IBM MQ Lab Tour – Paris – 26 September 2017

IBM MQ
High Availability

Jamie Squibb
jamie_squibb@uk.ibm.com

Notices and disclaimers

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law

Notices and disclaimers (continued)

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services ®, Global Technology Services ®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli®, Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

Agenda

- What is high availability (HA)?
- High availability considerations for messaging
- IBM MQ HA capabilities
  - HA clusters
  - Multi-instance queue managers
  - IBM MQ Appliance
  - Queue manager clusters
  - Queue-sharing groups
- Application connectivity
- High availability in cloud environments
- Putting it all together

Concept: High availability

Availability

A system is said to be available if it is able to perform its required function, such as successfully process requests from users.

Usually measured as a percentage of time that a system is functional in a month or a year.

An essential element of Service Level Agreements (SLAs) between service providers and service consumers.

High availability (HA)

A requirement, or a capability, of a system to be operational for a greater proportion of time than is common for other, less important, systems.

High availability

- Applications usually depend upon multiple services.
  - It is important to consider the complete workflow.
- HA solutions employ:
  - Redundancy to eliminate single points of failure – high speed failover.
  - Dynamic routing to redistribute work in response to failures.
- Aspects to consider for each component:
  - Power supply – UPS, dual supplies
  - System hardware
  - Disk storage – RAID, SAN
  - Networking connectivity – adapters, cables, switches, routers, DNS
  - Application environment – operating system, app server/container, application
  - External dependencies

Notes

A system is said to be available if it is able to perform its required function, such as successfully process requests from users. The term High Availability (HA) is used to identify a requirement, or a capability, of a system to be operational for a greater proportion of time than is common for other, less important, systems. Availability is usually measured as a percentage of time that a system is functional in a month or a year, and it is typically an essential element of Service Level Agreements (SLAs) between service providers and service consumers.

Availability is dependent upon a number of factors, so achieving a highly available system usually requires the introduction of redundancy to remove single points of failure, and also the ability to redistribute work elsewhere in response to failures.

Increasing the availability of a system incurs additional cost, so employing the highest level of availability is often reserved for the most critical systems of an enterprise. However, technology is constantly evolving to provide increased resiliency, which means that some capabilities traditionally identified with HA solutions are now commonplace.

Business applications typically depend upon multiple services that are hosted throughout an enterprise. They can also depend upon external parties, such as business partners and public services. It is important to consider the complete workflow of an application when determining its availability, because an application's availability is determined by its weakest point. However, the availability requirements for each service used by the application need not all be the same.

When considering how to improve the availability of a system it is important to consider all aspects of the system's environment. These aspects include, the power supply, the system hardware, disk storage, networking and the applications themselves,

It is also important to remember that technology is not a substitute for good planning, system management and testing.

Some HA messaging considerations

- What happens if a messaging application fails?
  - Can another instance takeover, or must it be restarted?
- How do applications connect to the messaging infrastructure?
- How do messages flow through the messaging infrastructure?
  - Can messages be queued or must the complete workflow remain available?
  - Can messages be routed elsewhere or is message sequence important?
  - Do applications have affinities to messaging resources?
- How quickly must failover occur?
- Is the loss, or temporary inaccessibility, of messages acceptable?

We discuss these considerations, and others, during this presentation.

Notes

When thinking about how to improve the availability of messaging infrastructure it is important to consider a number of factors that might not be initially apparent. This is because there are a number of different approaches to improving high availability. Each approach has its strengths and weaknesses, so it is important to identify the most appropriate option, or combination of options, for each application or messaging deployment. Some of the considerations are:

- Application failures
  - What happens if a messaging application fails?
  - Can another instance of the application take over, or must the original application be restored?
  - Does the application have an affinity to the local system or other specific resources?
  - If an application is restarted is resynchronization necessary to resolve partially completed transactions?

- Application connectivity
  - How do applications tolerate loss of connectivity? How do they reconnect?
  - Do applications connect to messaging resources, such as a queue manager, using a local bindings connection or do they connect as a client over the network?
  - Applications that connect locally have a dependency on local resources, so what happens if those resources fail? Can the application be moved?
  - How do client applications determine which resource to connect to? Can they reconnect elsewhere?

Notes

- Message flow
  - How do messages flow through the messaging infrastructure? Do they take a specific route to a specific destination, or can they be rerouted in the event of a failure?
  - Do messages flow in one direction, or must replies be returned? How does an HA solution affect this?
  - Can messages be queued if their destination is not available, or inaccessible, or is it critical to deliver them immediately? Not all workflows require immediate processing, so the ability to provide maximal availability for those systems might not be necessary.
  - Can messages be routed elsewhere or do they have an affinity to a particular target instance? For example, do the messages represent a conversation that another target instance would be unaware of? Is message sequence important, which requires all messages to be delivered to the same target instance in order?

- Failover
  - How quickly must failover occur?
  - Is there time to restart a new instance, or must a ‘hot standby’ be ready to takeover immediately?
  - Is the loss, or temporary inaccessibility, of messages acceptable? Some messages have a limited time to live, such as query requests that can be resubmitted readily. Sometimes it is more important to support the flow of new work than it is to recover work that is already in the system.
  - Can messages held by a messaging resource, such as a queue manager, be processed later when the resource has been recovered, or must they be available immediately after failover?

IBM MQ HA capabilities

- Support for HA clusters and network storage
- Multi-instance queue managers (Windows, Linux, UNIX)
- IBM MQ Appliance

- Queue manager clusters
- Queue-sharing groups (z/OS)

- Client connectivity (auto-reconnect, CCDTs, pre-connect exit)

Notes

- IBM MQ has a number of capabilities that can be used to build highly available solutions. The capabilities help address different aspects of a messaging solution, including the availability of queue managers and how they communicate with each other and applications.
- We will now discuss each of these capabilities in turn, then how they can be used in combination to build resilient solutions.

HA clusters

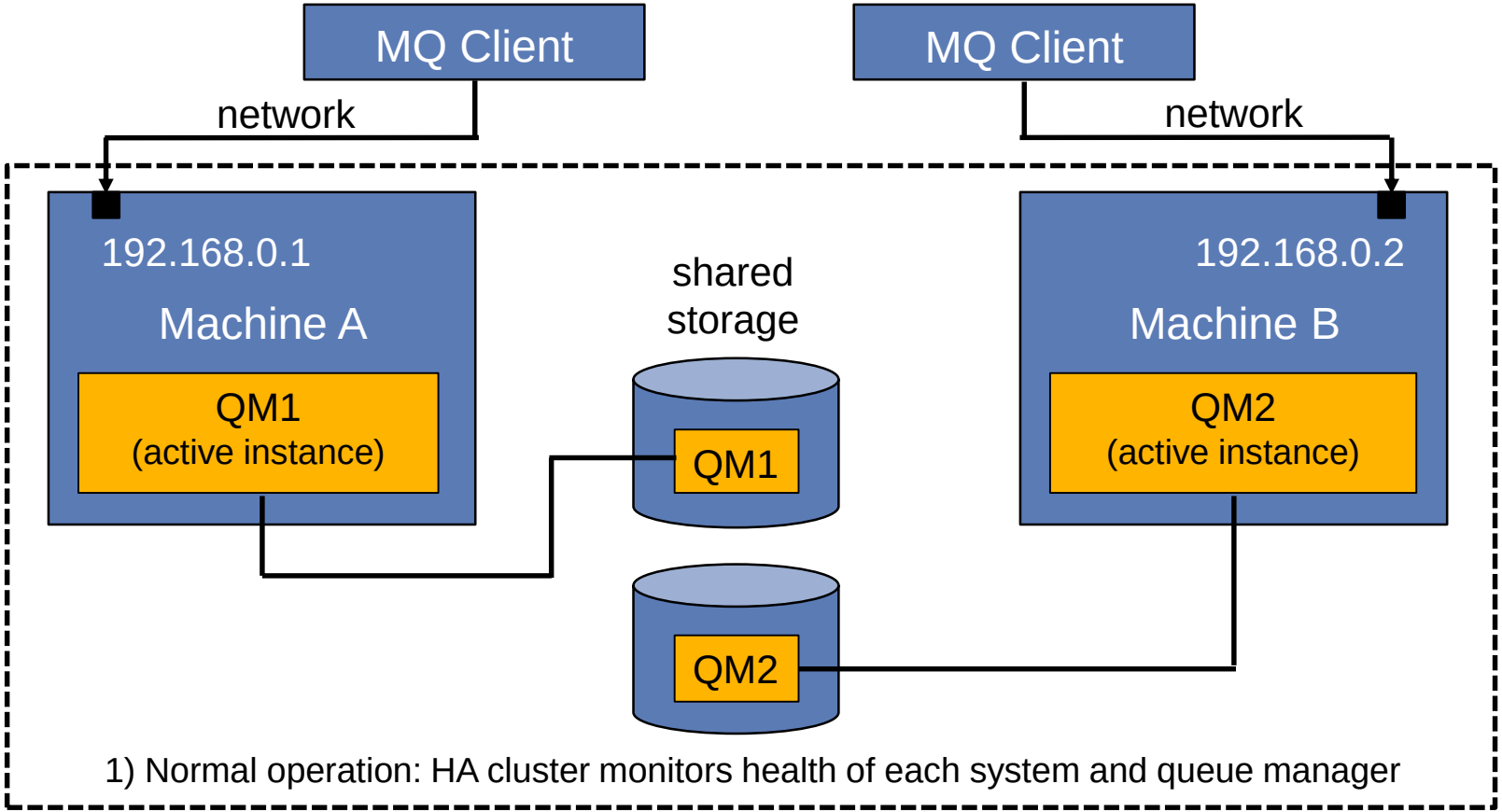
- MQ traditionally made highly available using an HA cluster
  - IBM PowerHA for AIX (formerly HACMP)
  - Veritas Cluster Server
  - Microsoft Cluster Server, etc.

- HA clusters can:
  - Coordinate multiple resources such as application server, database
  - Consist of more than two machines
  - Fail over more than once without operator intervention
  - Take over IP address as part of failover
  - Resilient to both MQ and operating system failures

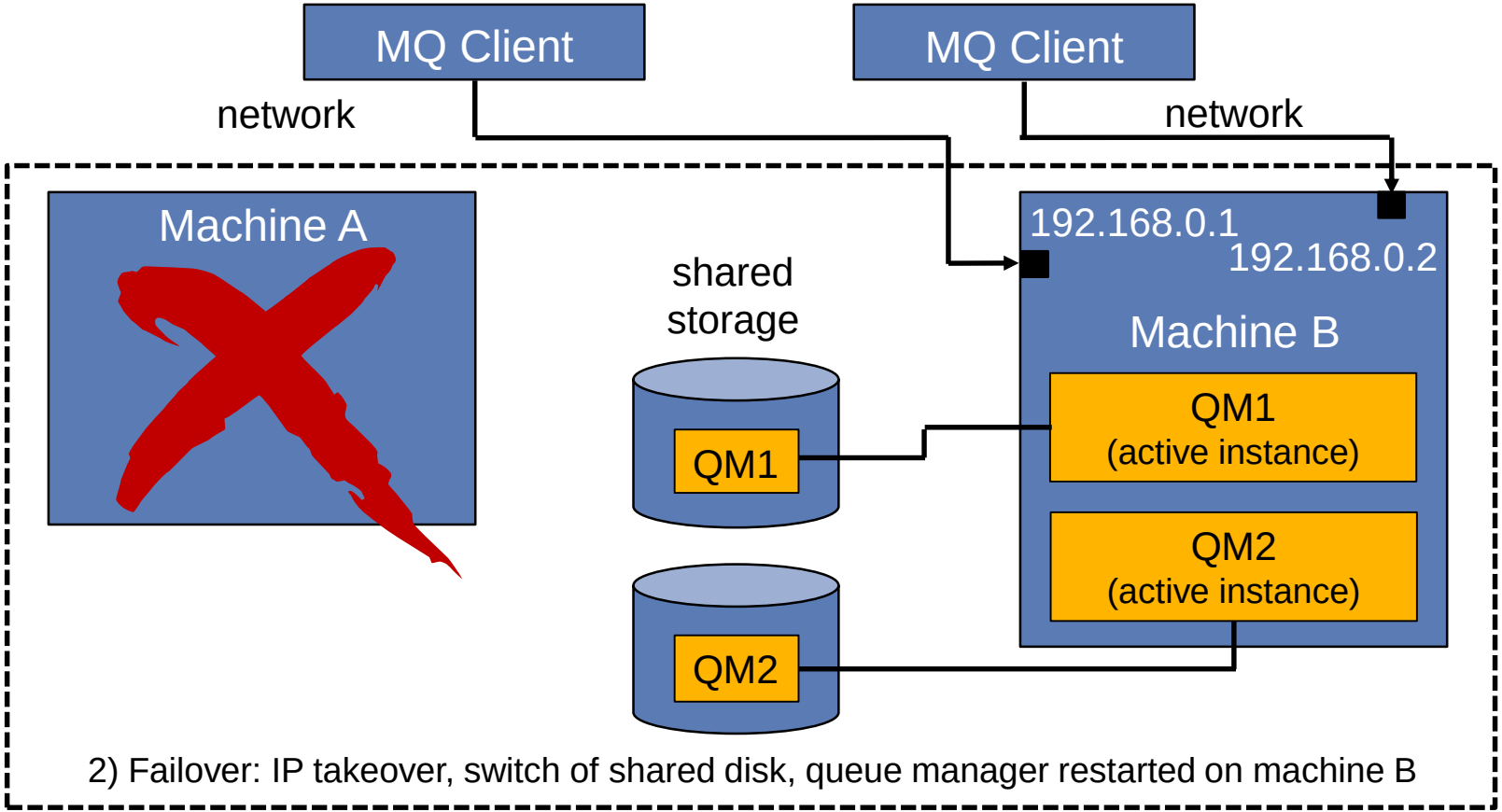
Notes

- You can use IBM MQ with a high availability (HA) cluster on UNIX and Linux platforms. For example, PowerHA for AIX (formerly HACMP), Veritas Cluster Server, HP Serviceguard, or a Red Hat Enterprise Linux cluster with Red Hat Cluster Suite.
- HA clusters have the concept of a unit of failover. This is a set of definitions that contains all the resources that make up the highly available service. The unit of failover includes the service itself and all other resources upon which it depends.
- The smallest unit of failover for IBM MQ is a queue manager. Typically, the resource group containing the queue manager also contains shared disks in a volume group or disk group that is reserved exclusively for use by the resource group, and the IP address that is used to connect to the queue manager. It is also possible to include other IBM MQ resources, such as a listener or a trigger monitor in the same resource group, either as separate resources, or under the control of the queue manager itself.
- A queue manager that is to be used in an HA cluster must have its data and logs on disks that are shared between the nodes in the cluster. The HA cluster ensures that only one node in the cluster at a time can write to the disks. The HA cluster can use a monitor script to monitor the state of the queue manager.
- If the HA cluster contains more than one queue manager, the HA cluster configuration might result in two or more queue managers running on the same node after a failover. Each queue manager in the HA cluster must be assigned its own port number, which it uses on whichever cluster node it happens to be active at any particular time.

