#SHAREorg

SHARE
Technology · Connections · Results

Highly Available Messaging Rock solid MQ

Session #11511

Paul S Dennis
dennisps@uk.ibm.com

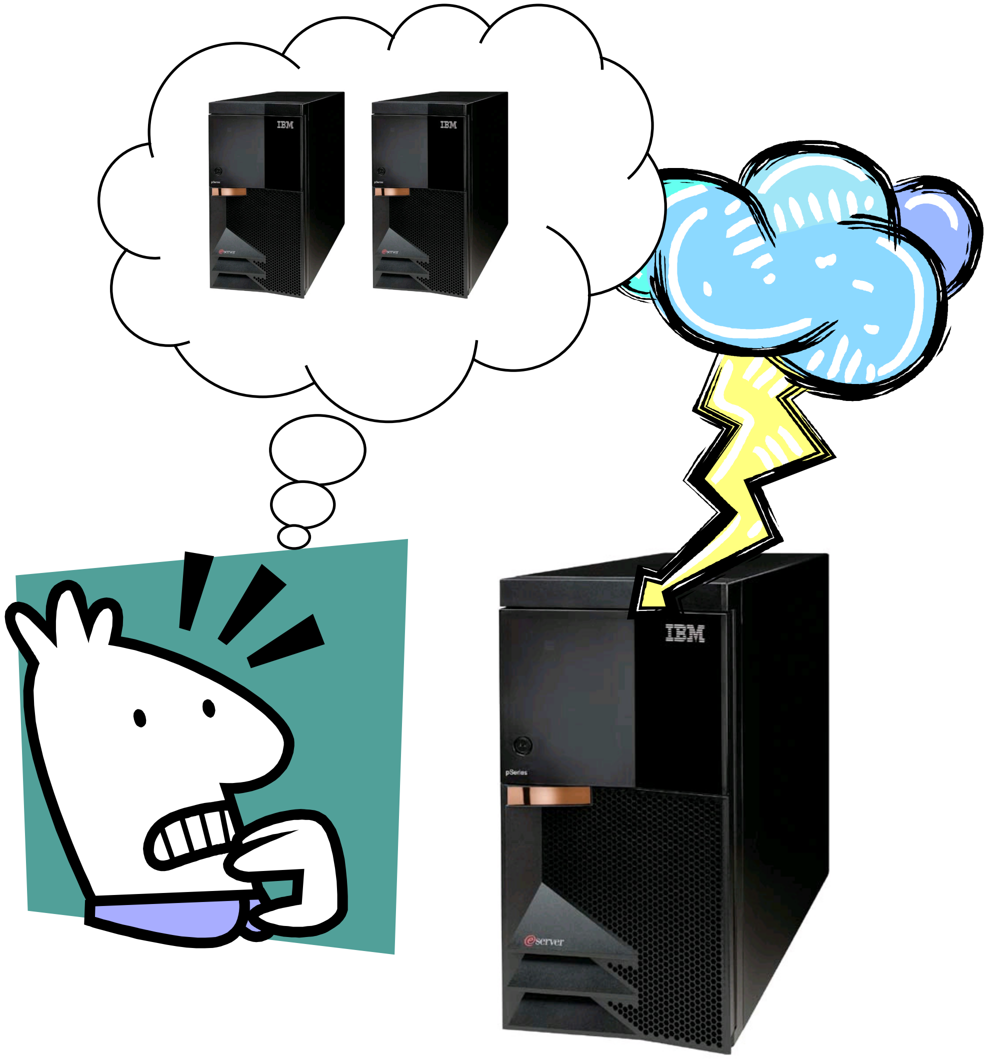
SHARE in Anaheim 2012

Agenda

- **Introduction to HA**
- WebSphere MQ HA technologies
- Using MQ in an HA cluster
- HA and WebSphere Message Broker
- Application considerations

Introduction

- Techniques and technologies to ensure availability of messaging
- Anything that can cause an outage is significant
  - e.g. an overloaded system
- You can have the best HA technology in the world, but you have to manage it correctly
- HA technology is not a substitute for good planning and testing!



Availability objective

- The objective is to achieve 24x7 availability of messaging

- Not always achievable, but we can get close
  - 99.9% availability = 8.76 hours downtime/year
  - 99.999% = 5 minutes
  - 99.9999% = 30 seconds

- Potential outage types:
  - 80% scheduled downtime (new software release, upgrades, maintenance)
  - 20% unscheduled downtime (source: Gartner Group)
    - 40% operator error
    - 40% application error
    - 20% other (network failures, disk crashes, power outage etc.)

Single Points of Failure

- With no redundancy or fault tolerance, a failure of any component can lead to a loss of availability
- Every component is critical. The system relies on the:
  - Power supply, system unit, CPU, memory
  - Disk controller, disks, network adapter, network cable
  - ...and so on
- Various techniques have been developed to tolerate failures:
  - UPS or dual supplies for power loss
  - RAID for disk failure
  - Fault-tolerant architectures for CPU/memory failure
  - ...etc
- Elimination of SPOFs is important to achieve HA

Availability objective

N O T E S

- The objective is to achieve 24x7 availability of messaging. Applications should be processing messages continuously, regardless of any failures in any component. This presentation concentrates on the MQ and MB element, but they are not the only areas to think about.
- Availability is not necessarily the same as ensuring processing of each and every message. In some situations, some limited message loss is acceptable provided that availability is maximised. For example, a message might expire or be superseded during an outage. Here, the important thing is to ensure that messages are still getting through.
- Service Level Agreements (SLAs) should define what level of availability your applications and services should provide. The level of availability is often measured by the number of 9s.
- HA solutions should increase availability given scheduled or unscheduled downtime. Scheduled downtime is more common than unscheduled. Availability issues usually involve a multitude of hardware and software systems.
- Avoid application awareness of availability solutions and aim to have little or no code in the application managing the environment. That's a task better left to systems administrators.
- The applications also need to be resilient to failures, since the messages will only flow if the applications are available to produce and consume them.

Agenda

- Introduction to HA
- **WebSphere MQ HA technologies**
- Using MQ in an HA cluster
- HA and WebSphere Message Broker
- Application considerations

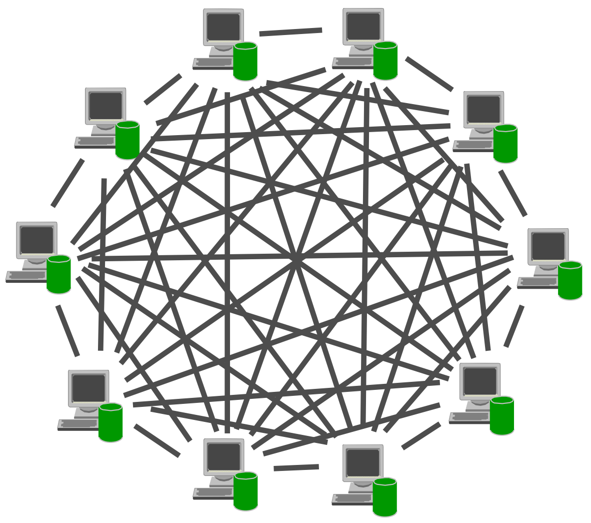
WebSphere MQ HA technologies

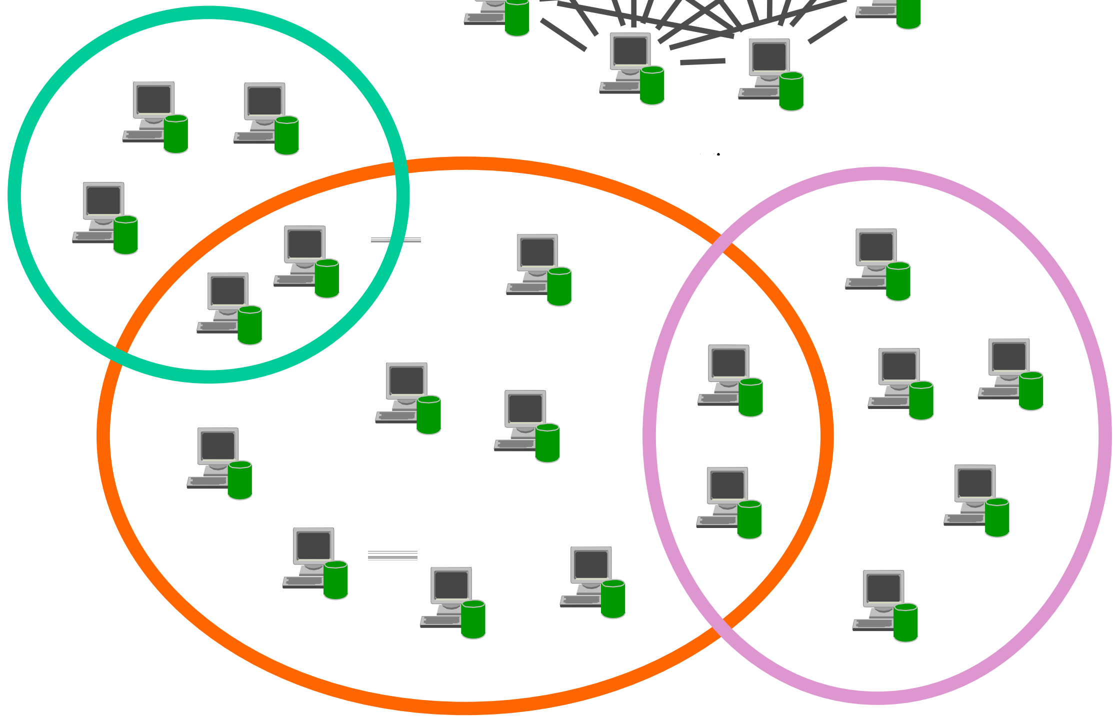
- Queue manager clusters
- Queue-sharing groups
- Support for networked storage
- Multi-instance queue managers
- HA clusters
- Client reconnection

