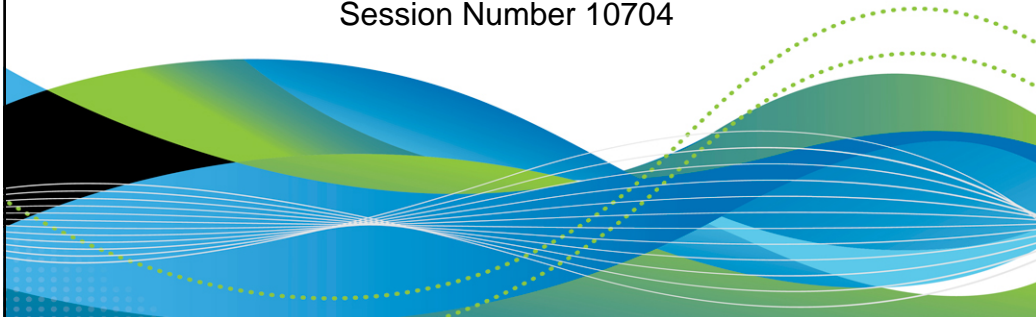


WebSphere MQ: Highly Scalable Publish/Subscribe Environments

Ralph Bateman

ralph@uk.ibm.com

Session Number 10704



Agenda

- What is Publish/Subscribe?
- Recap WebSphere MQ V7.0
Publish/Subscribe
- What is Multicast?
- Publish/Subscribe in WebSphere MQ V7.1
using Multicast

What is Publish/Subscribe ?



WebSphere MQ Publish/Subscribe

- N** • In WebSphere MQ V7 Publish/Subscribe becomes an in-built part of the MQ API (Application Programming Interface) and the administration model of WebSphere MQ.
- O** • WebSphere MQ V7 extends the MQ API (Application Programming Interface) to allow application programmers to use the publish/subscribe application model with ease. New verbs and changes to existing verbs are introduced in this presentation.
- T**
- E** • WebSphere MQ V7 also extend the administrative interfaces (MQSC and PCF) to allow administrators to manage Publish/Subscribe applications.
- S** • In WebSphere MQ V7.1, Publish/Subscribe is extended to include the Multicast transport.

What is Publish/Subscribe ?

Publish/Subscribe is a term used to define an application model in which the provider of some information is decoupled from the consumers of that information.

- providers of information need have no knowledge of consumers
- consumers of information need have no knowledge of providers
- new providers/consumers can be added without disruption
- Providers of information are called **publishers**
- Consumers of information are called **subscribers**

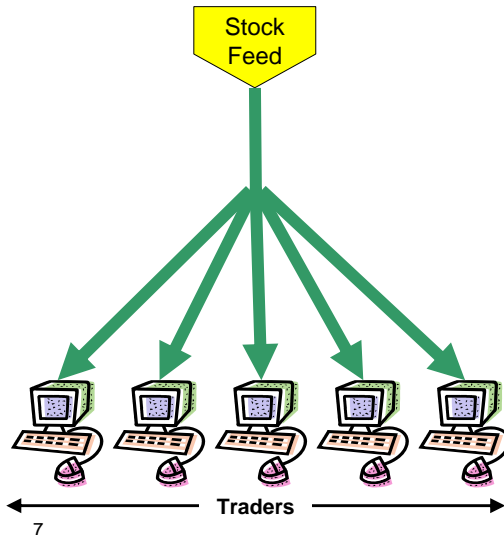
5

What is Publish/Subscribe? - Notes

- N**
- O**
- T**
- E**
- S**
- Publish/subscribe systems have become very popular in recent years as a way of distributing data messages from publishing computers to subscribing computers. Such systems are especially useful where data supplied by a publisher is constantly changing and a large number of subscribers needs to be quickly updated with the latest data. Perhaps the best example of where this is useful is in the distribution of stock market data.
 - In such systems, publisher applications of data messages do not need to know the identity or location of the subscriber applications which will receive the messages. Similarly, the subscribing applications do not need to know the identity or location of the publishing application providing the information. In this sense the providers and consumers are said to be loosely-coupled.

6

The classic example



- A "feed" provides a continuous flow of information which is pushed to interested parties
- Traders consume this information and use it as a basis for the buying and selling stock

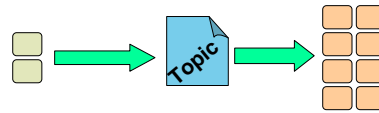
7

The classic example - Notes

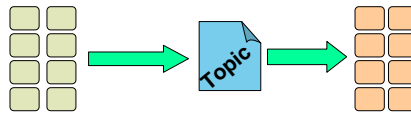
- N**
- Perhaps the most-commonly quoted example of a Publish/Subscribe system is one which provides stock-market information. Here a "feed" provides (publishes) a continuous flow of information containing the latest stock prices. The latest stock prices are required by traders who need this information in order to conduct trades. Traders register their interest in (subscribe to) particular stock prices and receive updates as prices change. Traders can be added/removed without disruption to the providers of the information who have no knowledge of who is receiving their information.
- O**
- The terms "push" and "pull" are also becoming increasingly popular when describing the flow of information between applications. If we concentrate on this example, traders receive new information from the stock-feed as soon as a stock price changes. In this sense the information can be thought of as being pushed directly to them. This pushing of information from provider to consumer is one of the major differentiators between publish/subscribe and more conventional systems. Our stock market example could have equally been designed in such a way that updated stock prices only flowed to the traders when they specifically requested, or pulled them from a central repository (server) of all stock prices. In such a system, the emphasis would instead be on the traders, to request a refresh of their stock prices on a continual basis.
- T**
- In fact WebSphere MQ supports both modes of operation.
- E**
- S**

8

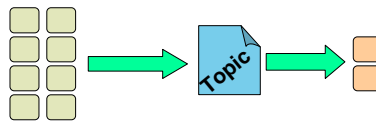
Loose-coupling with Publish/Subscribe



Few-to-many: Research, news tickers



Many-to-many: Prices and Quotes



Many-to-few: Orders

9

Loose-coupling with Publish/Subscribe - Notes

- N**
- In the WebSphere MQ Publish/Subscribe model the only thing which connects publishing and subscribing applications is the topic or subject which the publisher associates with his information. Publishers and subscribers need only agree on the topic to become connected to one another. Each different piece of information has its own topic associated with it. Subscribers nominate which types of information they want to receive by subscribing to specific topics.
- O**
- Publishers of information are unaware of subscribers to the extent that they may publish information even if there are no subscribing applications requiring it. Publishing and subscribing are completely dynamic processes. New subscribers and new publishers can be added to the system without disruption.
- T**
- With respect to a given topic, or piece of information, all possible combinations of publishers/subscribers are possible, that is:
 - information about each topic may be provided by a single or multiple publishing applications
 - the information may be received and processed by one or more subscribing applications
- E**
- The number of publishers and subscribers connected by a single topic depends upon the type of information which is flowing between them. As we will see later, WebSphere MQ supports both state and event based information, or topics.
- S**

10

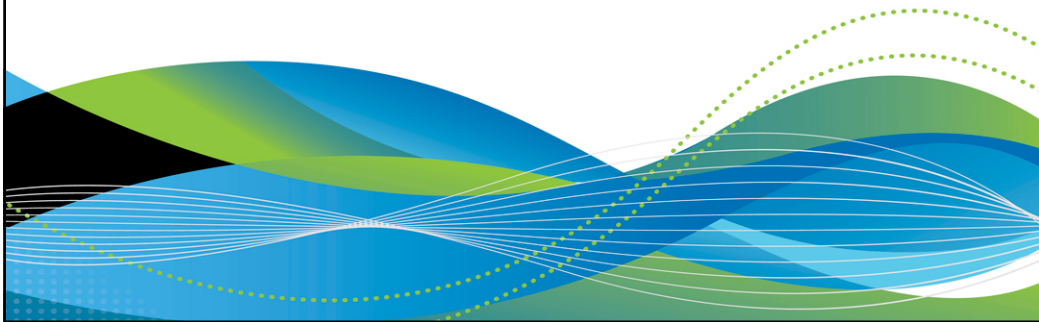
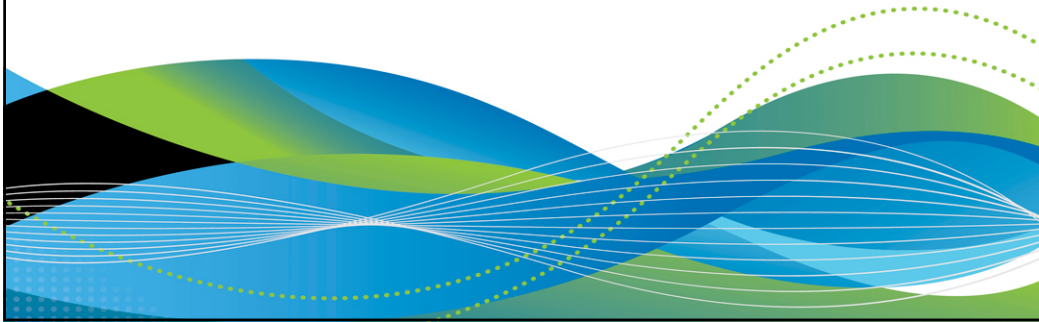
Publications and subscriptions

