IBM Software Group

WebSphere MQ 7.0 Capabilities

e business on demand software

WebSphere MQ

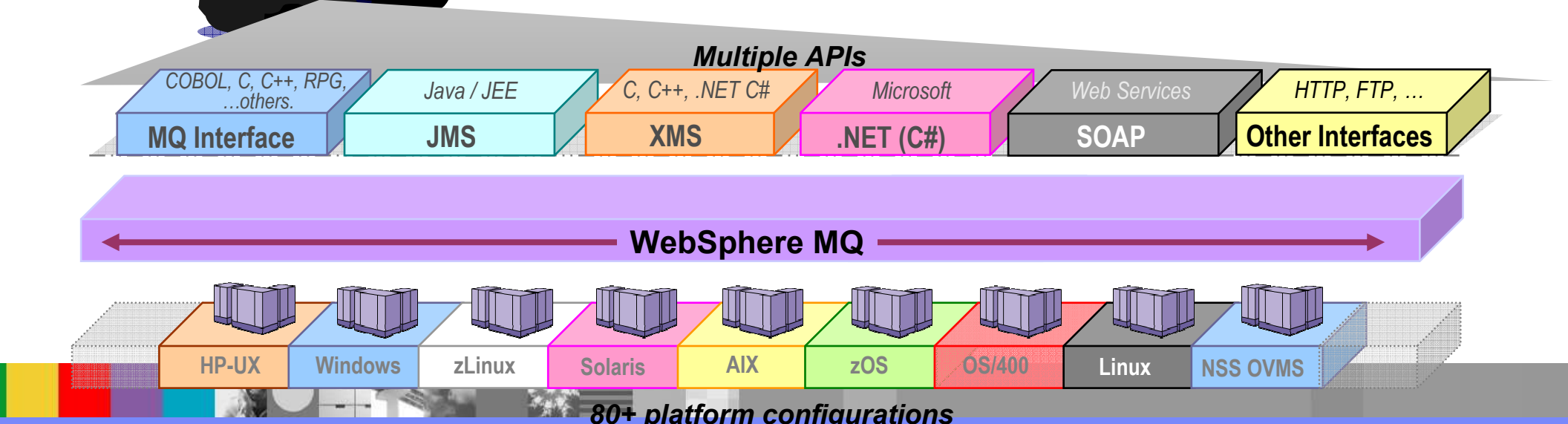
- Makes it easy to connect applications and systems
- Once-only assured delivery of data
- Many environments
  - Broad range of operating system and hardware platforms
  - Supports range of programming languages
  - Communications Protocols
  - Point-to-point and publish-subscribe styles
  - ... all available through simple APIs
- The industry standard for industrial messaging
- Recently celebrated 10th anniversary!

WebSphere MQ: The Universal Messaging Backbone

WebSphere MQ can dramatically reduce application infrastructure costs by providing a single manageable distributed infrastructure for all application messaging traffic.

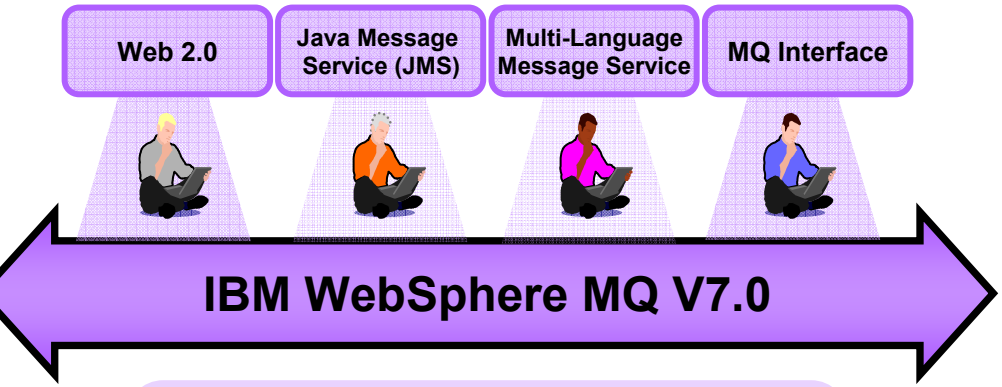
Features:

- *Supports the broadest range of APIs, programming languages and OS platforms*
- *Provides the only JMS engine that can be implemented on "any" standards-compliant JEE server*
- *Provides rich web services interfaces meeting customer needs for WS-Reliability*
- *Offers a broad range of qualities of service and messaging methods including publish/subscribe*
- *Provides Low Latency and Extended Security editions*
- *Offers the most scalable, most manageable messaging system available*
- *Assures transactional message delivery end-to-end.*

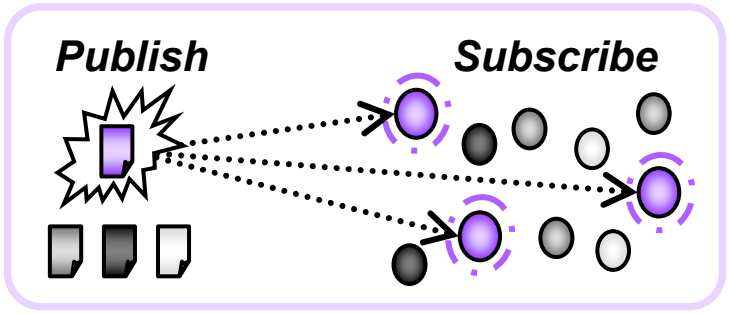


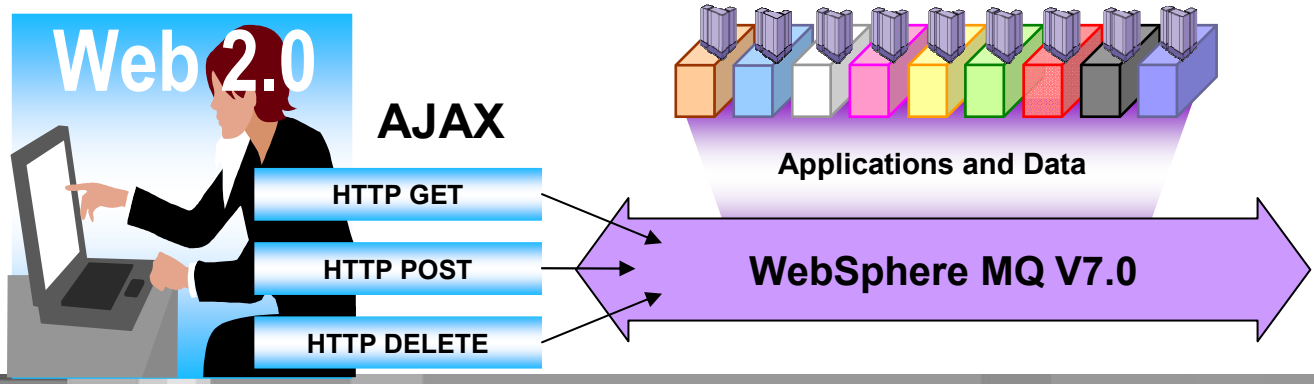
80+ platform configurations

IBM WebSphere MQ V7.0



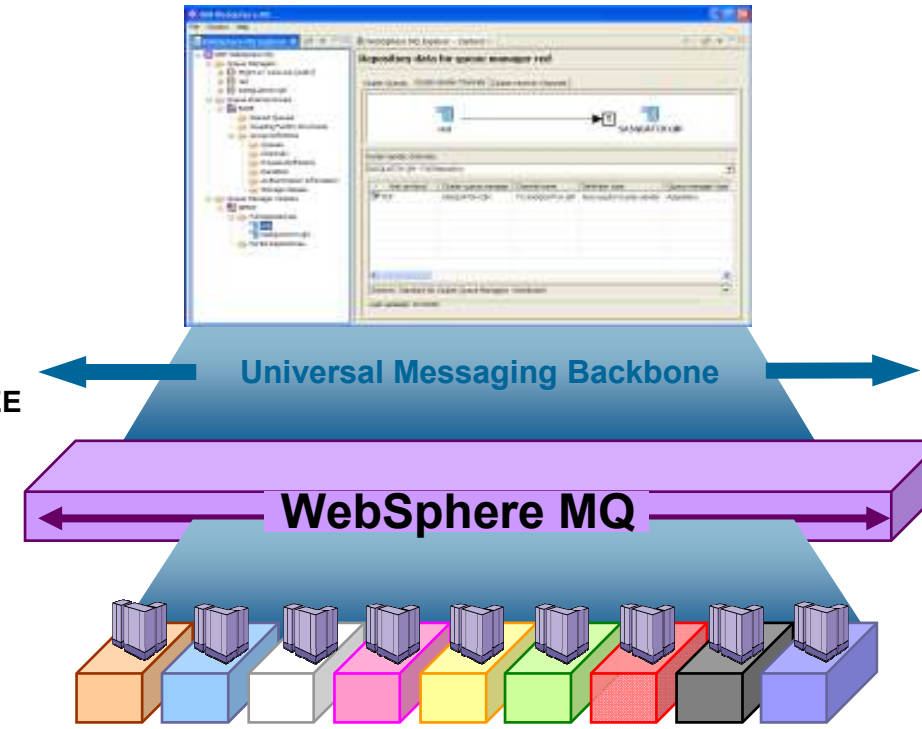
Universal Messaging Backbone for SOA and Web 2.0





