IBM Software Group

A Day in the Life of a WebSphere MQ Transmission Queue

Angel Rivera (rivera@us.ibm.com)
WebSphere MQ Unix® Level 2 Support
26 April 2011

WebSphere® Support Technical Exchange

ON DEMAND BUSINESS™

Agenda

- Overview of distributed messaging
- Components:
  - Remote Queue Definition
  - Transmission Queue
  - MCA
  - Sender/Receiver Channels
  - Heartbeat
- System Cluster Transmit Queue
- Dead Letter Queue

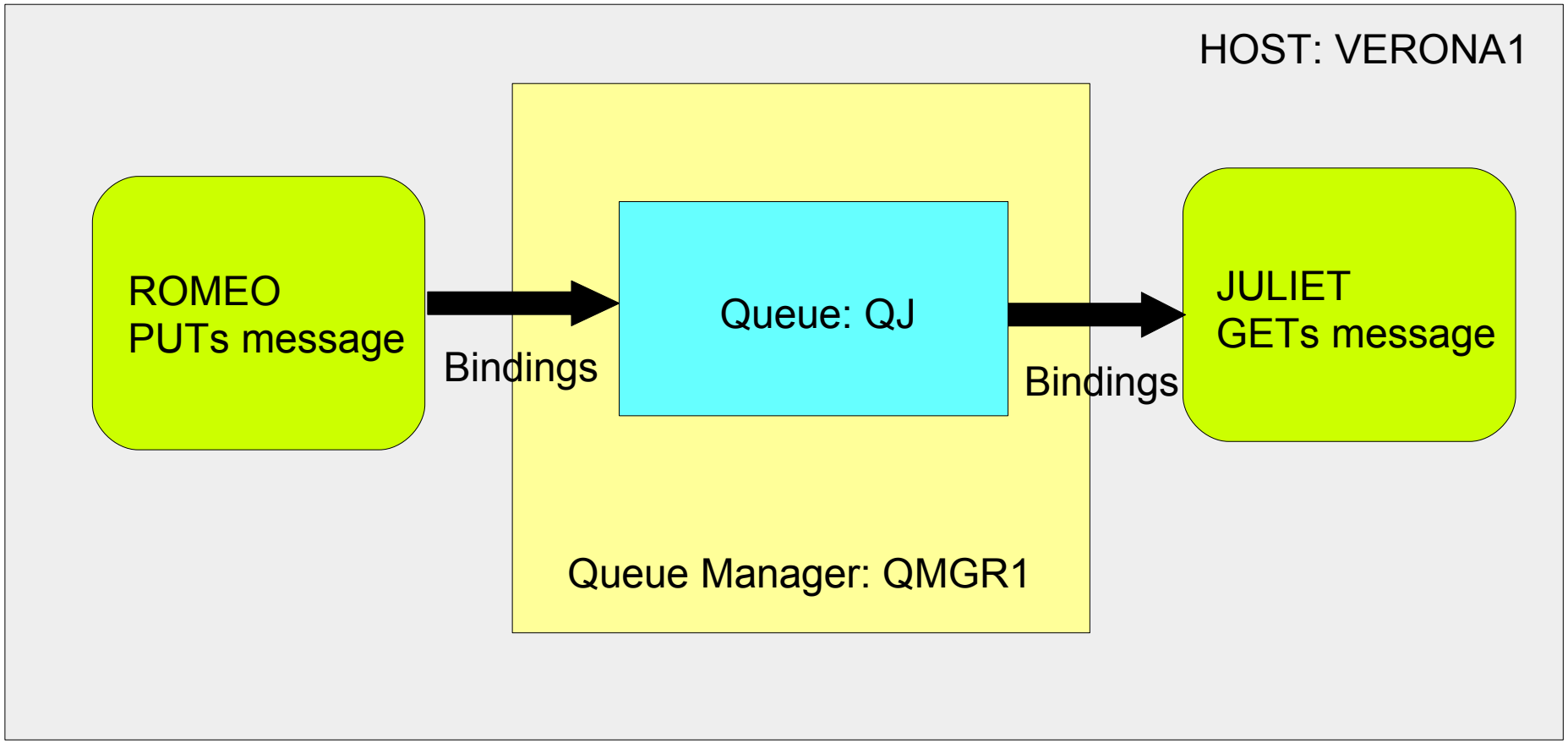
Having fun preparing for this presentation

- I will use a bit of “Romeo and Juliet” to illustrate some of the main points for this presentation.

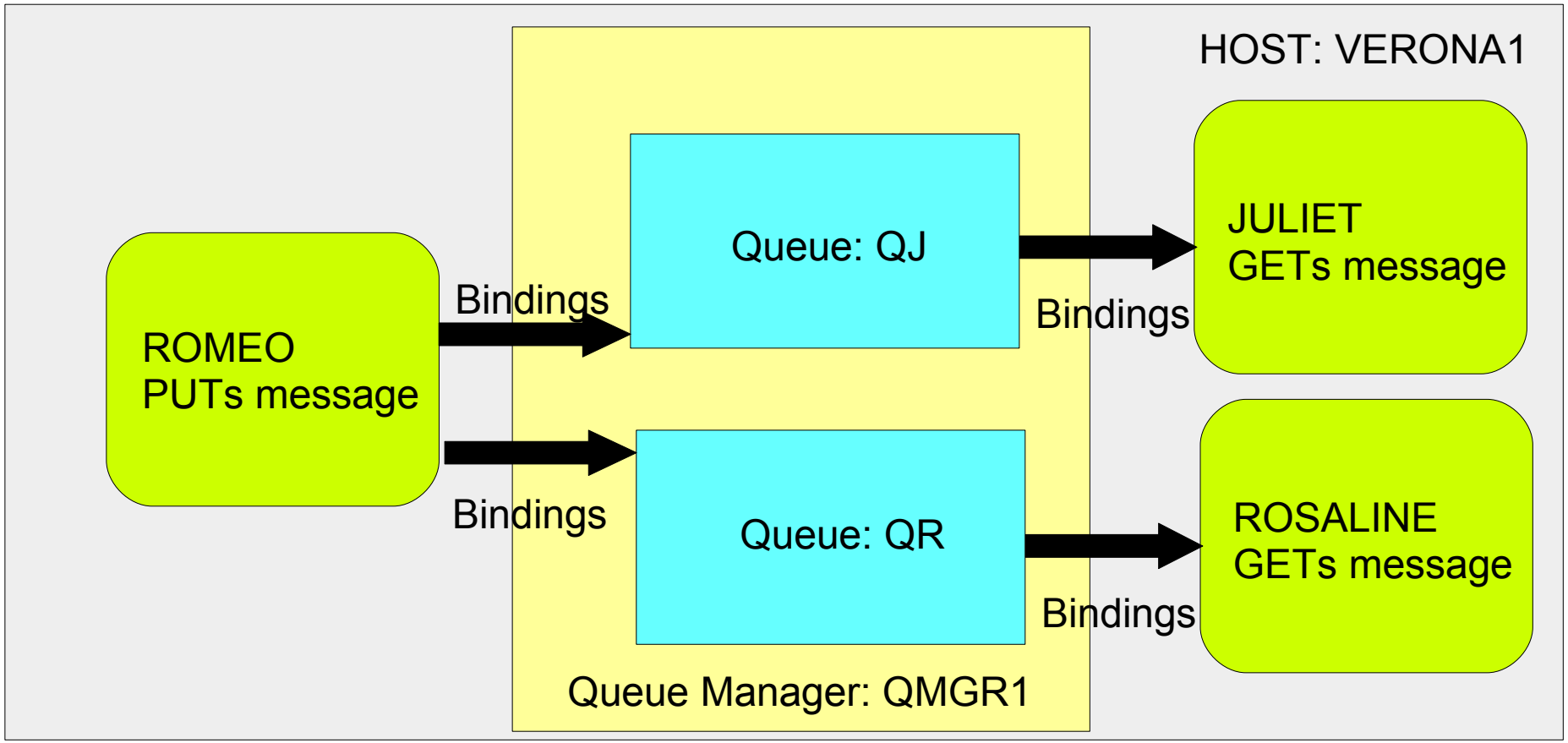
How to pronounce XMITQ

- This presentation uses the term XMITQ.
- Some novice MQ users do not know how to pronounce it. Is it ...?
- Zai-mit queue - Such as in Xylophone? ˈzaɪləfəʊn
- Ecs-mit queue - Such as in X-Ray, X-Men?
- Christ-mit queue - Such as in Xmas (Christmas)?
- Cross-mit queue - Such as in Xing (Crossing)?
- Answer:
  - **Trans-**mit Queue or Transmission Queue

