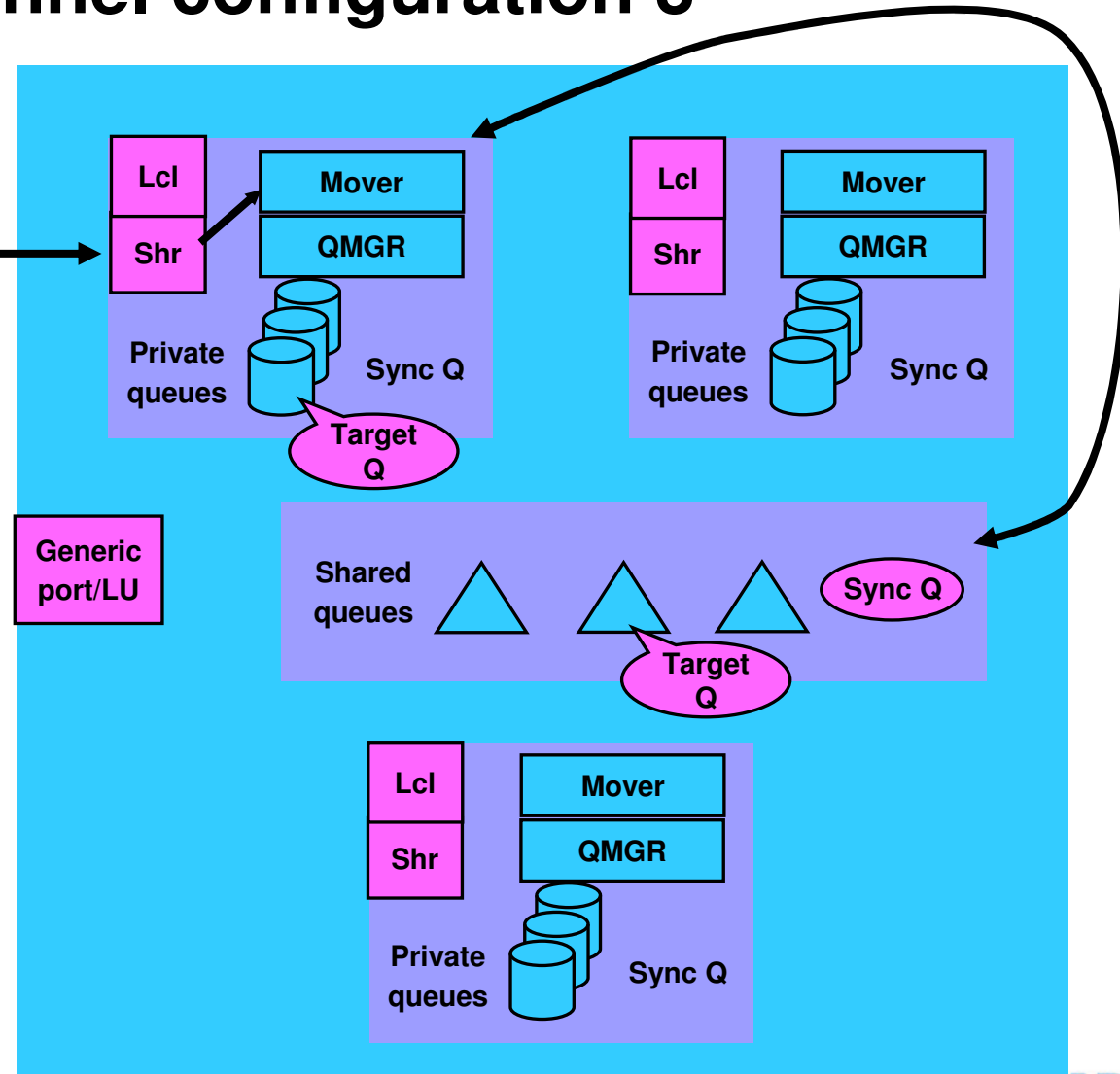
S

You can configure VTAM generic resources or dynamic DNS to use the z/OS workload manager (WLM) to select the "least busy" mover -- providing load balancing.