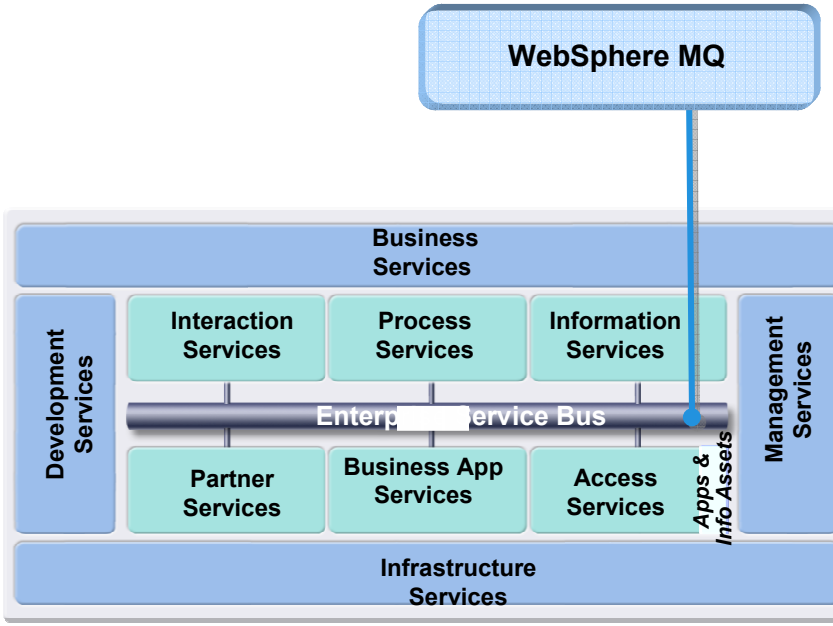
WebSphere MQ

- **Delivers the reliable Universal Messaging Backbone**
  - Industry standard JMS (Java Message Service) messaging
  - Flexible Publish/Subscribe messaging
  - Reliable SOAP transport for Web services
- **Offers a range of Qualities of Service**
  - Guaranteed to semi-persistent delivery
- **Connects virtually any commerical IT system**
  - Broadest platform support in the market including native z/OS
  - Connects your most valuable application assets including **CICS, IMS, DB2 and JEE Application Servers**
- **Easy to manage**
  - Configure entire MQ network remotely and securely with interactive and visual Eclipse-based Explorer tool
- **Shields developers from networking complexities**
  - Provides a consistent, rich MQI interface and JMS API
  - Provides a choice of APIs for Java, C, C++,C#, COBOL developers
- **Extends the reach of IBM's ESB offerings**
  - Providing a ubiquitous transport layer



WebSphere MQ V7.0 Features

- **Provides New and Enhanced Publish and Subscribe messaging**
  - New support for Publish and Subscribe for z/OS
  - Increased performance for distributed platforms by up to 60%*
  - Enables flexible message distribution based on matching topics or keywords
- **Enhances Java Message Service (JMS) support**
  - Increases performance of selectors by up to 250%*
  - Increases listener throughput by up to 45% and reduces latency*
- **Enhances Ease-of-Use**
  - Enables graphical configuration of JMS and Publish and Subscribe via Eclipse-based Explorer tooling
  - Common admin and security for Publish and Subscribe
- **Provides new verbs and behaviors for MQ Interface**
  - Helps improve developer productivity
- **Enhances WebSphere MQ clients with new Qualities-of-Service**
  - Increases throughput of non-persistent messaging by up to 300%*
  - Increases resilience and availability
- **Provides Web 2.0 support to help create richer user experience**
  - Enables Web 2.0 developers to connect to core applications using Ajax skills and RESTful approaches
  - Bridges HTTP applications with Ajax and REST to the MQ messaging backbone



* Distributed platforms

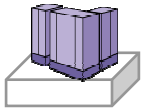
MQ goes Web 2.0 – The WebSphere MQ Bridge for HTTP

A simple way of connecting to WebSphere MQ from HTTP – the transport used today by most Web applications

ynamic Web applications
Speeds and eases integration of new Web apps with enterprise applications and data
Deliver richer content to Web users
No MQ skills needed

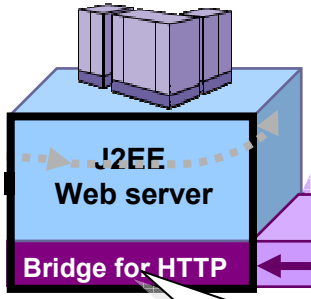
No MQ client footprint
- Simplifies deployment and maintenance of large scale distributed applications

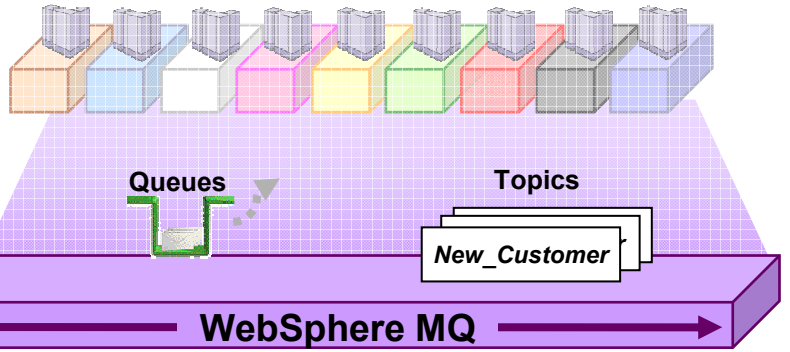
"Zero client" applications



Web 2.0

HTTP



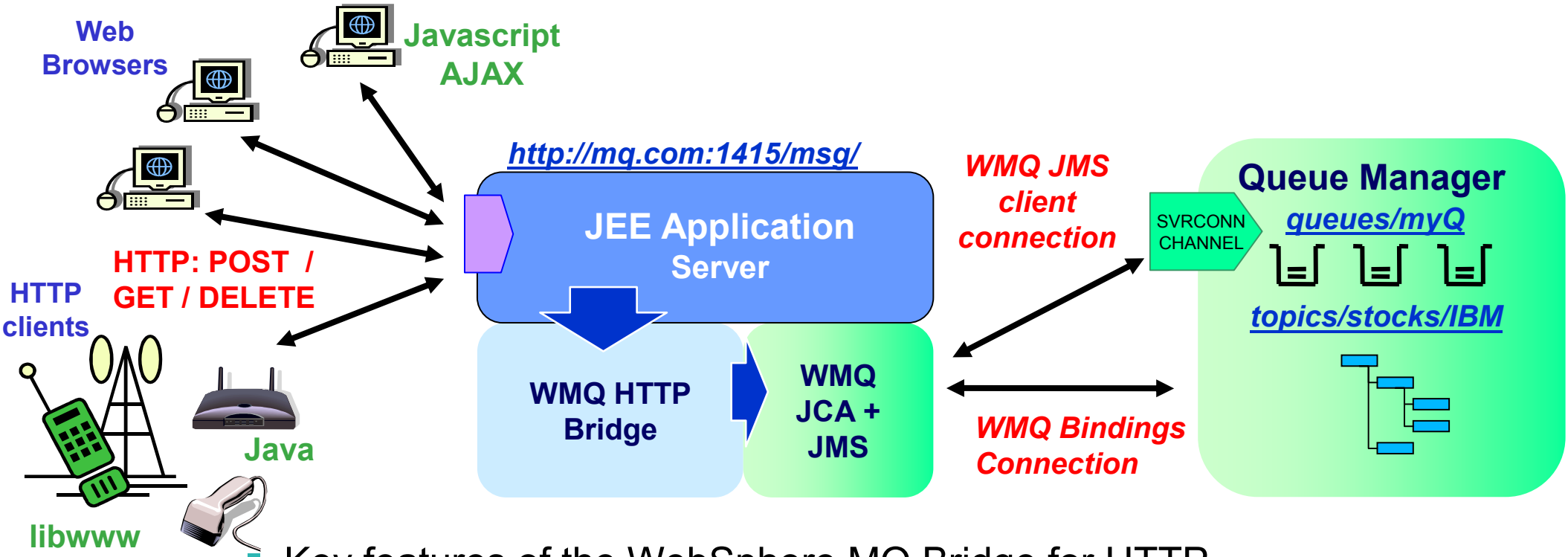


Easy access to enterprise applications and ESB
Reliable delivery of SOAP across MQ backbone
Pub/Sub distribution
Throttling of service requests via Queuing

REST-based Web services

- **Bridge for HTTP** runs in a J2EE App Server
- Maps HTTP traffic to MQ queues and topics

WebSphere MQ Bridge for HTTP - Architectural Overview



- Key features of the WebSphere MQ Bridge for HTTP -
  - Maps URIs to queues and topics
  - Enables MQPUT and MQGET from
    - Web Browser
    - Lightweight client
- Alternative non-servlet implementation available as MA94

More in 2256

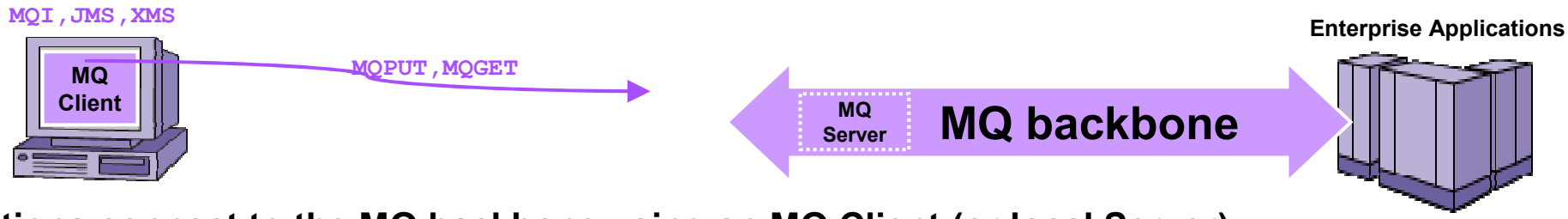
HTTP-MQI Verb / Resource Mapping

- Define URI to identify queue (or topic)
- Modelled on REST principles
  - Simple translation of HTTP to MQI
- Message Format:
  - Header fields (MQMD) conveyed in HTTP headers
  - Body is passed in HTTP entity body
  - Message type is conveyed in HTTP Content-Type
    - "text/plain" or "text/html" equate to WMQ string messages (MQFMT_STRING)
    - All other media types map to WMQ binary messages (MQFMT_NONE)