HA clusters


MQ Client
MQ Client
network
network
192.168.0.1
Machine A
QM1
(active instance)
shared
storage
QM1
QM2
192.168.0.2
Machine B
QM2
(active instance)
1) Normal operation: HA cluster monitors health of each system and queue manager

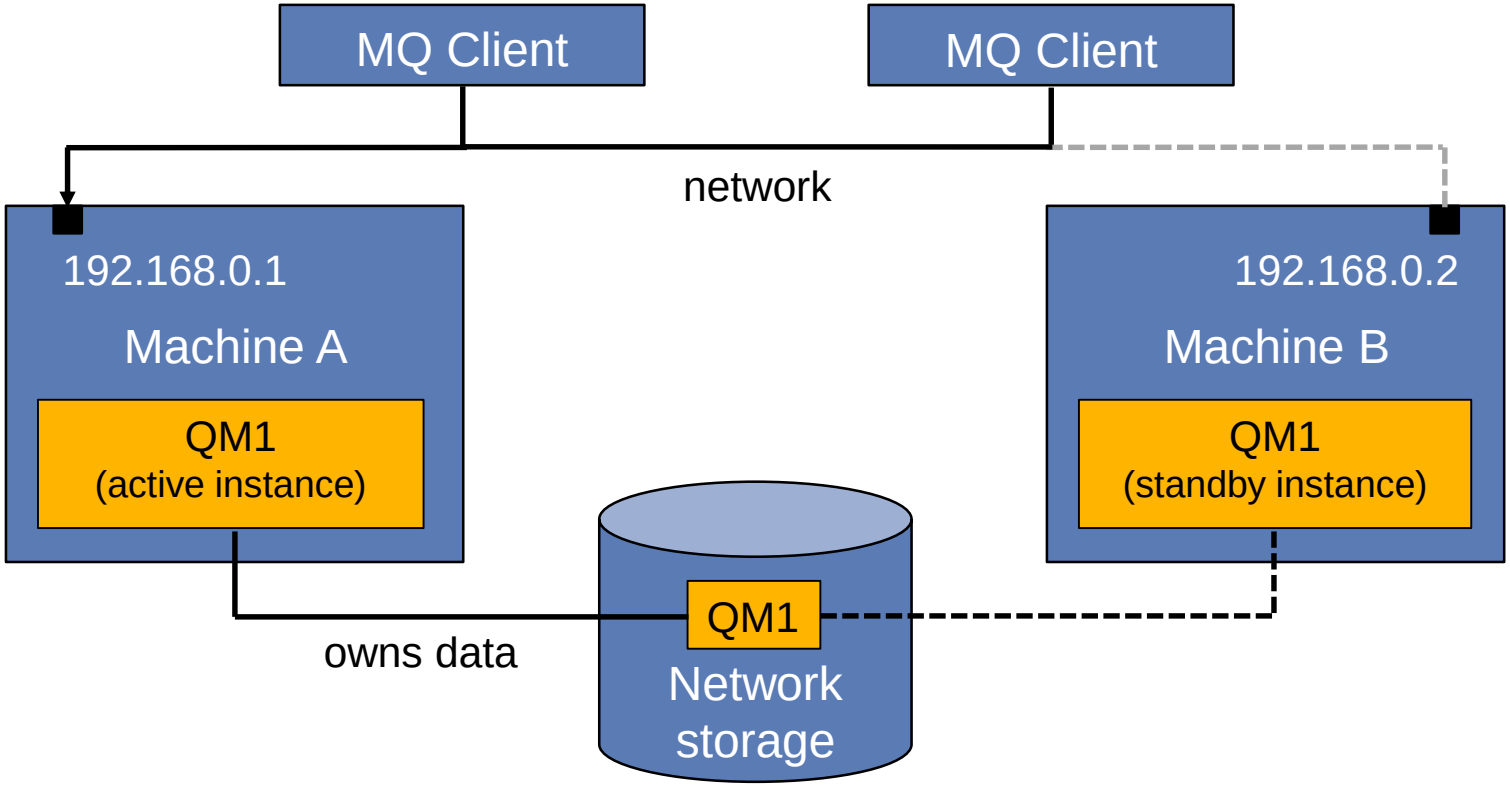
HA clusters


MQ Client
MQ Client
network
network
Machine A
192.168.0.1
192.168.0.2
shared
storage
Machine B
QM1
QM1
(active instance)
QM2
QM2
(active instance)
2) Failover: IP takeover, switch of shared disk, queue manager restarted on machine B

Multi-instance queue managers

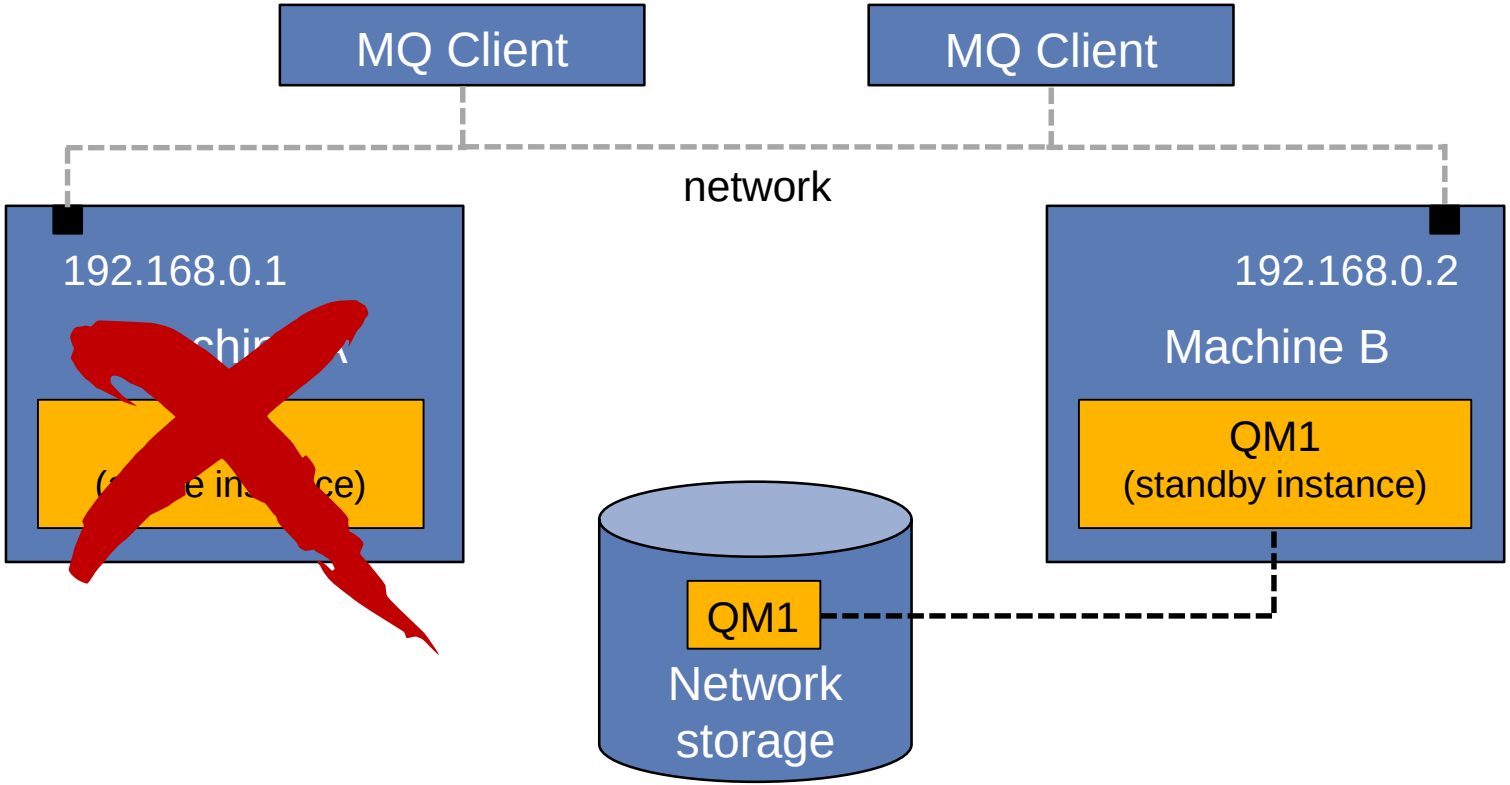
- Provides basic failover support without HA coordinator (HA cluster)
  - Faster takeover - fewer moving parts
  - Cheaper - no specialised software or administration skills needed
  - Windows, Unix, Linux platforms
- Queue manager data held in networked storage
  - NFS, GPFS, etc. - more than one machine sees the queue manager data (same data = same queue manager)
  - Formal support for networked storage even without failover configuration
- Multiple (2) instances of the SAME queue manager on different machines
  - One is “active” instance; other is “standby” instance
  - Active instance “owns” the queue manager’s files and will accept application connections
  - Standby instance does not “own” the queue manager’s files and applications cannot connect to it
  - If the active instance fails, standby instance performs queue manager restart and becomes active

Multi-instance queue managers



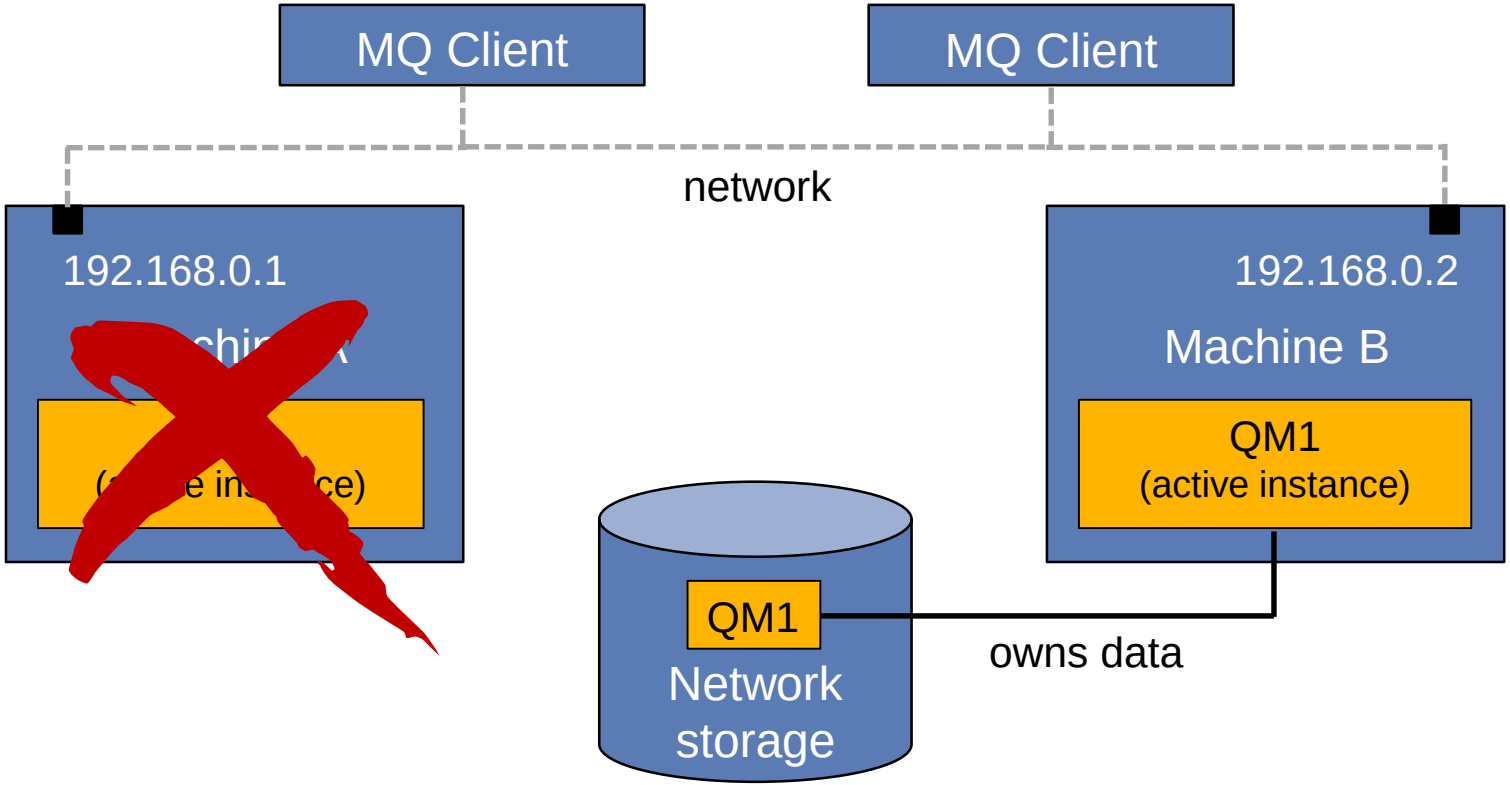
1) Normal operation: Queue manager active on machine A with a standby instance ready on machine B