Queue Manager Clusters

- Sharing cluster queues on multiple queue managers prevents a queue from being a SPOF

- Cluster workload algorithm automatically routes traffic away from failed queue managers





Queue Manager Clusters

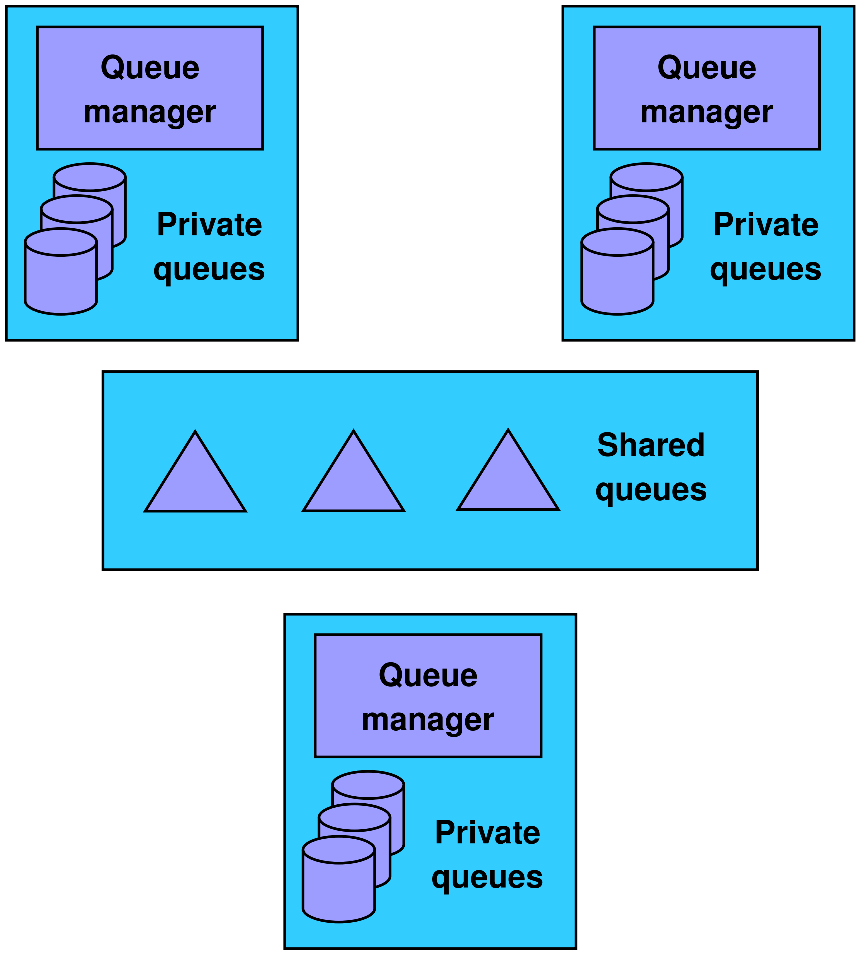
N O T E S

- Although queue manager clustering does provide some facilities useful in maintaining availability of messaging, it is primarily a workload distribution feature. It is simple to deploy extra processing power in the cluster to process more messages.

- If a queue manager in a cluster fails, the failure can be mitigated by other cluster queue managers hosting instances of the cluster queues. Messages are marooned on the failed queue manager until it restarts, but messaging through the cluster is still operational.

Queue-Sharing Groups

- On z/OS, queue managers can be members of a queue-sharing group
- Shared queues are held in a coupling facility
  - All queue managers in the QSG can access the messages
- Benefits:
  - Messages remain available even if a queue manager fails
  - Apps can connect to any QM in the queue-sharing group



Queue-Sharing Groups

N O T E S

- In the queue-sharing group environment, an application can connect to any of the queue managers within the queue-sharing group. Because all the queue managers in the queue-sharing group can access the same set of shared queues, the application does not depend on the availability of a particular queue manager; any queue manager in the queue-sharing group can service any queue. This gives greater availability if a queue manager stops because all the other queue managers in the queue-sharing group can continue processing the queue.

- To further enhance the availability of messages in a queue-sharing group, WebSphere MQ detects if another queue manager in the group disconnects from the Coupling Facility abnormally, and completes units of work for that queue manager that are still pending, where possible. This is known as peer recovery.

Introduction to Failover and MQ

- Failover is the automatic switching of availability of a service
  - For MQ, the “service” is a queue manager
- Traditionally the preserve of an HA cluster, such as HACMP
- Requires:
  - Data accessible on all servers
  - Equivalent or at least compatible servers
    - Common software levels and environment
  - Sufficient capacity to handle workload after failure
    - Workload may be rebalanced after failover requiring spare capacity
  - Startup processing of queue manager following the failure
- MQ offers two ways of configuring for failover:
  - Multi-instance queue managers
  - HA clusters

Introduction to Failover and MQ

N O T E S

- Requirement to access data
  - Networked storage for a multi-instance queue manager
  - Shared disks for an HA cluster, usually “switchable” between the servers
- Requirement for client connectivity
  - IP address takeover (IPAT) is generally a feature of failover environments
  - If a queue manager changes IP address, intelligent routers can hide this or MQ network configuration can be defined with alternative addresses
- Servers must be equivalent
  - Common software levels – or at least compatible, to allow for progressive upgrade of the servers
  - Common environments – paths, userids, security
- Sufficient capacity to handle workload
  - Often, workload will be redistributed following a failover. Often, the systems are configured for mutual takeover where the workload following failover is doubled since the surviving servers must handle the traffic intended for both.

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
  - Nonpersistent messages, nondurable subscriptions discarded

- For fastest times, ensure that queue manager restart is fast
  - No long running transactions, for example

Support for networked storage

- Queue manager data can be placed in networked storage
  - Data is available to multiple machines concurrently
  - Networked storage can be NAS or a cluster file system
    - Already have SAN support
  - Protection against concurrent starting two instances of a queue manager using the same queue manager data
  - On Windows, support for Windows network drives (SMB)
  - On Unix variants, support for Posix-compliant filesystems with leased file locking
    - NFS v4 has been tested by IBM

- Some customers have a “no local disk” policy for queue manager data
  - This is an enabler for some virtualized deployments
  - Allows simple switching of queue manager to another server following a hardware failure

Support for networked storage

N
O
T
E
S

- While not directly an HA technology, this is an enabler for customers who want to place all of the data remote from their servers such that it becomes possible to replace one server with another in the event of a failure.

- Support has been added for networked storage for queue manager data and logs. Previously, it's been supported for error and trace directories, and for installation binaries.

- On Windows, we support Windows network drives (SMB).

- On Unix platforms, we support Posix-compliant filesystems which supports lease-based file locking. The lease-based locking ensures that files unlock when the server running a queue manager fails. This rules out NFS v3 for use in an HA environment because the file locks are not released automatically for some failures and this will prevent failover.

- On Unix, we have provided a test program (amqmfsck) which checks out the filesystem's behavior. If the tests do not pass, a queue manager using the filesystem will not behave correctly. Output from this program can be used to diagnose a failure.

Multi-instance Queue Managers

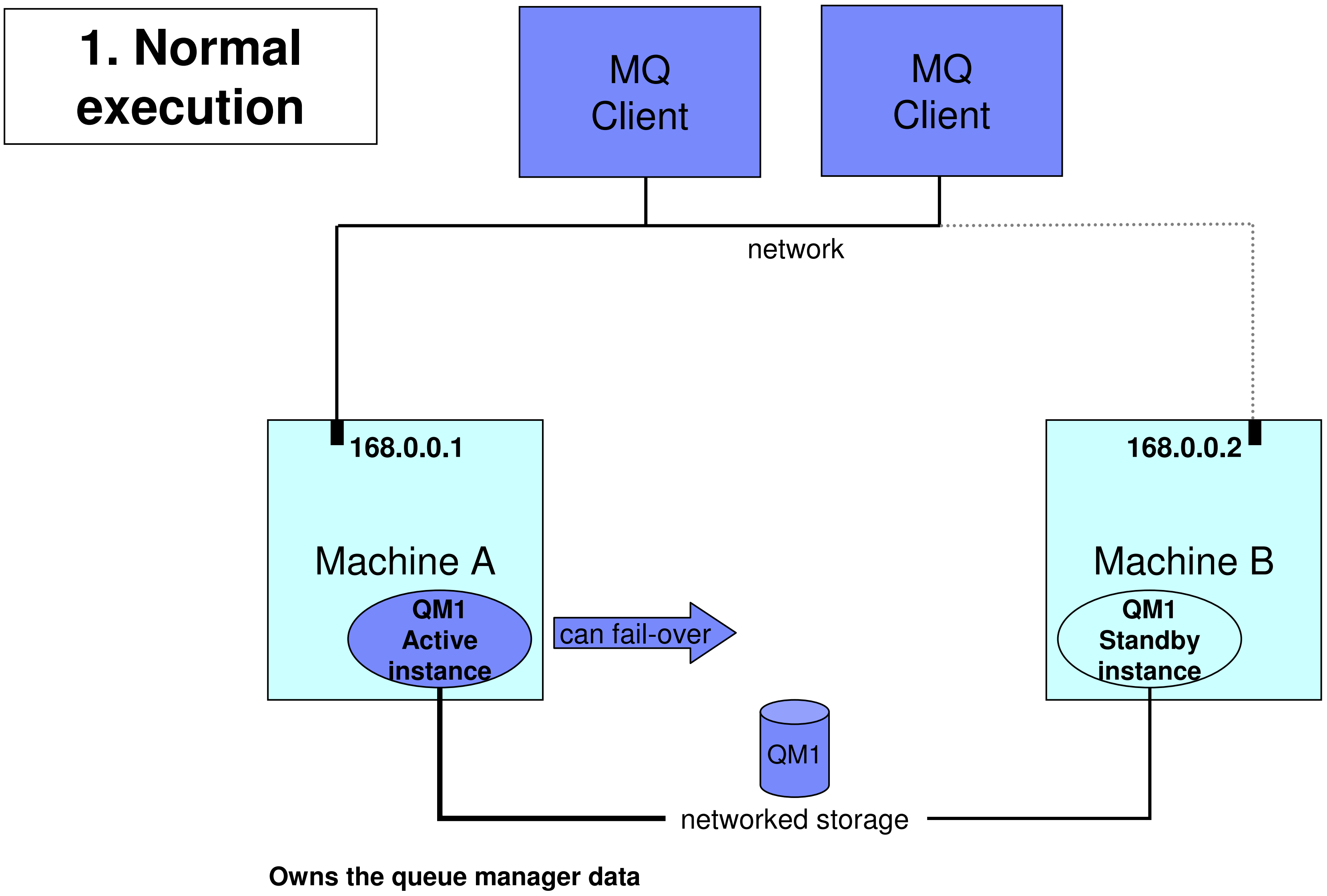
- Basic failover support without HA cluster
- Two instances of a queue manager on different machines
  - One is the “active” instance, other is the “standby” instance
  - Active instance “owns” the queue manager’s files
    - Accepts connections from applications
  - Standby instance monitors the active instance
    - Applications cannot connect to the standby instance
    - If active instance fails, standby performs queue manager restart and becomes active
- Instances are the SAME queue manager – only one set of queue manager data
  - Queue manager data is held in networked storage