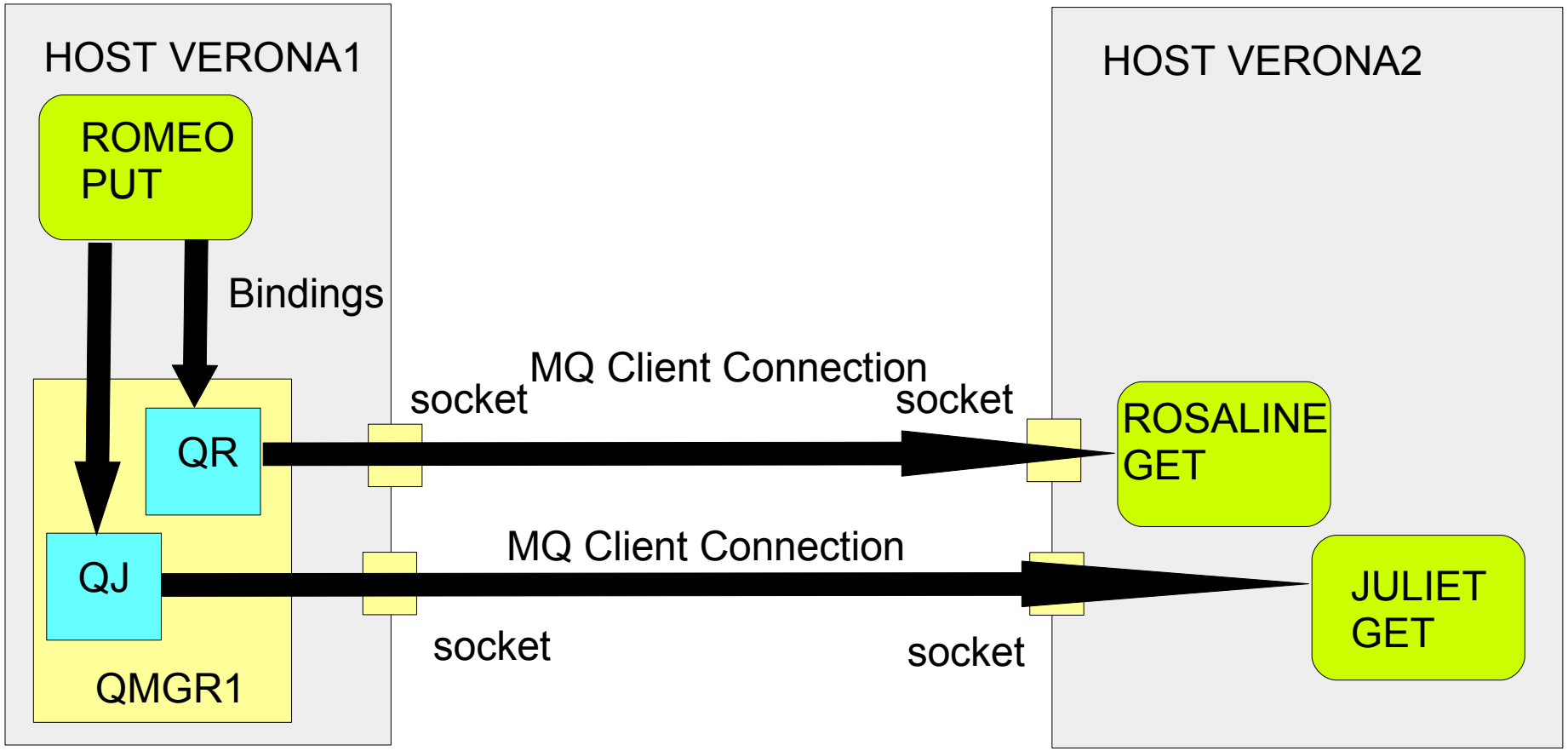
Simple Put/Get using Bindings



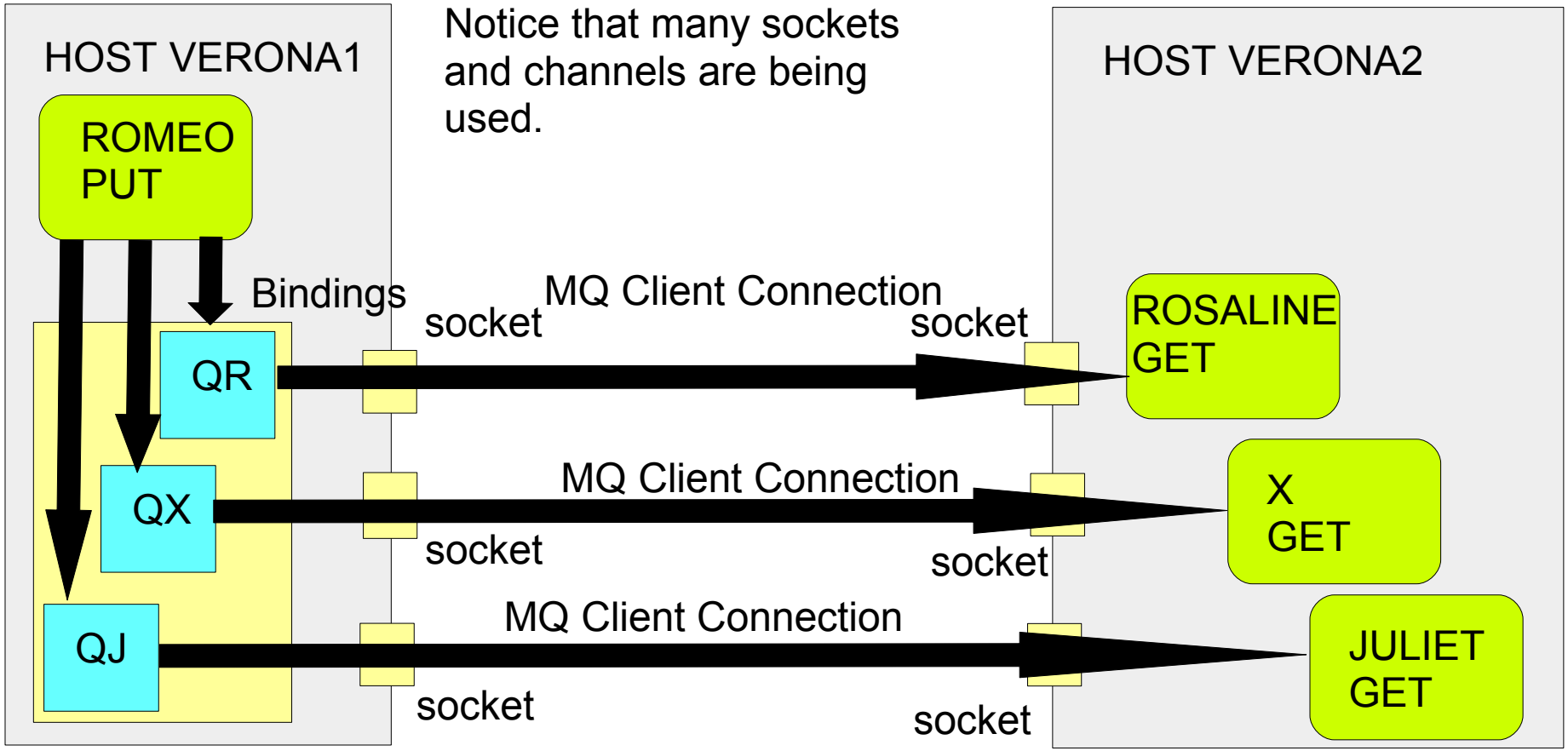
Multiple clients using Bindings



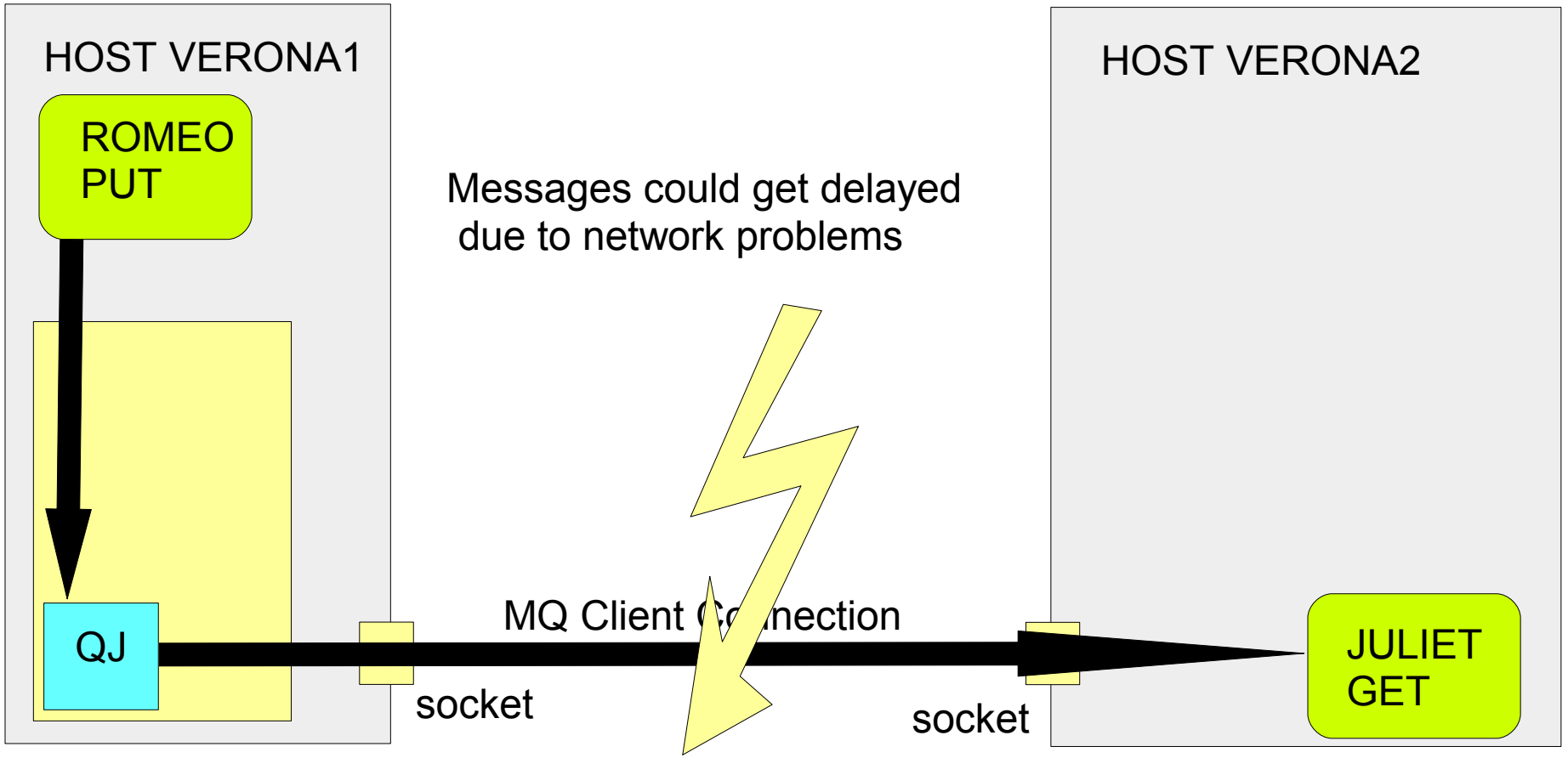
Multiple clients using Client Connection