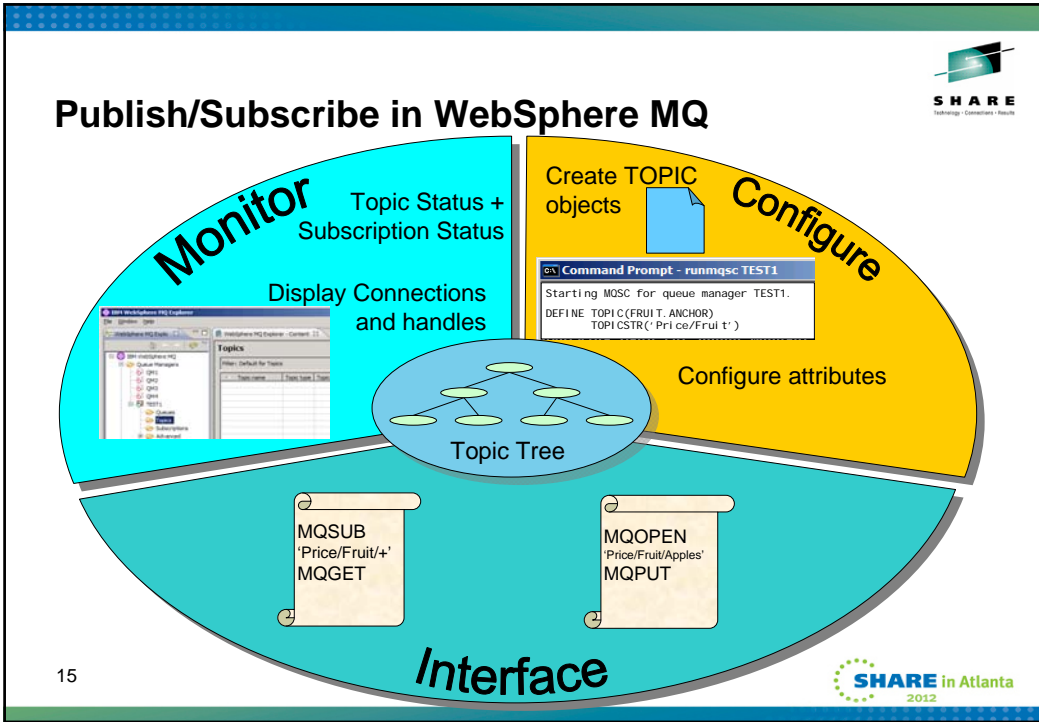
- Subscribers make **subscriptions** with the queue manager to register their interest in information relating to specific topics.
 - They use the MQSUB verb
- Publishers provide information about specific topics by sending **publications** to the queue manager
 - They use the MQPUT verb
- The queue manager forwards each publication it receives to all subscribers with a subscription which matches the associated topic

Publications and subscriptions - Notes

- N**
- Just to recap, applications which provide information are called publishers. Applications which consume information are called subscribers.
- O**
- A subscriber specifies the topic it is interested in receiving information about by specifying it on the MQSUB verb. A subscriber may make multiple subscriptions to the queue manager.
- T**
- A publisher publishes its information by putting a message to a topic.
 - It is the job of the queue manager, or queue manager network if multiple queue managers have been connected together, to ensure that all subscribing applications with matching subscriptions to the topic being published on receive the publisher's message, known as a publication.
- E**
- There is a separate presentation about publish/subscribe in a multiple queue manager scenario.
- S**

Re-cap WebSphere MQ V7 Publish/Subscribe





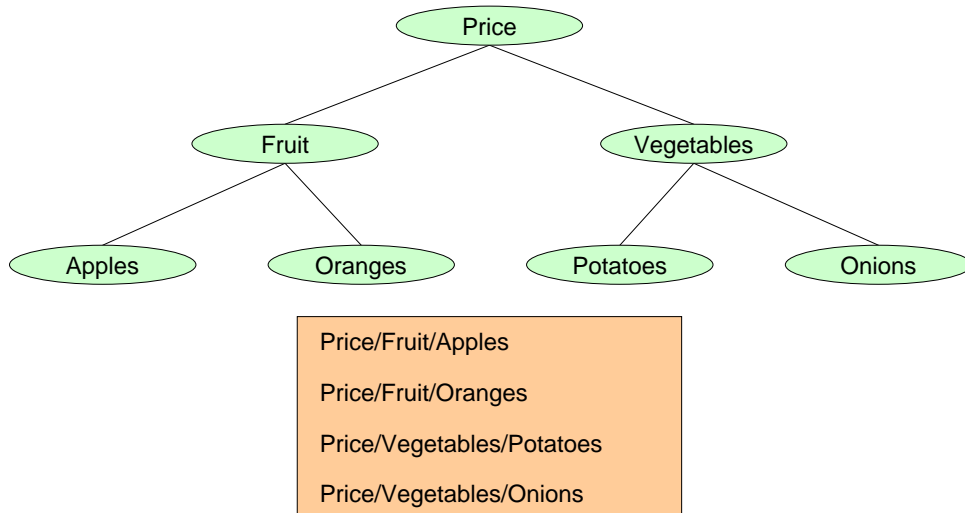
Publish/Subscribe in WebSphere MQ - Notes

N
O
T
E
S

- The queue manager holds a view of all the topic strings you are using in a hierarchical construct known as the topic tree (see next page). This topic tree is the central control point for all publish/subscribe. As a user you will interact with the topic tree in several different ways.
- You can configure the behaviour of the topic tree by defining topic objects and changing attributes on them. Of course you only need to do this if you want to change the default behaviour. You may not need any topic objects – we will look at this in more detail later.
- You can programmatically interface with the topic tree as a subscriber using MQSUB and as a publisher using MQOPEN and MQPUT.
- You can monitor the use of your topic tree by such applications using the Topic status command, the Subscription status command and the commands to display connections and their handles.

16

Topic strings and topic tree



17

Topic strings and topic tree - Notes

- N** • Topic strings can be any characters you choose. You can, and should, add structure to you topic strings using the ‘/’ character. This produces a topic tree with a hierarchical structure, as the example on this foil shows. Although this hierarchical topic tree was created by the use of the topic strings shown, we generally picture it as a tree such as this.
- O**
- T** • There are some special characters, apart from the ‘/’ character that you should avoid in your topic strings. These are ‘#’, ‘+’, ‘*’ and ‘?’ which are the symbols that are treated as wildcards for subscribers to use.
- E**
- S**

18

Topic Objects

- Not necessary for Publish/Subscribe
- Provide an administrative control point for your topic tree
 - Configuration attributes
 - Security profiles
 - Topic tree isolation

MY.TOPIC.OBJECT

Topic Objects - Notes

- N
O
T
E
S
- Topic objects are a new construct in WebSphere MQ V7. They can be used to control the behaviour of your topic tree.
 - You do not need to define any topic objects in order to use Publish/Subscribe with WebSphere MQ V7, however you may want to define some if you need to configure the topic tree to use non-default attributes; if you want to apply different security profiles to parts of your topic tree; or if you want to isolate you applications from administrative changes to the topic tree – rather like you do when you use remote queue and alias queue definitions.

Subscribing Application

- MQSUB verb
- Subscription Descriptor (MQSD) describes the topic
 - MQSD.ObjectString
 - MQSD.ObjectName
- Consume publications from the returned hObj
 - when MQSO_MANAGED used

```
MQSUB ( hConn,
        &SubDesc,
        &hObj,
        &hSub,
        &CompCode,
        &Reason);
```

```
MQGET ( hConn,
        hObj,
        &MsgDesc,
        &gmo,
        strlen(pBuffer),
        pBuffer,
        &DataLength,
        &CompCode,
        &Reason);
```

```
MQSD SubDesc = {MQSD_DEFAULT};
SubDesc.ObjectString.VSPtr = "Price/Fruit/Apples";
SubDesc.ObjectString.VSLength = MQVS_NULL_TERMINATED;
SubDesc.Options = MQSO_CREATE
                  | MQSO_MANAGED
                  | MQSO_FAIL_IF_QUIESCING;
```