Multi-instance Queue Managers

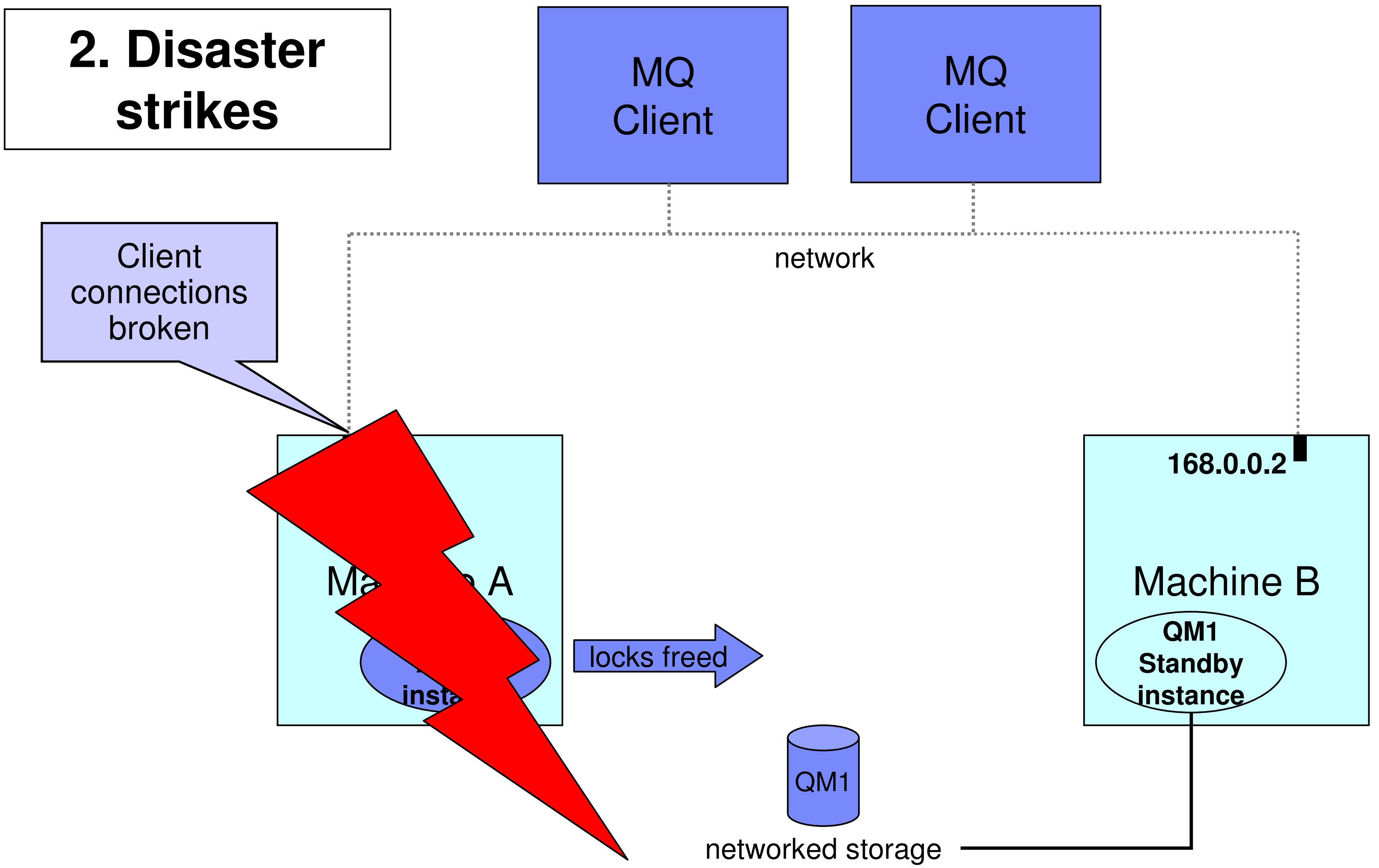
N O T E S

- "Basic failover": no coordination with other resources like disks, IP addresses, databases, user applications. There is also no sophisticated control over where the queue managers run and move to (like a 3-node HA cluster, for example). Finally, once failover has occurred, it is necessary to manually start a new standby instance.
- Architecturally, this is essentially the same as an existing HACMP/VCS setup, with the data shared between systems. It does not give anything "stronger" in terms of availability – but we do expect the typical takeover time to be significantly less. And it is much simpler to administer.
- Just as with a configuration using an HA cluster, the takeover is in essence a restart of the queue manager, so nonpersistent messages are discarded, queue manager channels go into retry, and so on.