If the target queue for a put is not shared then the same private queue must be defined on each of the queue managers in the QSG.

Inbound channel configuration 3

Inbound connection to shared port/LU of specific queue manager's mover



Inbound channel configuration 3

N

This chart shows that you can connect directly to the shared port of a specified mover -- that is, you do not have to use VTAM generic resources or dynamic DNS.

O

This allows you to use an "external" router to select which mover to connect to.

T

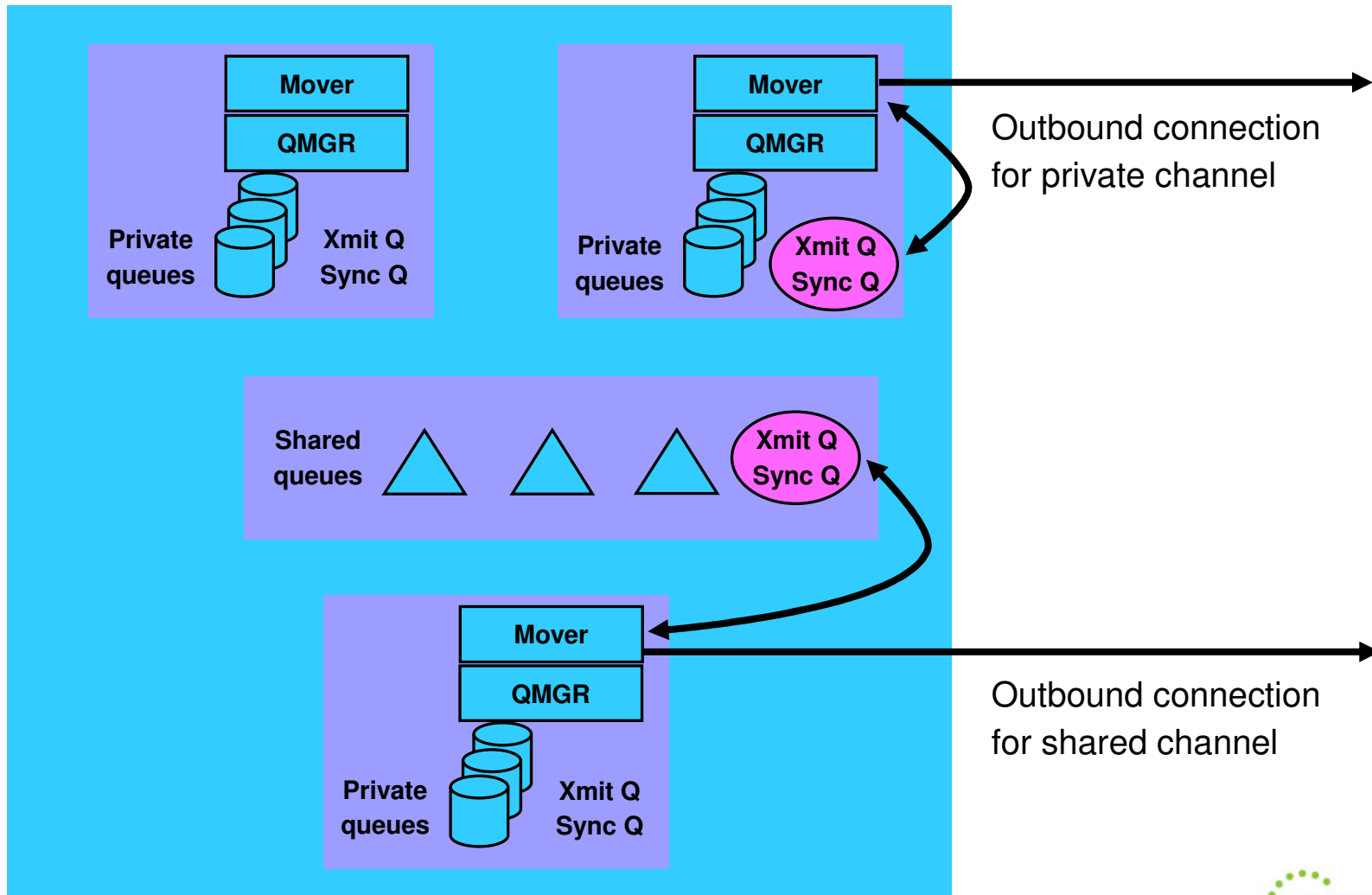
By connecting to the shared port the connection uses the shared sync queue. This allows correct resynchronization if the channel fails and reconnects to the shared port of a different mover.