		HTTP verb mapping			
Resource	**Sample URIs**	**GET**	**POST**	**PUT**	**DELETE**
Messages	**http://host/msg/queue/*qname/*** **http://host/msg/topic/*topic_path/***	**MQGET w. browse**	**MQPUT**	**-**	**MQGET**

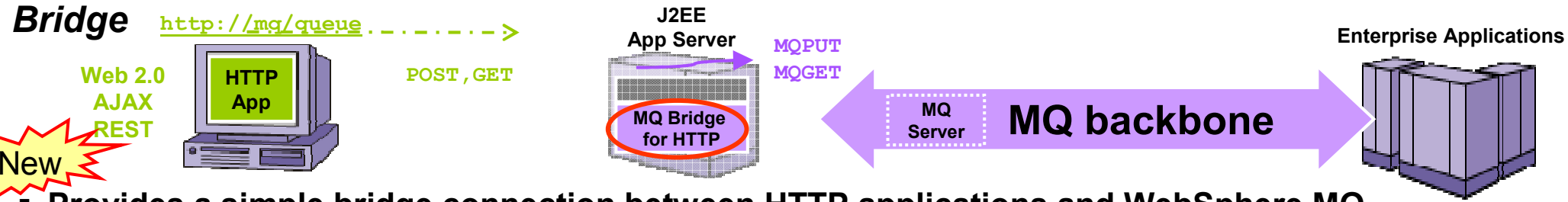
The MQ Client positioned with WebSphere MQ Bridge for HTTP

Client



- **Applications connect to the MQ backbone using an MQ Client (or local Server)**
- **MQ Clients are free BUT their deployment can create an inhibitor to MQ adoption**
  - *Clients need to be installed and configured – a potential provisioning challenge for large numbers*
  - *They have a small footprint (Approx. 3-10 MB) – but this could still pose problems for some applications / users*

Bridge



- **Provides a simple bridge connection between HTTP applications and WebSphere MQ**
  - *Maps messages from HTTP URLs to MQ Queues and MQ Topics (provides Pub/Sub)*
- Zero footprint on client - requires a J2EE Application Server – Connects via JCA
  - *Works with <u>any</u> J2EE Application Server with JCA support including WAS Community Edition*

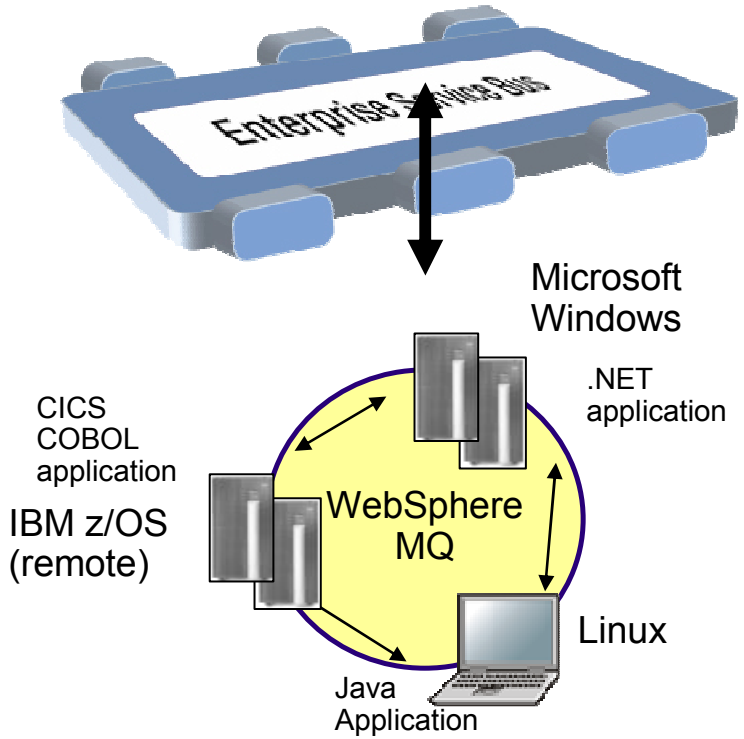
Key Benefits of the MQ Bridge

- **Zero client solution for connecting to an MQ network** – Alternative to MQ client for simple access
- **Easy way for Web developers** that create Web 2.0, AJAX, REST Web services to connect to MQ
  - e.g. Web pages can dynamically update themselves on-the-fly using data from the MQ backbone

The 12 Advantages of WebSphere MQ over other JMS Providers

1. **Built in transaction coordinator (a unique feature among all messaging vendors)**
2. **Multi-CPU and multi-system scaling**
3. **Very high speed performance between queue manager servers.**
4. **Dynamic publish / subscribe routing across servers.**
5. **Queue-level Clustering**
6. **MQ Explorer administration**
7. **Rich API support – MQI, JMS, XMS, .NET, Web Services; 80+ configurations**
8. **Managed File Transfer Extension (PM4Data)**
9. **Extended Security Extension (MQESE)**
10. **Eclipse based tooling**
11. **SOAP over a reliable messaging backbone.**
12. **Built-in transactional connectivity to WAS v6, WESB v6, and WPS v6.**

Application Connectivity Solution



More detail in Backup

Improved Availability for Distributed Platforms

- Log file information provided to permit Disaster Recovery solutions to be built
- Queue manager reports on required, in-use, no-longer-needed log files
  - When you are using linear logging
- Command to force advance of log to new log file
  - Can now get a consistent copy or backup of log files at a specific time
- Ability to replay log files on a DR site, on a “backup” queue manager before any disaster requires it to be done
  - Reduces time needed for restart in a live situation
- Also increased the amount of active log space that can be configured

More status

- Problem determination (and problem avoidance) indicators
  - Show "instantaneous" information about activity
- DISPLAY CHSTATUS
  - Cluster transmission queue subdepth and average throughput times
  - Average batch size
  - "Substate" - sending data on wire, waiting for nameserver response, in an exit
  - Response time from remote end
- DISPLAY QSTATUS
  - Average time messages lived on queue (short & long-term averages)
  - Last GET/PUT times
  - Oldest message still on queue

```
DISPLAY QSTATUS(TESTQ)
   AMQ8450: Display queue status details
   QUEUE(TESTQ)                IPPROCS(1)           OPPROCS(1)
   CURDEPTH(24)                UNCOM(NO)
   LPUTDATE(2004-04-16)        LPUTTIME(18:09:16)
   MSGAGE(323)                 QTIME(1340,4540)
   MEDIALOG(S0000001.LOG)
```

System Statistics

- Statistics are useful for checking what applications and the queue managers are doing. They could be used for charge-back or capacity planning.
- This release puts SMF-style information in the hands of Distributed administrators, as it is now reported via PCF Event Messages

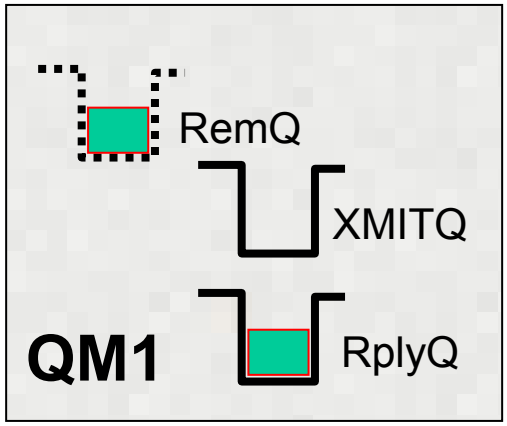
```
QueueManager: klein
IntervalEndDate: 2004-08-10
IntervalEndTime: 14:39:50
CommandLevel: 600
SeqNumber: 0
ApplName: amqsput.exe
ApplicationPid: 9408
ApplicationTid: 1
UserId: 'metaylor'
ObjectCount: 1
```

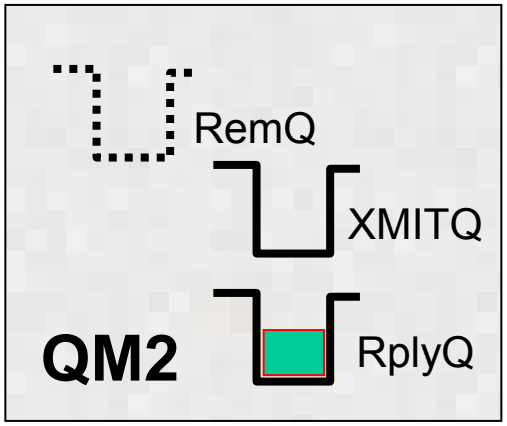
```
OBJECTS:
QueueName: 'APP.QUEUE.X'
  QueueType: Predefined
  QueueDefType: Local
  OpenCount: 1
  OpenDate: 2004-08-10
  OpenTime: 14:39:49
  CloseCount: 1
  CloseDate: 2004-08-10
  CloseTime: 14:39:50
```

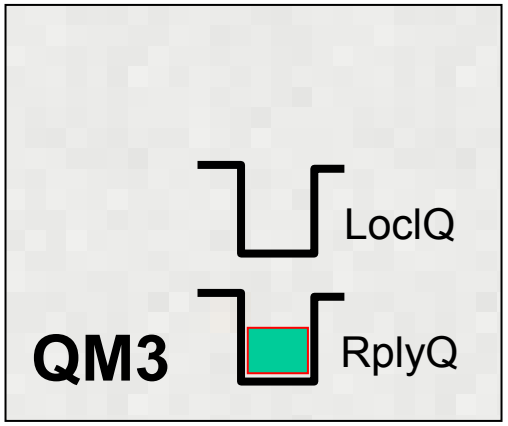
```
PutCount: [0, 1]*
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [0, 4]
PutMinBytes: [0, 4]
PutMaxBytes: [0, 4]


*array shows non-
persistent, persistent
```