Multi-instance queue managers



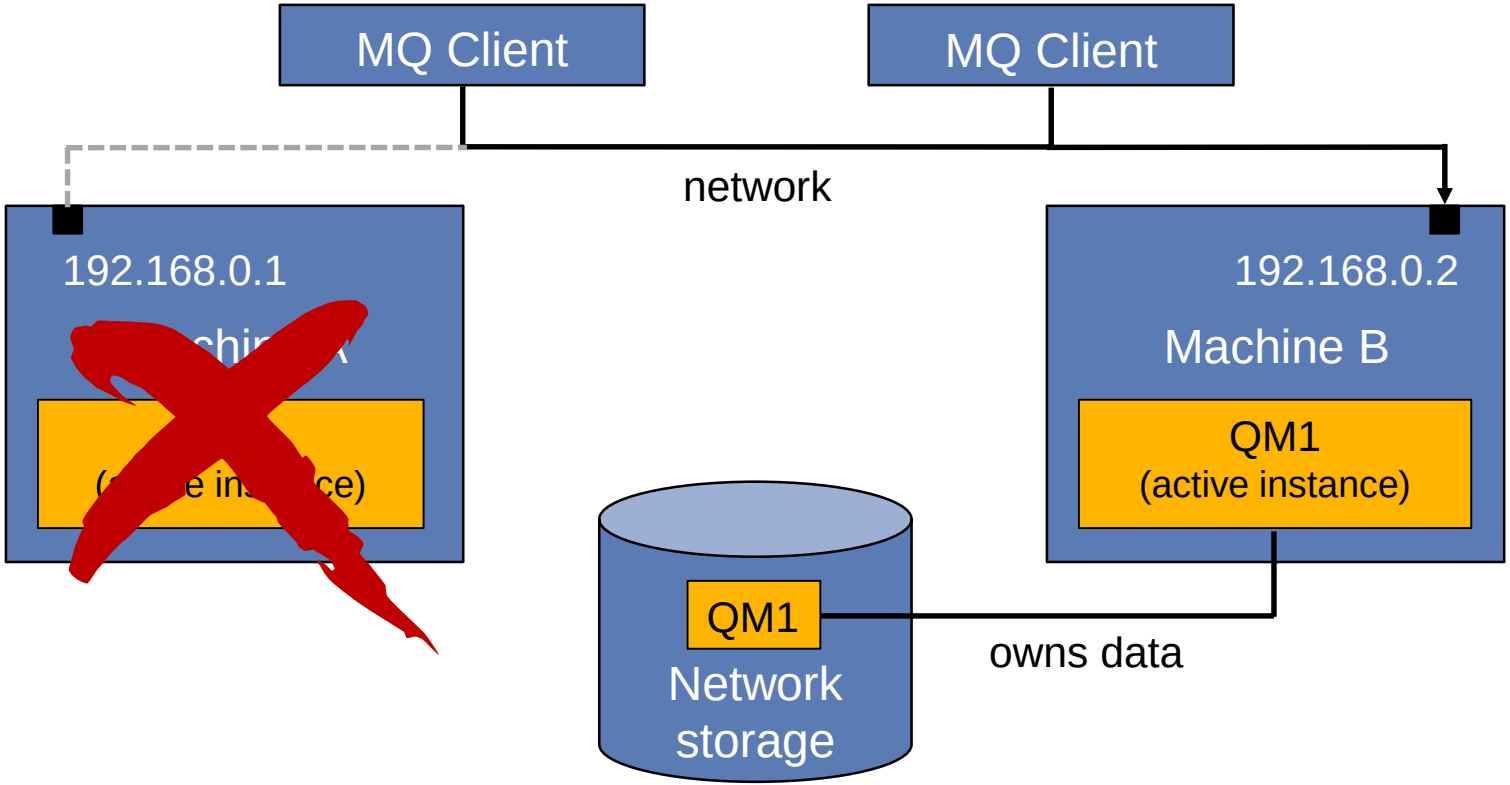
2) Outage of machine A: Client connections broken and locks released on network storage

Multi-instance queue managers



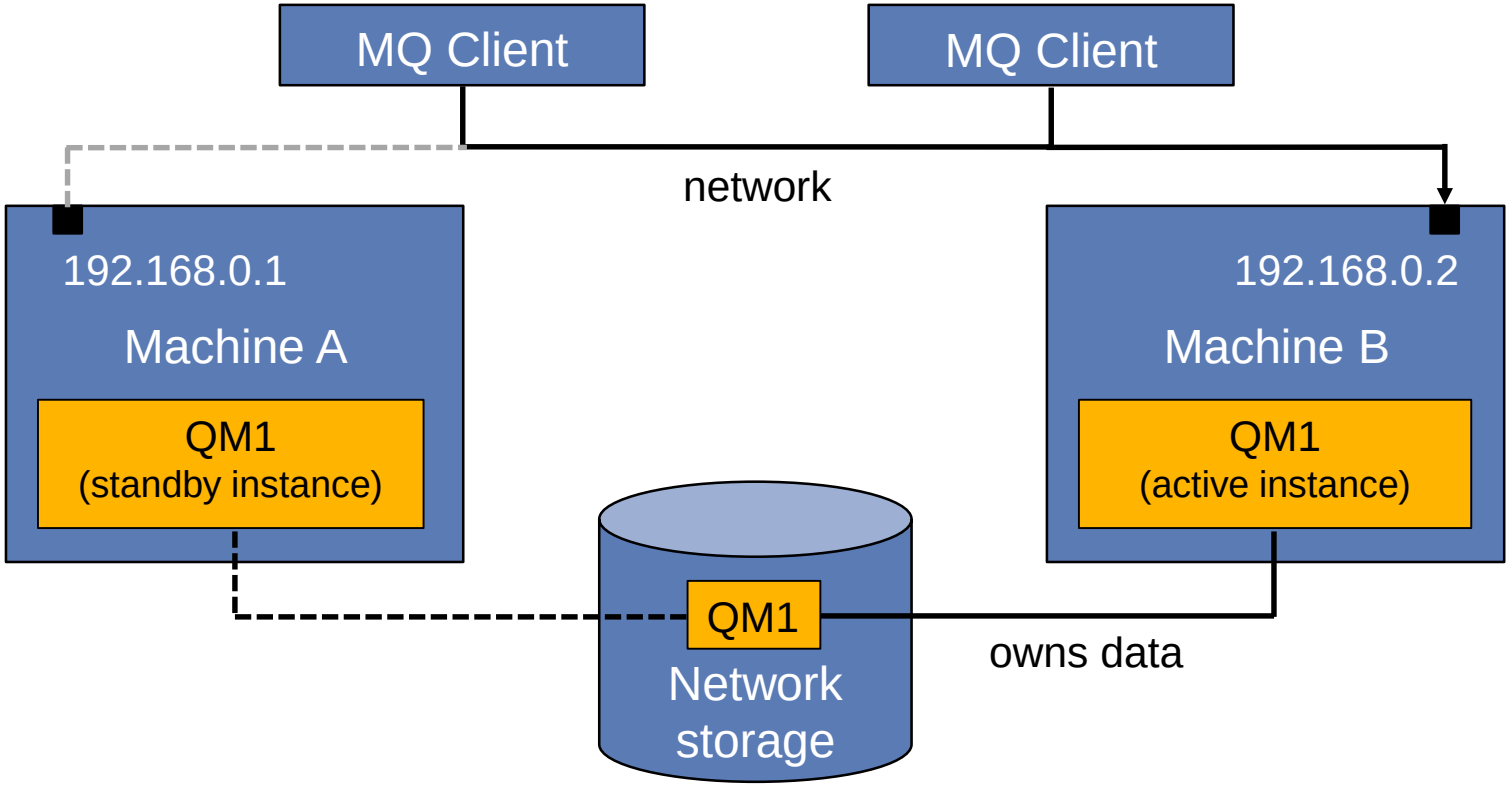
3) Standby instance takes over: Client connections still broken

Multi-instance queue managers



4) Recovery completes: Clients able to reconnect to machine B

Multi-instance queue managers



5) Machine A recovered: QM1 instance restarts in standby mode

Multi-instance queue managers

- No IP address take-over
  - Applications and channels need to know the set of possible IP addresses for the queue manager unless you use an intelligent router
- Important notes for Linux / UNIX:
  - Ensure the mqm UID and GID are the same on both servers
  - Ensure the queue manager data and logs are mounted in the same location
  - Ensure a hard mount is used not a soft mount
- Filesystem requirements
  - Invest in reliable storage and network access that is backed up (as necessary)
  - Can be provided by a networked storage device (such as NAS) or a cluster file system
  - Must satisfy three criteria:
    - Data write integrity, Guaranteed exclusive access to files, Release locks on failure

Notes

- Application recovery
  - Failover is not invisible to applications - need to reconnect (discussed later in this presentation)
    - Uncommitted (in-flight) transactions are backed out
    - Persistent messages are preserved but non-persistent messages are lost
  - Consider using automatic client reconnect (MQCNO_RECONNECT / MQCNO_RECONNECT_Q_MGR)
    - Alternatively, applications can be restarted or written to manually reconnect
  - For server-side applications can configure applications as a queue manager service
  - Detailed information about application recovery considerations:
    - http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.con.doc/q018380_.htm
- IBM is not able to test every possible shared filesystem
  - Documented support position: http://www.ibm.com/support/docview.wss?uid=swg21392025
  - Documented test statement: http://www.ibm.com/support/docview.wss?rs=0&q1=multi-instance&uid=swg21433474&loc=en_US&cs=utf-8&cc=us&lang=en
  - Note mount options and versions (e.g. NFS v4)
- Use amqmfsck with amqsfhac to check a shared filesystem meets MQ's requirements
  - www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.pla.doc/q005820_.htm

HA cluster or multi-instance queue manager?

- Multi-instance queue manager
  - Integrated in to the MQ product
  - Faster failover – delay before queue manager restarted is less
  - Runtime performance dependent on network storage
  - Typically uses NAS (e.g. NFS)
- HA cluster
  - Capable of handling wider range of failures
  - Failover historically rather slow, but improving
  - Capable of more flexible configurations
  - Extra product purchase and skills required
  - Typically uses SAN

Failover considerations

- Failover times are made up of three parts:
  - Time taken to notice the failure
    - Heartbeat missed
    - Bad result from status query
  - Time taken to establish the environment before activating the service
    - Switching IP addresses and disks, and so on
  - Time taken to activate the service
    - This is queue manager restart
- Failover involves a queue manager restart
  - Non-persistent messages, non-durable subscriptions discarded
- For fastest times, ensure that queue manager restart is fast
  - For example, no long running transactions

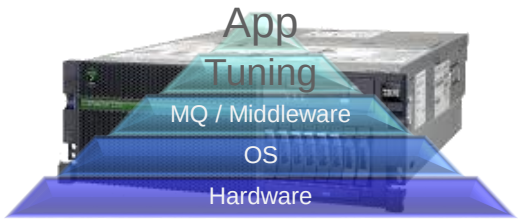
What's an MQ Appliance?

- The scalability and security of IBM MQ
- Integrates seamlessly into MQ networks and clusters
  - Familiar administration model for administrators with MQ skills
- The convenience, fast time-to-value and low total cost of ownership of an appliance
- Ideal for use as a messaging hub running queue managers accessed by clients, or to extend MQ connectivity to a remote location
- Familiar feel for existing MQ users – application interfaces, administration, networking/clustering, security….
- Plus new appliance specific features – e.g. built in high availability

Key differences with appliance form-factor

IBM MQ Appliance

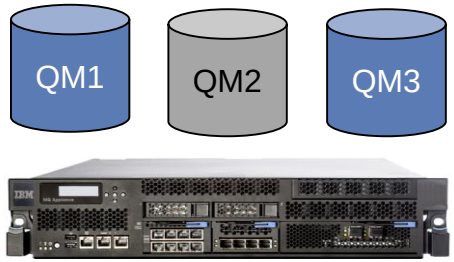
- Prebuilt for hub pattern – no apps on device
- No additional software installation
  - No user exits in MQ
  - Monitoring agents must be remote
- High availability out-of-the-box
- Pre-tuned
- Single firmware update for whole appliance
  - Firmware update inc. appliance and MQ fix pack
  - Can be rolled back as an single unit



IBM MQ on custom server

- DIY hub or generic server – apps + middleware
- Install any software
  - Build & maintain your own custom extensions
  - Add local monitoring agents
- Needs HA cluster SW or network storage for HA
- Custom tuning for each layer (OS/middleware)
- Discrete maintenance for each layer
  - MQ fix packs
  - OS maintenance, security patches, etc.