E

A disadvantage of an external router is that it does not use the workload manager to identify the "best" (least busy) mover.

S

Outbound channel configurations



Outbound channel configurations

N

This chart shows the two possible configurations for outbound channels.

At the top is a *private* (or *local*) outbound channel. It works exactly the way existing channel configurations work:

- Private transmission queue local to the mover
- Private synchronization queue local to the mover.

O

Below is a *shared* outbound channel:

- Shared transmission queue -- any application in the QSG can use it
- Shared synchronization queue.

T

A shared outbound channel can start on any mover -- WebSphere MQ selects the "best" (least busy) mover.

E

If a shared outbound channel fails (communication, mover, or queue manager failure), the channel can restart automatically on another mover. This is called *peer channel recovery*.

S

Shared queue restrictions apply to shared transmission queues, for example:

- CF capacity is limited (compared to DASD).

Client Channels

- Client channels are stateless, so don't use synchronization queues
 - Only benefit of using a shared channel is the shared status
 - Can cause performance issues if using shared channel
 - Needs to update DB2 status for each connect/disconnect
- Can configure a generic port to point at INDISP(QMGR) listener on each queue manager
 - Can still benefit from failover and balancing of client connections without using a shared channel, and can still use QSG name on the MQCONN
- Will not work for Extended Transactional Client (including WAS 2-Phase Commit over client conn) until at V7.0.1

Client Channels

N

O

T

E

S

As client channels are stateless, they don't use a synchronization queue. The only benefit of using a shared channel for client channels is the shared status information. However, the use of a shared server-connection channel has drawbacks as it means each connection/disconnect will cause the queue manager to update the shared channel status, which is held in DB2. This could lead to performance issues if there are lots of clients connecting.

It is still possible to use a generic port to provide workload distribution and failover in the QSG, but rather than targeting an INDISP(SHARED) listener on each queue manager, the INDISP(QMGR) listener should be targeted.

When using client channels into a QSG it is not possible to use the Extended Transactional Client (or client connections from WAS) if you are using 2-phase commit, unless you are connecting into a V7.0.1 queue manager

2-Phase Commit Client Connections



- When setting up the connection, specify the QSG name rather than QMGR name
 - In MQConnectionFactory if using JMS under WAS, you must ensure that you are only using shared resources
 - This causes a UR with GROUP disposition to be created, rather than QMGR
 - A GROUP UR can be inquired and resolved via any member of the QSG
 - If there is a failure, the transaction manager will reconnect to the QSG and request a list of in-doubt transactions. GROUP URs will be reported back no matter what QMGR they were started on