Network Configuration Checker - Traceroute for WMQ







Put to REMQ, which is XMITQ on QM1

Taken from XMITQ, put to RemQ on QM2 which is XMITQ on QM2

Taken from XMITQ, put to LoclQ on QM3

- Reports generated showing message flowing
- Transmission queues, channel names, final queue
- Timestamps
- Could be used to validate cluster workload distributions

Standards - IPv6

- The new version of the TCP/IP protocol
- IPv6 expected to get more important during the lifetime of this release
  - Allows many more addresses
- Extensions to WMQ attributes & parameters to allow IPv6 addresses
  - eg instead of 9.20.3.4 you can have fe80:43e4:0204:acff:fe97:2c34:fde0:3485
  - 32 bits extended to 128 bits
- IPADDR determines which stack(s) to listen to
- CONNAME and LOCLADDR determines which stack to send on
- Clusters will work with IPv6 stacks

Channel Compression

- All platforms now include optional channel compression
  - Particularly useful with some types of data - text, XML
  - Makes best use of network bandwidth
  - Should give better performance for encrypted (SSL) channels
- Channel compression allows V6 queue managers and clients to communicate using compressed data on a per channel basis
- Header data compression and/or message data compression can be selected
- Message data is compressed using either RLE or ZLIB compression
  - Options for ZLIB to prefer faster or better compression

Channel Compression

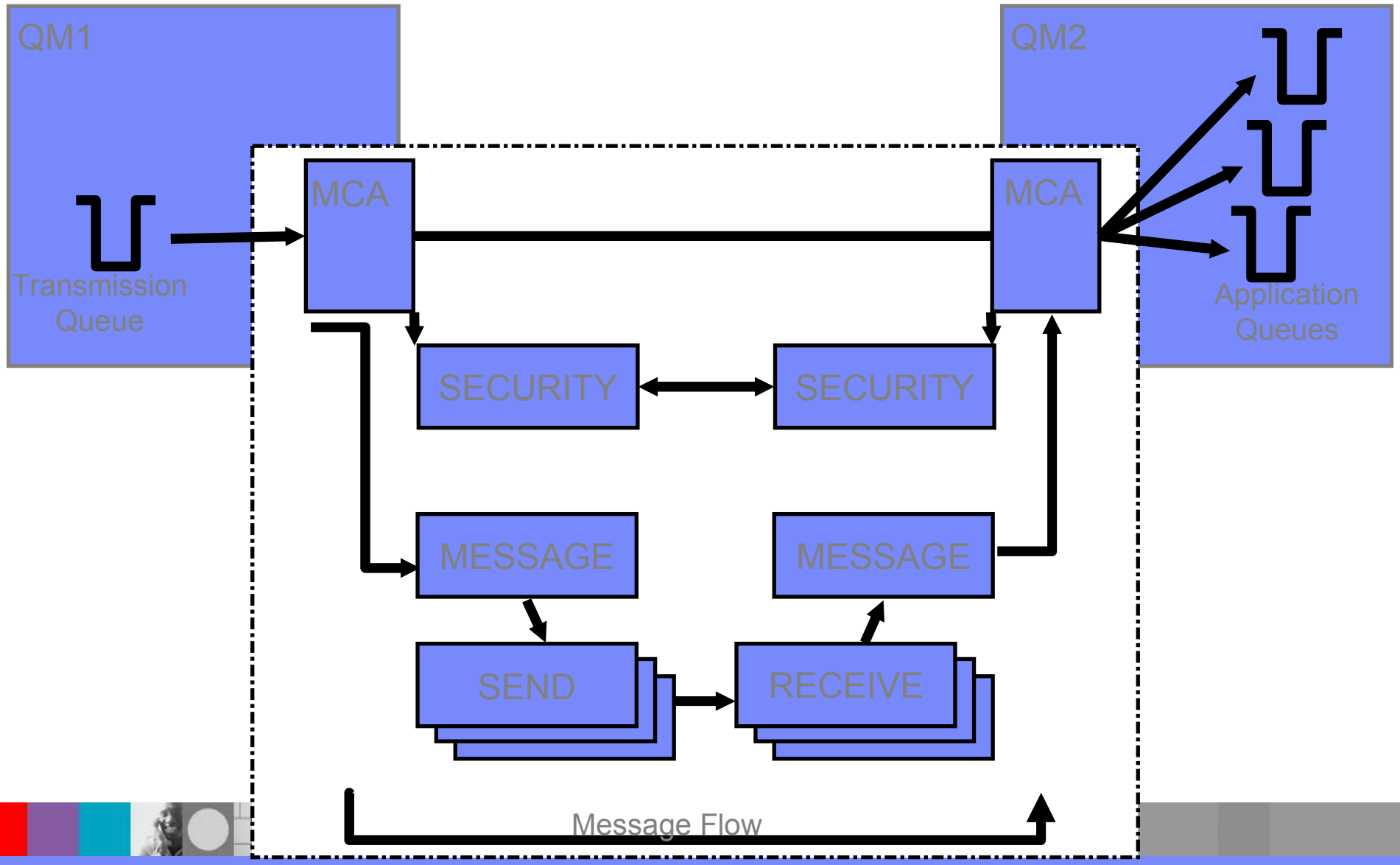
- Compression on channels has a selection of algorithms, to allow trade-offs between the CPU cost of compression, and the benefit of sending fewer bytes on the network. The algorithms have been chosen to deal well with text and XML messages. The CPU cost of compression ought to be paid back if you are also using encryption on SSL channels. This will work on both qmgr channels and client connections. When channel monitoring is enabled, information about compression percentages and time taken is shown in channel status outputs.
- Note that compression will alter the data passed to send and receive exits, but not message exits
- SDR - RCVR channel with COMPHDR(SYSTEM)
  - Header length reduced from 476 bytes to 66 bytes (86 %)
- CLNTCONN - SVRCONN channel with COMPHDR(SYSTEM) - MQPUT
  - Header length reduced from 500 bytes to 89 bytes (82 %)
- Message Compression
  - Message Compression rates are dependent on the form of the data
  - Example below of blank padded text strings (2590 bytes)

	CompRate	CompTime
RLE	55%	220
ZLIBFAST	81%	1300
ZLIBHIGH	82%	1700

Native WMQ Security

	Native WebSphere MQ
Data Protection	SSL services at channel level only & *all-or-nothing* Only protects messages while in transit
Data Protection Policy	Statically set at channel level
WMQ Client Access	Limited control over client connections
Administration	Every queue must have a unique ACL set on it Access control policy separate from data protection and audit
Identities	Authentication solely based on local OS identities

Current Channel Security



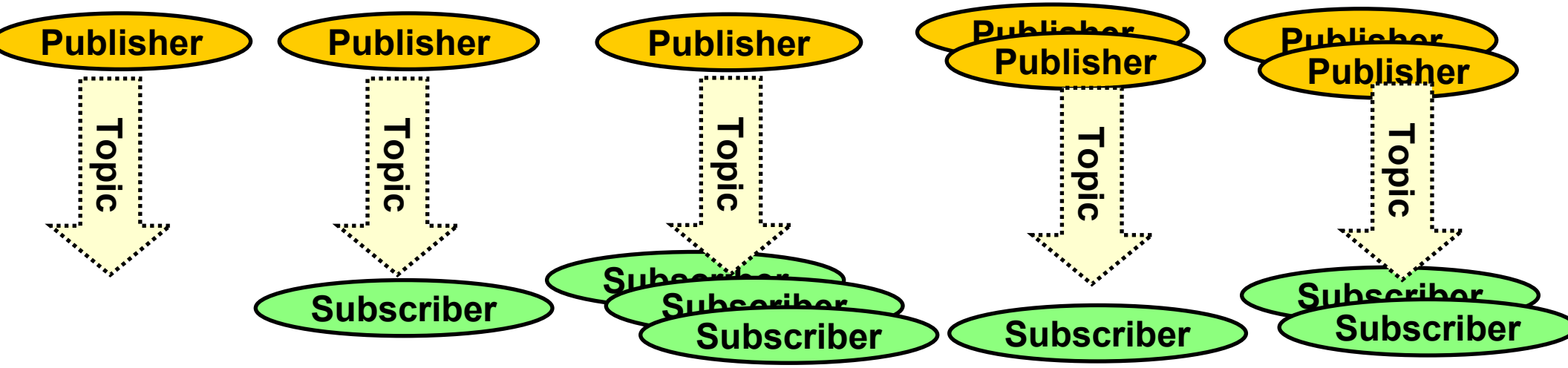
WMQ Channel Security Exits

- Security Exit
  - Called after initial channel negotiation
  - Useful for partner authentication
- Message Exit
  - Operates on the full message
  - Useful for encryption of the full message before transmission
- Send/Receive Exit
  - Operates on the transmission buffer
  - Useful for compression and decompression
  - Useful for encryption

Publish/Subscribe

- A natural part of the JMS API
  - Combines both Publish/Subscribe and Point-to-Point patterns
  - Now also a natural part of the native MQI
- Point-to-point asynchronous messaging decouples applications
  - But still implies a one-one relationship between sender and receiver
- Publish/subscribe is a further stage of decoupling
  - Sender has no direct knowledge of how many (if any) apps will see a message
  - Link between applications is a Topic, not a Queue
- WMQ V6 (Distributed) included a Publish/Subscribe broker (formerly MA0C)
  - Compatibility mode available in V7
- Implementation substantially improved with V7
  - And is available for the first time on z/OS