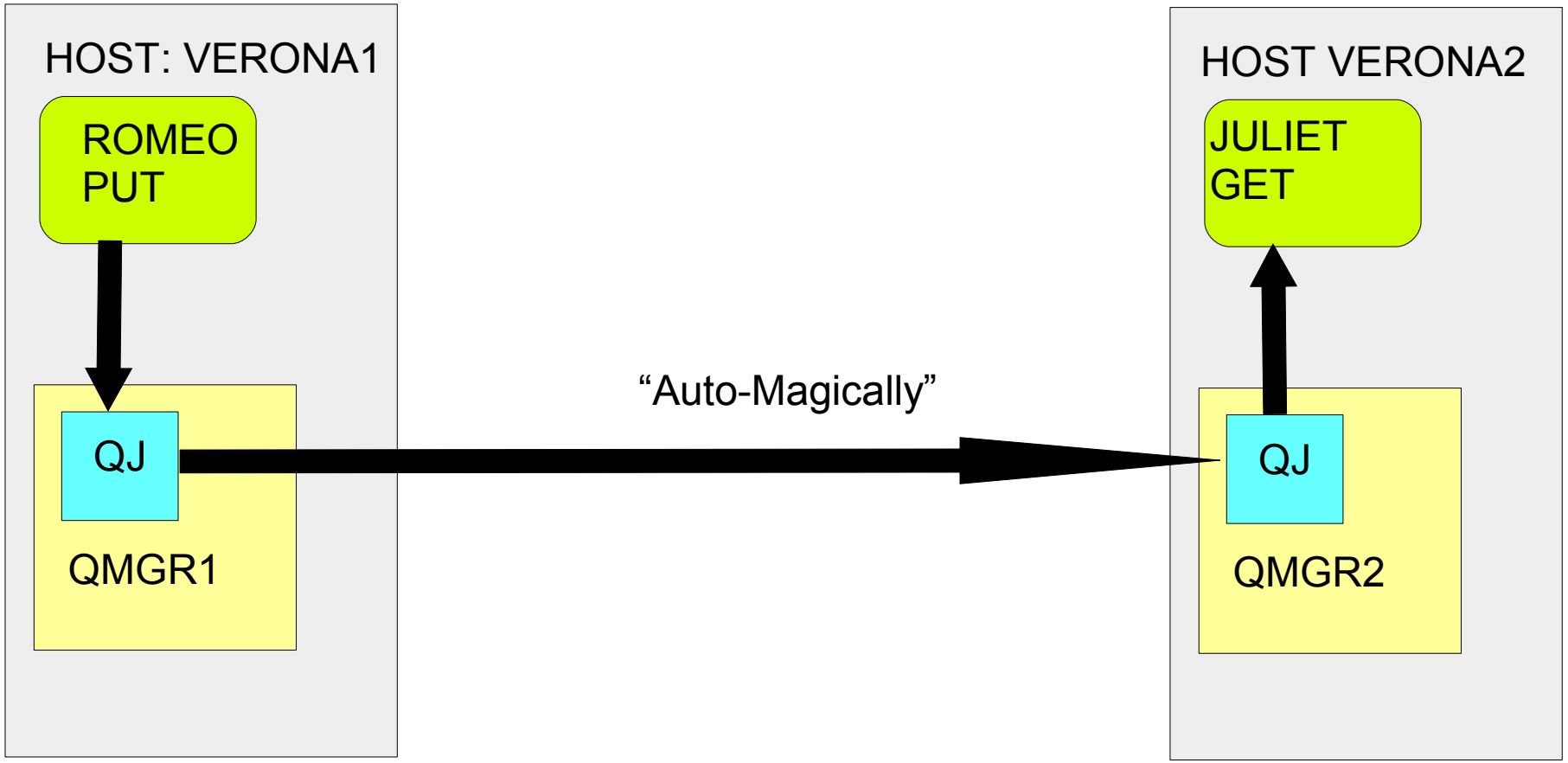
Many more clients connecting remotely



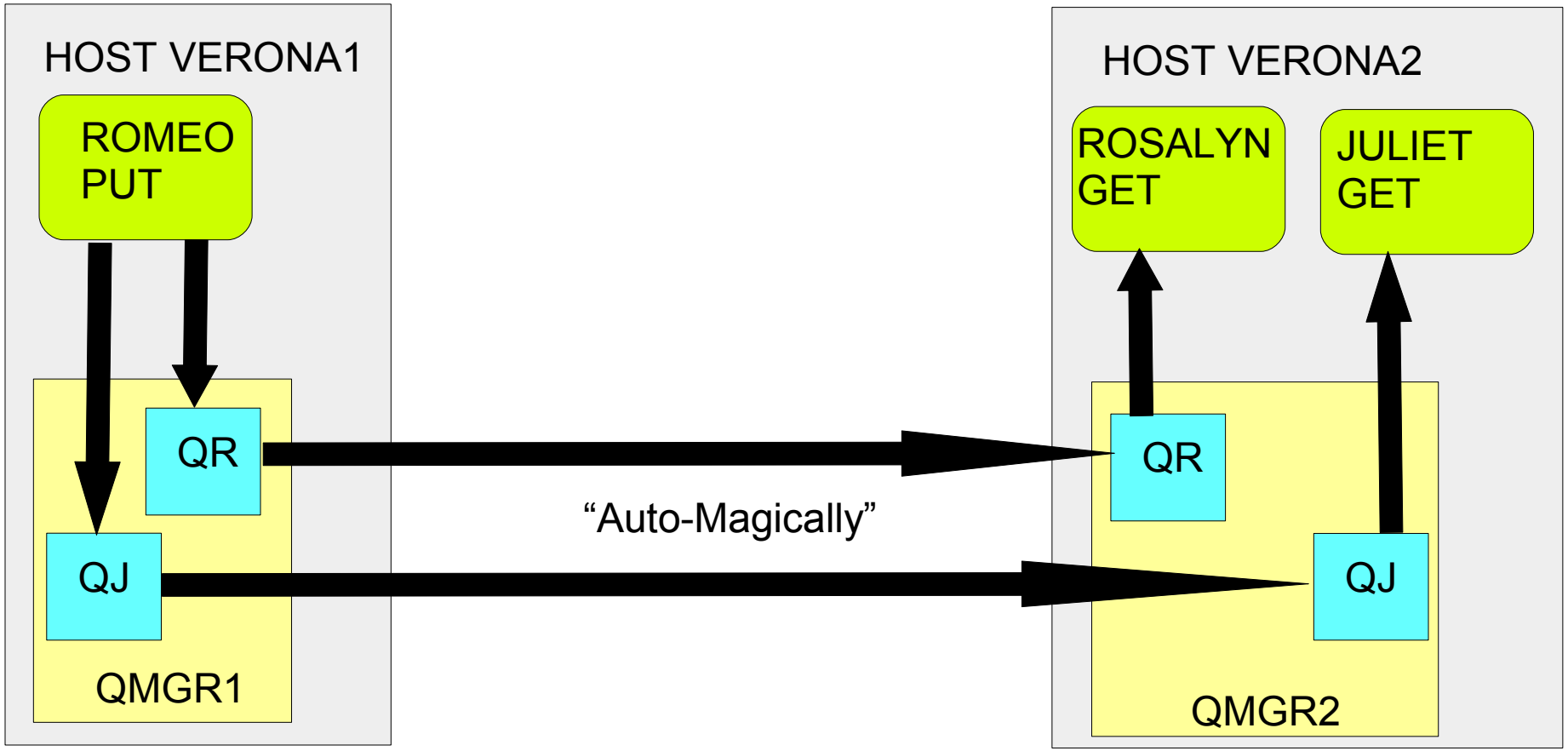
Network delays - Client Connection



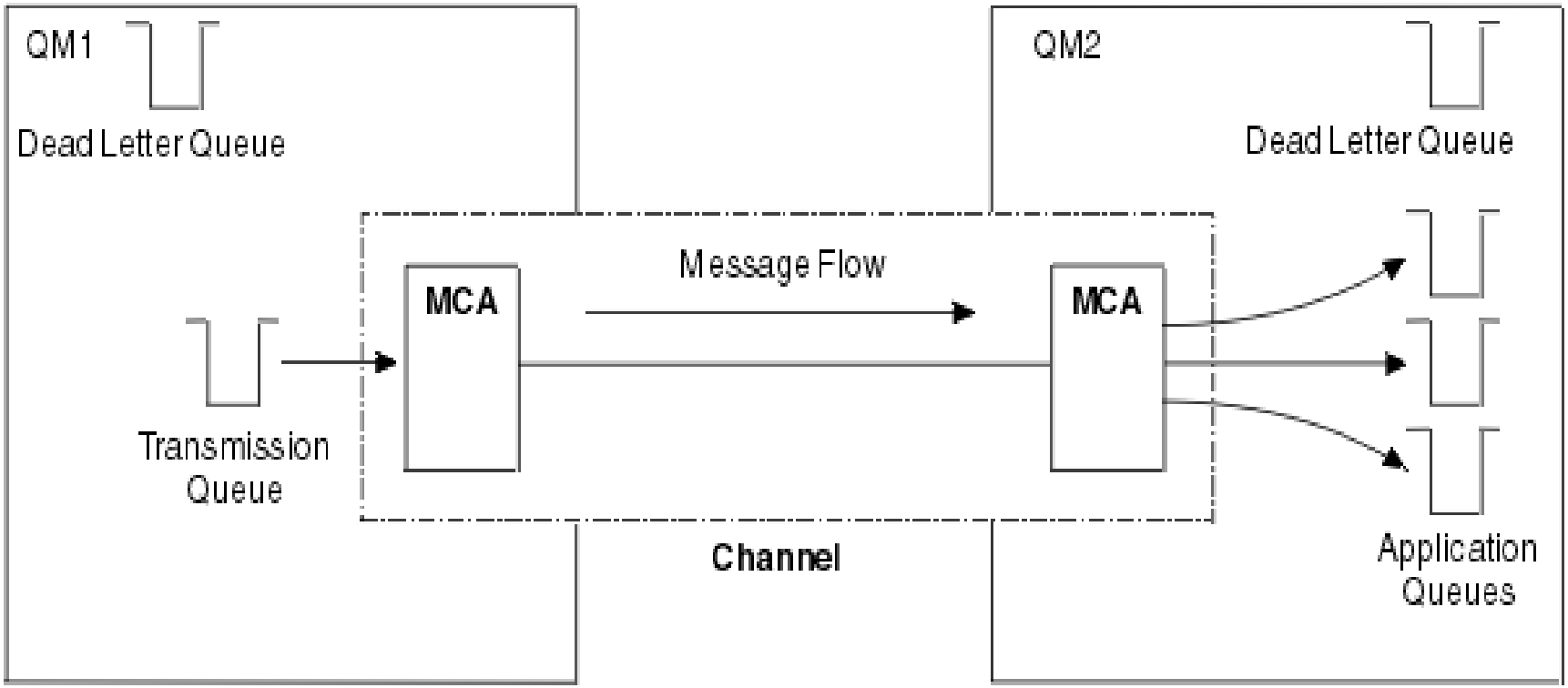
Preparing for distributed messaging - 1



Preparing for distributed messaging - 2



Sending messages



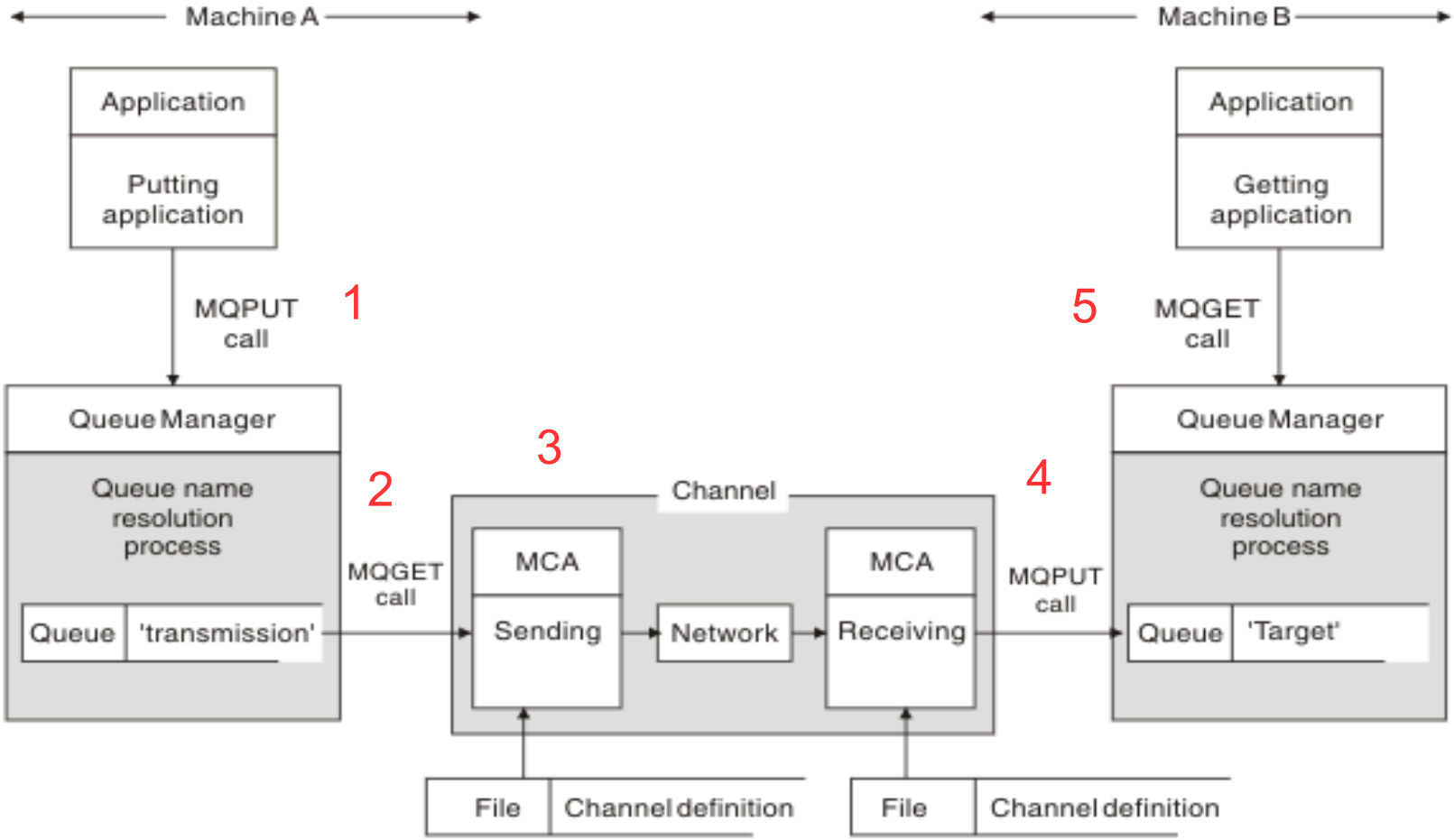
Notes: Sending messages

notes

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzae.doc/ic10270_.htm
MQ V7 Information Center: Components needed to send a message

- If a message is to be sent to a remote queue manager, the local queue manager needs definitions for a transmission queue and a channel.
- Each end of a channel has a separate definition, defining it, for example, as the sending end or the receiving end. A simple channel consists of a sender channel definition at the local queue manager and a receiver channel definition at the remote queue manager.
  - These two definitions must have the same name, and together constitute one channel.
- There is also a message channel agent (MCA) at each end of a channel.
- Each queue manager should have a dead-letter queue (also known as the undelivered message queue). Messages are put on this queue if they cannot be delivered to their destination.