2-Phase Commit Client Connections

V7.0.1

N

O

T

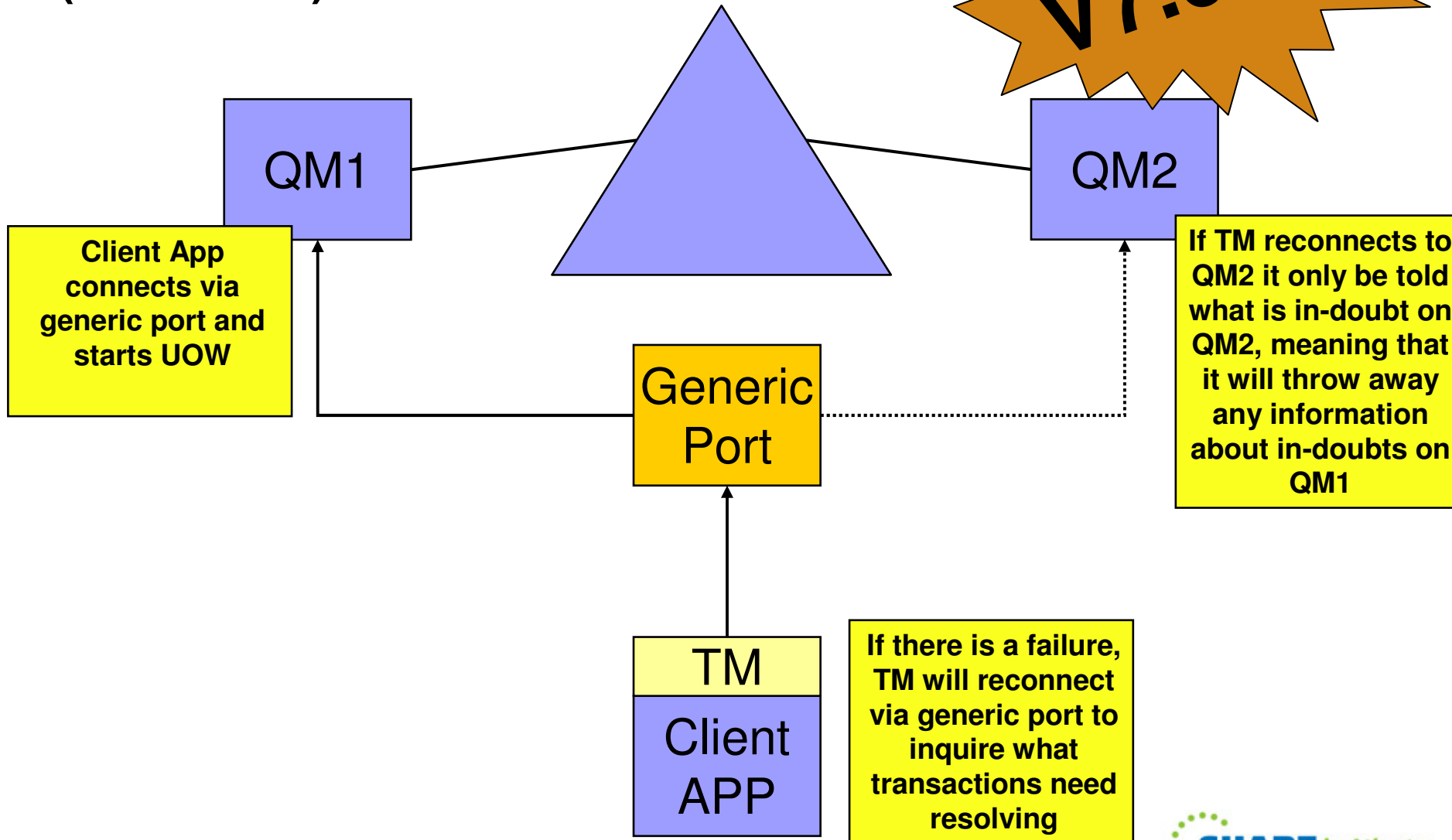
E

S

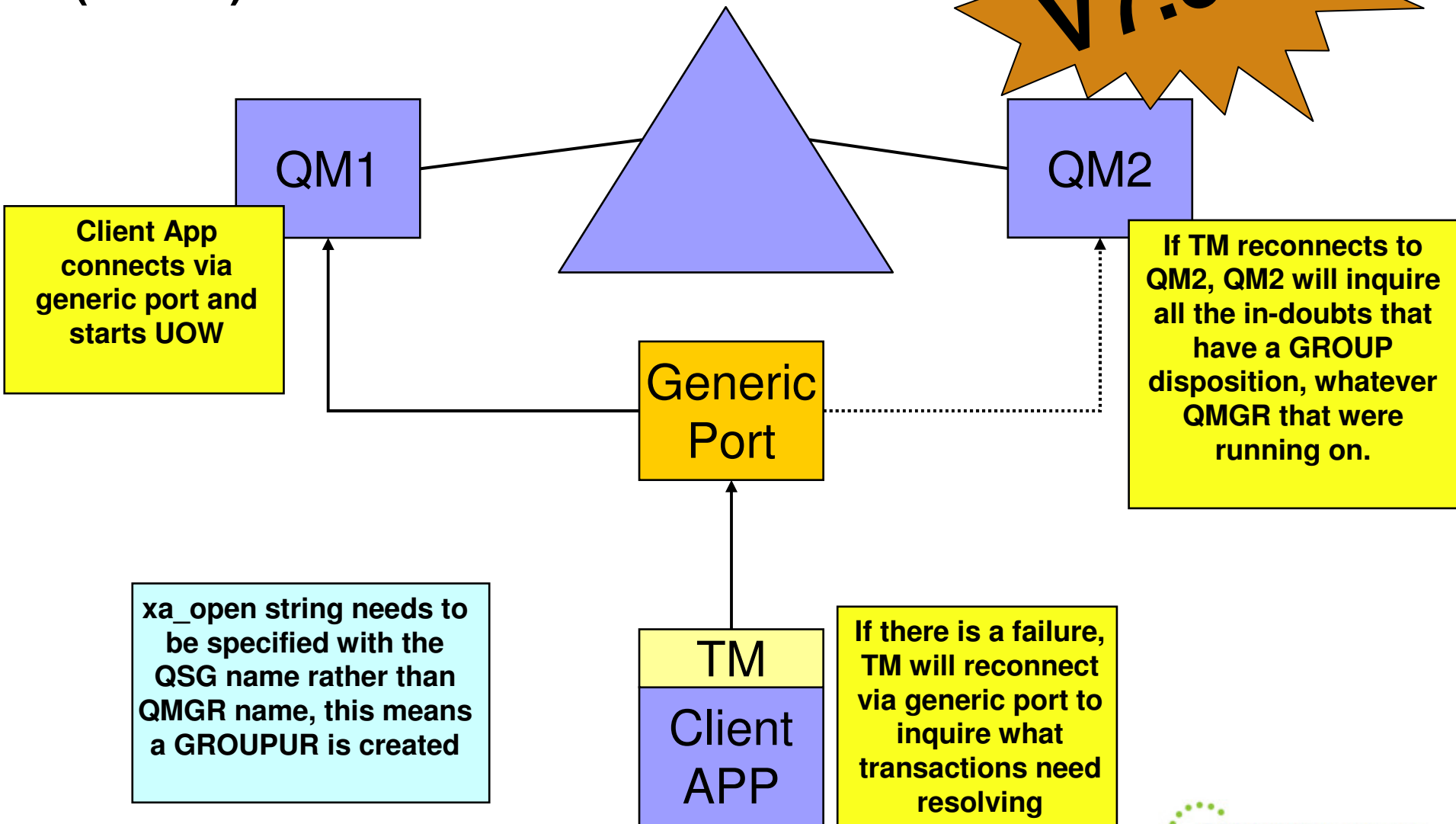
When using the Extended Transactional Client, or the JMS transactional client (under WAS), it is possible to use 2-phase commit applications in a QSG. When specifying the connection options to the Transaction Manager it is necessary to provide the QSG name rather than the QMGR name, and also configure the client channel to be routed to a suitable (V7.0.1 or higher qmgr) in the QSG. When using this configuration, any Unit Of Recovery (UR) that is created will have a GROUP disposition. This means that it can be inquired and resolved on any qmgr in the QSG.

If a connection fails for some reason, and the TM reconnects to the QSG, it can inquire and resolve the transactions no matter which qmgr it is now connected to, and where the transactions were originally started.