Multi-instance Queue Managers



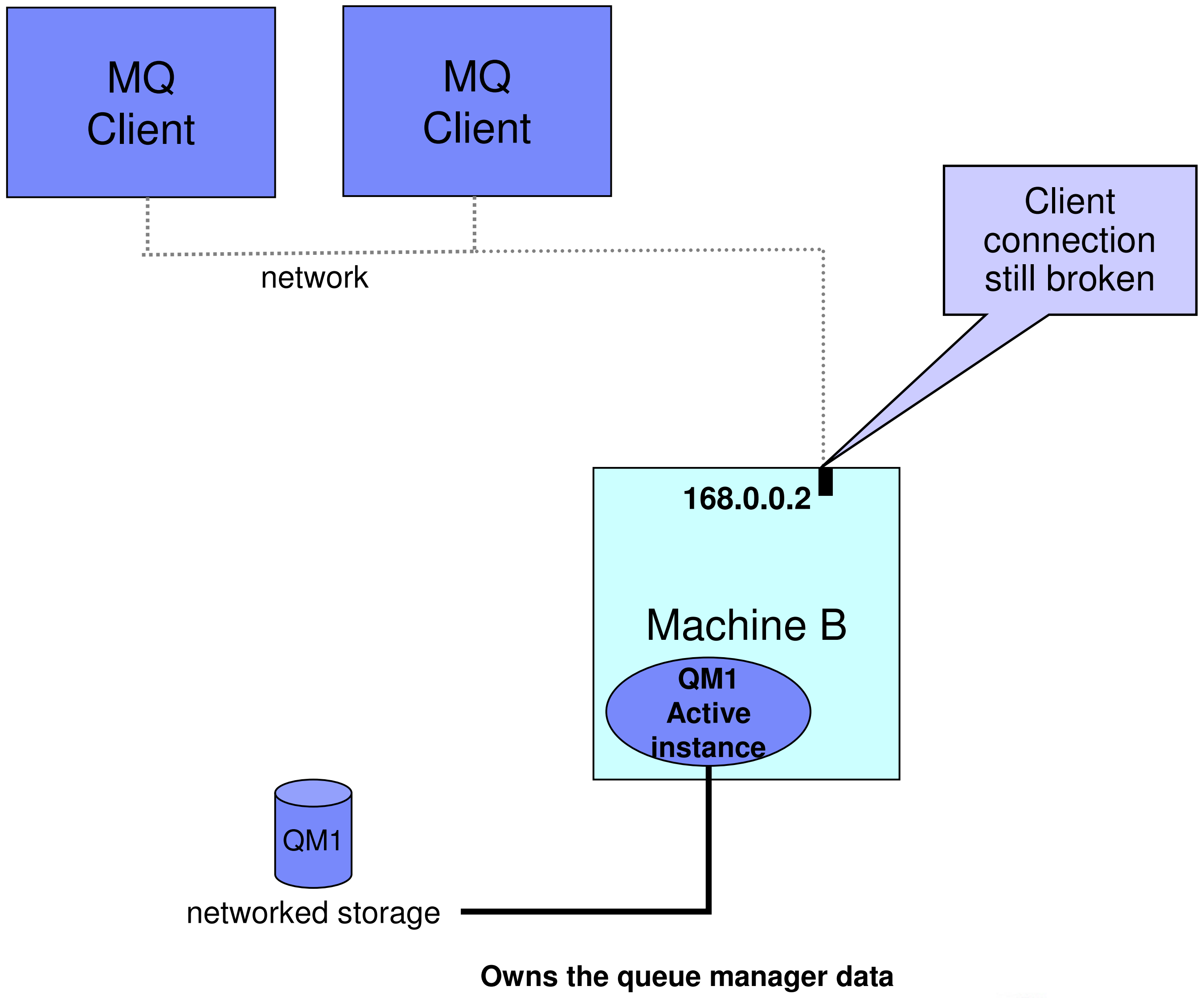
Multi-instance Queue Managers



Multi-instance Queue Managers

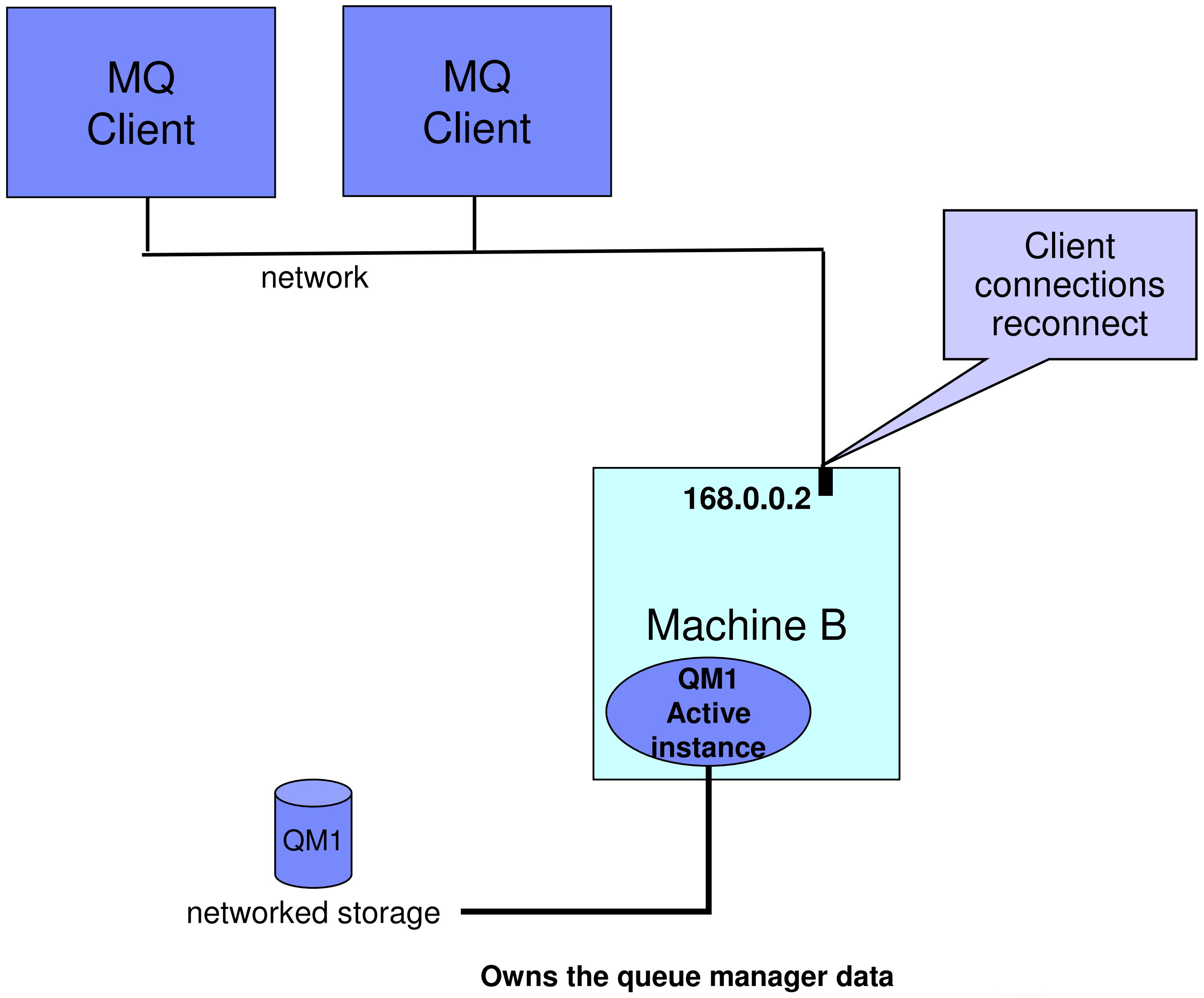
3. FAILOVER

Standby becomes active



Multi-instance Queue Managers

4. Recovery complete



Multi-instance Queue Managers

- MQ is NOT becoming an HA cluster
  - If other resources need to be coordinated, you need an HA cluster
  - WebSphere Message Broker will integrate with multi-instance QM
  - Queue manager services can be automatically started, with limited control

- System administrator is responsible for restarting another standby instance when failover has occurred

- The IP address of the queue manager changes when it moves
  - MQ channel configuration needs list of addresses unless you use external IPAT or an intelligent router
  - Connection name syntax extended to a comma-separated list
    - CONNAME(‘168.0.0.1,168.0.0.2’)
    - In MQ 7.1, extended to LOCLADDR too

New in MQ 7.1

Multi-instance and Queue Manager Clusters

- Must handle the IP address of the queue manager changing
- Method 1 – List of connection names in CLUSRCVR
  - DEFINE CHL('TO_MIQM') CHLTYPE(CLUSRCVR) CONNAME('MA,MB')
  - Only works if all other cluster members are v7.0.1 or later
- Method 2 – Blank connection names in CLUSRCVR
  - DEFINE CHL('TO_MIQM') CHLTYPE(CLUSRCVR) CONNAME(' ')
- Causes QM to re-advertise its local IP address when it restarts
  - In MQ 7.1, you can supply just a port number in brackets
    - DEFINE CHL('TO_MIQM') CHLTYPE(CLUSRCVR) CONNAME('(1415)')

New in MQ 7.1

Administering a Multi-instance Queue Manager

- All queue manager administration must be performed on the active instance
- dspmq enhanced to display instance information

```
$ hostname
staravia
$ dspmq -x
QMNAME(MIQM)          STATUS(Running as standby)
    INSTANCE(starly)        MODE(Active)
    INSTANCE(staravia)      MODE(Standby)
```

  - dspmq issued on “staravia”
  - On “staravia”, there’s a standby instance
  - The active instance is on “starly”

Agenda

- Introduction to HA
- WebSphere MQ HA technologies
- **Using MQ in an HA cluster**
- HA and WebSphere Message Broker
- Application considerations

HA clusters

- MQ traditionally made highly available using an HA cluster
  - IBM PowerHA for AIX (formerly HACMP), Veritas Cluster Server, Microsoft Cluster Server, HP Serviceguard, …
- HA clusters can:
  - Coordinate multiple resources such as application server, database
  - Consist of more than two machines
  - Failover more than once without operator intervention
  - Takeover IP address as part of failover
  - Applicable to more use-cases than multi-instance queue managers
- The disks in an HA cluster are switchable shared disks
  - Not networked storage as used by multi-instance queue managers
- Most customers using MQ and HA clusters use MC91
  - This has been withdrawn – still downloadable, but no further updates

HA clusters

- In an HA cluster, queue manager data and logs are placed on a shared disk
  - Disk is switched between machines during failover
- The queue manager has its own “service” IP address
  - IP address is switched between machines during failover
  - Queue manager’s IP address remains the same after failover
- The queue manager is defined to the HA cluster as a resource dependent on the shared disk and the IP address
  - During failover, the HA cluster will switch the disk, take over the IP address and then start the queue manager