21



Subscribing Application - Notes

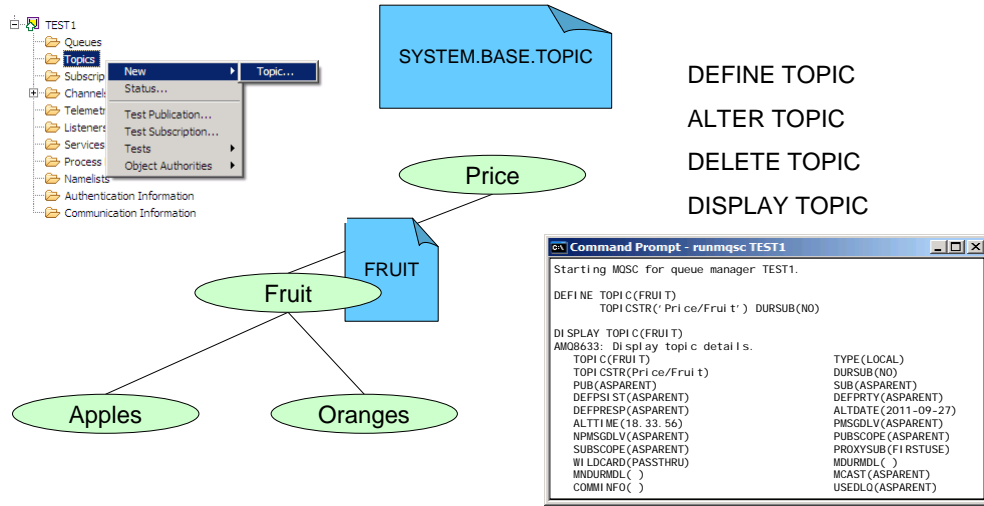
N
O
T
E
S

- An application that wants to register an interest in information about a certain topic needs to 'subscribe' to that topic. This can be done using the MQ API verb MQSUB. MQSUB can be thought of rather like MQOPEN. It details the resource you wish to use, and it is the point where security checks are done.
- The main structure that you need to be familiar with when using MQSUB is the MQSD (subscription descriptor). This structure is where you define the topic you are interested in; the options to use when making a subscription; and any other interesting changes to the way the subscription is made.
- When specifying the topic string you wish to subscribe for, you can provide the whole topic string, or an anchoring topic object which defines a certain point in the topic tree and then the remaining part of the topic string to be appended to that which the topic object represents.
- Here we see an example of MQSUB using the option MQSO_MANAGED. This is the option to use when the application wishes the queue manager to look after the storage of publication messages. On return from the MQSUB call your application is given two handles, an hSub and an hObj. The hSub is the handle to the subscription. The hObj is the handle which you can consume publications from, i.e. using MQGET or MQCB.
- Once you have successfully done an MQSUB, you can start to consume any publication messages that are being sent to you.

22



Defining a topic object



SYSTEM.BASE.TOPIC

Price

Fruit

Apples

Oranges

```

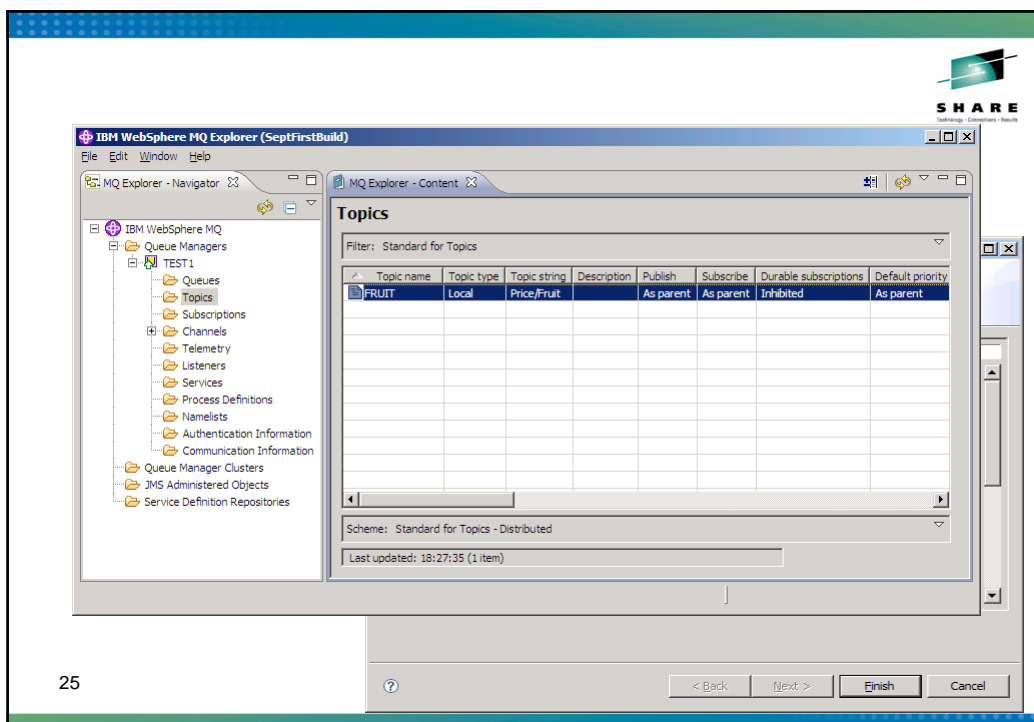
DEFINE TOPIC(FRUIT)
  TOPICSTR('Price/Fruit') DURSUB(NO)

DISPLAY TOPIC(FRUIT)
AMQ8633: Display topic details.
TOPIC(FRUIT)                                TYPE(LOCAL)
TOPICSTR('Price/Fruit')                     DURSUB(NO)
PUB(ASPARENT)                               SUB(ASPARENT)
DEFPSIST(ASPARENT)                         DEFPRTY(ASPARENT)
DEFPRESP(ASPARENT)                         ALTDTE(2011-09-27)
ALTTIME(18.33.56)                          PMSGDLV(ASPARENT)
NPMSGDLV(ASPARENT)                         PUBSCOPE(ASPARENT)
SUBSCOPE(ASPARENT)                         PROXYSUB(FIRSTUSE)
WILDCARD(PASSTHRU)                          MDURMDL( )
MNDURMDL( )                                 MCAST(ASPARENT)
COMMINFO( )                                 USEDLO(ASPARENT)
CUSTOM( )
  
```

23

- DEFINE TOPIC(FRUIT)
TOPICSTR('Price/Fruit') DURSUB(NO)
 - DISPLAY TOPIC(FRUIT)
AMQ8633: Display topic details.
- | | |
|-------------------------|--------------------|
| TOPIC(FRUIT) | TYPE(LOCAL) |
| TOPICSTR('Price/Fruit') | DESCR() |
| CLUSTER() | DURSUB(NO) |
| PUB(ASPARENT) | SUB(ASPARENT) |
| DEFPSIST(ASPARENT) | DEFPRTY(ASPARENT) |
| DEFPRESP(ASPARENT) | ALTDTE(2011-09-27) |
| ALTTIME(18.33.56) | PMSGDLV(ASPARENT) |
| NPMSGDLV(ASPARENT) | PUBSCOPE(ASPARENT) |
| SUBSCOPE(ASPARENT) | PROXYSUB(FIRSTUSE) |
| WILDCARD(PASSTHRU) | MDURMDL() |
| MNDURMDL() | MCAST(ASPARENT) |
| COMMINFO() | USEDLO(ASPARENT) |
| CUSTOM() | |

24



Defining a topic object - Notes

N

- Let's say you need to disallow the creation of durable subscriptions for one half of the topic tree. We can create one TOPIC object at the highest point where we need this behaviour to start, and that behaviour will be inherited by the nodes in the topic tree below that point without the need for any further TOPIC object definitions.

O

- As you might expect, this new object type has DEFINE, ALTER, DELETE and DISPLAY commands. One thing to note about ALTER is that the TOPICSTR parameter of a TOPIC object cannot be altered. Think of this attribute as the other name of the TOPIC object – you cannot alter the name of an object, you must delete and redefine the object to do that.

T

- Looking at the DISPLAY output from the object we just defined, we can see that many of the attributes that we didn't specify have the value ASPARENT (or for the character strings – have blanks, which means the same thing as the ASPARENT value). ASPARENT means that the value for this attribute is taken from the next TOPIC object found by walking up the topic tree. If the next TOPIC object found also says ASPARENT for the value that is being resolved we carry on up the tree – eventually we may get to the very top and thus use the values in the SYSTEM.BASE.TOPIC.