GROUPUR – The Problem (Pre V7.0.1)



GROUPUR – The Solution (V7.0.1)

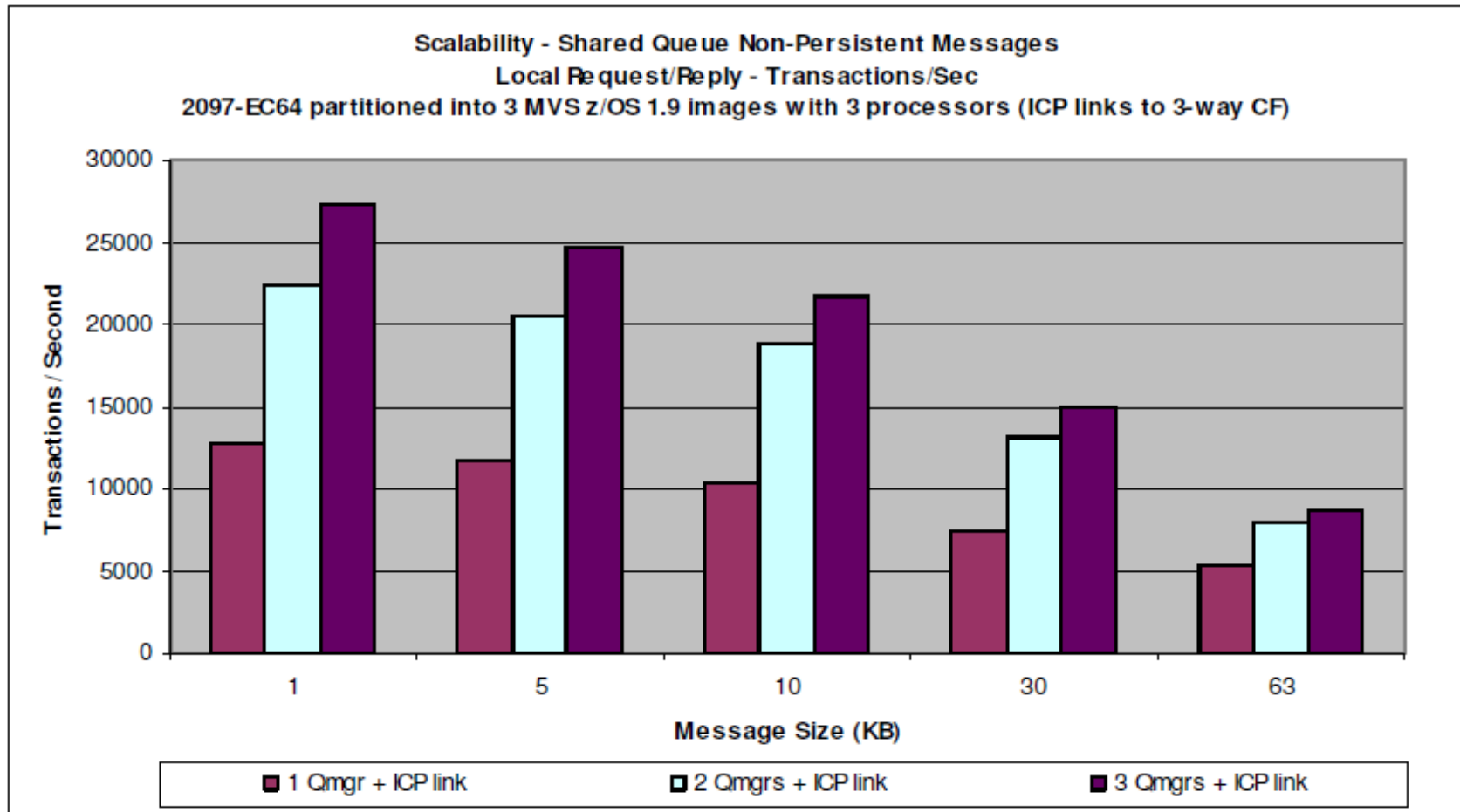


Agenda

- What shared queues are
 - Shared queues
 - Queue-sharing groups
 - Coupling Facility (CF) structures
 - Persistence and transaction integrity
- Configuring channels with shared queues
 - Inbound channel configurations
 - Outbound channel configurations
- **Exploiting shared queues**
 - Scalability

This page intentionally blank

Shared queue scaling – non persistent



Shared queue scaling – non persistent

N

Chart shows measured results on lab setup -- actual numbers of messages processed will vary depending on the equipment and configuration used.