High availability – concept

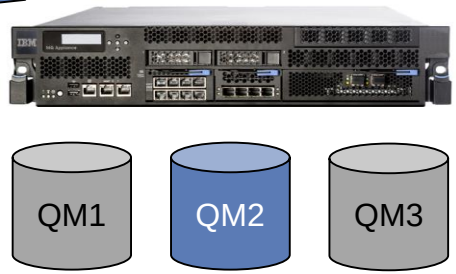


No persistent data loss on failure

No external storage

No additional skills required

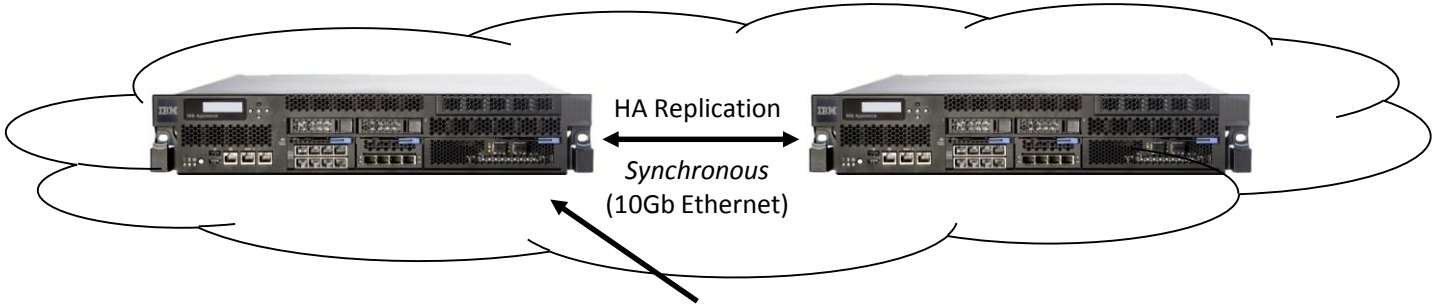
Fully synchronous replication



Manual control of failover for migration/maintenance

Queue manager level active/passive (i.e. both appliances can run workload)

High availability – floating IP support (9.0.1)



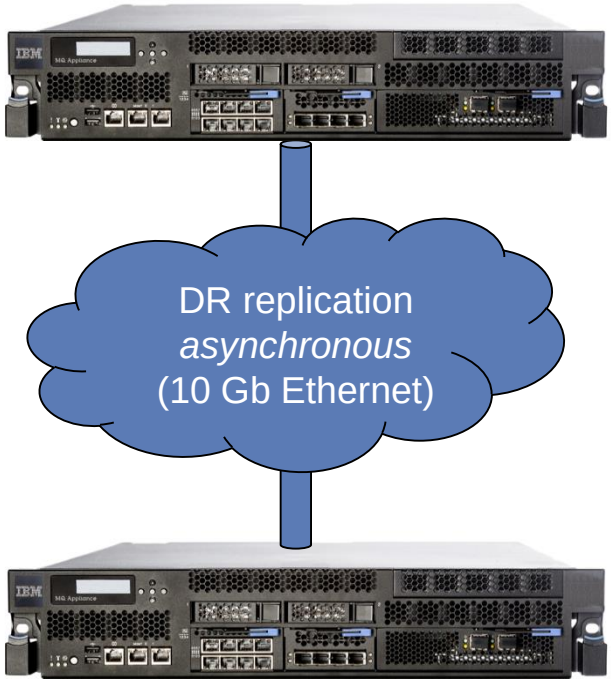
In version 8 of the MQ Appliance, clients connecting to HA queue managers must be aware of all possible IP addresses (e.g. via comma separated list or CCDT)

Particularly useful when replacing existing standalone queue managers with HA Appliance queue managers, requiring no changes on the application side

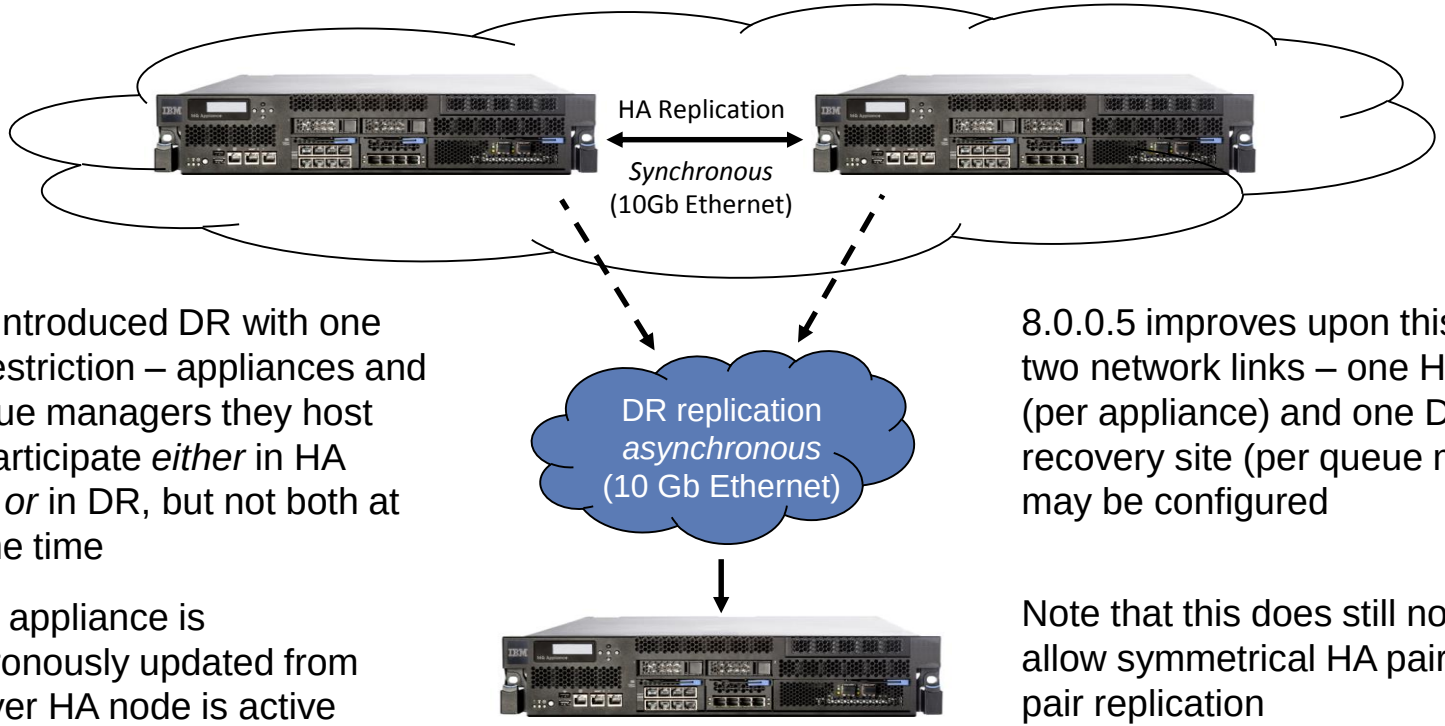
Client applications now able to use a single IP address associated with the queue manager - automatically adopted by whichever instance is currently active

Disaster recovery (8.0.0.4) – concept

- Provides for longer distance recovery than HA
  - e.g. out-of-region standby site
- Still ultimately requires high bandwidth connectivity as all persistent data is fully mirrored
- But - asynchronous so better choice than HA for higher latency, ‘bursty’ or ‘lossy’ networks
  - Means most recent messages are potentially lost on fail-over and application logic must consider this
- Manual interaction required to trigger fail-over / fail-back



Disaster recovery for HA groups (8.0.0.5)



8.0.0.4 introduced DR with one major restriction – appliances and the queue managers they host could participate *either* in HA groups, *or* in DR, but not both at the same time

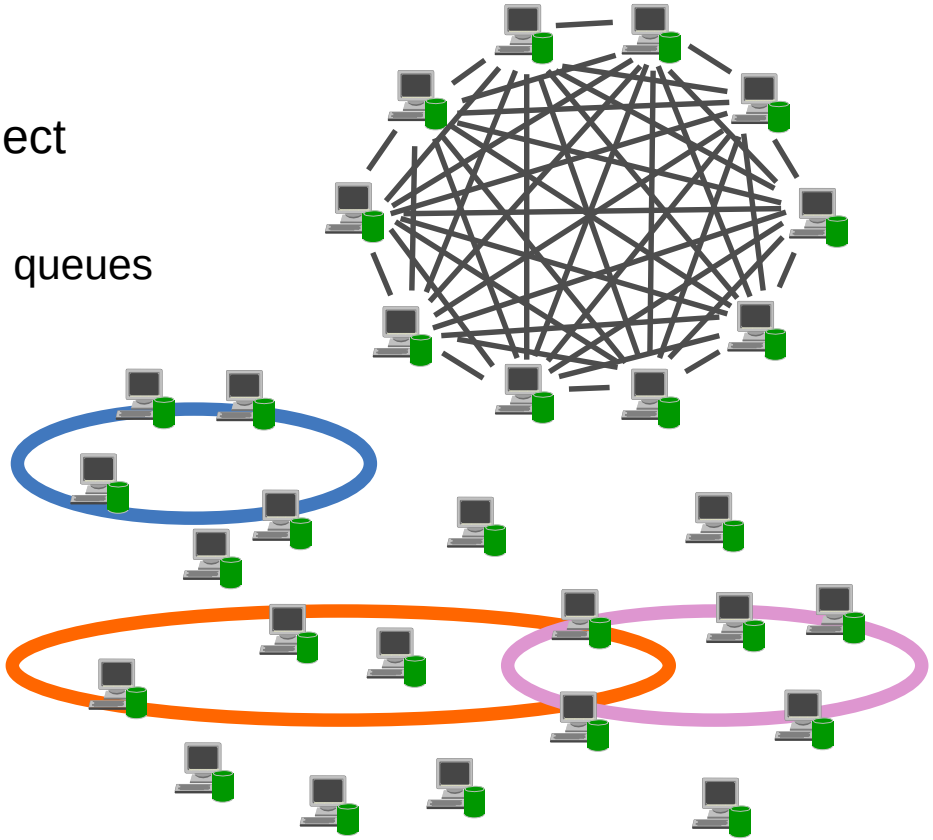
The DR appliance is asynchronously updated from whichever HA node is active

8.0.0.5 improves upon this using two network links – one HA partner (per appliance) and one DR recovery site (per queue manager) may be configured

Note that this does still not (yet) allow symmetrical HA pair to HA pair replication

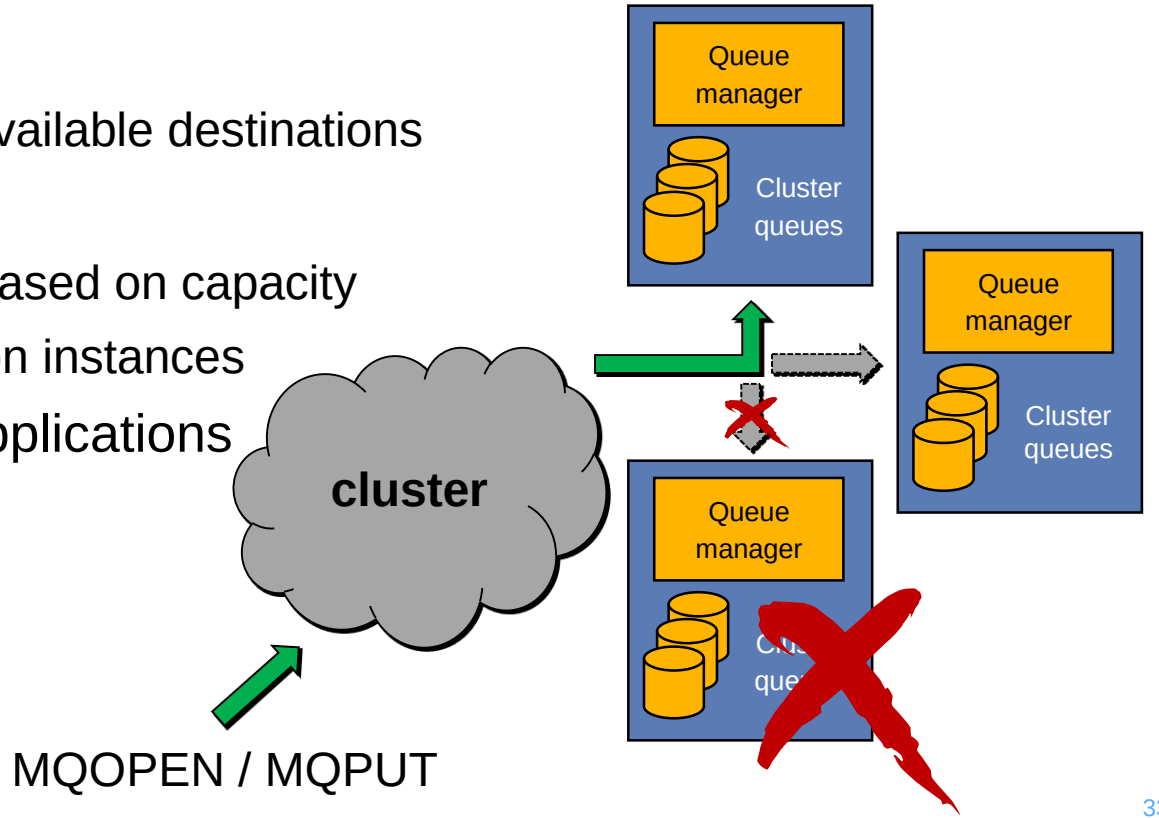
Queue manager clusters

- Simplified administration
  - Large MQ networks require many object definitions
    - Channels, transmission queues, remote queues
- Workload balancing
  - Spread the load
  - Route around failures
- Flexible connectivity
  - Overlapping clusters
  - Gateway queue managers
- Publish/subscribe clusters



Queue manager clusters

- Multiple instances of a queue
  - Failure isolation
  - Messages routed only to available destinations
- Scalable throughput
  - Servers can be weighted based on capacity
  - Can add/remove destination instances
- Changes transparent to applications
- Definition through usage
  - Resolved at MQOPEN
- MQGET always local



Notes

Consider a client using the black queue that is available in the cluster on three server queue managers. A message is MQPUT by the client and is delivered to *one* of the servers. It is processed there and a response message sent to a ReplyToQueue on the client queue manager.