Overview of distributed queuing



Notes: Overview of the components of distributed queuing

notes

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzae.doc/ic19690_.htm
Intercommunication Manual - Queue name resolution
.
In larger networks, the use of queue managers has a number of advantages over other forms of communication. These advantages derive from the name resolution function and the main benefits are:
• Applications do not need to make routing decisions nor know the network structure
• Network links are created by systems administrators
• Network structure is controlled by network planners
• Multiple channels can be used between nodes to partition traffic

Referring to Figure 1, the basic mechanism for putting messages on a remote queue, as far as the application is concerned, is the same as for putting messages on a local queue:
• The application putting the message issues MQOPEN and MQPUT calls to put messages on the target queue.
• The application getting the messages issues MQOPEN and MQGET calls to get the messages from the target queue.

Notes: Overview of the components of distributed queuing - 2

notes

If both applications are connected to the same queue manager then no inter-queue manager communication is required, and the target queue is described as local to both applications.

However, if the applications are connected to different queue managers, two MCAs and their associated network connection are involved in the transfer, as shown in the figure. In this case, the target queue is considered to be a remote queue to the putting application.

The sequence of events is as follows:

1. The putting application issues MQOPEN and MQPUT calls to put messages to the target queue.

2. During the MQOPEN call, the name resolution function detects that the target queue is not local, and decides which transmission queue is appropriate. Thereafter, on the MQPUT calls associated with the MQOPEN call, all messages are placed on this transmission queue.

Notes: Overview of the components of distributed queuing - 3

notes

3. The sending MCA gets the messages from the transmission queue and passes them to the receiving MCA at the remote computer.

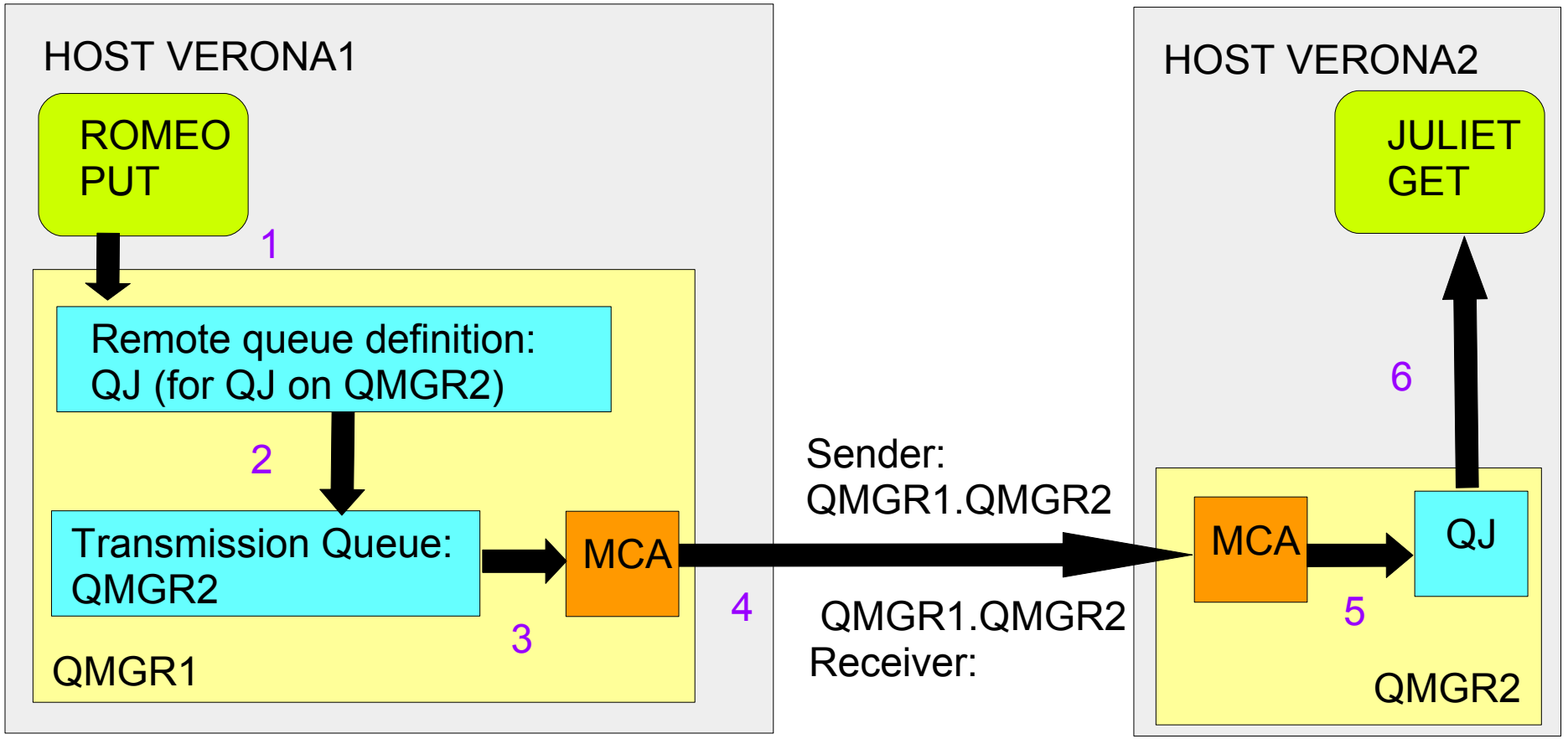
4. The receiving MCA puts the messages on the target queue, or queues.

5. The getting application issues MQOPEN and MQGET calls to get the messages from the target queue.

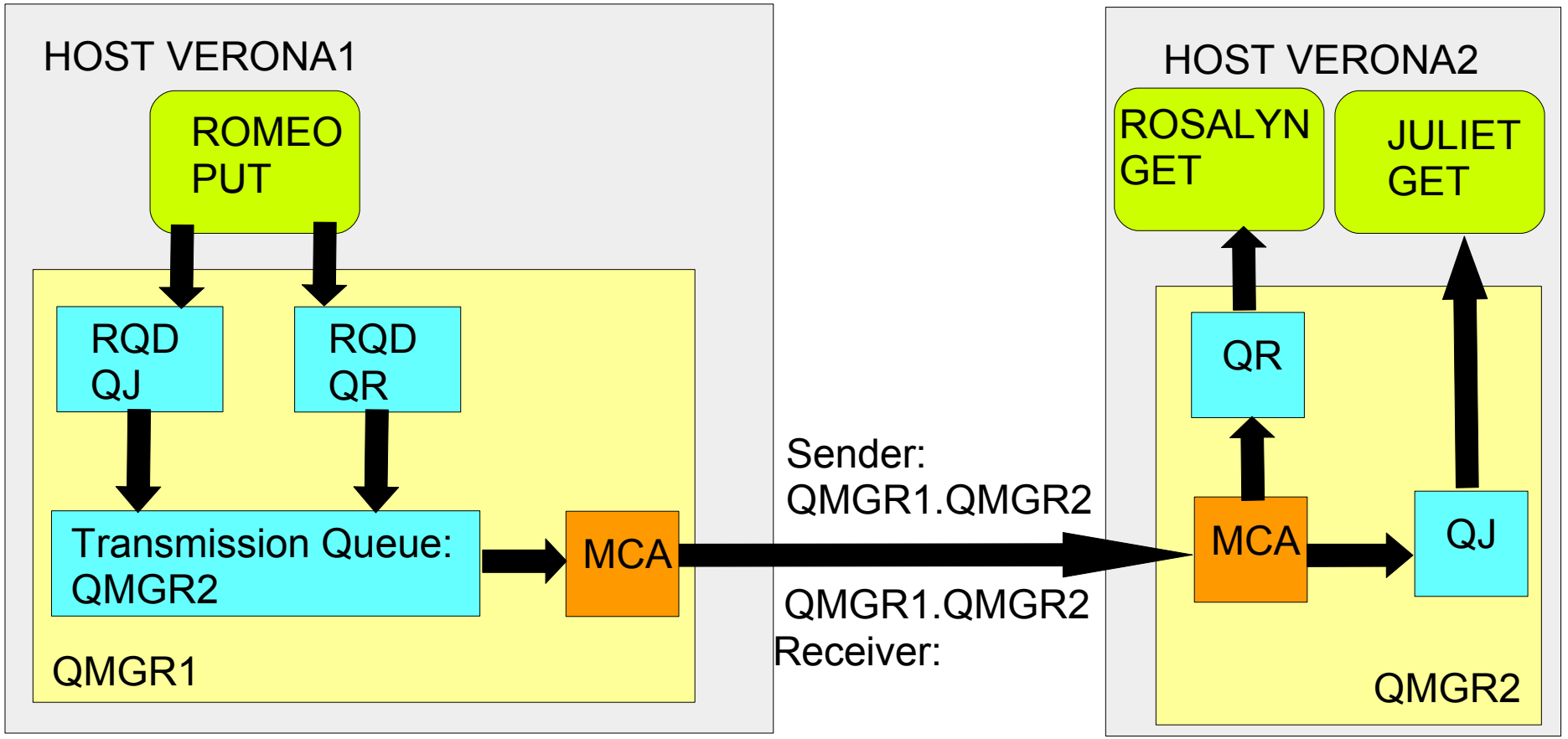
Note: Only step 1 and step 5 involve application code; steps 2 through 4 are performed by the local queue managers and the MCA programs. The putting application is unaware of the location of the target queue, which could be in the same processor, or in another processor on another continent.