O

Lab setup does not include business logic.

T

All messages are nonpersistent.

E

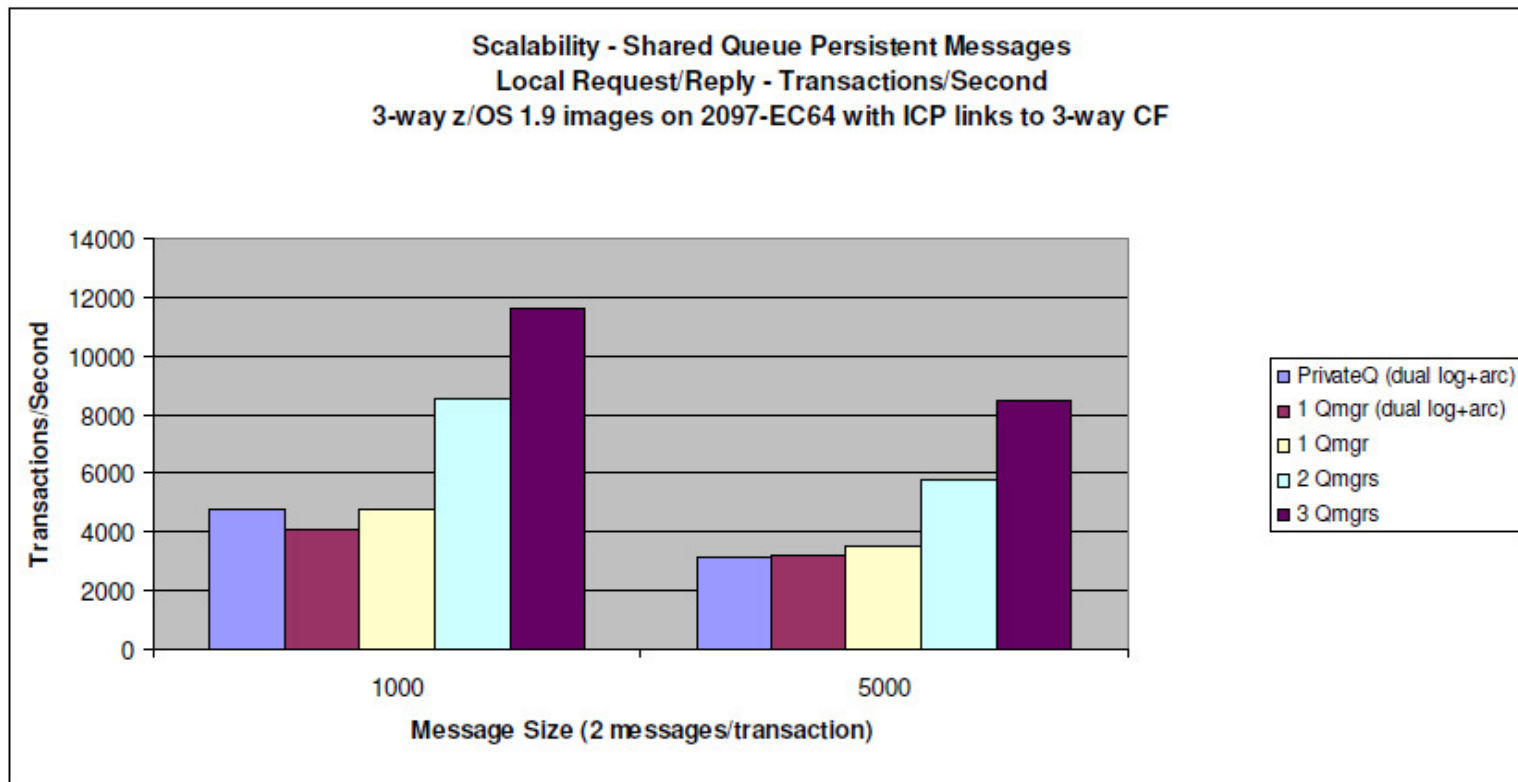
In all cases one queue manager per z/OS image.