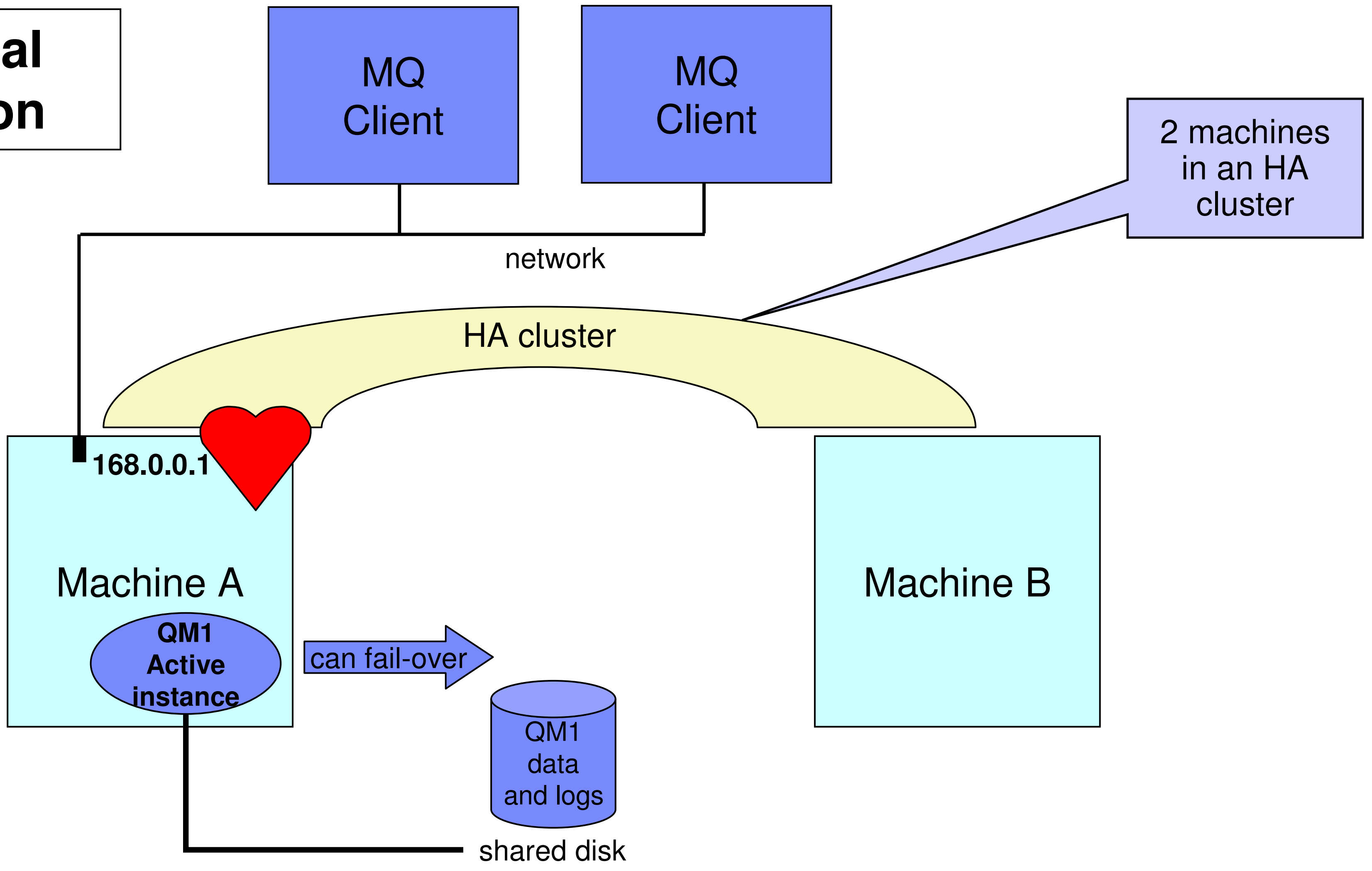
HA clusters

N O T E S

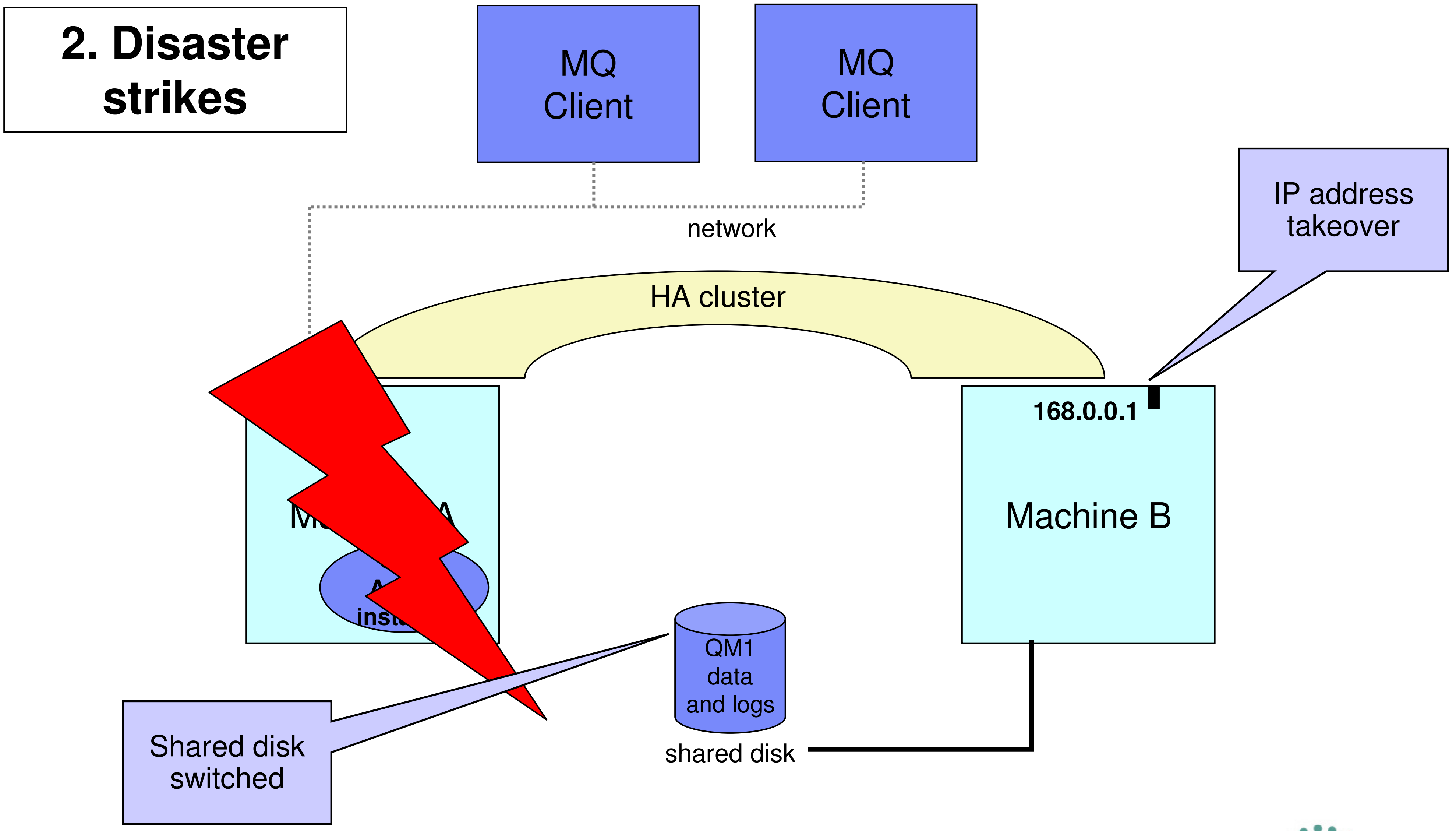
- The collection of servers that makes up a failover environment is known as a cluster. The servers are typically referred to as nodes.
- One nodes runs an application or service such as a queue manager, while the HA cluster monitors its health. The following example is called a cold standby setup because the other nodes are not running workload of their own. The standby node is ready to accept the workload being performed by the active node should it fail.
- A shared disk is a common approach to transferring state information about the application from one node to another, but is not the only solution. In most systems, the disks are not accessed concurrently by both nodes, but are accessible from either node, which take turns to "own" each disk or set of disks. In other systems the disks are concurrently visible to both (all) nodes, and lock management software is used to arbitrate read or write access.
- Alternatively, disk mirroring can be used instead of shared disk. An advantage of this is increased geographical separation, but latency limits the distance that can be achieved. But for reliability, any synchronous disk writes must also be sent down the wire before being confirmed.