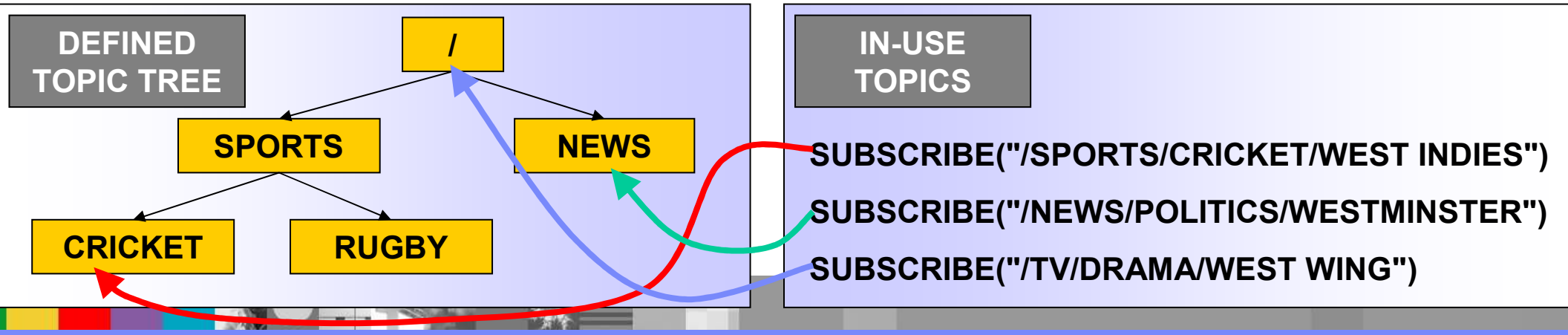
Loose Coupling with Publish/Subscribe



1:None **1:1** **1:Many** **Many:1** **Many:Many**

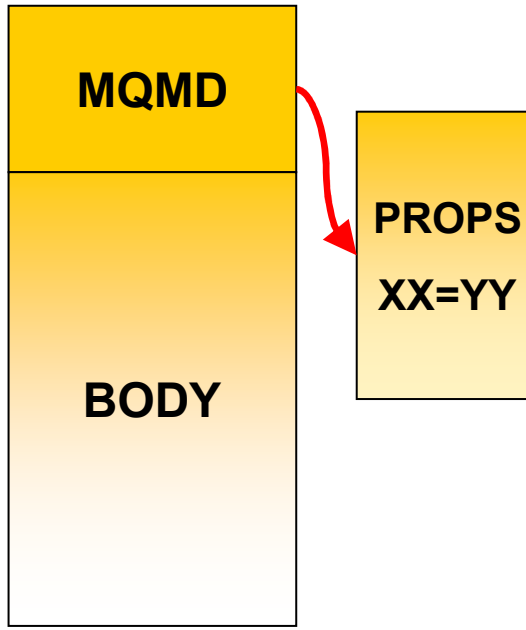
Publish/Subscribe Administration

- Based on Topic Strings
- Topic Objects
  - New object type, like queue or channel definitions
  - A 48-character name which has a longer attribute for full **topic string**
  - Defines major points in a topic tree
  - No additional definitions needed before applications can start using pub/sub
- In-use topics
  - The topic strings that applications are publishing or subscribing on
  - Inherit attributes (eg security) from the "closest" defined topic object
  - Not defined administratively, but can be viewed



Message Properties

- Arbitrary values associated with the message but not part of the body
  - Like a user-extendable MQMD
  - Already part of JMS
- New verbs including **MQSETMP** and **MQINQMP**
  - Properties can be integers, strings, boolean, etc.
- Easier to use than RFH2 folders
  - Receiving apps do not see them unless they want
  - No need to parse and skip over message headers
- Configuration options for compatibility
  - Queue and channel attributes define behaviour
  - Defaults will create RFH2 folders
- Permits explicit statement of relationships between messages
  - eg Message X is a **REPLY** to Message Y
  - Messages referred to by handles



Client Performance

- Traditional WMQ non-persistent messages more reliable than some need
- "Read Ahead" for Receiving Messages/Publications:
  - Messages sent to a client in advance of MQGET, queued internally
  - Administrative choice – no application changes needed
  - Higher performance in client
- "Asynchronous Put" for Sending/Publishing Messages:
  - Application can indicate it doesn't want to wait for the real return code
    - Maybe look for return code later – **MQSTAT** verb
  - Maintains transactional semantics
  - Higher performance in client

Client Connection Management

- Shared Client Conversations
  - Several connections from the same process can be handled on the same socket
  - Faster startup for the second and subsequent connections
- Implementation also gives us more heartbeat opportunities
  - Faster failure notification for clients
- Client Connections
  - Automatic workload distribution via CCDT
  - Control number of connected clients at a queue manager
- Free connections to z/OS for administration programs like WMQ Explorer
  - Limited number of clients permitted by V7 license without CAF

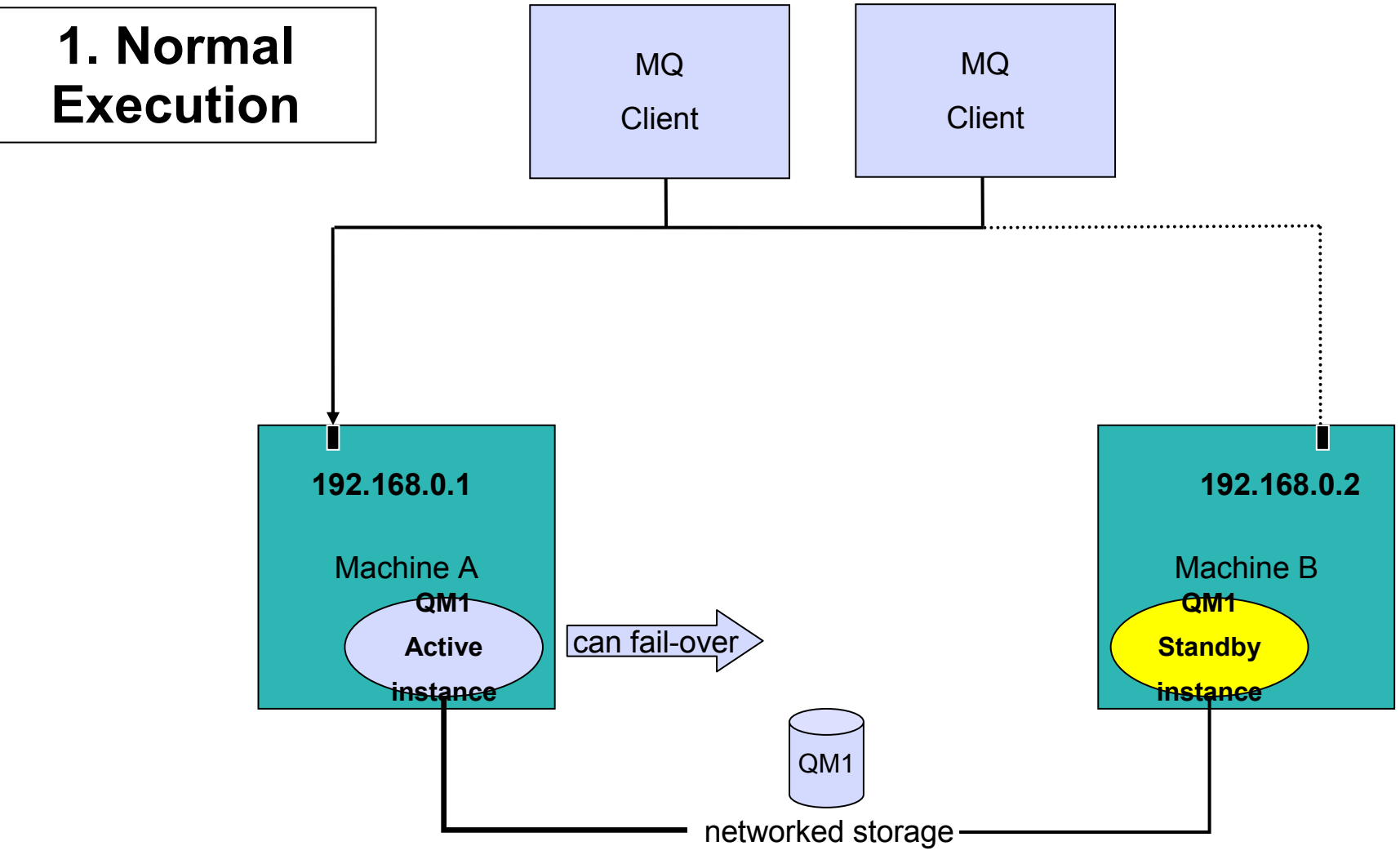
WMQ V7.0.1 Content Summary

New Feature	*Benefits*	*Details*
Multi-Instance Queue Managers	Increases availability Does not require specialist skills Can help ease system maintenance	Enables automatic failover to a standby Queue Manager instance in the event of an incident or planned outage
Automatic Client Reconnect	Increases availability Simplifies programming	Provides Client-connected applications with automatic detection of failures and reconnects to alternative Queue Managers
Enhanced Governance	Increases visibility of changes Enables SOA Governance	Emits events whenever configuration changes are made or commands are run Service Definition wizard generates WSDL describing MQ apps
Enhanced SSL Security	Simplifies security certificate management	Supports certificate checks with Online Certificate Status Protocol (OCSP) as well as to Certificate Revocation Lists (CRL)
Enhanced .NET support	Increases ease-of-use for .NET developers	Provides IBM Message Service Client for .NET developers Supports use of WebSphere MQ as custom channel within Windows Communication Foundation
Increased 64-bit z/OS exploitation	Increased use of z/OS system resources Provides constraint relief for virtual storage	Extends use of 64-bit storage by Queue Manager enabling more capacity such as number of open queues
z/OS Log Compression	Increased use of z/OS system resources Increased log performance & bandwidth	Compresses message logs produced by persistent messages
z/OS Group Units of Work	Increased resilience	Enables Units of Work to be owned collectively by Queue Sharing Groups so that any Queue Manager in the group can process two-phase transactions from clients
Publish/Subscribe Interfaces	Additional control of pub/sub behaviour Simplified integration for Message Broker	Exit point to dynamically modify routing and content Tools to migrate pub/sub state from MB to MQ