S

Measurements are transactions/sec, which is made up of 2 messages.

Shared queue scaling – persistent

- 23,000+ persistent messages/sec using 3 qmgrs
- log DASD still likely to be first limit on throughput



Shared queue scaling – persistent

N

Chart shows measured results on lab setup -- actual numbers of messages processed will vary depending on the equipment and configuration used.

O

Lab setup does not include business logic.

T

All messages are persistent.

E

In all cases one queue manager per z/OS image.

S

DASD configuration enforced single logs and no archiving

CF Link types / z/OS XCF heuristics

N

Throughput and CPU cost depend on load

CFP slowest, most expensive - up to 27 Km

CBP 'ClusterBus' - 10 metres

O

ICP only to CF LPAR'd within same box

CBP performs more like ICP than CFP? (did on 9672 anyway)

T

CF calls can be synchronous or async according to z/OS heuristics

Async likely to mean order of 130+ microsec wait

E

Sync means z/OS CPU is busy while call data sent, CF processes, return data received

S

typically 50+ microsecs

Technology trends - CPU speed increasing faster than link speed

This page intentionally blank

Shared queue benefits

- ✓ **No mover between servers in the QSG.**
- ✓ **Pull load-balancing for servers.**
- ✓ **Availability from multiple servers.**
- ✓ **Workload-balancing for movers.**
- ✓ **Availability from shared channels.**
- ✓ **Simplified configuration management from shared object definitions and command scoping.**
- ✓ **Flexible capacity management.**

Shared queue benefits

N

This chart summarizes benefits from using shared queues . Mostly these are things we have discussed in this presentation.

O

Using shared queues for communication within the sysplex is faster and simpler than using the mover.

T

Multiple servers get better performance from sharing the same request queue (pull load balancing) than from separate queues.

Multiple servers provide better availability than single servers.

E

Each mover runs on the "best" (least busy) queue manager.

Shared channels provide better availability than private channels (peer channel recovery and so on).

S

Configuration management is simplified by sharing the same object definitions across many queue managers and by commands which act on more than one queue manager (command scoping, see *MQSC Command Reference*).

Capacity can be increased (or decreased) nondisruptively by adding or upgrading processors, disks, or whatever.

More Information

- WebSphere MQ for z/OS Concepts and Planning Guide
- SupportPacs MP16, MP1E, MP1F, MQ1G
 - www.ibm.com/software/integration/support/supportpacs/perfreppacs.html
- RedPaper 3636 – WebSphere MQ Queue Sharing Group in a Parallel Sysplex environment
 - www.redbooks.ibm.com/redpieces/pdfs/redp3636.pdf

Questions?



Copyright Information

© Copyright IBM Corporation 2011. All Rights Reserved. IBM, the IBM logo, ibm.com, AppScan, CICS, Cloudburst, Cognos, CPLEX, DataPower, DB2, FileNet, ILOG, IMS, InfoSphere, Lotus, Lotus Notes, Maximo, Quickr, Rational, Rational Team Concert, Sametime, Tivoli, WebSphere, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at ibm.com/legal/copytrade.shtml.

Coremetrics is a trademark or registered trademark of Coremetrics, Inc., an IBM Company.

SPSS is a trademark or registered trademark of SPSS, Inc. (or its affiliates), an IBM Company.

Unica is a trademark or registered trademark of Unica Corporation, an IBM Company.

Java and all Java-based trademarks and logos are trademarks of Oracle and/or its affiliates. Other company, product and service names may be trademarks or service marks of others. References in this publication to IBM products and services do not imply that IBM intends to make them available in all countries in which IBM operates.