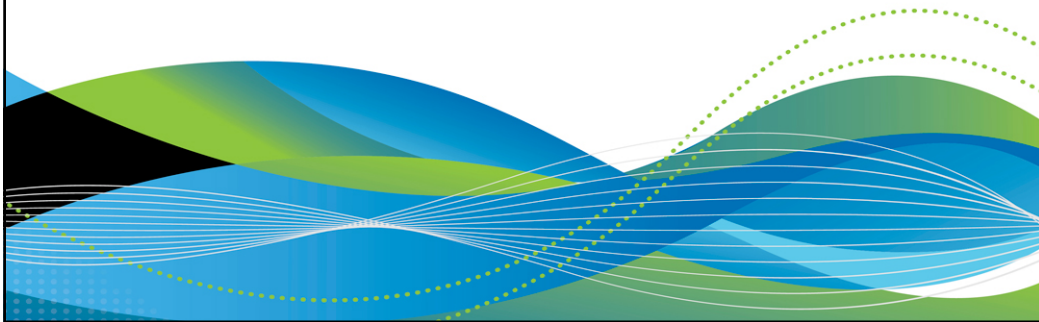
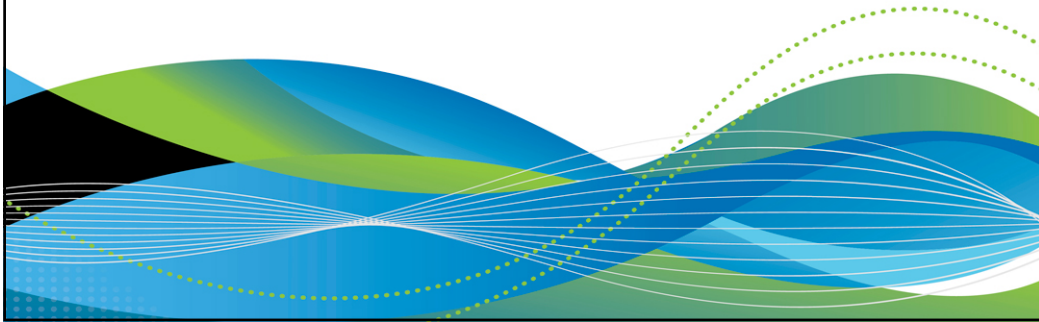
E

S

26

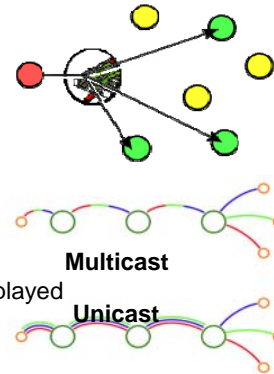
SHARE in Atlanta 2012

What is Multicast?



Multicast – Technical Overview

- IP Multicast is a low level form of pub /sub implemented in NICs and routers
 - Multicast group addresses
 - For IPv4 this is 224.0.0.0 to 239.255.255.255 ([RFC3171](#))
- Can be more efficient than unicast pub/sub
 - Scaling to a high number of subscribers
- But Multicast is more complex to set up than unicast
- Basic multicast is unreliable
 - No retries, no persistence
 - Data can be lost
- Reliable Multicast
 - Uses sequence numbers and NAKs
 - Allows receivers to request missed messages to be replayed
- Reliable protocols not interoperable across vendors
 - 'PGM' provides common denominator and is supported by routers



29

Multicast – Technical Overview - Notes

- N**
- IP Multicast is a low level form of pub /sub implemented in NICs and routers. Receivers register their interest in receiving data on pre-defined set of multicast group addresses). For IPv4 this is 224.0.0.0 to 239.255.255.255 ([RFC3171](#)). Senders send datagrams to the multicast address and then network cards/routers make copies of data and send to receivers who have registered for an address.
- O**
- Multicast can be more efficient than traditional uni-cast pub/sub, scaling to a high number of subscribers, due to removing the duplication of sending each subscriber its own copy of the data.
- T**
- However, multicast is more complex to set up than unicast. Routers need to be configured to pass multicast traffic. This is gradually getting easier, since ISPs are starting to enable routers – allowing multicast over the internet.
- E**
- The basic multicast transport is unreliable – no retries, persistence etc. This means data can be lost. However, there is a layer provided over the basic multicast transport called Reliable Multicast Messaging which uses sequence numbers and (negative) acknowledgements (ACKS/NAKs), allowing receivers to request messages to be replayed if they are missed. Publisher and subscriber speed mismatches can lead to broadcast storms and flooding. NAK suppression and aggregation is used to circumvent this problem.
- S**

30

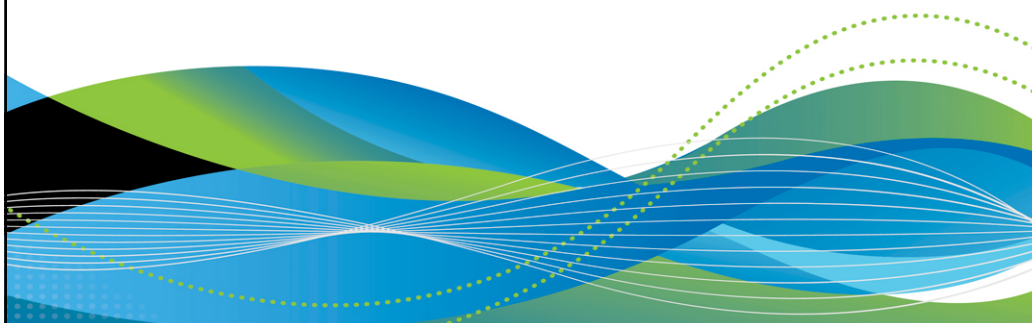
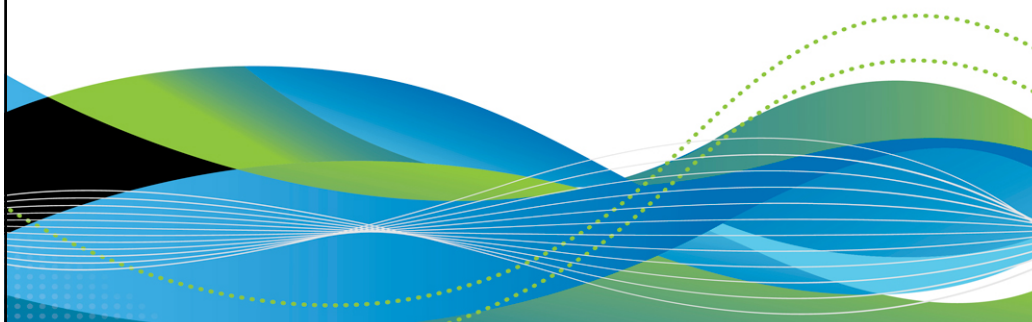
Multicast – What are the Benefits?

- Low latency
 - Much higher volumes than standard non-persistent messaging
 - Messages do not pass through queues
 - Peer to peer communication
- High Scalability
 - Additional subscribers cause no slow down
 - Reduced network traffic
- 'Fair delivery' of data
 - Each subscriber 'sees' the data at the same time
 - Fair delivery is critical to ensure that no recipient gains an advantage
 - Multicast offers near simultaneous delivery
- High availability
 - Multicast uses the network so does not need a pub/sub engine to fan-out data
 - Once a Topic is mapped to a group address there is no need for a Qmgr
 - Publishers and subscribers can operate in a "peer-to-peer" mode
 - Allows load to be reduced on Qmgr servers
 - Qmgr servers not a single point of failure

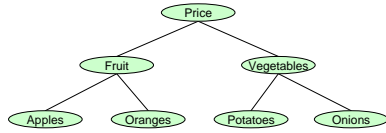
31



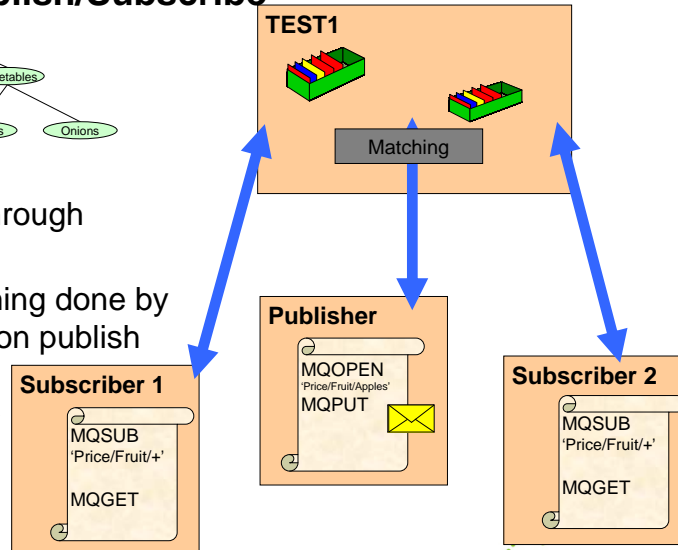
Publish/Subscribe in WebSphere MQ V7.1 using Multicast



Current MQ Publish/Subscribe



- Messages flow through Queue Manager
- Subscriber matching done by Queue Manager on publish
- Each subscriber sent its own copy of message