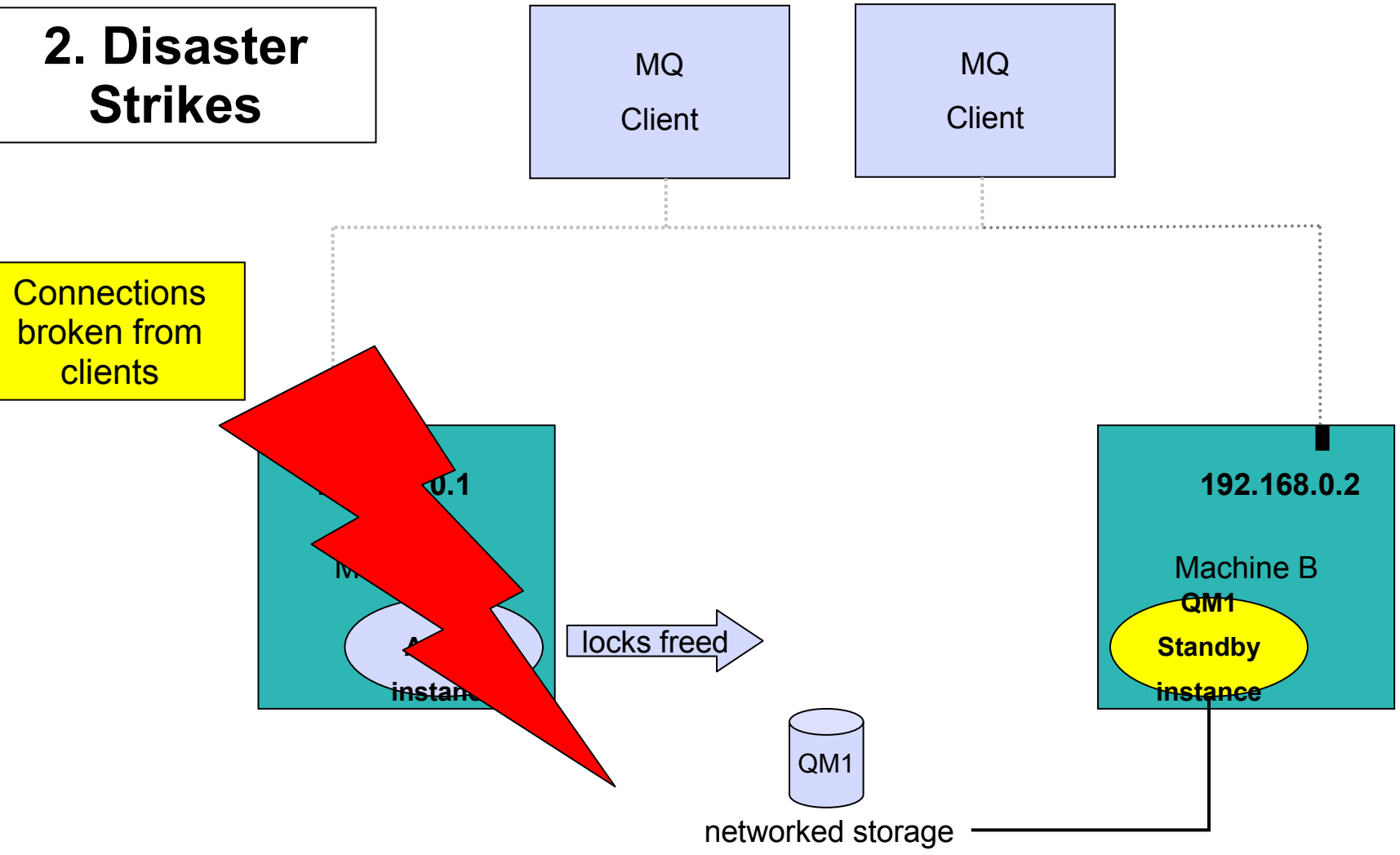
Distributed Platforms: Multi-instance Queue Managers

- Basic failover support without HA coordinator
  - Faster takeover: fewer moving parts
  - Cheaper: no specialised software or administration skills needed
  - Windows, Unix, Linux platforms
- Queue manager data is held in networked storage
  - NAS, NFS, GPFS etc so more than one machine sees the queue manager data
  - Improves storage management options: formal support for these even without failover config
- Multiple (2) instances of a queue manager on different machines
  - One is “active” instance; other is “standby” instance
  - Active instance “owns” the queue manager’s files and will accept app connections
  - Standby instance does not “own” the queue manager’s files and apps cannot connect
    - If active instance fails, standby performs queue manager restart and becomes active
- Instances share data, so it’s the SAME queue manager