MQ in an HA cluster – Cold standby

1. Normal execution

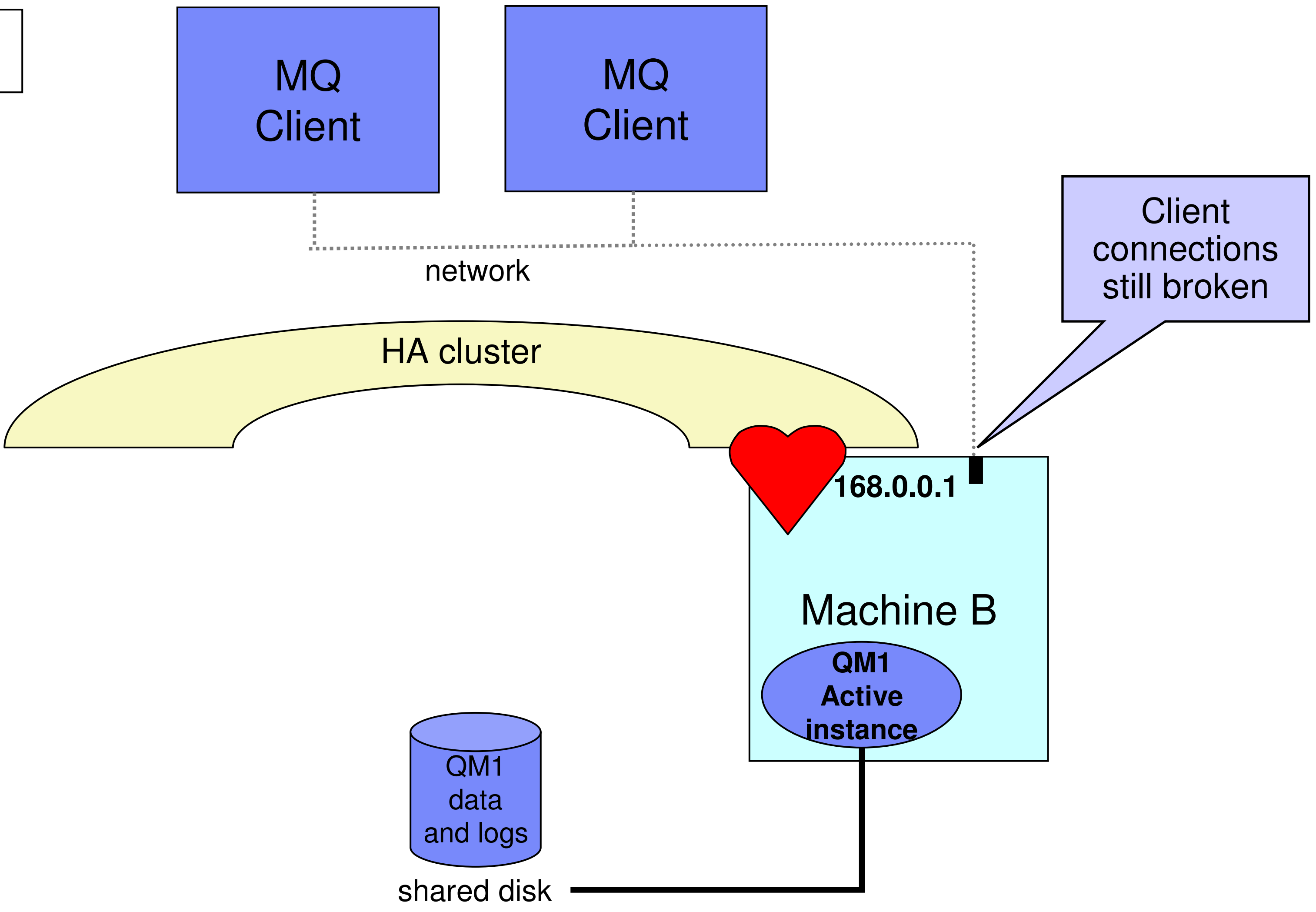


MQ in an HA cluster – Cold standby



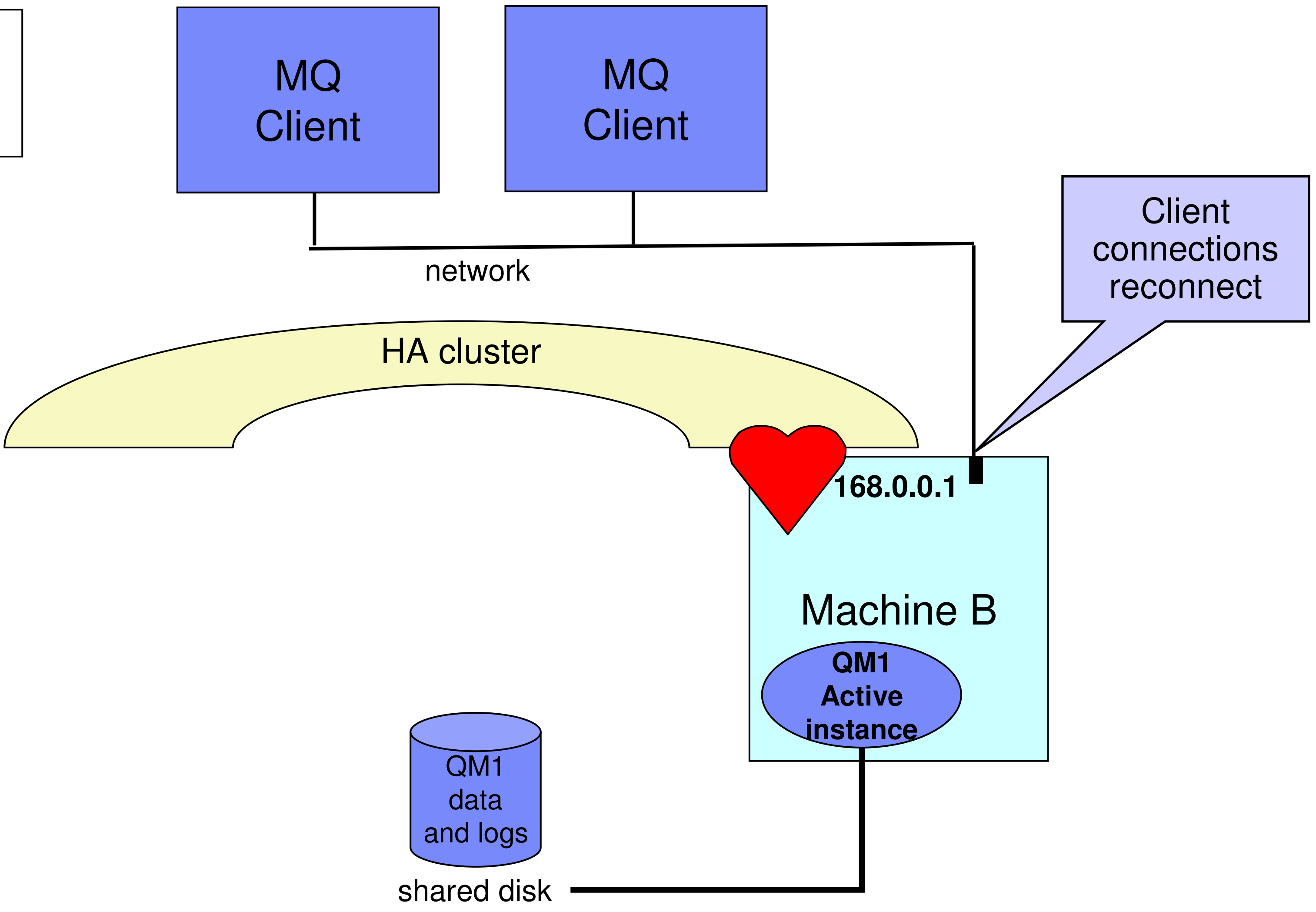
MQ in an HA cluster – Cold standby

3. FAILOVER



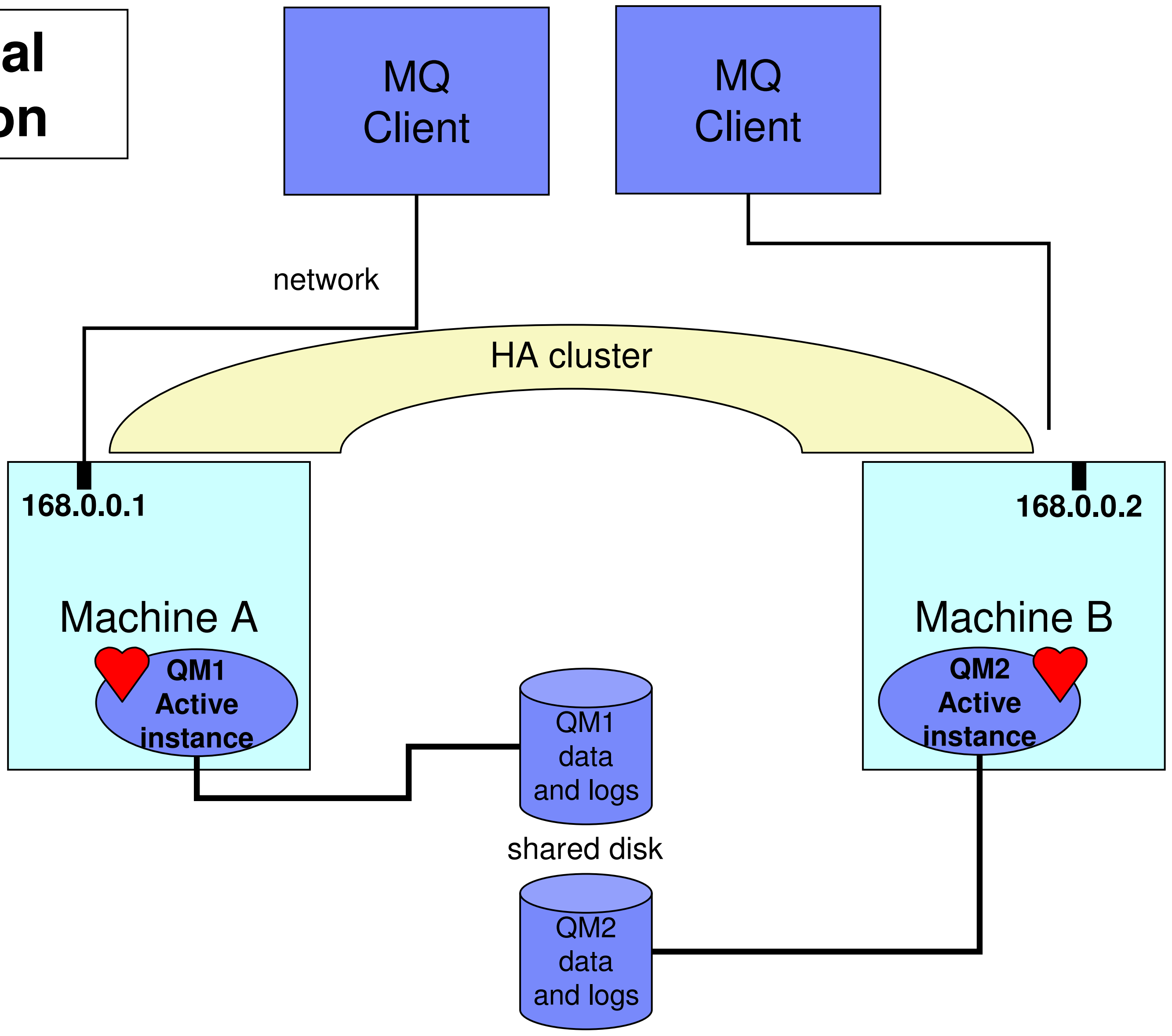
MQ in an HA cluster – Cold standby

4. Recovery complete

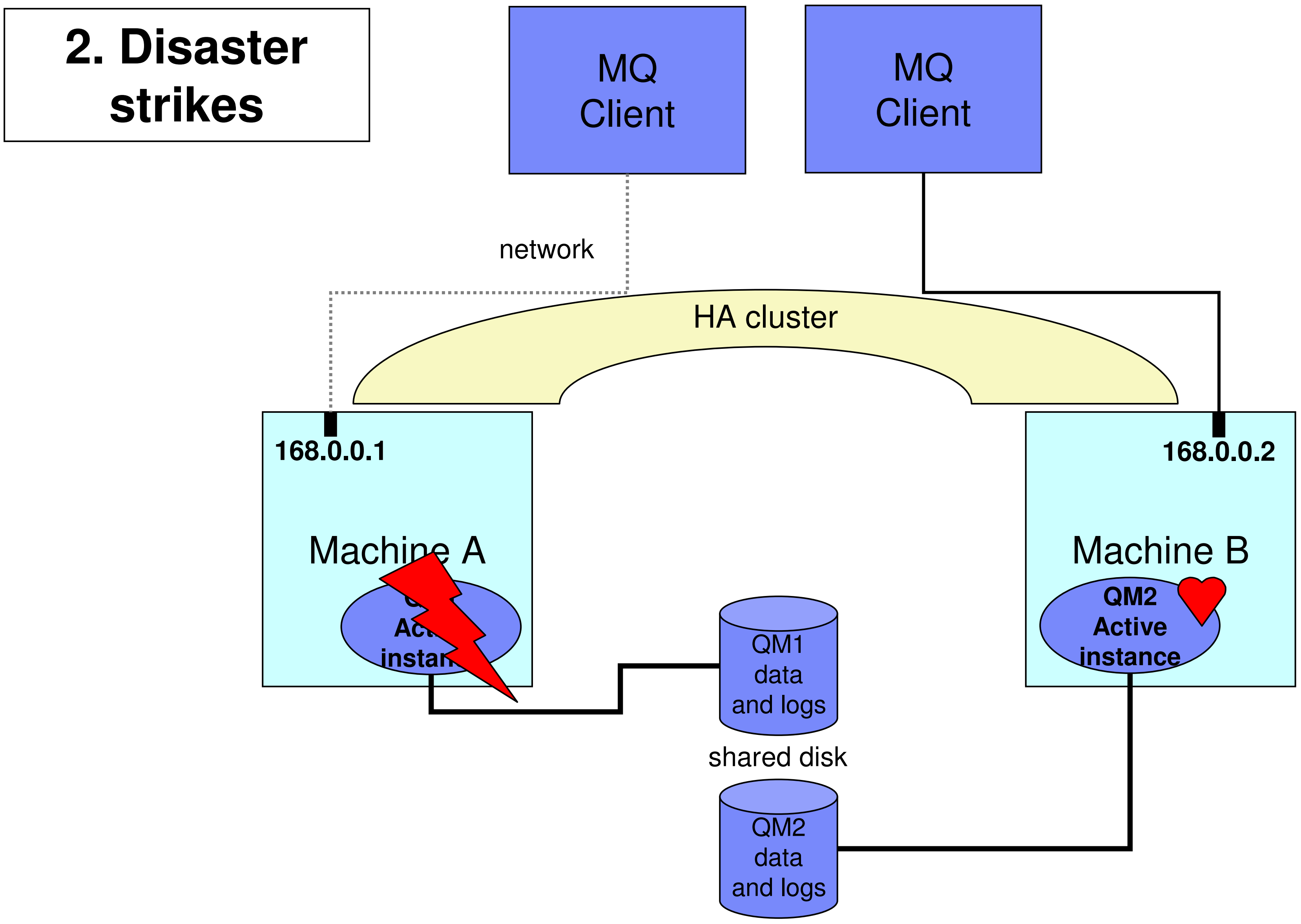


MQ in an HA cluster – Active/active

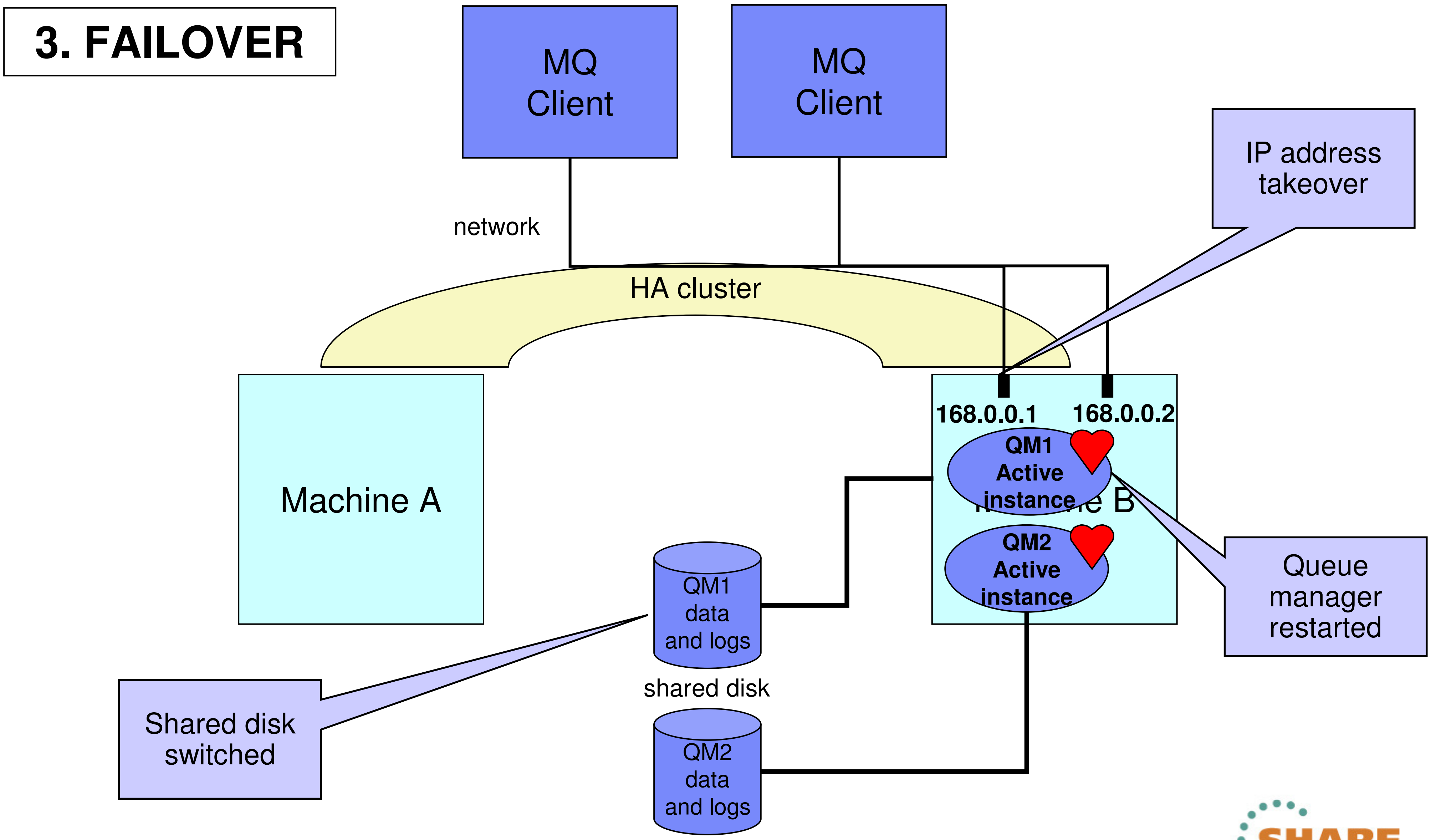
1. Normal execution



MQ in an HA cluster – Active/active



MQ in an HA cluster – Active/active



MQ in an HA cluster – Active/active

N O T E S

- This configuration is also sometimes called a "mutual takeover" system
- In normal operation, both machines are running independent queue managers. If one of the systems fails, then this configuration can migrate the failed queue manager to the working machine. So it still appears to applications outside the cluster that you have 2 queue managers. The throughput of each queue manager may degrade (depending on how heavily loaded they run) but at least the work is still getting done.
- With this kind of setup, you probably have a failback capability so that the queue manager can be sent back to its original node when the failure has been corrected. Whether the failback is automatic or not may be your choice, but I'd strongly recommend that it's done manually so that applications which have already connected to the running queue manager do not have their connections broken arbitrarily. You probably want to monitor the workload and only failback when there's not too much work that's going to be disrupted by the failback.
- A number of variations on the themes of cold and hot standby are also possible, for example having 3 nodes to host 2 queue managers (an "N+1" configuration). The availability of these options will depend on the facilities available in your cluster software.
- In this configuration, we've shown the IP address associated with each queue manager being migrated. You will also need to keep a port number reserved for each queue manager (the same number on both machines in the cluster), and have appropriate setup for runmqlsr or a queue-manager listener object.

Creating a QM in an HA cluster

- Create filesystems on the shared disk, for example
  - /MQHA/QM1/data for the queue manager data
  - /MQHA/QM1/log for the queue manager logs

On one of the nodes:

- Mount the filesystems
- Create the queue manager
  - `crtmqm –md /MQHA/QM1/data –ld /MQHA/QM1/log QM1`
- Print out the configuration information for use on the other nodes
  - `dspmqinf –o command QM1`

On the other nodes:

- Mount the filesystems
- Add the queue manager's configuration information
  - `addmqinf –s QueueManager –v Name=QM1 –v Prefix=/var/mqm –v DataPath=/MQHA/QM1/data/QM1 –v Directory=QM1`