The combination of sending MCA, the network connection, and the receiving MCA, is called a message channel, and is inherently a unidirectional device. Normally, it is necessary to move messages in both directions, and two channels are set up for this, one in each direction.

Sending 1 message via XMITQ-MCA

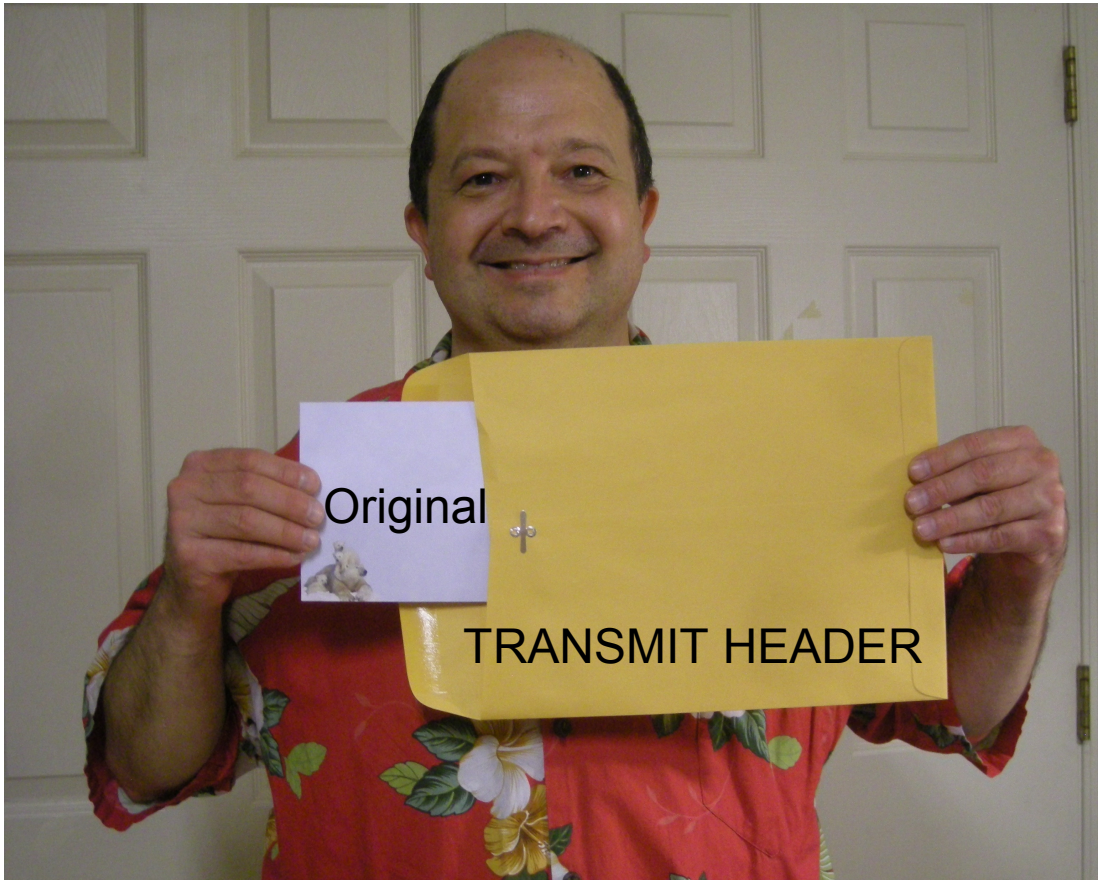


Sending 2 messages via same XMITQ-MCA



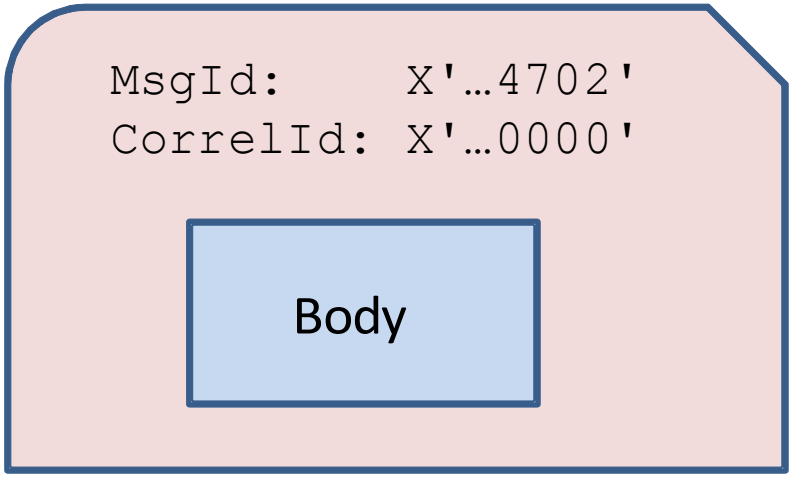
Using analogy of small and big envelope

- Hopefully this analogy of a small envelope inside a big envelope could be helpful to understand the usage of the Transmit Header
- This analogy does not use standard terminology for MQ.



Step 1, Message is placed in a queue

Step 1: Application puts a text message into a local queue or a remote queue definition
The queue manager obtains a logical envelope to put the message
The envelope has a MsgId assigned by the queue manager, such as: X'414D5120514D5F414E47454C49544F20B5493E4C2000**4702**'
The CorrelId is initially 0.
The text message is placed inside the logical envelope.



Step 2 If RQD message is moved to XMITQ

If it is a local queue, then no further processing on the envelope is needed and the envelope waits for an MQ client to pick the envelope.

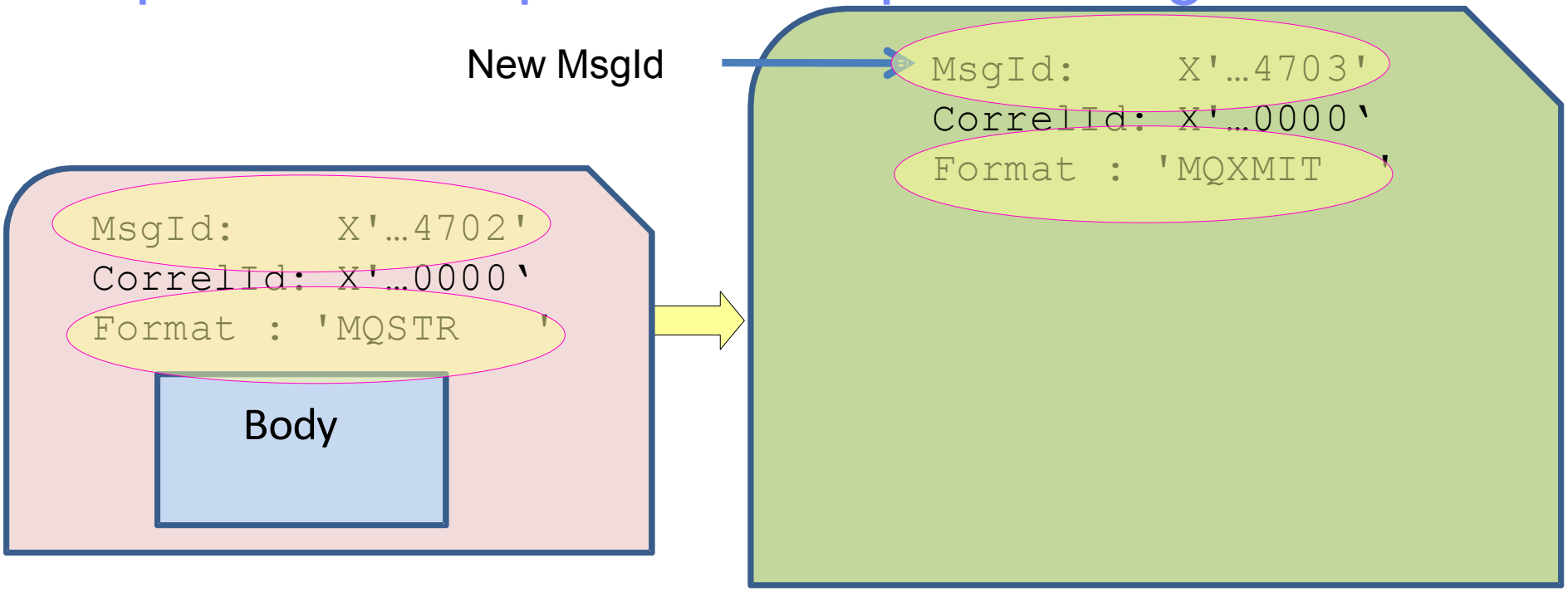
Step 2: If it is remote queue definition (RQD), then the message is quickly moved to a transmission queue.
The RQD does NOT store messages. You cannot browse or get messages from an RQD.