Multi-instance Queue Managers

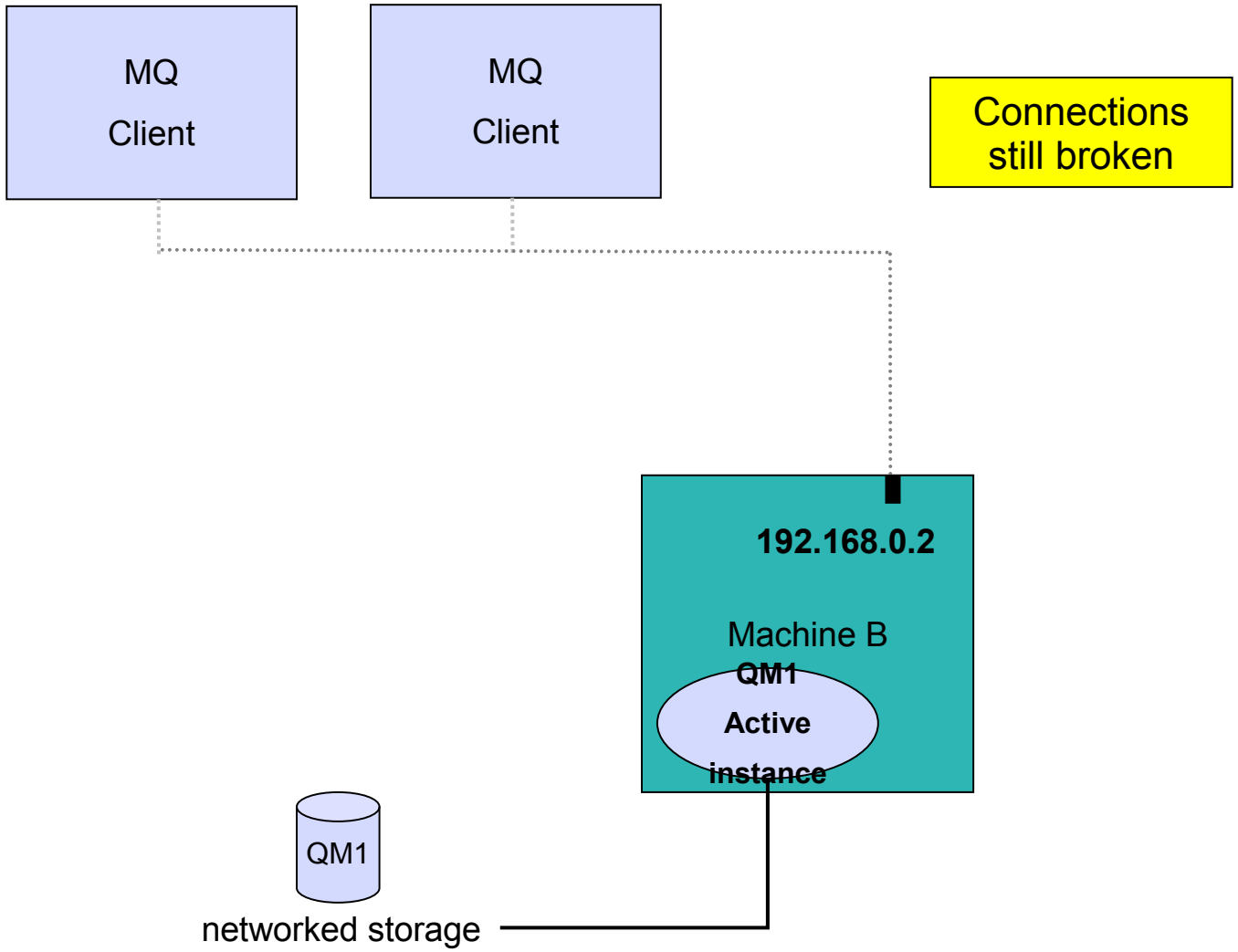


Multi-instance Queue Managers



Multi-instance Queue Managers

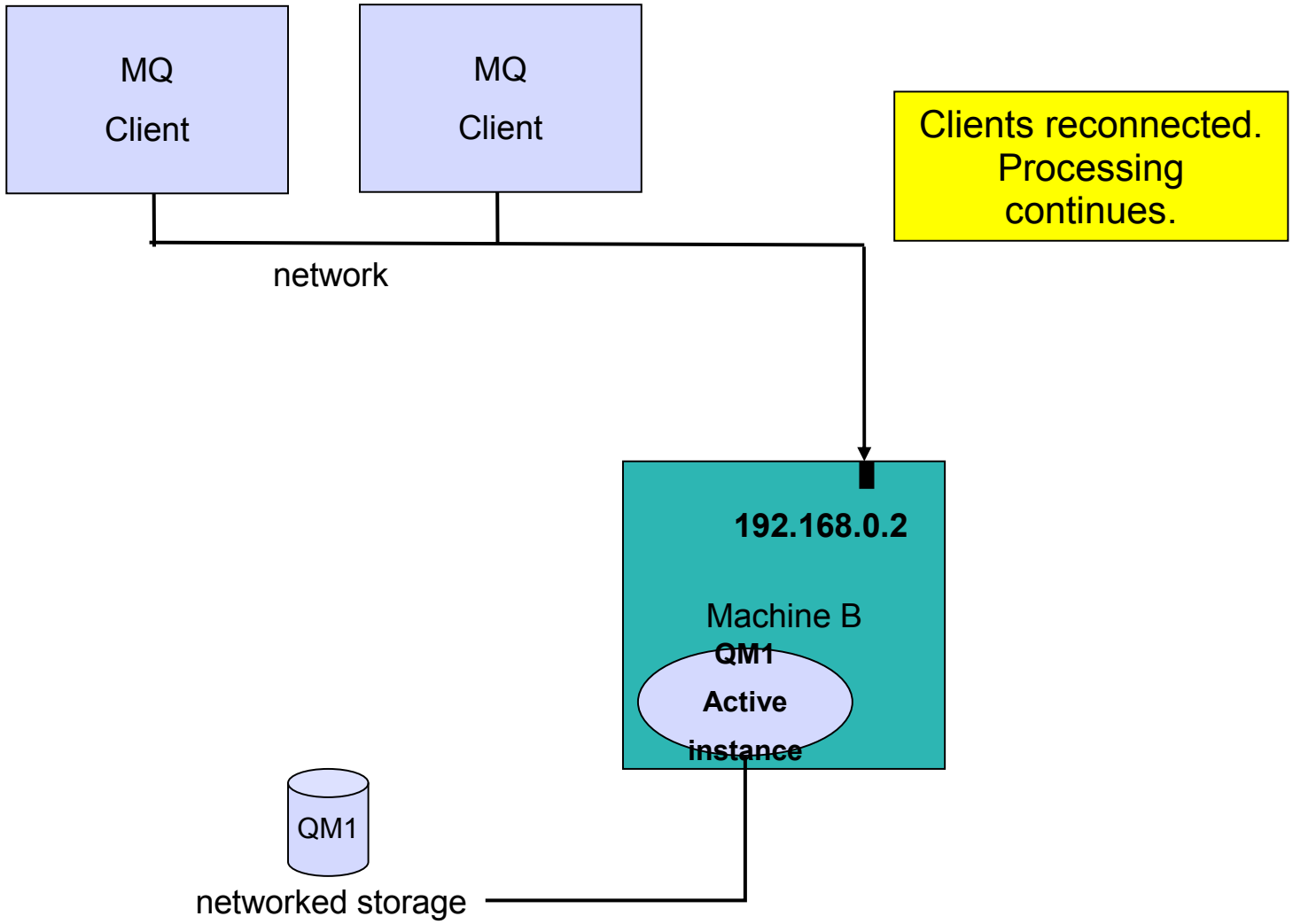
3. Standby Comes to Life



Owns the queue manager data

Multi-instance Queue Managers

4. Recovery Complete



Owns the queue manager data

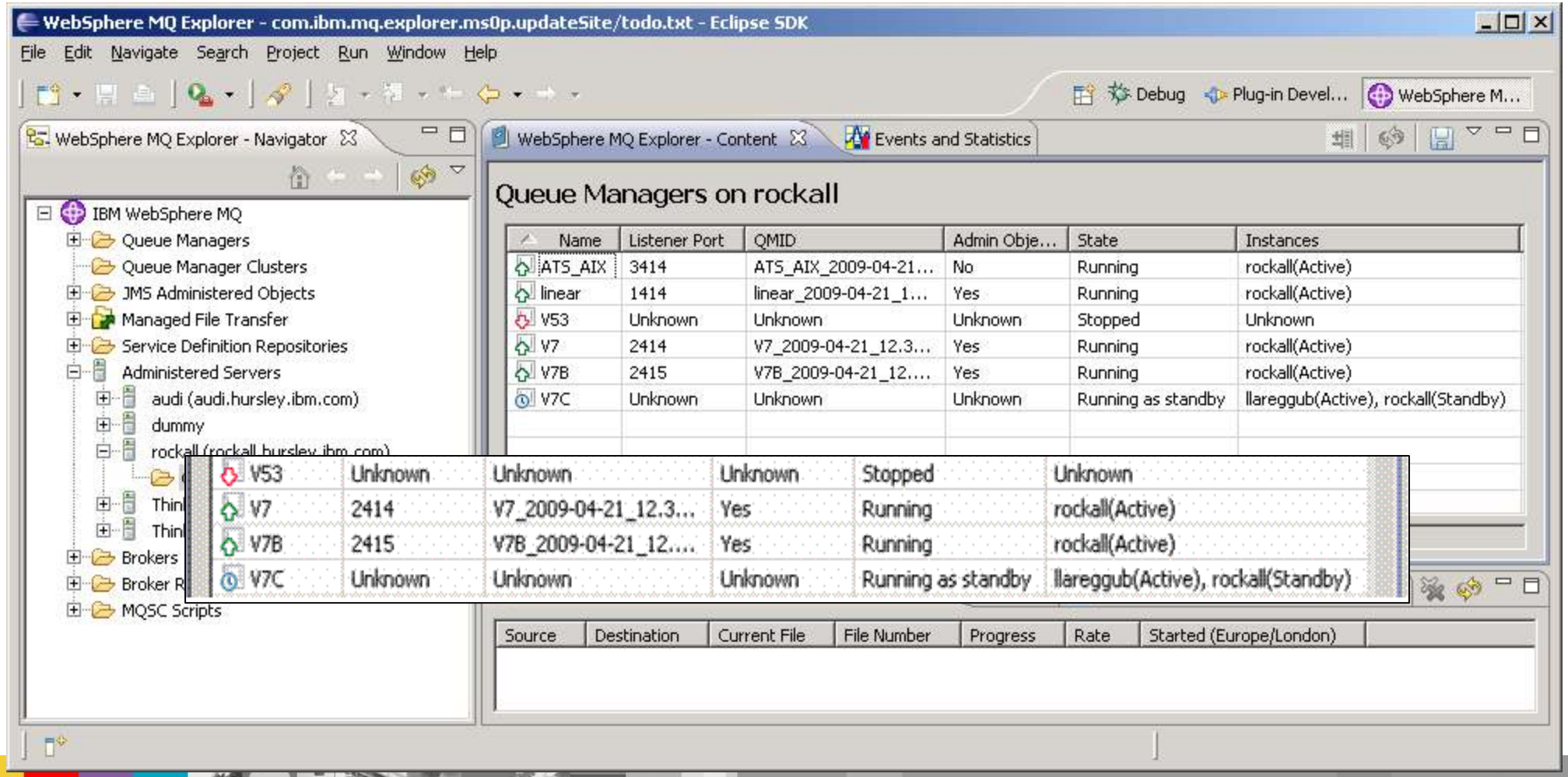
Multi-instance queue managers: How it looks

- Enhanced dspmq
- New option for dspmq to output English-only text
  - Useful for programmable parsing

```
$ hostname
rockall
$ dspmq -x
QMNAME(V7)            STATUS(Running)
     INSTANCE(rockall)    MODE(Active)
QMNAME(V7B)           STATUS(Running)
     INSTANCE(rockall)    MODE(Active)
QMNAME(V7C)           STATUS(Running as standby)
     INSTANCE(llareggub)  MODE(Active)
     INSTANCE(rockall)    MODE(Standby)
```