Using MQ in an HA cluster

NOTES

- The list of queue managers on a machine is stored in /var/mqm/mqs.ini. Each queue manager is described by a QueueManager stanza which lists its name and where its data lives.
  - crtmqm creates the data for a queue manager and adds it to mqs.ini
  - addmqinf simply adds a queue manager to mqs.ini

- Some of the files that MQ uses must be local to each node. These files are used to derive keys using the ftok() function call, and are based on inode numbers in filesystems. If the inodes are on different filesystems, then their numbers might be the same and key clashes may occur. A good way to ensure all keys are on the same filesystem is to leave the Prefix attribute of all of the queue managers with its default value of /var/mqm. This will result in all of these key files being stored underneath /var/mqm/sockets on the node's local disk. If it is necessary to override this default, the MQSPREFIX environment variable can be used.

- A queue manager will not restart if it believes there are still-attached processes. You might need to change the MC91 example start scripts to kill any of your application processes which might be running. Many customers use client applications in HA configurations which eliminates the possibility of an application process preventing queue manager restart. Another alternative is isolated bindings for local applications.

- Most queue managers will require a listener. A common way of controlling this in an HA cluster is to put the listener itself under control of the HA cluster as a separate resource dependent on the queue manager. An alternative is to have the queue manager automatically start the listener, but a running queue manager will not restart a listener automatically if it fails.

MC91 SupportPac

- Scripts for IBM PowerHA for AIX, Veritas Cluster Server and HP Serviceguard
  - The scripts are easily adaptable for other HA cluster products
- Scripts provided include:
  - hacrtmqm – Create queue manager
  - hadltmqm – Delete queue manager
  - halinkmqm – Link queue manager to additional nodes
  - hamqm_start – Start queue manager
  - hamqm_stop – Stop queue manager
  - hamigmqm – Used when migrating from V5.3 to V6

Equivalents to MC91 facilities

MC91	Using MQ 7.0.1 and later
hacrtmqm to create queue manager on shared disk and point symbolic links back to node’s /var/mqm	New crtmqm –md option
halinkmqm	New addmqinf command
hadltmqm	New rmvmqinf command to remove queue manager from a node, dltmqm to delete the queue manager
hamqm_start	Use the MC91 hamqm_start
hamqm_stop	Use the MC91 hamqm_stop
rc.local script	Part of MC91 hamqm_start
hamqm_applmon	Use the MC91 hamqm_applmon, or a script more tailored to your needs

What does the HA cluster support add?