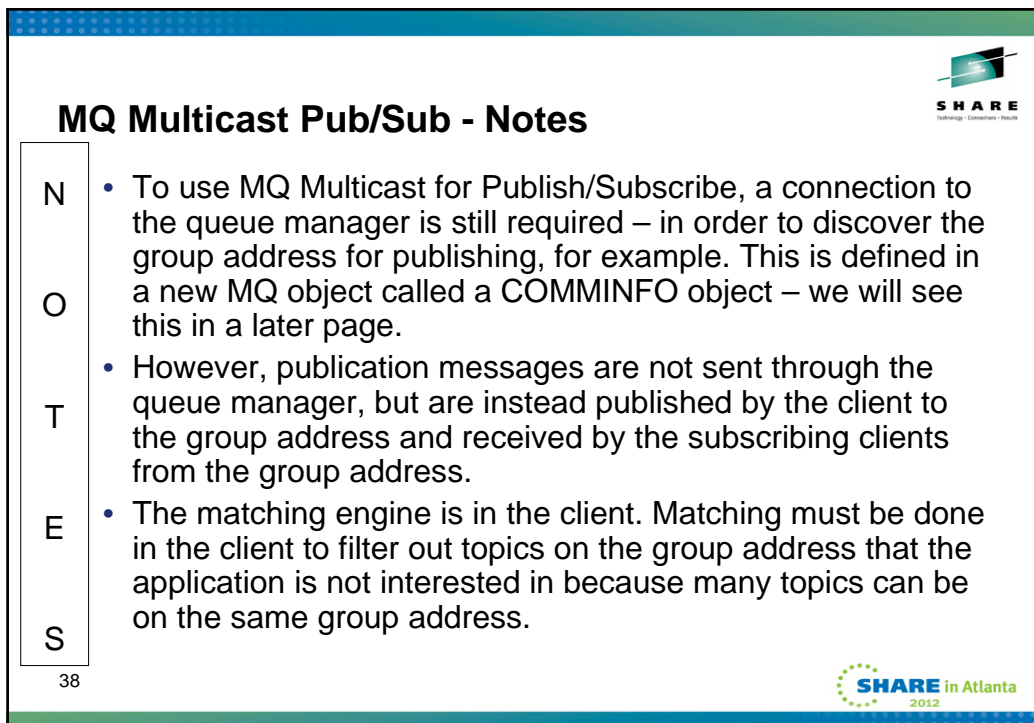
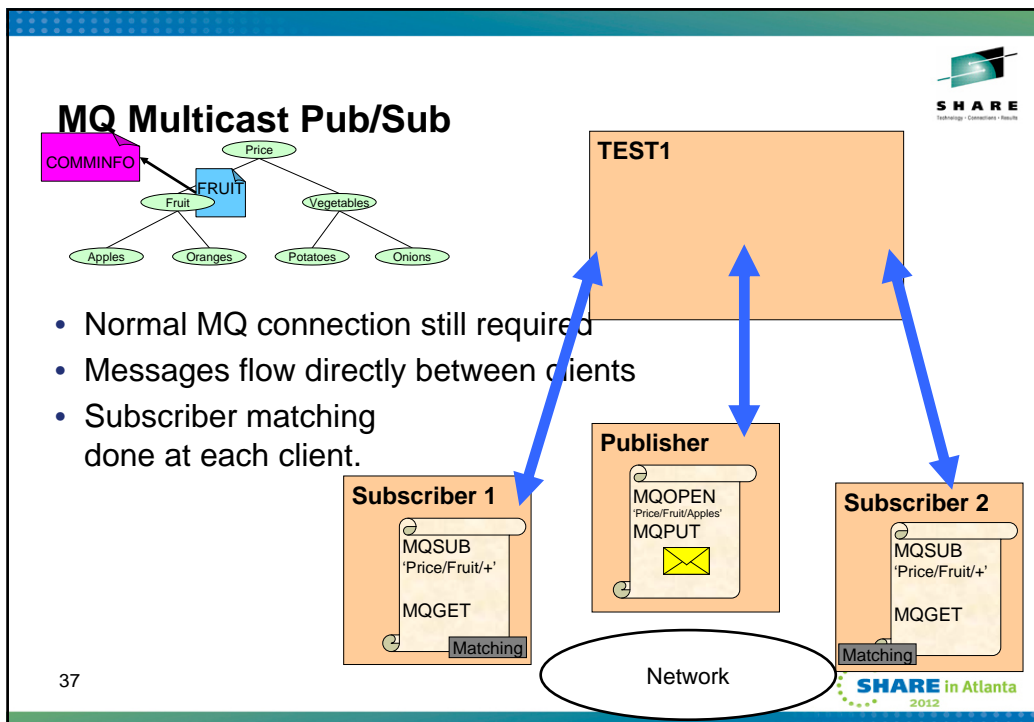



35

Current MQ Publish/Subscribe - Notes

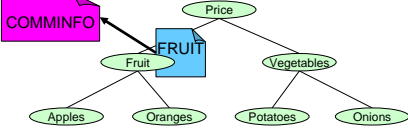
- N
O
T
E
S
- Using the MQ Publish/Subscribe feature in V7.0, publication messages are stored on queues for the subscriber to consume. Applications which publish messages send them to the queue manager, and applications which consume messages retrieve them from the queue manager.
 - The matching engine is in the queue manager. At publish time this is where the subscribers are identified and copies of the publication messages are sent to each subscriber's queue.

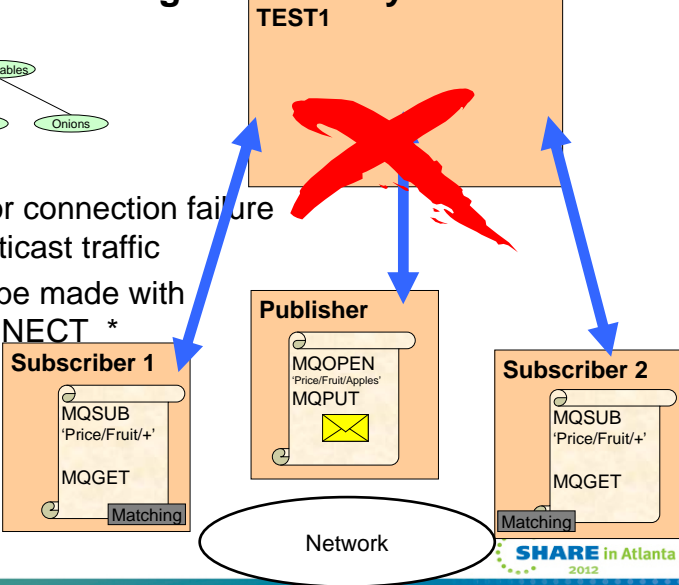
36






MQ Multicast Pub/Sub High availability






- Queue Manager or connection failure does not stop multicast traffic
- Connection must be made with MQCNO_RECONNECT *




39



MQ Multicast Pub/Sub High availability - Notes

N
O
T
E
S

- Using the multicast transport gives your applications high availability – specifically the ability to keep on sending and receiving publication messages even whilst the queue manager is unavailable.
- This is achieved because the messages are not going through the queue manager, and is enabled when using a reconnectable client, that is a client using one of the MQCNO_RECONNECT_* options. If connection with the Queue Manager is lost this option will cause the client to try to reconnect to the queue manager in the background. During this reconnect process multicast traffic is able to continue.



40

Changes to your application to use Multicast

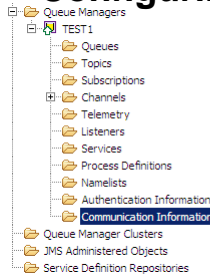
An application written in V7.0 using these criteria will not require any changes in V7.1 to use Multicast

- Is a threaded client application
- Uses managed handles
- Uses non-durable subscriptions
- Uses a topic string length 255 or less
- Uses re-connectable clients (for high availability) – V7.0.1 feature
- Doesn't use transactions
- Doesn't use persistent messages
- Doesn't use message grouping or segmentation

Changes to your application to use Multicast - Notes

- N
- O
- T
- E
- S
- You can write an application in WebSphere V7.0 that will not require any changes to run as a multicast application in WebSphere V7.1.
 - Of course not every application will meet these criteria, for example an application that is linked using local bindings and doing point-to-point transactional work. The criteria that are required are shown on this slide.
 - Search the MQ Information Centre for “Multicast and the Message Queue Interface” for more details.

Configuring Pub/Sub to use Multicast



- New MQ object – COMMINFO
 - Defines the behaviour of the multicast traffic

```
DEFINE COMMINFO
ALTER COMMINFO
DELETE COMMINFO
DISPLAY COMMINFO
```

- SYSTEM.BASE.TOPIC points to SYSTEM.DEFAULT.COMMINFO.MULTICAST
 - Default group address is 239.0.0.0
 - Usable as soon as MCAST(ENABLED)
- Ensure topic strings length 255 or less

```
Command Prompt - runmqsc TEST1
Starting MOSC for queue manager TEST1.
DEFINE COMMINFO(MC) GRPADDR(239.0.0.0)

DISPLAY COMMINFO(MC)
AMQ8861: Display comminfo details.
COMMINFO(MC)                                TYPE(MULTICAST)
DESCR( )                                     ALTDTE(2011-09-27)
ALTTIME(18.39.50)                           BRIDGE(ENABLED)
CCSID(ASPUB)                                 COMMEV(DISABLED)
ENCODING(ASPUB)                             GRPADDR(239.0.0.0)
MCHBIT(2000)                                 MCPROP(ALL)
MONITOR(60)                                  MSGHST(100)
NSUBHST(NONE)                                PORT(1414)

ALTER TOPIC(FRUIT) COMMINFO(MC) MCAST(ENABLED)
```