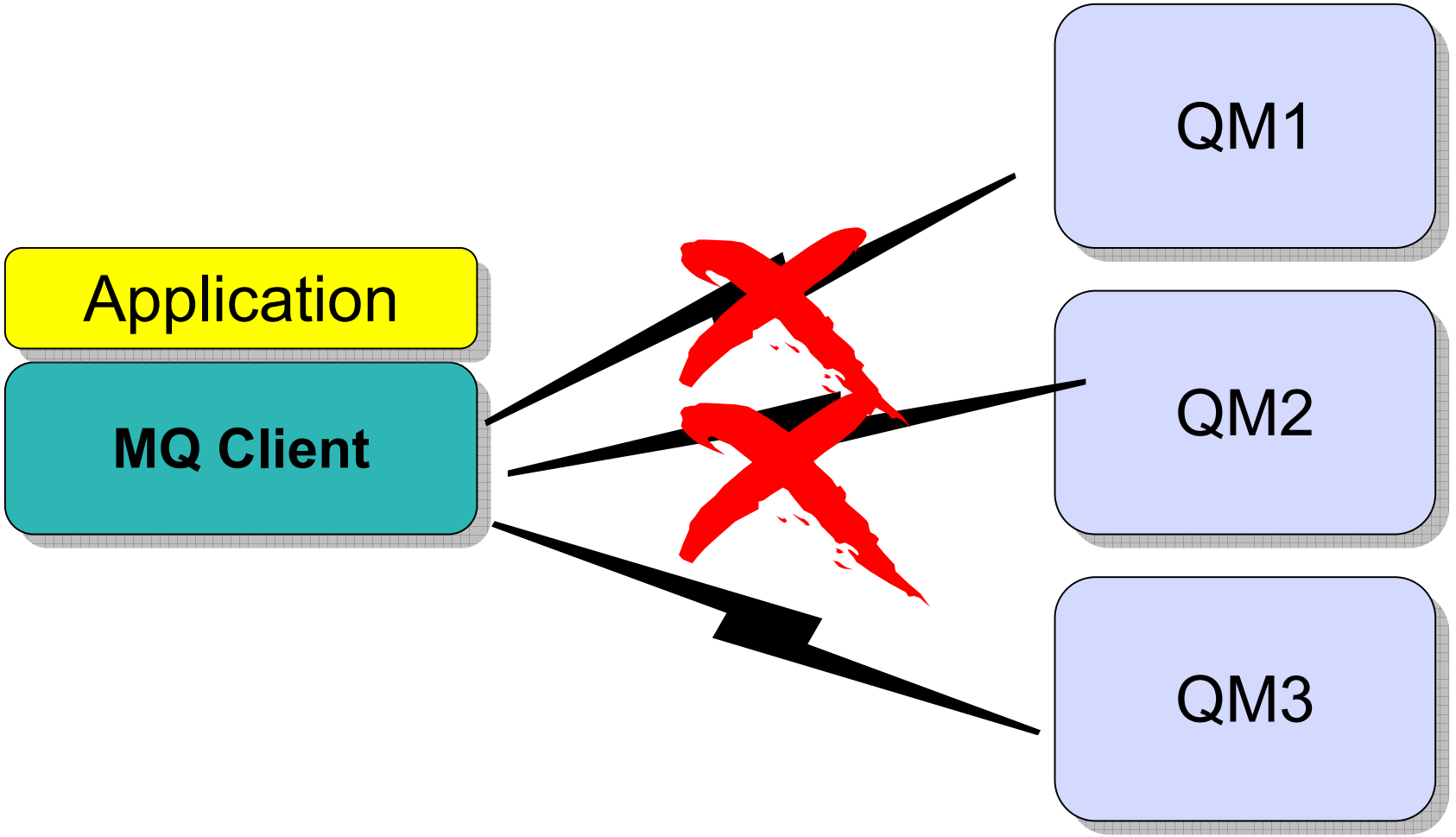
Multi-instance queue managers: How it looks

- As a graphical example, SupportPac MS0P V7.0.1



Automatic Client Reconnection

- Client library provides necessary reconnection logic on detection of a failure
- Hides failure from application code



Automatic Client Reconnection

- Tries to hide queue manager failures by restoring current state automatically
  - For example, if MQPUT returns error, client reruns MQCONN/MQOPEN/MQPUT internally
- Uses the list of addresses in CONNAME to find queue manager
  - MQSERVER environment variable also understands list
  - MQSERVER=SYSTEM.DEF.SVRCONN/TCP/host1(1414),host2(1414)
- Can reconnect to the same or different Queue Manager
- Re-opens queues and other qmgr objects, re-establishes subscriptions
- Reconnection interval is backed off exponentially on each unsuccessful retry
  - Total timeout is configurable – default 30 minutes

Multi-instance queue managers: Details

- MQ is NOT becoming an HA coordinator
  - Generally, if other resources also required, use an HA coordinator such as HACMP
  - Service objects can restart applications with qmgr but limited control
  - Message Broker will integrate with and exploit this MQ function
- The IP address is not taken over
  - Channel config needs all possible addresses unless you use external IPAT or intelligent router
  - CONNAME('host1(port1),host2(port2)') syntax extension on all platforms including z/OS
- Support for networked storage over modern network file system protocols
  - For example, NFS v4 (not v3)
  - Tool (amqmfsck) shipped to validate configuration
- New options for crtmqm/strmqm/endmqm to control operations
  - Cannot guarantee which instance becomes the primary
- Removes need for MC91, which will be withdrawn
  - crtmqm now does equivalent of MC91's hacrtmqm

Automatic Client Reconnection: Details

- Enabled in application code or ini file
  - Event Handler callback shows reconnection is happening if app cares
- Tries to keep dynamic queues with same name
  - So replies may not be missed
- Not all MQI is seamless, but majority repaired transparently
  - eg a browse cursor would revert to the top of the queue, non-persistent messages will have been lost during restart, non-durable subscriptions may miss some messages, in-flight transactions backed out, hObj values maintained
- Some MQI options will fail if you have reconnection enabled
  - Using MQGMO_LOGICAL_ORDER, MQGET gives MQRC_RECONNECT_INCOMPATIBLE
- Initially just in MQI and JMS – not the other OO classes
  - Requires both client and server to be V7.0.1 level with SHARECNV>0
  - Server can be z/OS

Simplified Administration

- Remote runmqsc no longer requires default qmgr
  - New optional “-m” flag to specify a local qmgr when “–w” also used
  - Useful in active/active HA configurations where no default qmgr may exist
- Can set dsp permissions for any user on SYSTEM.AUTH.DATA.QUEUE
  - Making it easier to have “read-only” administration
  - No more Authorisation Failure Events if someone tries to access this queue
  - Also being retro-fitted to V6 via APAR IZ52608: expected in 6.0.2.8

Publish/Subscribe Enhancements

- Option to discover if no subscribers (user or proxy) during MQPUT/PUT1
  - MQPMO_WARN_IF_NO_SUBS_MATCHED
  - MQRC_NO_SUBS_MATCHED
- Will guarantee that no one has received the publication
  - But does NOT guarantee that anyone will definitely receive the publication
  - For example, it is not returned if the target queue is full
- Publish Exit
  - When a publication is made, this exit is invoked for each valid subscriber
    - Runs “inside” the queue manager
  - Can change routing destination
  - Can change contents of message
  - Can change contents of message descriptor
  - Can inhibit publication

Automatic Client Reconnection: Details

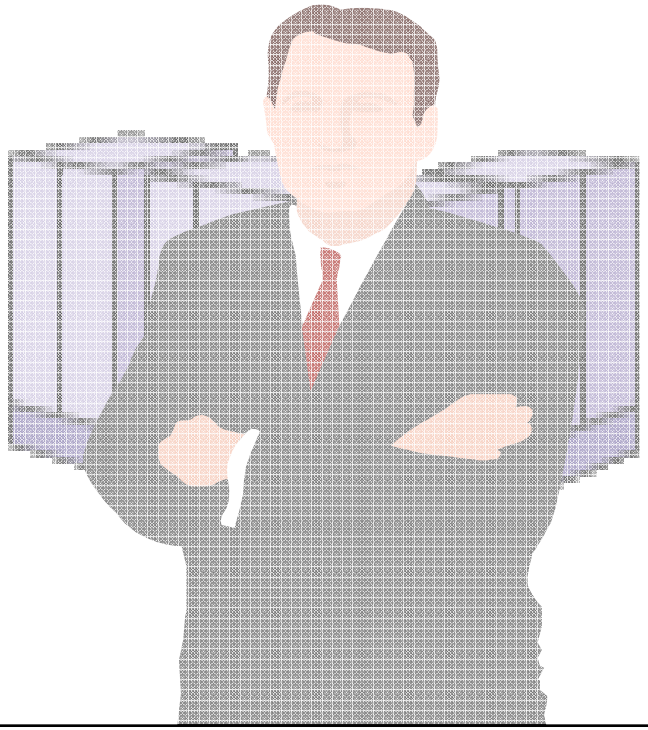
- Enabled in application code or ini file
  - Event Handler callback shows reconnection is happening if app cares
- Tries to keep dynamic queues with same name
  - So replies may not be missed
- Not all MQI is seamless, but majority repaired transparently
  - eg a browse cursor would revert to the top of the queue, non-persistent messages will have been lost during restart, non-durable subscriptions may miss some messages, in-flight transactions backed out, hObj values maintained
- Some MQI options will fail if you have reconnection enabled
  - Using MQGMO_LOGICAL_ORDER, MQGET gives MQRC_RECONNECT_INCOMPATIBLE
- Initially just in MQI and JMS – not the other OO classes
  - Requires both client and server to be V7.0.1 level with SHARECNV>0
  - Server can be z/OS

IBM WebSphere MQ V7.0

- Enhanced JMS
  - More applications being written to use this API
  - Underpins many SOA/ESB solutions needing access to messaging
  - Improved performance & ease-of-use
- Enhanced Publish-and-subscribe
  - Ease-of-use
  - New support for z/OS
- Extended verbs and behaviors for MQI programming interface
- Enhanced MQ clients for increased throughput resilience and availability
- Web 2.0 support to help create richer user experience
- Evolutionary – if you know V6, you will know V7

Why Clients bet their business on WebSphere MQ

- ✓ **Proven Scalability**
  - – Grow your network incrementally one server at a time
- ✓ **Performance**
  - – Many clients are moving millions of messages per day
- ✓ **Administer massive networks**
  - – Cross-platform, remote configuration tooling
  - – Tivoli CAM for enterprise-wide systems administration
- ✓ **Support for virtually any commercial IT platform**
- ✓ **MQ for zOS**
  - – Built from the ground up to exploit zSeries platform
  - – Consistent with MQ on distributed platforms
- ✓ **Clustering on distributed, shared queues on zOS**
  - – For High-Availability and workload balancing
  - – Easier to set up than you may think!
- ✓ **Multi-threading**
  - – Exploits multi-processors for high-speed throughput
- ✓ **Security**
  - – Industry-standard SSL support
  - – Certified for Common Criteria
  - – Policy-based security with MQ Extended Security Edition
- ✓ **IBM's worldwide 24x7 support**



- 90% of the Fortune 100
- 300 of the Fortune 500
- 66% of NA and European banks
- Banking clients move transactions worth $35 Trillion over MQ
- Government clients move 675+ million messages per day over MQ

WebSphere MQ: Summary

- Continued enhancements to the messaging backbone
- Early delivery of new function – no need to wait for the next full version
- Simplification for administrators and application developers
- Improved system exploitation