- Queue manager start and stop scripts are more resilient than vanilla strmqm/endmqm
  - For example, endmqm could get stuck in extreme cases
- Monitoring script for health-checking of queue manager by HA cluster
  - Uses runmqsc `PING QMGR`
  - A new alternative is `dspmq –n <qmname> | grep "RUNNING"`

Multi-instance QM or HA cluster?

- Multi-instance queue manager
  - ✓ Integrated into the WebSphere MQ product
  - ✓ Cheaper – special "idle-standby" licensing terms
  - ✓ Faster failover than HA cluster and MC91
    - Delay before queue manager restart is much shorter
  - ✗ Runtime performance of networked storage
  - ✗ Less able to react to an unresponsive queue manager

- HA cluster
  - ✓ Capable of handling a wider range of failures
  - ✓ Failover historically rather slow, but some HA clusters are improving
  - ✗ Some customers frustrated by unnecessary failovers
  - ✗ Require MC91 SupportPac or equivalent configuration
  - ✗ Extra product purchase and skills required

Comparison of Technologies

	Access to existing messages	Access for new messages
Shared Queues, HP NonStop Server	continuous	continuous
MQ Clusters	none	continuous
	automatic	continuous
HA Clustering, Multi-instance	automatic	automatic
No special support	none	none

Comparison of Technologies

N
O
T
E
S

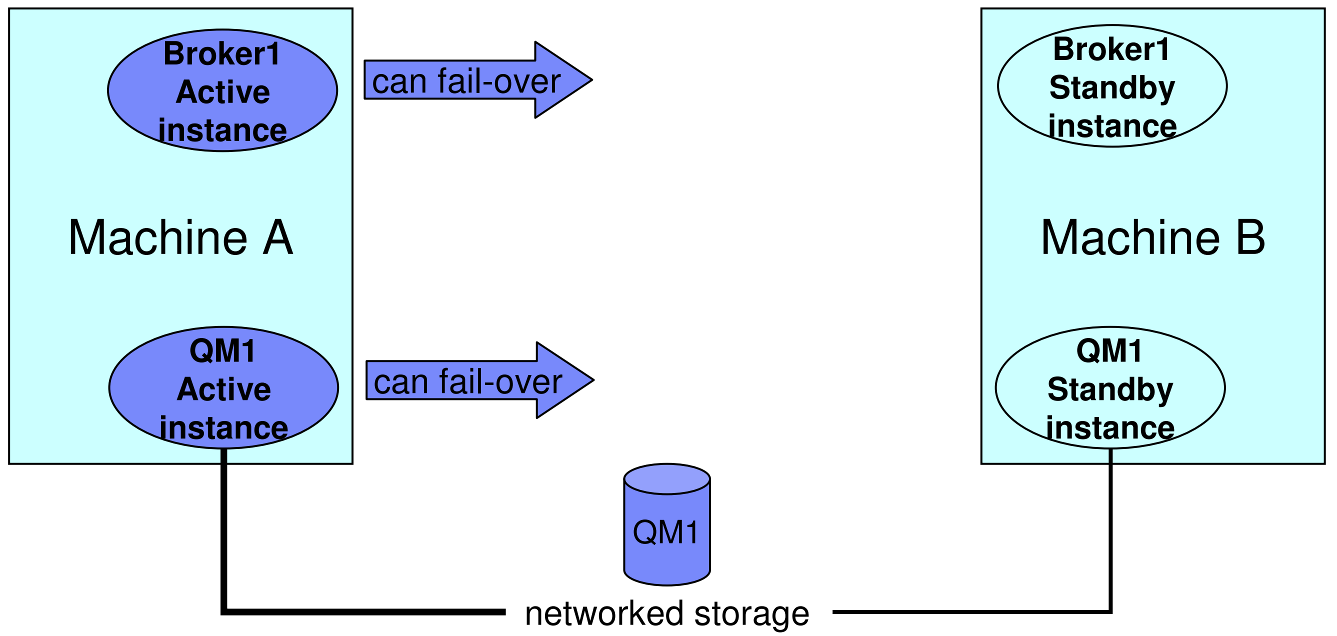
- This picture shows one view of the different capabilities. However you also need to consider factors such as:
  - total hardware/software price
  - the requirement for nonpersistent message availability (remember that they are discarded by a failover-restart)
  - the requirement for persistent message availability (not if you're using the shared queue support)

Agenda

- Introduction to HA
- WebSphere MQ HA technologies
- Using MQ in an HA cluster
- **HA and WebSphere Message Broker**
- Application considerations

WebSphere Message Broker – Multi-instance brokers

- WebSphere Message Broker v7 is designed to work well with multi-instance queue managers
  - Standby broker instance can be started on a standby QM instance
  - An alternative is to make the broker a queue-manager service



Owns the queue manager data

WebSphere Message Broker – HA Clusters

- Unit of failover is a broker
  - Broker depends on a queue manager
  - May also be a user database, but no broker database any more (v7)

- Lots of configurations available, because of database topologies
  - SupportPacs list common/recommended setups

- SupportPac IC91 available (still at MB v6 currently)
  - Support for Windows MSCS, AIX PowerHA for AIX, Veritas Cluster Server, HP Serviceguard, Linux HA
  - They prereq the equivalent MQ and DB2 packages
    - MC91
    - DB2 ships the support on the product CDs
  - There are scripts to make the Config Manager highly available if you require

Agenda

- Introduction to HA
- WebSphere MQ HA technologies
- Using MQ in an HA cluster
- HA and WebSphere Message Broker
- **Application considerations**

How to make your own applications HA

- Application environment
  - What does the application depend on
- Application design
  - Affinities implicit in the application design
  - Message loss
- MQ connectivity
  - What happens when the application loses connectivity

HA applications – Application environment

- Simple applications only need a queue manager connection
- Many business applications have dependencies, such as:
  - Database instance, message broker, application server
  - Configuration information
  - Some data is machine-specific, other data is server-specific
  - Get the ordering of dependencies and timing correct
- How can you tell if it's working
  - Such as PING QMGR
  - Remember that restart might take a little while
- Start/stop operations need to be robust
  - Don't rely on anything!
  - Remember that a 'stop' command might erroneously block
- If you want to put your app in an HA cluster, you'll need to answer these

HA applications – Message loss

- Does your application really need every message delivered?
  - If so, it's quite fragile to failures
  - Queue manager restart will lose nonpersistent messages
  - Message expiry discards old messages
  - Typically, disaster recovery (DR) situations involve message loss

- By careful design of the messages and applications, it is often possible to keep messaging even without failover
  - Some customers use workload balancing and application redundancy instead

HA applications – Application affinities

- Affinities reduce availability
- Affinities can be introduced by:
  - The need to continue using the same instance of a service
  - Multiple messages sent to the same application process
  - Conversational-style applications
- Carry any transient state in the message
  - And replicate frequently-read data
- Let other components handle partitioning or sharing of data
  - e.g. store state in a parallel database
- MQ clusters can handle application affinity requirements
  - Use BIND_ON_OPEN option
  - Maybe create a workload exit to remember previous messages

HA applications – MQ connectivity

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

Automatic client reconnection

- MQ client automatically reconnects when connection broken
  - MQI C clients and JMS clients
- Reconnection includes reopening queues, remaking subscriptions
  - All MQI handles keep their original values
- Can connect back to the 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
  - Nonpersistent 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

Automatic client reconnection

- Enabled in application code or ini file
  - MQI: MQCNO_RECONNECT, MQCNO_RECONNECT_Q_MGR
  - JMS: Connection factories/activation specification properties
  - In MQ 7.1, a new DEFRECON channel attribute lets you control reconnection administratively

New in MQ 7.1

- Plenty of opportunity for configuration
  - Reconnection timeout
  - Frequency of reconnection attempts

- Requires:
  - Threaded client
  - At least 7.0.1 server
  - Full-duplex client communications (SHARECNV >= 1)

Summary

- MQ and operating system products provide lots of options to assist with availability
  - Many interact and can work well in conjunction with one another

- But it's the whole stack which is important ...
  - Think of your application designs
  - Ensure your application works in these environments

- Decide which failures you need to protect against
  - And the potential effects of those failures

- Also look for RedBooks and read the MQ HA whitepaper
  - www.ibm.com/developerworks/websphere/library/techarticles/0505_hiscock/0505_hiscock.html

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00					Free MQ! - MQ Clients and what you can do with them
09:30	Clustering – the easier way to connect your Queue Managers	MQ on z/OS – vivisection	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation		
11:00		Diagnosing problems for Message Broker	Lock it down - WebSphere MQ Security	Using IBM WebSphere Application Server and IBM WebSphere MQ Together	Spreading the message – MQ pubsub
12:15	Highly Available Messaging - Rock solid MQ	Putting the web into WebSphere MQ: A look at Web 2.0 technologies	The Doctor is In and Lots of Help with the MQ family - Hands-on Lab		
01:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in the WebSphere MQ Product Family	Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud	MQ Performance and Tuning on distributed including internals	
03:00	First steps with WebSphere Message Broker: Application integration for the messy	What's new in Message Broker V8.0	Under the hood of Message Broker on z/OS - WLM, SMF and more	The Do's and Don'ts of z/OS Queue Manager Performance	
04:30	The MQ API for Dummies - the Basics	What the **** is going on in my Queue Manager!?	Diagnosing problems for MQ	Shared Q using Shared Message Data Sets	
06:00			For your eyes only - WebSphere MQ Advanced Message Security	MQ Q-Box - Open Microphone to ask the experts questions	

#SHAREorg

SHARE
Technology · Connections · Results

Any questions?

Please fill in evaluations at
share.org/AnaheimEval
Session #11511

SHARE in Anaheim 2012

Copyright Information