In this system, if a server becomes unavailable, then it is not sent any further messages. If messages are not being processed quickly enough, then another server can be added to improve the processing rate.

It is important that both these behaviours are achieved by existing MQI applications, i.e. without change. It is also important that the administration of clients and servers is easy. It must be straight forward to add new servers and new clients to the server.

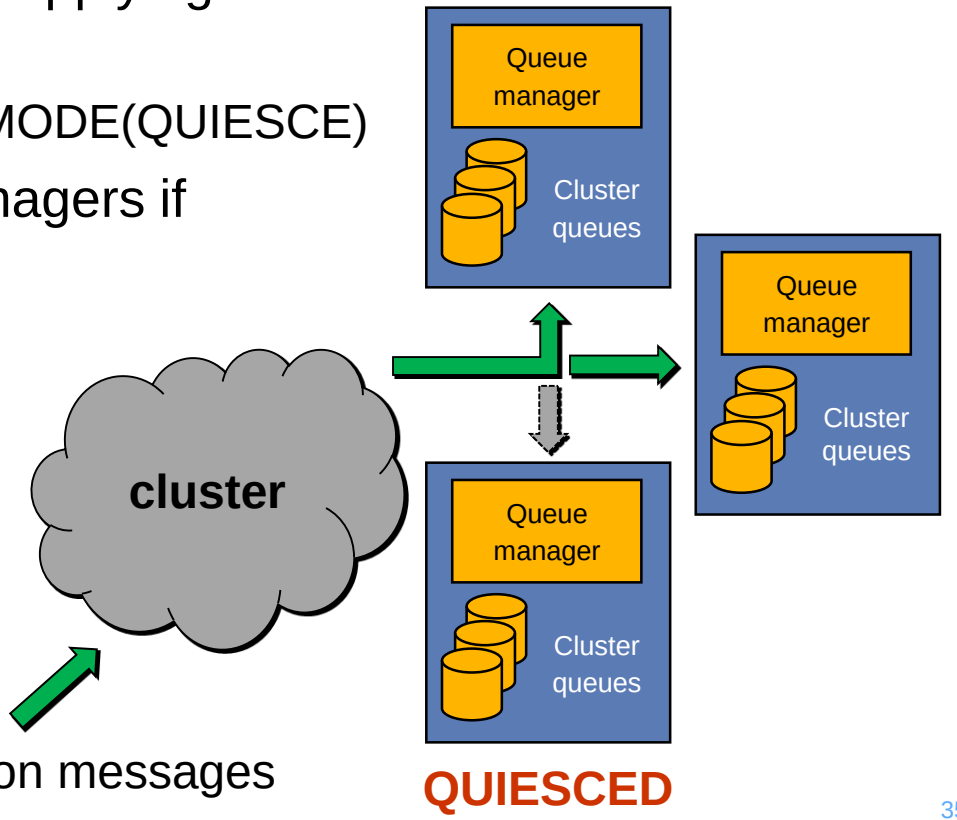
We see how a cluster can provide a highly available and scalable message processing system. The administration point in processing is MQOPEN because this is when a queue or queue manager is identified as being required by an application.

Note that only one message is sent to a server; it is not replicated three times, rather a specific server is chosen and the message sent there. Also note that MQGET processing is still local, MQGET is not extended into the network.

Clustering only provides queue availability, not necessarily message availability. Once a cluster queue instance has been selected and a message has been routed to a queue manager that message can only be accessed from that queue manager. If the queue manager becomes unavailable the message is ‘marooned’ until the queue manager is available once more. This problem can be overcome by also using a queue-sharing group (discussed next). A shared queue can also be clustered, providing multiple routes to both the queue and its messages.

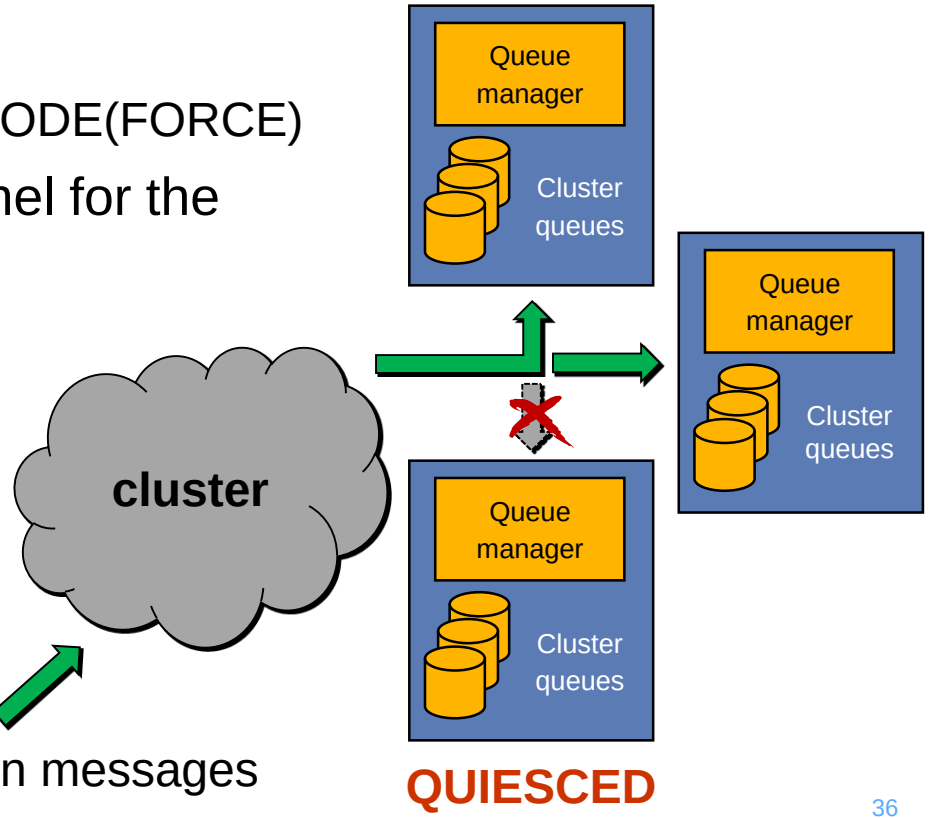
Cluster maintenance

- Suspend a queue manager prior to applying maintenance
  - SUSPEND QMGR CLUSTER(xxx) MODE(QUIESCE)
- Messages sent to other queue managers if possible
- Allows messages already sent to be processed to drain cluster queue instance(s)
- Resume a queue manager once the maintenance is complete
  - RESUME QMGR CLUSTER(xxx)



Cluster maintenance

- Can forcibly prevent further communication if required
  - SUSPEND QMGR CLUSTER(xxx) MODE(FORCE)
- Forcibly ends cluster receiver channel for the cluster
- Messages to be routed to quiesced queue manager are queued within the cluster



Notes

The previous two slides illustrate how a queue manager can be suspended within a cluster to quiesce or terminate workload routed to it prior to applying maintenance.

If a queue manager is suspended using MODE(QUIESCE), which is the default if the mode is not specified, messages are routed to other queue managers in the cluster wherever possible. If messages are specifically sent to the quiesced queue manager, or there are no other available instances of a cluster queue, messages may still be sent to it.

To prevent all incoming communication from the cluster MODE(FORCE) can be specified. This will result in messages being queued on the cluster transmission queues of other queue managers in the cluster if they must be sent to the quiesced destination.

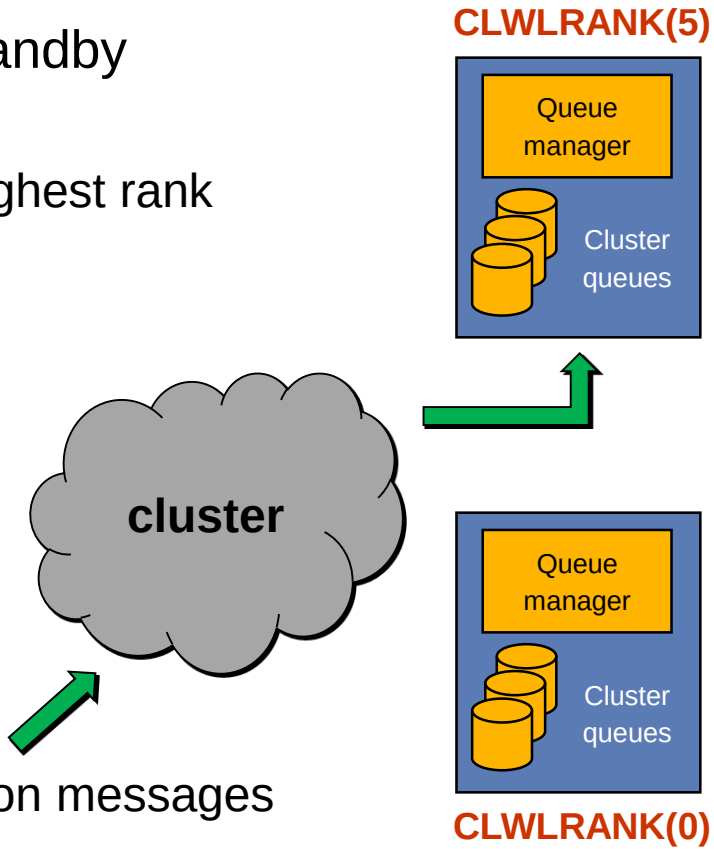
After disconnecting the server application instance for the quiesced queue manager the upgrade, or other maintenance, can be performed.

After ensuring the maintenance has been completed the queue manager can be resumed in the cluster to restore normal operation.

This sequence of actions can then be repeated for each other queue manager.

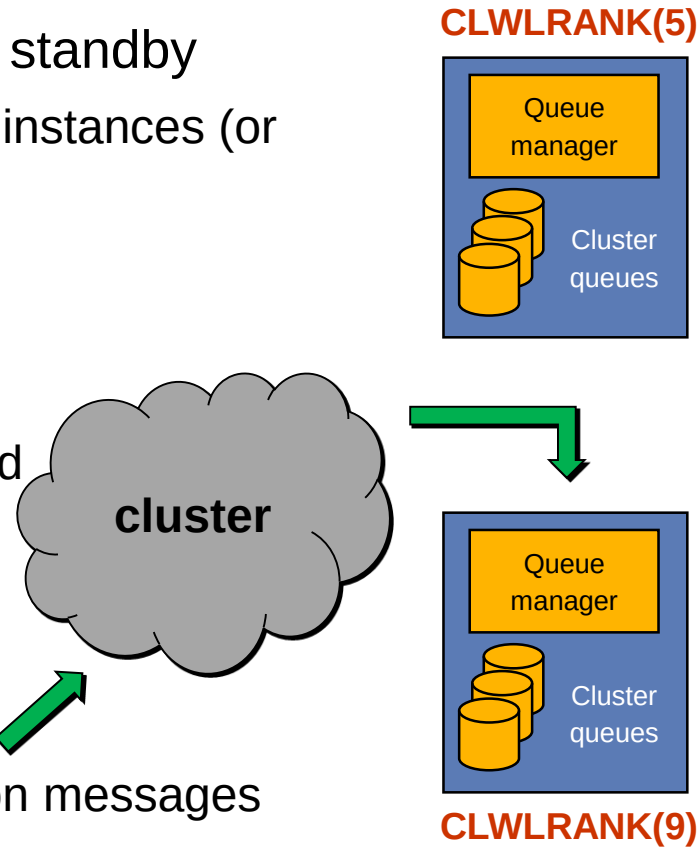
Active-standby using clustering

- CLWLRANK can be used for an active-standby configuration
  - Messages routed to destinations with the highest rank
  - Lower ranked destinations are not used
  - Can use lower ranks for standby instances
  - Workload distributed between destinations with the same rank
- Specify on the cluster-receiver channel
  - Cluster-sender value likely to be ignored



Active-standby using clustering

- Update CLWLRANK to switch workload to standby
  - Maintenance can then be applied to primary instances (or sites)
  - Restore ranking to switch back
  - Provides manual administrative control
  - CLWLPRTY can also be used but channel status affects the chosen destinations instead of administrative control



Notes

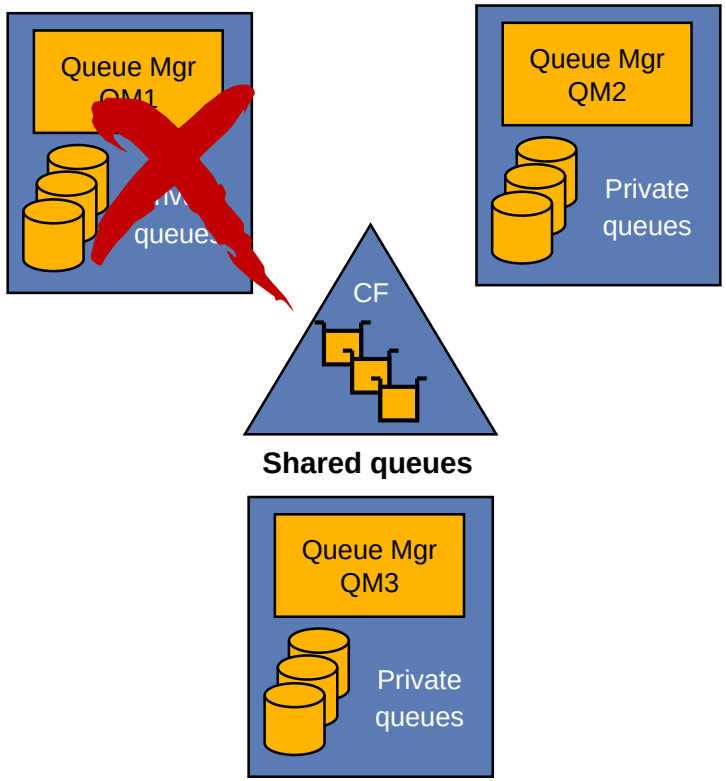
The previous two slides illustrate how cluster workload attributes can be used to control to which queue managers messages are routed. These attributes can be used to configure an active-standby configuration where messages are routed to a subset of queue managers. A second subset, which may be located at a different site, can be used instead by updating them to have a more preferable rank. This approach can be used to switch between them when maintenance is scheduled on one or the other.