Configuring Pub/Sub to use Multicast - Notes

NOTES

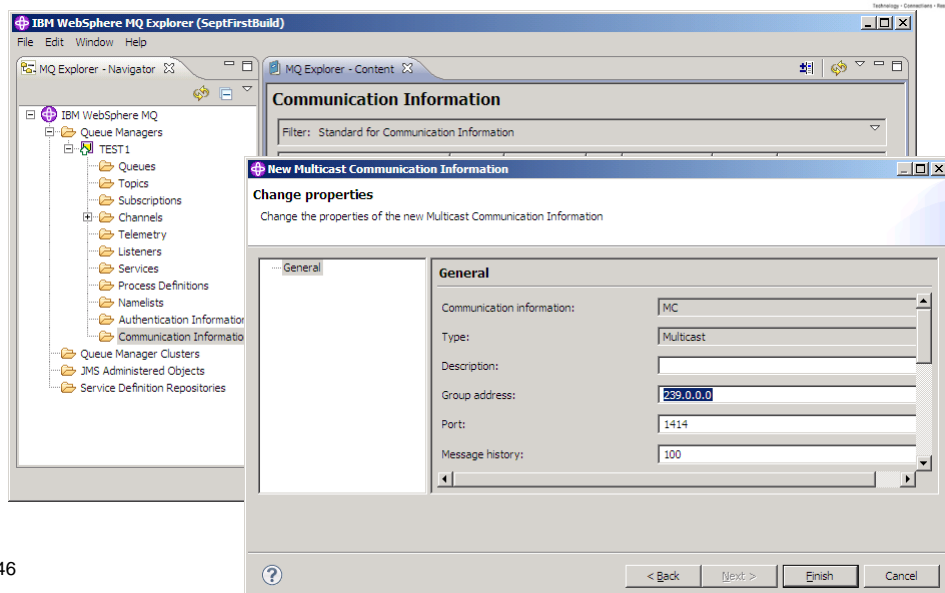
- In order to configure multicast there is a new MQ Object, the Communication Information (COMMINFO) object. This object allows the tuning of behaviour of the multicast traffic including the Group address; whether message properties are propagated; the reliability setting; attributes to control data conversion; communication Monitoring and queue manager bridging.
- The COMMINFO object is then referenced from the TOPIC object. By default MQ comes with the SYSTEM.DEFAULT.COMMINFO.MULTICAST object linked to the SYSTEM.BASE.TOPIC – so the whole topic tree is ready to use multicast on the default group address (239.0.0.0) – all that needs to be done to enable it is to alter the topic object you want to use multicast with to MCAST(ENABLED). This is useful for demos and playing with multicast initially, but unless you really want the whole topic tree to use the same group address and all the same settings, you would instead make your own COMMINFO objects, possibly using a different group address for each.
- Another point to bear in mind is that the topic string used with multicast is limited at 255 characters, so ensure to topics you define stay below this length limitation.

- DEFINE COMMINFO(MC) GRPADDR(239.0.0.0)

- DISPLAY COMMINFO(MC)
AMQ8861: Display comminfo details.
- COMMINFO(MC) TYPE(MULTICAST)
- DESCR() ALTDATA(2011-09-27)
- ALTTIME(18.39.50) BRIDGE(ENABLED)
- CCSID(AS PUB) COMMEV(DISABLED)
- ENCODING(AS PUB) GRPADDR(239.0.0.0)
- MCHBINT(2000) MCPROP(ALL)
- MONINT(60) MSGHIST(100)
- NSUBHIST(NONE) PORT(1414)

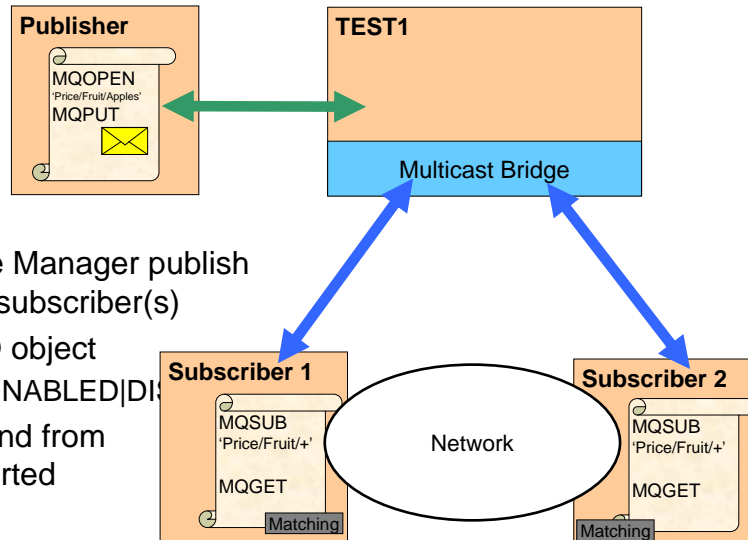
- ALTER TOPIC(FRUIT) COMMINFO(MC) MCAST(ENABLED)

45



46

Bridging from Queue Manager to Multicast



- From Queue Manager publish to Multicast subscriber(s)
- COMMINFO object
 - BRIDGE(ENABLED|DISABLED)
- Only outbound from QMgr supported

47

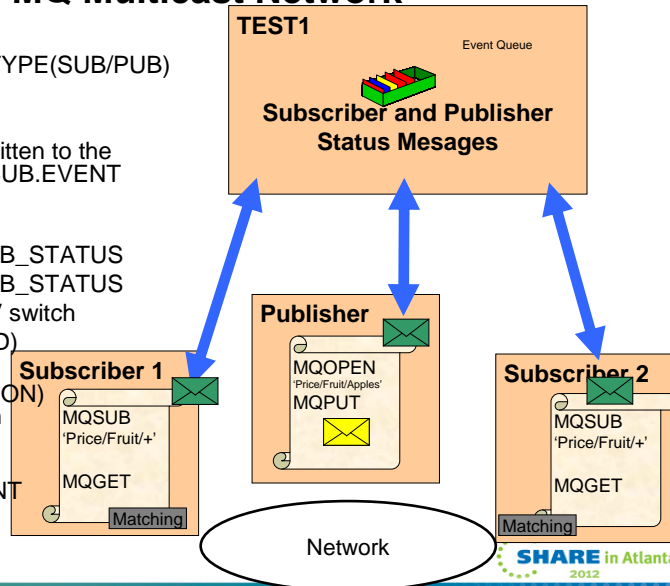
Bridging from Queue Manager to Multicast - Notes

- N**
- Publications made by applications that are not using multicast can be bridged to multicast subscribers by a component in the queue manager.
- O**
- Bridging is enabled on the COMMINFO object using the BRIDGE field.
- T**
- When an application publishes on a topic whose COMMINFO object is configured as BRIDGE(ENABLED) then these messages are published out on the group address for that TOPIC/COMMINFO.
 - Bridging is only available outbound from the queue manager.
- E**
- Bear in mind the multicast topic string length limitation of 255 characters when creating a topic to be bridged between MQ publishers and multicast subscribers.
- S**

48

Monitoring your MQ Multicast Network

- DISPLAY TPSTATUS TYPE(SUB/PUB)
 - All the usual status
 - Plus MCASTREL
- More detail in events written to the SYSTEM.ADMIN.PUBSUB.EVENT queue
 - Two event types
 - MQRC_MCAST_PUB_STATUS
 - MQRC_MCAST_SUB_STATUS
- Controlled by COMMEV switch
 - COMMEV(ENABLED) for all events
 - COMMEV(EXCEPTION) for events only when MCASTREL is 90% or less.
 - Events every MONINT



49

Monitoring your MQ Multicast Network - Notes

- N**
- WebSphere MQ V7.0 introduced the DISPLAY TPSTATUS command which shows status for the subscribers and publishers using a topic. Multicast publishers and subscribers provide this status information but in addition have one extra field called MCASTREL which is a pair of short and long term indicators showing the reliability (as a percentage) of the multicast transport. A value of 100% shows that all messages are being delivered successfully. A lower value indicates some issue with the transport that means some messages are not being delivered.
- O**
- To receive more detailed information about the problem you can turn on events which provide more detail. This is another field on the COMMINFO object, called COMMEV. This allows you to turn on events all the time, or just when a problem is seen. COMMEV(EXCEPTION) will only write events when MCASTREL indicates reliability is 90% or less. Events will be written at an interval defined by the MONINT field on the COMMINFO object.
- T**
- The information that is used by the queue manager to display the reliability of the multicast transport and the other status fields, periodically is sent from the clients (as the messages being monitored do not go through the queue manager). Clearly if the queue manager is not available to the clients and they are in the process of reconnecting, this information may be out of date.
- E**
- S**

50

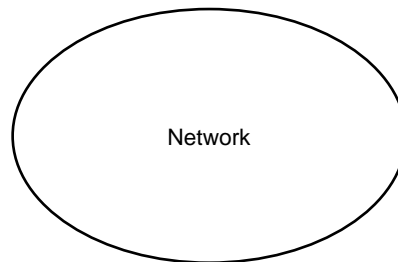
MQ Multicast interoperability with MQ LLM

WebSphere MQ

```
MQPUT
Data: "Hello World"
```

WebSphere MQ LLM

```
rmmRxInit
rmmRxJoinMulticastGroup
rmmRxCreateTopic
Data: "Hello World"
```