The transmission queue obtains a new (big) envelope, which has a transmission header:

Format : 'MQXMIT '

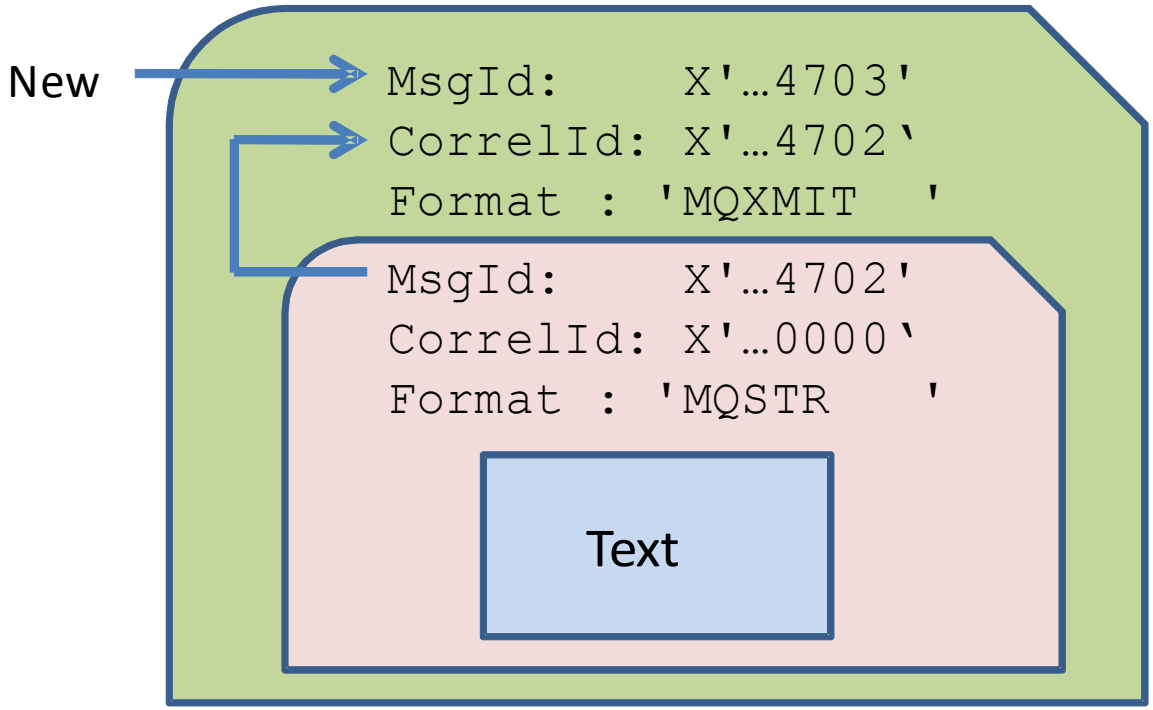
This header has a new MsgID and a CorrelId=0

Step 3: XMITQ puts envelope into big one



Notice that each new envelope has a new MsgId.
MsgId of the bigger envelope is different than the one from the smaller envelope
The small envelope has a format: MQSTR (MQ String)
The big envelope has a format: MQXMIT (MQ Transmission)

Step 4: CorrelId of big envelope is changed



The MsgId of the smaller envelope (x'...4702')
is used as the CorrelId of the larger envelope.

Notes: Real example of MQSTR header

notes

See the following IBM® Techdoc: 7021177
http://www.ibm.com/support/docview.wss?rs=171&uid=swg27021177
How the Message ID and Correlation ID in an MQ Message Descriptor are handled in a Transmission Queue
+ Example of small envelope, just by itself:
Sample: amqsbcg Queue QueueManager

```
.****Message descriptor****
  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 1208
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D5120514D5F414E47454C49544F20B5493E4C20001F02'
  CorrelId : X'0000000000000000000000000000000000000000000000000'
  BackoutCount : 0
…
****   Message      **** PAYLOAD
 length - 6 bytes
 00000000:  5465 7874 2D31                    'Text-1        '
****   END of Message      ****
```

Notes: Real example - small inside big

notes

+ Example of small envelope inside a bigger one:

BIGGER ENVELOPE (Notice the MQXMIT format and XQH "eye catcher")

```
****Message descriptor****
  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 437
   Format : 'MQXMIT  ' <== Notice the FORMAT
  Priority : 0  Persistence : 0
  MsgId :    X'414D5120514D5F414E47454C49544F20B5493E4C20004703'
  CorrelId : X'414D5120514D5F414E47454C49544F20B5493E4C20004702'
…
****   Message      ****
length - 434 bytes
00000000:  5851 4820 0100 0000 5132 2020 2020 2020 'XQH ....Q2       '
00000010:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000020:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000030:  2020 2020 2020 2020 514D 5F46 5231 2020 '        QM2  '
00000040:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
```

SMALLER ENVELOPE

```
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020 4D44 2020 0100 0000 '        MD  ....'
00000070:  0000 0000 0800 0000 FFFF FFFF 0000 0000 '........    ....'
00000080:  2202 0000 B804 0000 4D51 5354 5220 2020 '".......MQSTR   '
00000090:  0000 0000 0000 0000 414D 5120 514D 5F41 '........AMQ QM_A'
000000A0:  4E47 454C 4954 4F20 B549 3E4C 2000 4702 'NGELITO .I>L .G.'
000000B0:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
...
00000110:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000120:  2020 2020 2020 2020 2020 2020 7269 7665 '            rive'
00000130:  7261 2020 2020 2020 1601 0515 0000 00BA 'ra      ........'
00000140:  1E06 D260 3C35 1488 5EDB C2EF 0300 0000 '...`<5.ê^.......'
00000150:  0000 0000 0000 000B 2020 2020 2020 2020 '........        '
00000160:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000170:  2020 2020 2020 2020 0B00 0000 7265 204D '        ....re M'
00000180:  515C 6A61 7661 5C6A 7265 5C62 696E 5C6A 'Q\java\jre\bin\j'
00000190:  6176 6177 2E65 7865 3230 3130 3037 3134 'avaw.exe20100714'
000001A0:  3233 3538 3531 3933 2020 2020 5465 7874 '23585193    Text'
000001B0:  2D32                                   '-2              '
***   END of Message ***
```

Notes: Real sample – Big envelope

notes

+ Example of small envelope inside a bigger one

****Message descriptor**** BIGGER ENVELOPE (Format MQXMIT “and XQH "eye catcher")

```
  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 437
  Format : 'MQXMIT  '
  Priority : 0  Persistence : 0
  MsgId :    X'414D5120514D5F414E47454C49544F20B5493E4C2000 4703'
  CorrelId : X'414D5120514D5F414E47454C49544F20B5493E4C2000 4702'
…
****   Message      ****
length - 434 bytes
00000000:  5851 4820 0100 0000 5132 2020 2020 2020 'XQH ....Q2      '
00000010:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000020:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000030:  2020 2020 2020 2020 514D 5F46 5231 2020 '        QM2  '
00000040:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
```

Notes: Real sample – small envelope

notes

+ Example of small envelope inside a bigger one: (Continuation)