The CLWLRANK attribute has a higher precedence than the channel status. This means that if a channel is not available to the destination with the highest rank then messages are held on the transmission queue. The messages are only released when a channel becomes available. Specifically, messages are not routed to a destination with the next lowest rank. This provides a control point for maintenance to avoid messages being routed to destinations that are not ready to process them.

The CLWLPRTY attribute is also provided that has a lower precedence than the channel status. This can be used to route messages to destinations in preference to others, subject to their availability.

Queue-sharing groups (QSGs)

- Available with a z/OS SysPlex
- Shared queues held in a Coupling Facility
- All queue managers in a QSG can access the same shared queues
  - and their messages
- Benefits
  - Messages remain available even if a queue manager fails
  - Pull workload balancing
  - Applications can connect to the group using a QSG name
    - Removes affinity to a specific queue manager



Notes

A queue-sharing group (QSG) is a co-operating set of queue managers running in the same z/OS SysPlex. Each queue manager has its own private queues, but also has access to shared queues that are held in the Coupling Facility. Multiple queue-sharing groups can be defined in the same SysPlex, but each queue manager can belong to only one QSG.

An application that accesses shared queues can connect to any of the queue managers in the queue-sharing group. This is because all the queue managers in the queue-sharing group can access the same set of shared queues, so the application does not depend on the availability of a particular queue manager. If an individual queue manager or LPAR fails an application can reconnect to any other available queue manager in the group. Client applications can similarly connect to any available queue manager using a virtual IP address for the QSG.

Queue-sharing groups support group attach for locally connected BATCH applications and CICS regions. This removes the affinity to a single queue manager by allowing the QSG name to be specified instead. Connections are routed to any available queue manager in the QSG that is running on the same LPAR.

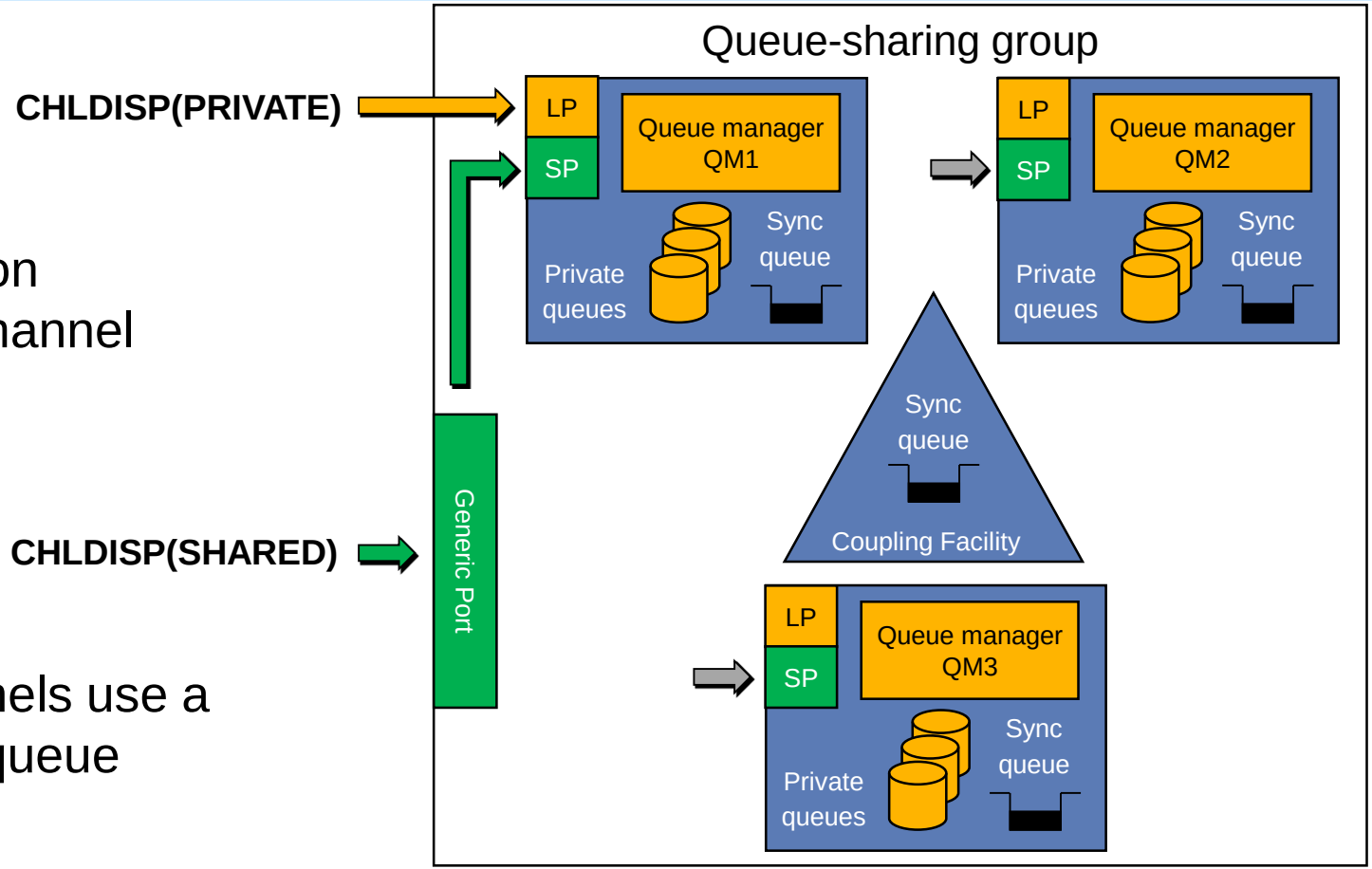
During a maintenance outage applications can re-connect to an alternative queue manager without change in response to the original queue manager being quiesced. The applications can then resume normal processing with access to the same shared queues and the messages on them.

Group attach also simplifies creating additional instances of an application, or moving an application to a different LPAR in the same SysPlex because the connection configuration details can remain the same.

Shared queue messages are only lost if the CF fails. CF duplexing can be configured to mitigate this and the BACKUP CFSTRUCT command can be used to backup persistent messages to a queue manager’s recovery log. The RECOVER CFSTRUCT command can be used to rebuild shared queues in the event of a failure.

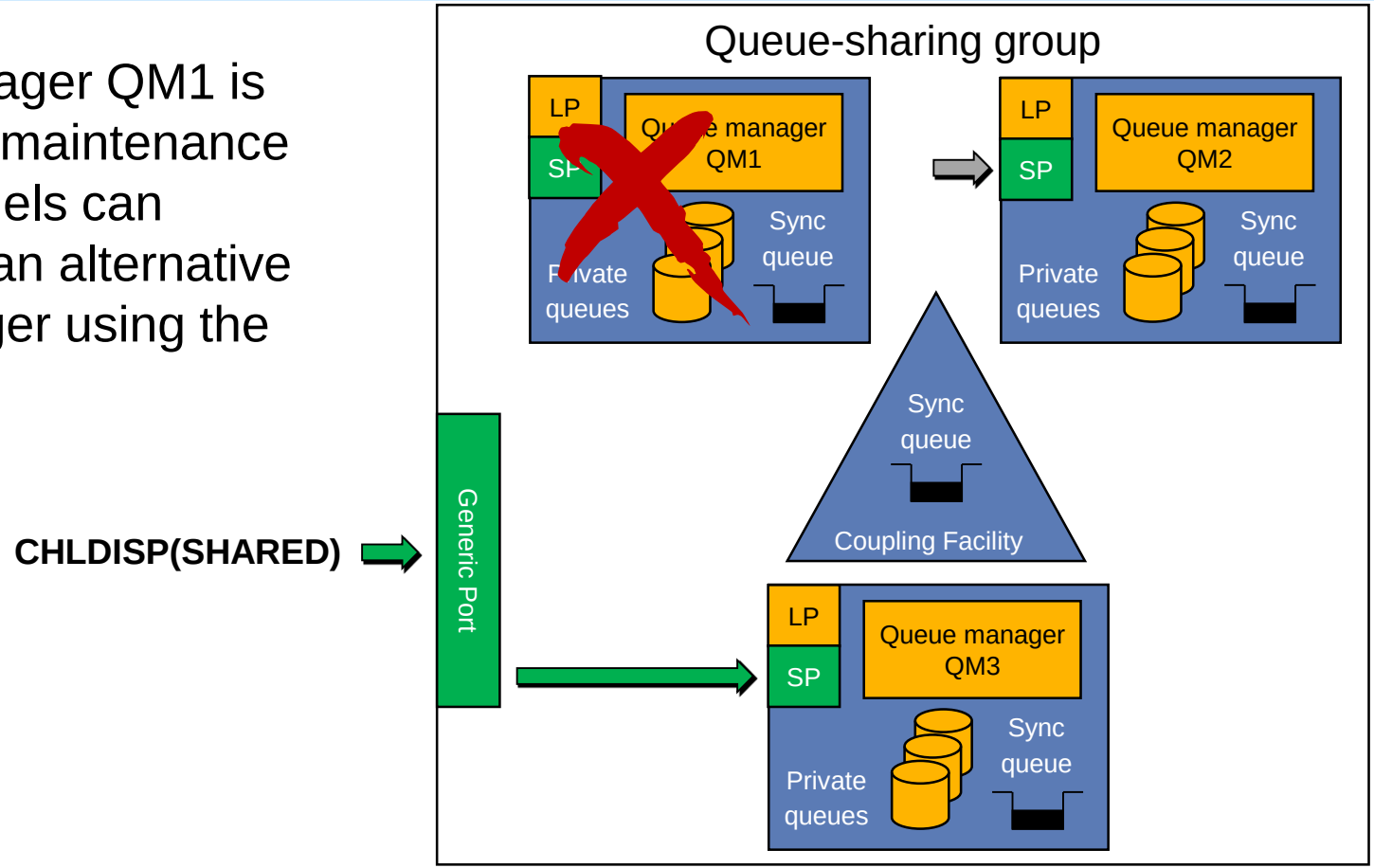
Inbound queue manager channels to a QSG

- Port disposition determines channel disposition
- Shared channels use a shared sync queue



Inbound queue manager channels to a QSG

- If queue manager QM1 is shutdown for maintenance shared channels can reconnect to an alternative queue manager using the generic port



Notes

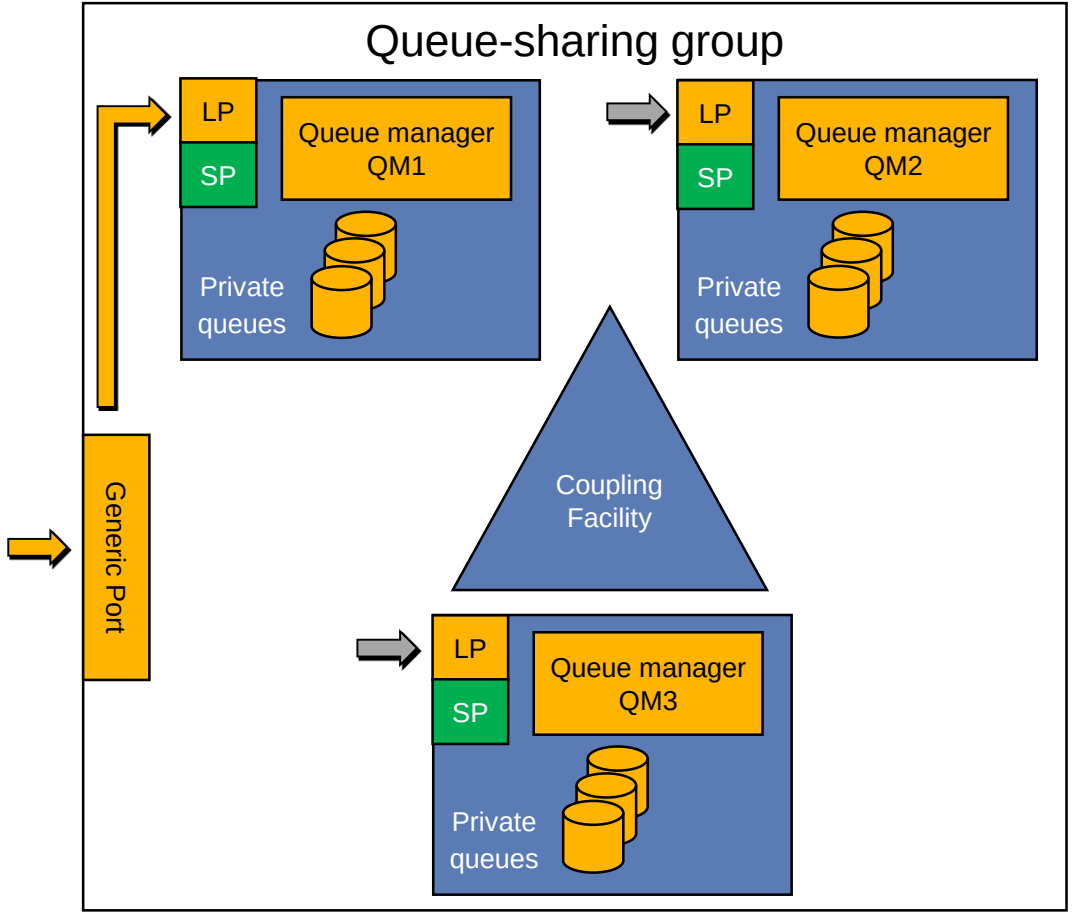
Queue managers in a queue-sharing group can listen on two different types of port. The first is a private local listener port (LP) that is available to queue managers on all platforms. A queue manager channel that connects using this port has an affinity to that queue manager.