51

MQ Multicast interoperability with MQ LLM

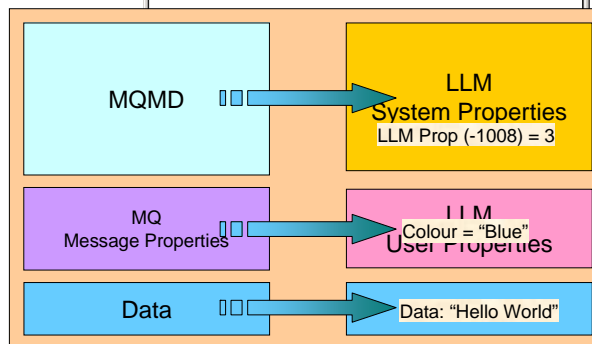
WebSphere MQ

```
MQPUT
MQMD.Priority = 3
MQMD.Colour = "Blue"
Data: "Hello World"
```

```
Command Prompt - runmqsc TEST1
Starting MQSC for queue manager TEST1.
ALTER COMMINFO(MC) MCPROP(ALL)
```

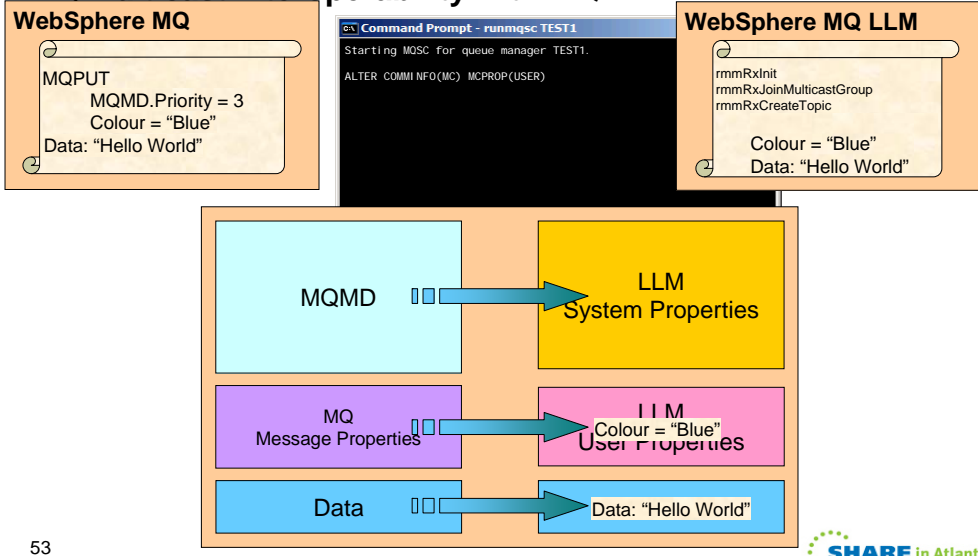
WebSphere MQ LLM

```
rmmRxInit
rmmRxJoinMulticastGroup
rmmRxCreateTopic
LLM Prop (1008) = 3
LLM Colour = "Blue"
Data: "Hello World"
```

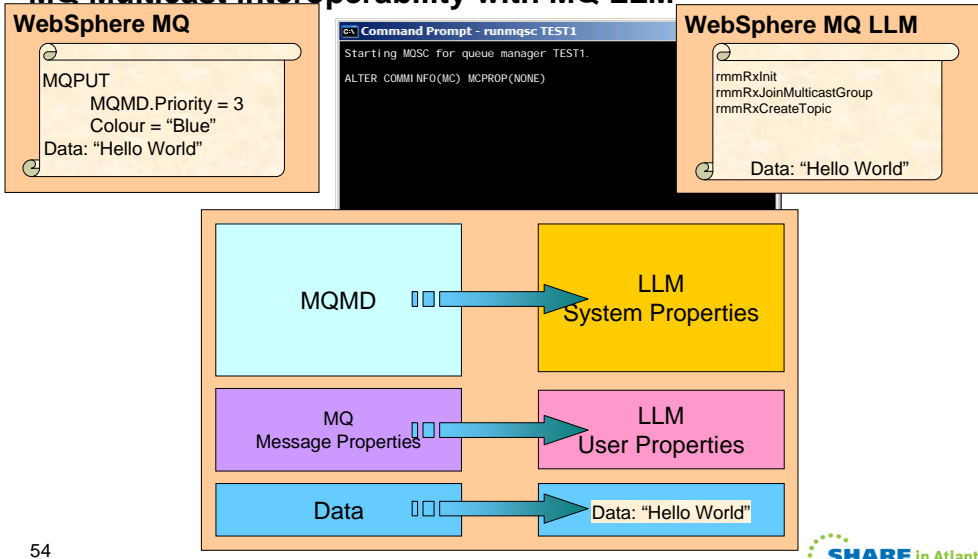


52

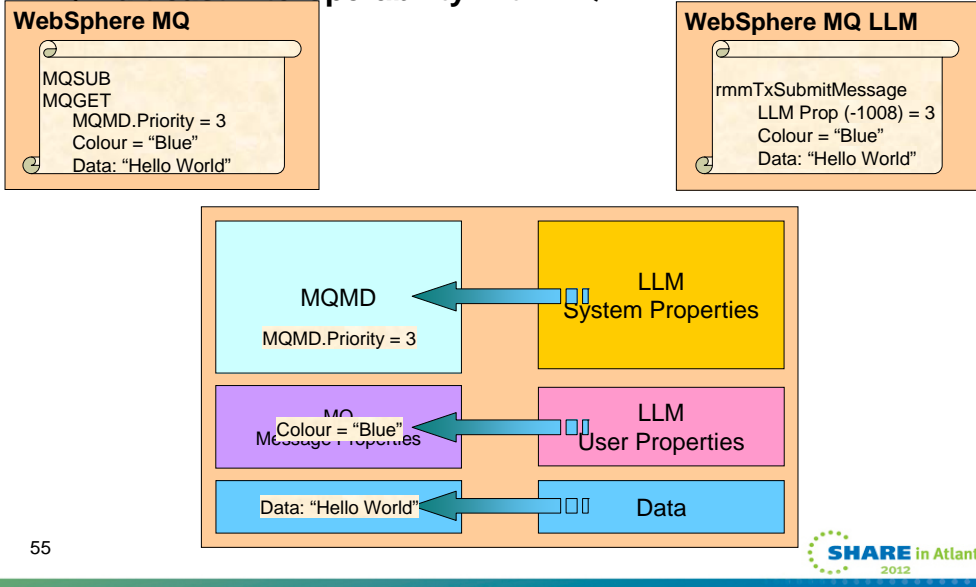
MQ Multicast interoperability with MQ LLM



MQ Multicast interoperability with MQ LLM



MQ Multicast interoperability with MQ LLM



MQ Multicast interoperability with MQ LLM - Notes

- N**
- The underlying technology in WebSphere MQ V7.1 multicast support and in WebSphere MQ Low Latency Messaging (LLM) is the same. Messages published by one can be subscribed by the other and vice versa. WebSphere MQ messages of course have a different look and feel to those published over LLM, so there are a few considerations. Additionally it is worth noting that a publisher using either interface is unaware of the subscribers and whether they are MQ or native LLM applications.
- O**
- The message descriptor (MQMD) is quite a large structure (428 bytes) and often much of this structure is the same for every message. Avoiding sending most/all of the MQMD reduces the size of the message which improves the speed, and additionally means an LLM native application is not presented with information it might not be expecting. MQMD fields can be sent however, and when they are they become LLM properties. All fields in the MQMD are defined as reserved LLM system properties.
- T**
- Only those parts of the MQMD that are different from a defined default set for MQ Multicast are sent (search MQ Info Centre for "Controlling the size of multicast messages"). When fields of the MQMD are sent, they are given to LLM as properties. You can control the fields that are sent using the COMMINFO field MCPROP.
- E**
- When receiving a message from a native LLM application, normally no MQMD fields will be there, so this defined default set also provides a way of constructing an MQMD when presenting the message to an MQ application. Of course, since the MQMD fields are defined as reserved LLM system properties, a native LLM application can set the appropriate property and populate an MQMD field in an MQ application (search MQ Info Centre for "Multicast interoperability with WebSphere MQ Low Latency Messaging" for the LLM system property values).
- S**