Each queue manager in the queue-sharing group can also connect to a generic port. A queue manager channel that connects to this port is connected to one of the available queue managers in the QSG via a shared listener port (SP). The generic port provides high availability because the connecting (sending) queue manager can resynchronise with any member of the queue-sharing group using the shared sync queue. The most common technique for implementing the generic port is Sysplex Distributor.

If a queue manager to which a shared inbound queue manager channel has been established is shutdown for maintenance the channel can reconnect to another available queue manager in the QSG and continue processing. No configuration changes are required.

Inbound client channels to a QSG

- Client channels are stateless so they do not have a sync queue
- Better to use a generic port targeting a private listener on each queue manager
- Can still specify the QSG name to connect to avoid queue manager affinity
- Requires GROUPUR support for transactional (XA) clients



Notes

Client channels do not require state to be stored on a synchronisation queue so there is limited benefit to connecting a client to a shared listener port (SP). A client connected to a shared listener port also incurs a performance cost as a result of updating the channel status in DB2.

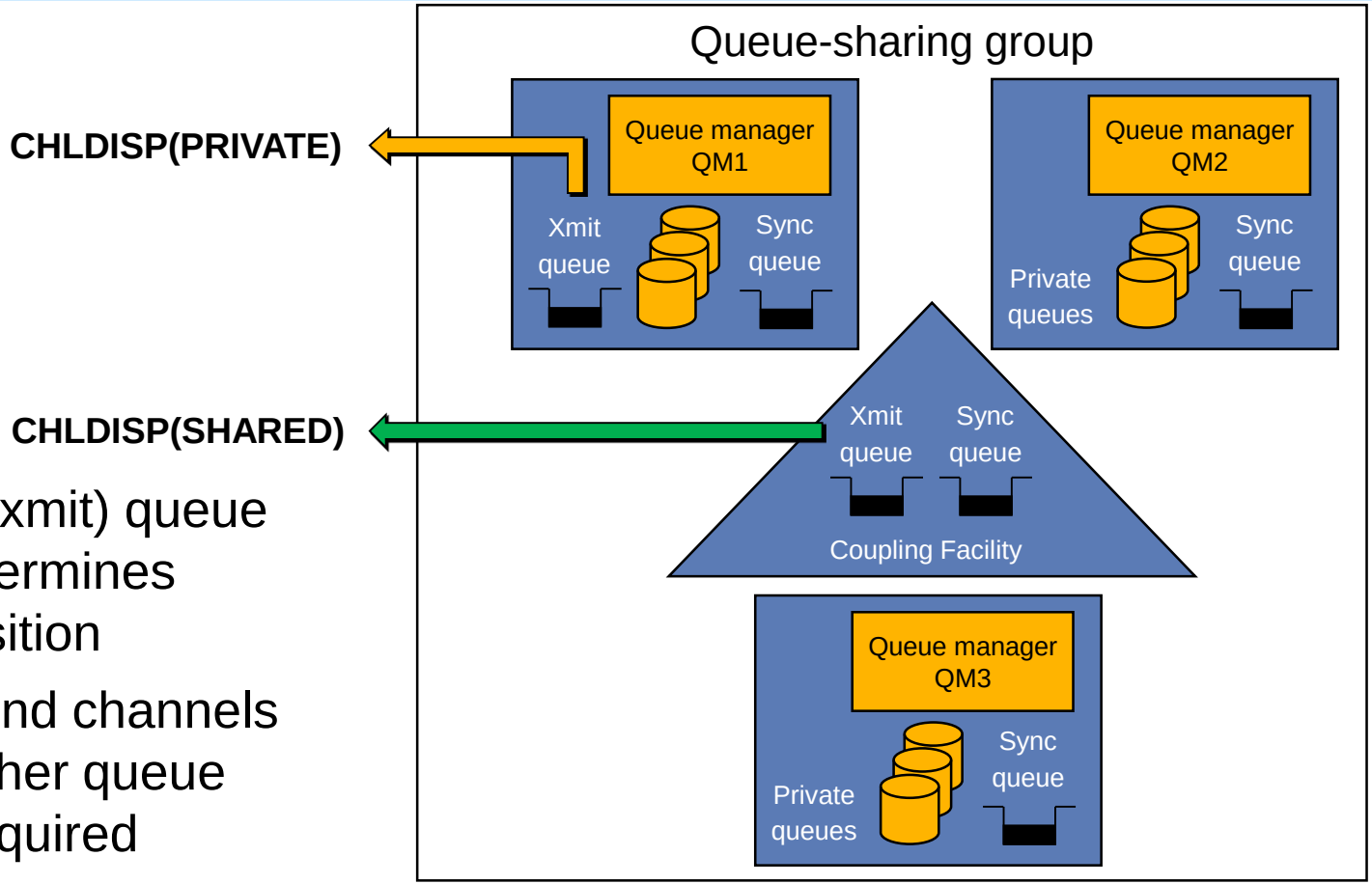
Client channels can still benefit from high availability to a queue-sharing group by connecting to a generic port, as per shared inbound queue manager channels. However, the generic port for client connections should target the local listener port that is private to each queue manager.

Clients connecting to the generic port should specify the QSG name because they do not have an affinity to a specific queue manager. If a queue manager is shutdown for maintenance clients are connected to one of the remaining members of the QSG targeted by the generic port.

Transactional (XA) clients can only connect using a generic port to queue managers at version 7.0.1 or higher that have group units of recovery (GROUPUR) enabled. This is required so the transaction manager is able to inquire and resolve any in-doubt transactions that might reside on another queue manager in the QSG.

Outbound queue manager channels from a QSG

- Transmission (xmit) queue disposition determines channel disposition
- Shared outbound channels restart on another queue manager as required



Notes

Queue managers in a queue-sharing group also support shared channels for outbound communication to another queue manager. A shared outbound channel sends messages from a shared transmission queue that resides in the Coupling Facility.

In the event of a queue manager failure, or shutdown due to maintenance, the outbound channel is restarted on another queue manager in the same QSG. The synchronisation state for shared channels is similarly stored on a shared queue so this information is available throughout the group.

Shared outbound channels are workload balanced across all the systems in the queue-sharing group to avoid all channels being started on a single queue manager unnecessarily.

Application connectivity

- If an application loses connection to a queue manager, what does it do?
  - End abnormally?
  - Handle the failure and retry the connection?
  - Reconnect automatically thanks to an application container?
    - WebSphere Application Server contains logic to reconnect JMS clients
  - Use MQ automatic client reconnection?

- Applications can also be triggered or started as queue manager services
  - These two techniques can be useful during queue manager failover

Automatic client reconnection

- MQ client can automatically reconnect when connection broken
  - MQI C clients and standalone JMS clients
  - JMS in app servers (EJB/Servlet, MDB) can't auto-reconnect
    - Not needed for MDB anyway as resource adapter does that
    - EJB/servlets need to retry
- Reconnection includes reopening queues, remaking subscriptions
  - All MQI handles keep their original values
- Can reconnect to same queue manager or another, equivalent queue manager
- MQI or JMS calls block until connection is remade
  - By default, will wait for up to 30 minutes
  - Long enough for a queue manager failover (even a really slow one)

Automatic client reconnection

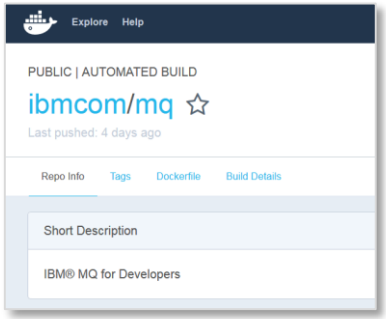
- Can register event handler to observe reconnection
- Not all MQI is seamless, but majority repaired transparently
  - Browse cursors revert to the top of the queue
  - Non-persistent messages are discarded during restart
  - Nondurable subscriptions are remade and may miss some messages
  - In-flight transactions backed out
- Tries to keep dynamic queues with same name
  - If queue manager doesn't restart, reconnecting client's TDQs are kept for a while in case it reconnects
  - If queue manager does restart, TDQs are recreated when it reconnects

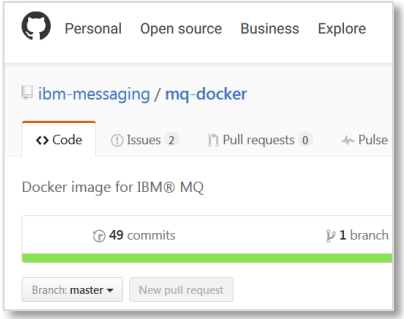
Client configurations for availability

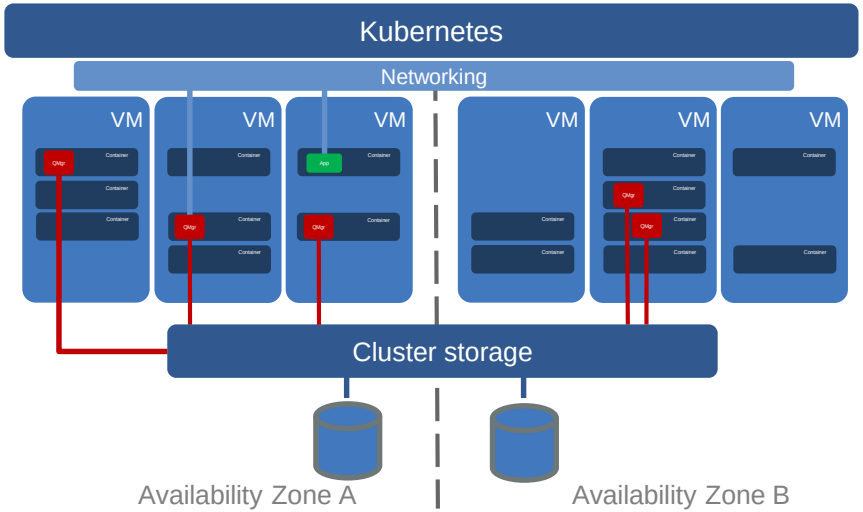
- Use wildcarded queue manager names in CCDT
  - Gets weighted distribution of connections
  - Selects a “random” queue manager from an equivalent set
- Use multiple addresses in a CONNAME
  - Can potentially point at different queue managers
  - More likely point at the same queue manager in a multi-instance setup
- Use automatic reconnection
- Pre-connect exit from V7.0.1.4
- Use IP routers to select address from a list
  - Based on workload or anything else known to the router
- Can use all of these in combination!

High availability in cloud environments

- IBM MQ is supported to run inside Docker containers, bringing the benefits of containers to MQ
  - Lightweight containers for running MQ
  - Predictable and standardized units for deploying MQ
  - Process, resource and dependency isolation
  - Best practice guidance
- IBM provided sample Docker files for customizing and building your own Docker images
- IBM MQ Advanced for Developers V9 available direct from Docker Hub
- Docker enables MQ deployments to be provisioned and managed within the same orchestration frameworks that make Docker so exciting
  - Kubernetes, Mesos, Swarm, Fleet, ...
  - or individual IaaS cloud container services
    - Bluemix, Amazon EC2, Azure, …