56

Security

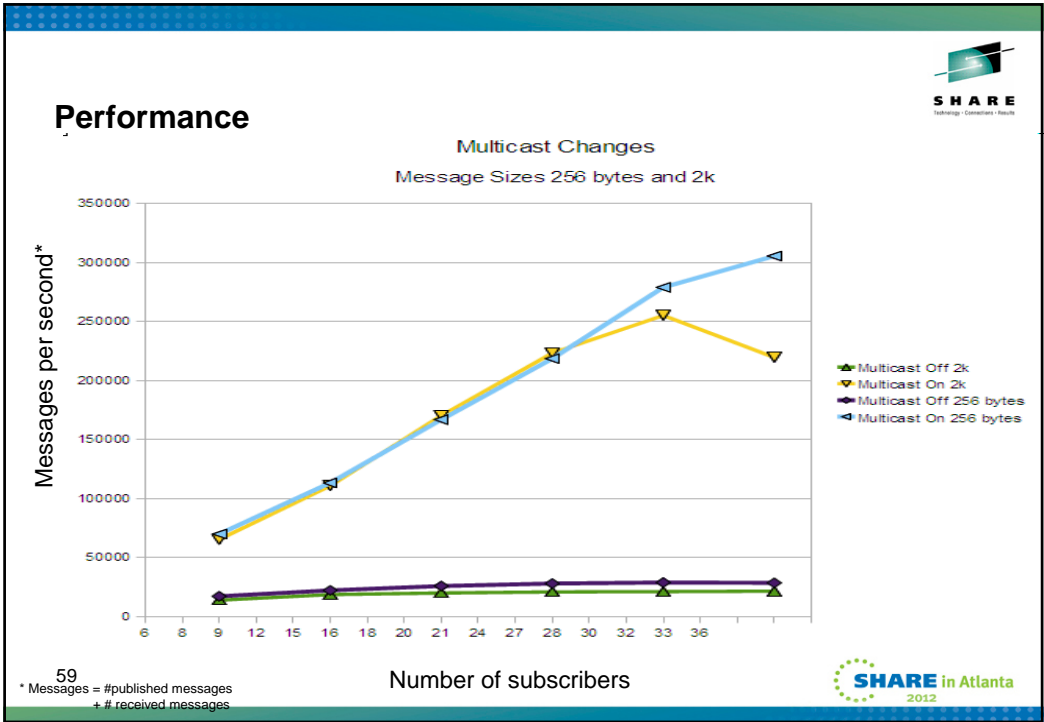
- MQ Security
 - MQOPEN
 - MQSUB
 - Controlled on the topic object
 - Doesn't stop native LLM applications from transmitting/receiving using same group address
- Physical Network Security
 - Controls who can use the subnet
- Mixed MQ and LLM applications
 - MQ API Exit on clients

57


Security – Notes

- N**
- Applications consuming the messages sent on a group address don't have to be MQ applications – as we have just discussed with LLM interoperability. Therefore securing access to the MQ topic object for subscribe and publish is not enough.
- O**
- It is expected that many multicast applications will run within a single subnet. Control on who can use that subnet can be restricted by network controls and further security is often not required.
- T**
- However, if applications are more widely spread than a small network domain, security needs to be on the data that is being sent, and to this end, you could use an API exit to encrypt the data before sending it on the group address, and only recipients with an appropriate decrypting API exit could then see the data. Equally, if you need to be sure that the data is sent from a trusted source, an API exit which signs the data, and an appropriate API exit which only accepts messages that are signed thus would allow the detection of rogue messages inserted by an unauthorized application. To this end, MQ V7.1 has added support for client side API exits.
- E**
- S**

58




Performance - Notes



N
O
T
E
S

- Here is a brief look at the sort of difference using multicast can make when using it for a large fan-out to bigger numbers of subscribers.
- In short, the more subscribers you have, the more messages can be sent per second as the publishing rate is independent of the number of subscribers.
- Full details will be published in the Performance Reports for WebSphere MQ V7.1.

60



Availability

- Clients and Server both must be WebSphere MQ V7.1
- Clients
 - 'C' Client only
 - Not Java
 - Not .Net
- Platforms
 - Unix and Windows
 - Not z/OS
 - Not IBM i
- WebSphere MQ LLM Interoperability
 - Need MQ LLM V2.4 minimum

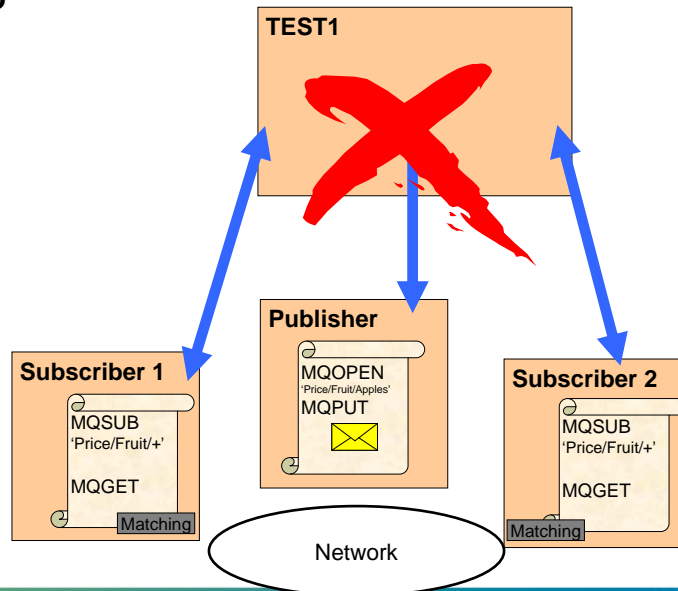
61

Availability – Notes

- N
O
T
E
S
- Multicast is available for use with the 'C' Client, that is the client implementation which underlies the procedural MQI client applications. It is not currently available in the Java Client or the .Net Client implementations.
 - Multicast is not available on z/OS or IBM i platforms.
 - If you wish to interoperate between WebSphere MQ V7.1 multicast client applications and native LLM applications you need a minimum of MQ LLM V2.4. You do not need any version of MQ LLM installed to just use MQ multicast however.

62

Demo



63

Demo - Notes

- N**
- We are going to show a demonstration of the scenario we talked about earlier where we can continue to publish messages to multicast subscribers even while the queue manager is down. We will use a very simple command line application from SupportPac MA01 - 'Q'. The commands for which are provided here for you to try out at home.
- O**
- Make sure you have MULTICAST enabled in the SYSTEM.BASE.TOPIC
 - Subscriber (as many as you want) – the demo will have two subscribers


```
q -m TEST1 -l mqic32 -xr -Scs:"Price/Fruit/+" -w60
```

 - which means connect to queue manager TEST1 using mqic32 library (client) and a re-connectable option; create a subscription to the topic string "Price/Fruit/+"; and wait for 60 seconds.
- T**
- Publisher


```
q -m TEST1 -l mqic32 -xr -Ts:"Price/Fruit/Apples" -M#100/-150/1000
```

 - which means connect to the queue manager TEST1 using mqic32 library (client) and a re-connectable option; open and put messages to the topic string "Price/Fruit/Apples"; there will be 100 messages, ranging in size from 1 to 150 bytes in length which will be put at intervals of 1000 ms.
- E**
- Once the publisher and subscribers are running we will end the queue manager using


```
endmqm -r -w TEST1
```

 - which tells the queue manager that although this is a graceful command driven way of stopping the queue manager, we should tell the clients to reconnect. If of course we had a failure, perhaps of the network connection, reconnection would also take place.
- S**

64

Summary – Pub/Sub in MQ V7.1 using Multicast

- Programming
 - Use managed handles
 - Use non-durable subscriptions
 - Use re-connectable clients for high availability
- Configuration
 - Enable on TOPIC object
 - Detailed tuning on COMMINFO object
- Other Considerations
 - Bridging
 - Monitoring
 - Interoperation with MQ LLM
 - Security

65

Summary - Notes

- N
- So today we have seen that writing an MQ application to use multicast requires no changes over and above the managed handles that could be used in WebSphere MQ V7.0.
- O
- Changes to enable multicast are all made by making administrative changes to the TOPIC object and tuning can be done with the new COMMINFO object.
- T
- Multicast is a new transport in MQ and we looked at some of the other considerations as well.
- E
- S

66


This was session WMQ 101 - The rest of the week

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00			Free MQ! - MQ Clients and what you can do with them.	MQ Performance and Tuning on distributed	
09:30		The MQ API for dummies - the basics	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation	The even darker arts of SMF	CICS Programs Using WMQ V7 Verbs
11:00		Putting the web into WebSphere MQ: A look at Web 2.0 technologies The Doctor is in. Hands-on Lab and Lots of Help with the MQ Family	Message Broker administration	The Do's and Don'ts of z/OS Queue Manager Performance	
12:15		WebSphere MQ: Highly scalable publish subscribe environments		MQ & DB2 - MQ Verbs in DB2 & Q-Replication	
13:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in WebSphere Message Broker V8.0	The Do's and Don'ts of Message Broker Performance	Diagnosing problems for MQ	
15:00	WebSphere Message Broker 101: The Swiss army knife for application integration	What's new in WebSphere MQ V7.1	WebSphere MQ Security - with V7.1 updates	Diagnosing problems for Message Broker	
16:30	Introduction to the WebSphere MQ Product Family - including what's new in the family products	Under the hood of Message Broker on z/OS - WLM, SMF and more	MQ Java zero to hero	Shared Q including Shared Message Data Sets	
18:00			For your eyes only - WebSphere MQ Advanced Message Security	MQ Q-Box - Open Microphone to ask the experts questions	

67



Questions & Answers



As a reminder, please fill out a session evaluation

68