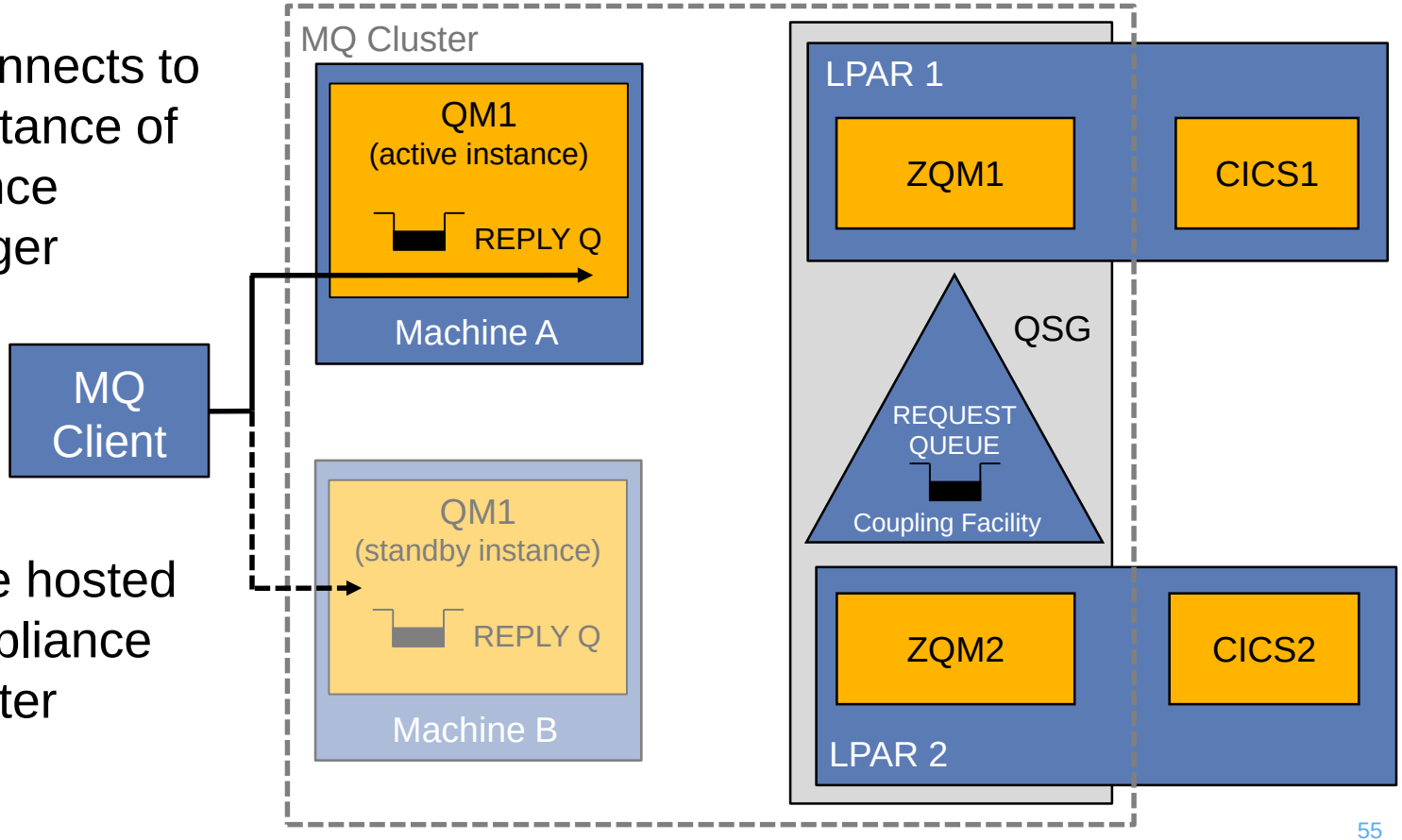




Putting it all together

MQ Client connects to the active instance of a multi-instance queue manager

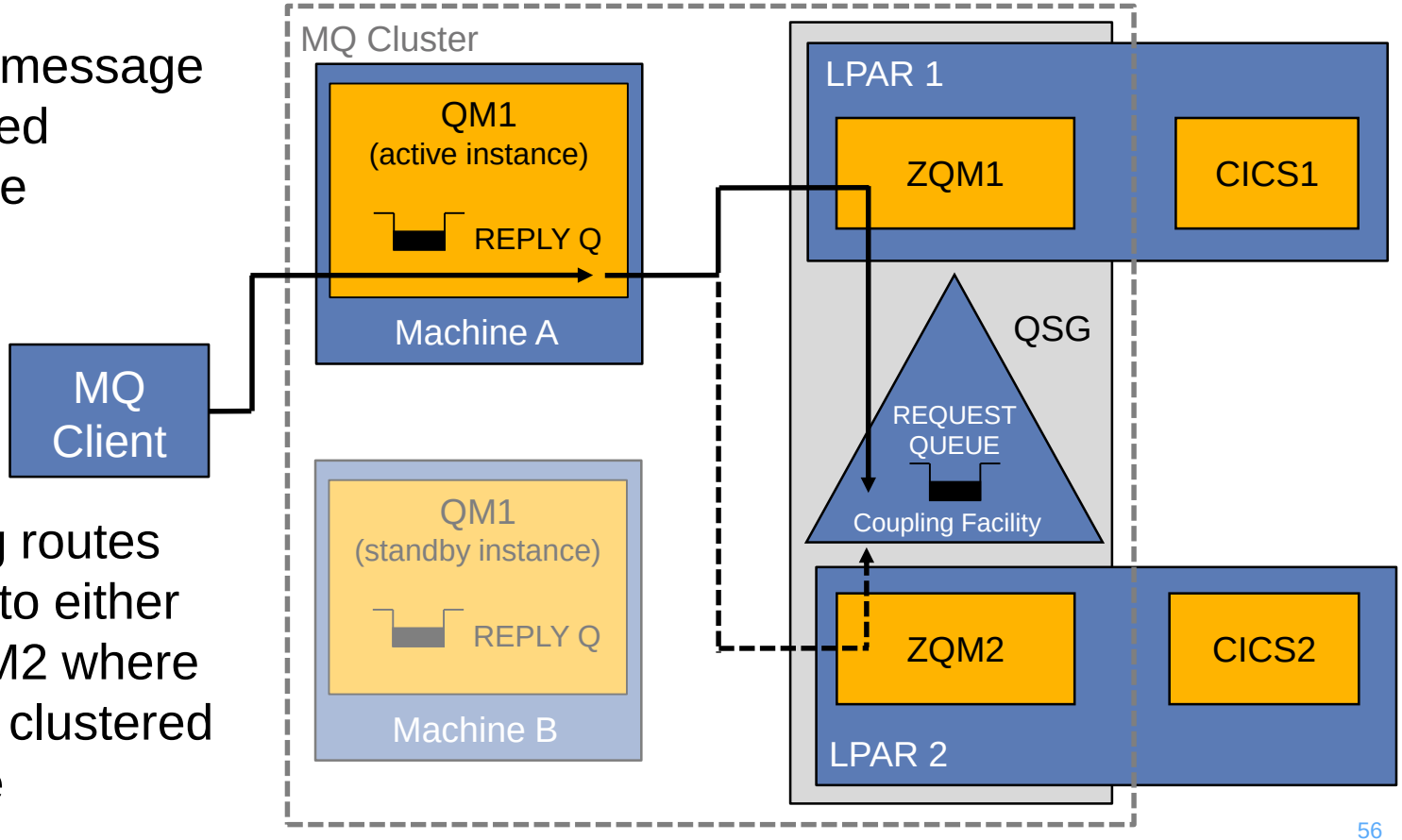
QM1 could be hosted on an MQ Appliance or an HA cluster instead



Putting it all together

Client puts a message to the clustered request queue

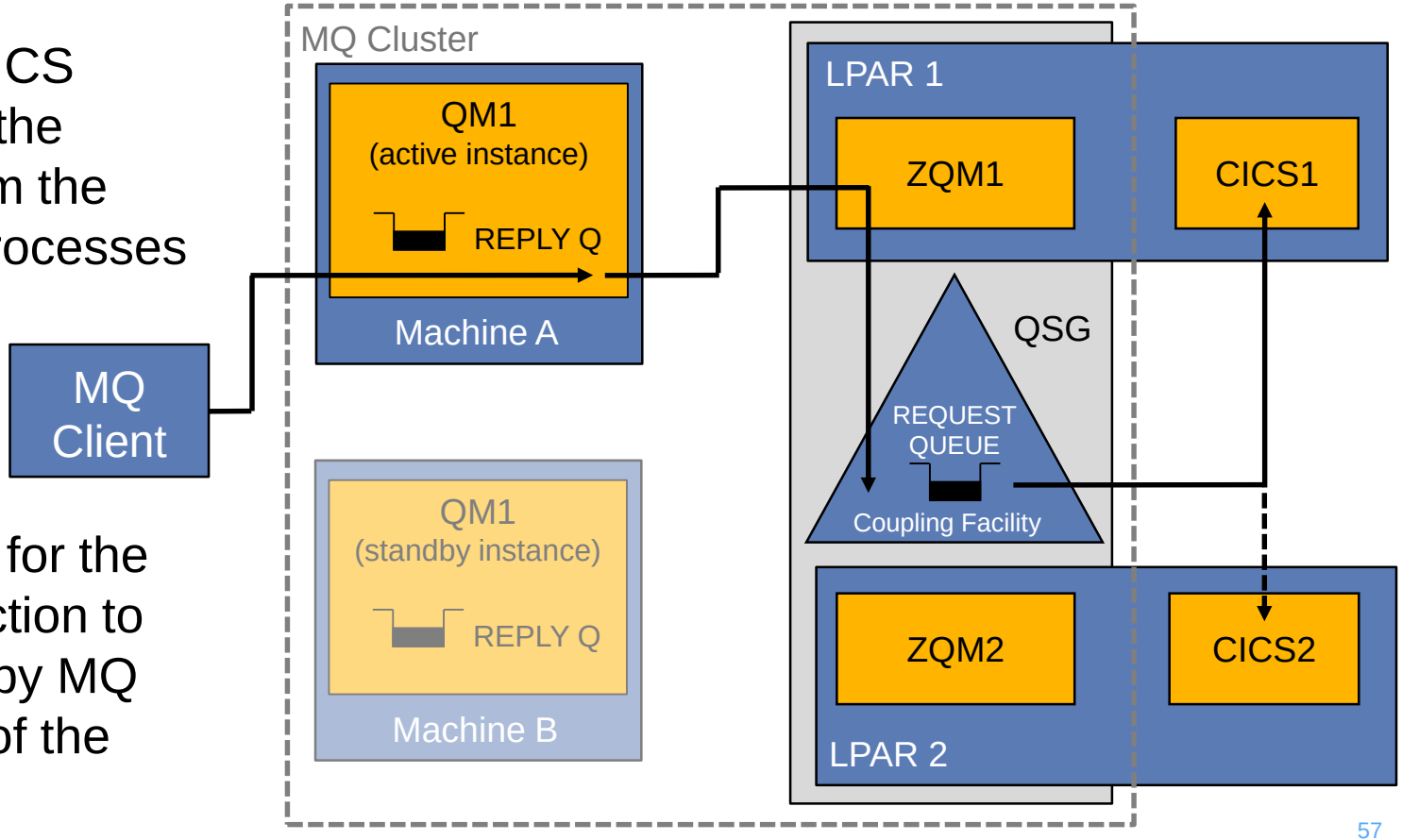
MQ clustering routes the message to either ZQM1 or ZQM2 where it is put to the clustered shared queue



Putting it all together

One of the CICS regions gets the message from the queue and processes the request

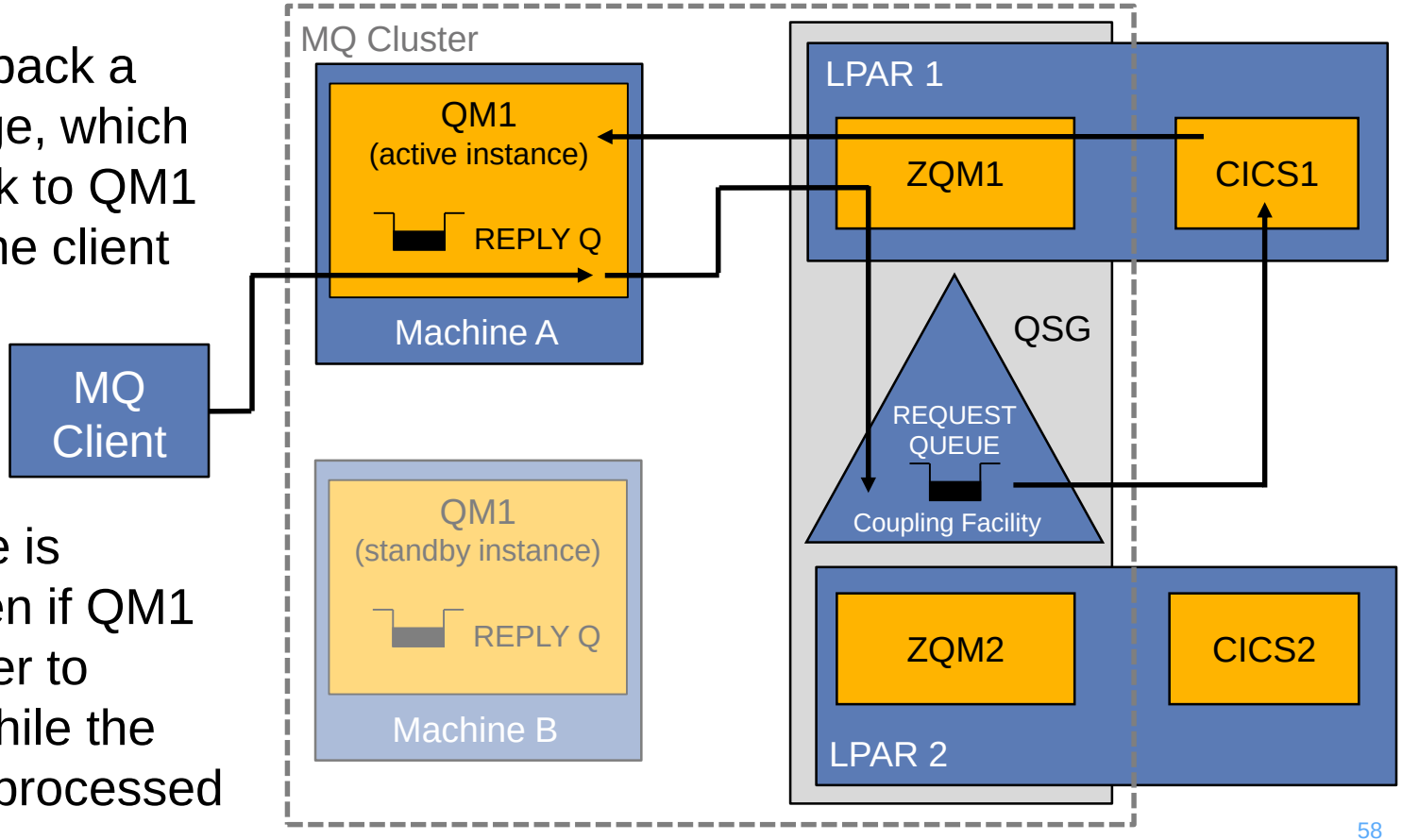
It is common for the CICS transaction to be triggered by MQ upon arrival of the message



Putting it all together

CICS sends back a reply message, which is routed back to QM1 then got by the client

The message is delivered even if QM1 has failed over to machine B while the request was processed



Summary

- HA solutions employ:
  - Redundancy to eliminate single points of failure – high speed failover.
  - Dynamic routing to redistribute work in response to failures.
- IBM MQ HA capabilities
  - HA clusters
  - Multi-instance queue managers
  - IBM MQ Appliance
  - Queue manager clusters
  - Queue-sharing groups
- Application connectivity

IBM MQ Lab Tour – Paris – 26 September 2017

Questions?