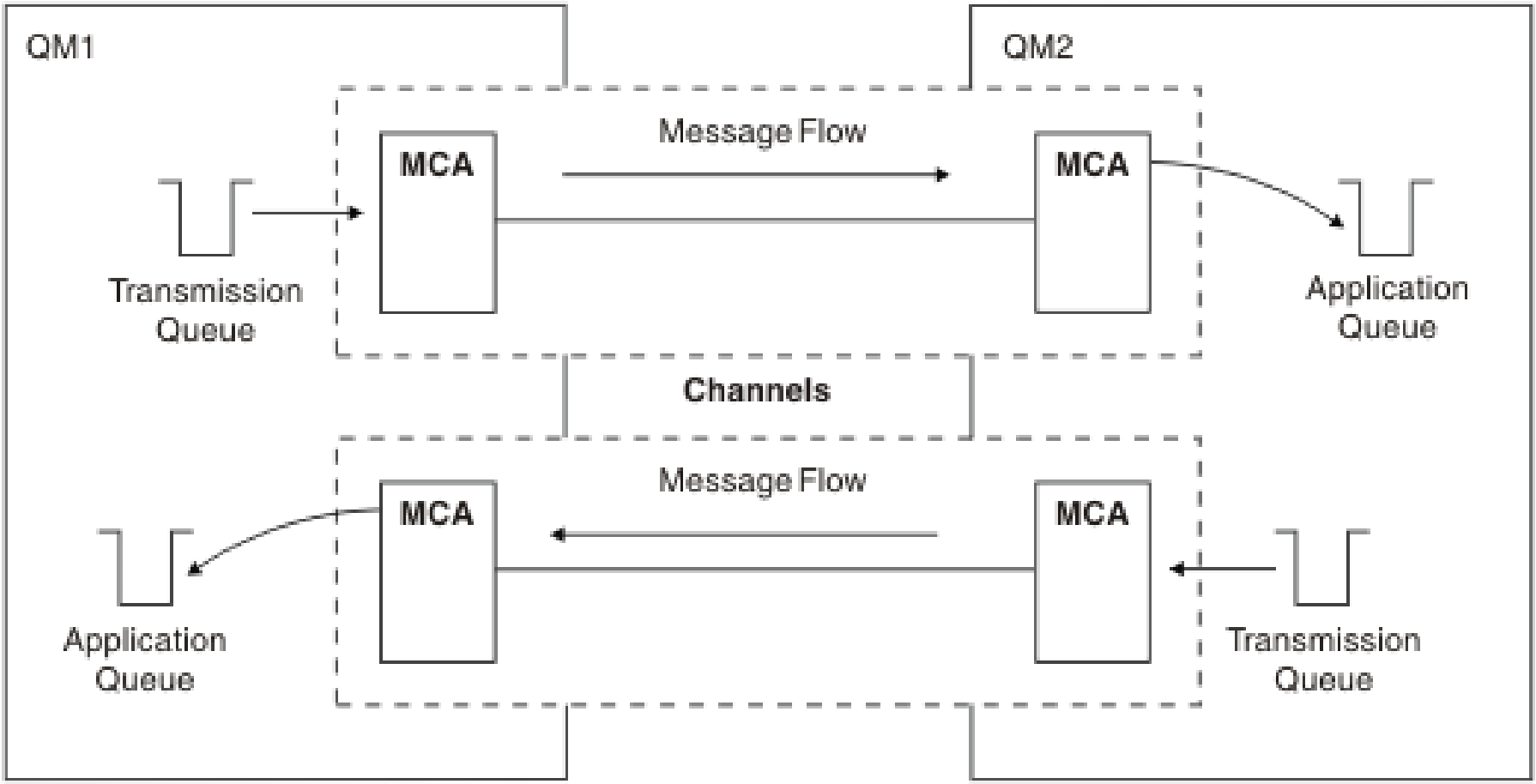
SMALLER ENVELOPE

```
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020 4D44 2020 0100 0000 '        MD  ....'
00000070:  0000 0000 0800 0000 FFFF FFFF 0000 0000 '........    ....'
00000080:  2202 0000 B804 0000 4D51 5354 5220 2020 '".......MQSTR   '
00000090:  0000 0000 0000 0000 414D 5120 514D 5F41 '........AMQ QM_A'
000000A0:  4E47 454C 4954 4F20 B549 3E4C 2000 4702 'NGELITO .I>L .G.' <= MessageId
000000B0:  0000 0000 0000 0000 0000 0000 0000 0000 '................'
000000C0:  0000 0000 0000 0000 0000 0000 2020 2020 '............    '
000000D0:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
000000E0:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
000000F0:  2020 2020 2020 2020 2020 2020 514D 5F41 '            QM_A'
00000100:  4E47 454C 4954 4F20 2020 2020 2020 2020 'NGELITO         '
00000110:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000120:  2020 2020 2020 2020 2020 2020 7269 7665 '            rive'
00000130:  7261 2020 2020 2020 1601 0515 0000 00BA 'ra      ........'
00000140:  1E06 D260 3C35 1488 5EDB C2EF 0300 0000 '...`<5.ê^.......'
00000150:  0000 0000 0000 000B 2020 2020 2020 2020 '........        '
00000160:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000170:  2020 2020 2020 2020 0B00 0000 7265 204D '        ....re M'
00000180:  515C 6A61 7661 5C6A 7265 5C62 696E 5C6A 'Q\java\jre\bin\j'
00000190:  6176 6177 2E65 7865 3230 3130 3037 3134 'avaw.exe20100714'
000001A0:  3233 3538 3531 3933 2020 2020 5465 7874 '23585193    Text'
000001B0:  2D32                                   '-2              '
***   END of Message ***
```

MCA sends big envelope

- Step 5: The MCA sends the big envelope using the Sender channel
- Step 6: The MCA of the Receiver channel obtains the big envelope and extracts the smaller envelope. Then it deletes the big envelope
- Step 7: The MCA places the message into the destination queue
- Step 8: The application gets the message

Returning a message



Notes: Returning a message

notes

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzae.doc/ic10280_.htm
MQ V7 Information Center: Components needed to send a message
Components needed to return a message

If your application requires messages to be returned from the remote queue manager, you need to define another channel, to run in the opposite direction between the queue managers,

Commands to setup 2-way communication

- Technote: 1470997
- http://www-01.ibm.com/support/docview.wss?uid=swg21470997
- Commands to setup both ways communication between 2 queue managers via Sender and Receiver channels

Notes: Commands for Qmgr 1

notes

In this example, the following queue managers are used:
QMgr1: QM_ANGELITO (in Windows®)
QMgr2: QM_VER (in Linux®)

*** QMgr1: QM_ANGELITO
* Define a local queue:
define qlocal(Q5)
* Define a local queue (used for transmission):
define qlocal(QM_VER) usage(xmitq)
* Define a remote queue definition by typing the following command:
define qremote(Q6_VER) rname(Q6) rqmname(QM_VER) xmitq(QM_VER)
* Define a receiving channel by typing the following command:
define channel(QM_VER.QM_ANGELITO) chltype(RCVR) trptype(TCP)
* Define a sender channel by typing the following command:
define channel(QM_ANGELITO.QM_VER) chltype(SDR) +
conname('veracruz.x.ibm.com(1414)') +
xmitq(QM_VER) trptype(TCP)
* Start the sender channel
start channel(QM_ANGELITO.QM_VER)

Notes: Commands for Qmgr 2

notes

*** QMgr2: QM_VER
* Define a local queue:
  define qlocal(Q6)
* Define a local queue (used for transmission):
  define qlocal(QM_ANGELITO) usage(xmitq)
* Define a remote queue definition by typing the following command:
  define qremote(Q5_ANGELITO) rname(Q5) rqmname(QM_ANGELITO) +
    xmitq (QM_ANGELITO)
* Define a receiving channel by typing the following command:
  define channel(QM_ANGELITO.QM_VER) chltype(RCVR) trptype(TCP)
* Define a sender channel by typing the following command:
  define channel(QM_VER.QM_ANGELITO) chltype(SDR) +
    conname('angelito.x.ibm.com(1414)') +
    xmitq(QM_ANGELITO) trptype(TCP)
* Start the sender channel
  start channel(QM_VER.QM_ANGELITO)

Notes: Miscellaneous

notes

*** To enable Triggering of the Sender Channels

* For example, from queue manager 1 (QM_ANGELITO), to start the Sender channel when the First message (depth of 1) arrives to the transmission queue for queue manager 2 (QM_VER)
Modify the XMITQ, as follows:

```
ALTER qlocal(QM_VER) usage(xmitq) TRIGGER TRIGTYPE(FIRST) TRIGDPTH(1) +
INITQ('SYSTEM.CHANNEL.INITQ')
```

*** Only for MQ V7:
How to specify a Sender channel to connect to multi-instance queue managers
* If queue manager 2 (QM_VER) was a multi-instance queue manager, then the Sender channel from the queue manager 1 (QM_ANGELITO) would need to exploit the expanded format for CONNAME, introduced in V7.
In that way, the queue manager 1 could reconnect to the new active instance after a failover of the multi-instance queue manager.

```
define channel(QM_ANGELITO.QM_VER) chltype(SDR) +
conname('veracruz.x.com(1414),cbeech.x.ibm.com(1414)') +
xmitq(QM_VER) trptype(TCP)
```

Notes: Testing – sending messages

notes

+++ Testing - Send messages to each other

+ QMgr1: QM_ANGELITO

C:\> amqsput Q6_VER QM_ANGELITO
Sample AMQSPUT0 start
target queue is Q6_VER
TEST-FROM-ANGELITO
Sample AMQSPUT0 end

+ QMgr2: QM_VER

rivera@veracruz: /home/rivera
$ amqsput Q5_ANGELITO QM_VER
Sample AMQSPUT0 start
target queue is Q5_ANGELITO
TEST-FROM-QMVER
Sample AMQSPUT0 end

Notes: Testing – browsing received msgs

notes

++ Browse the messages

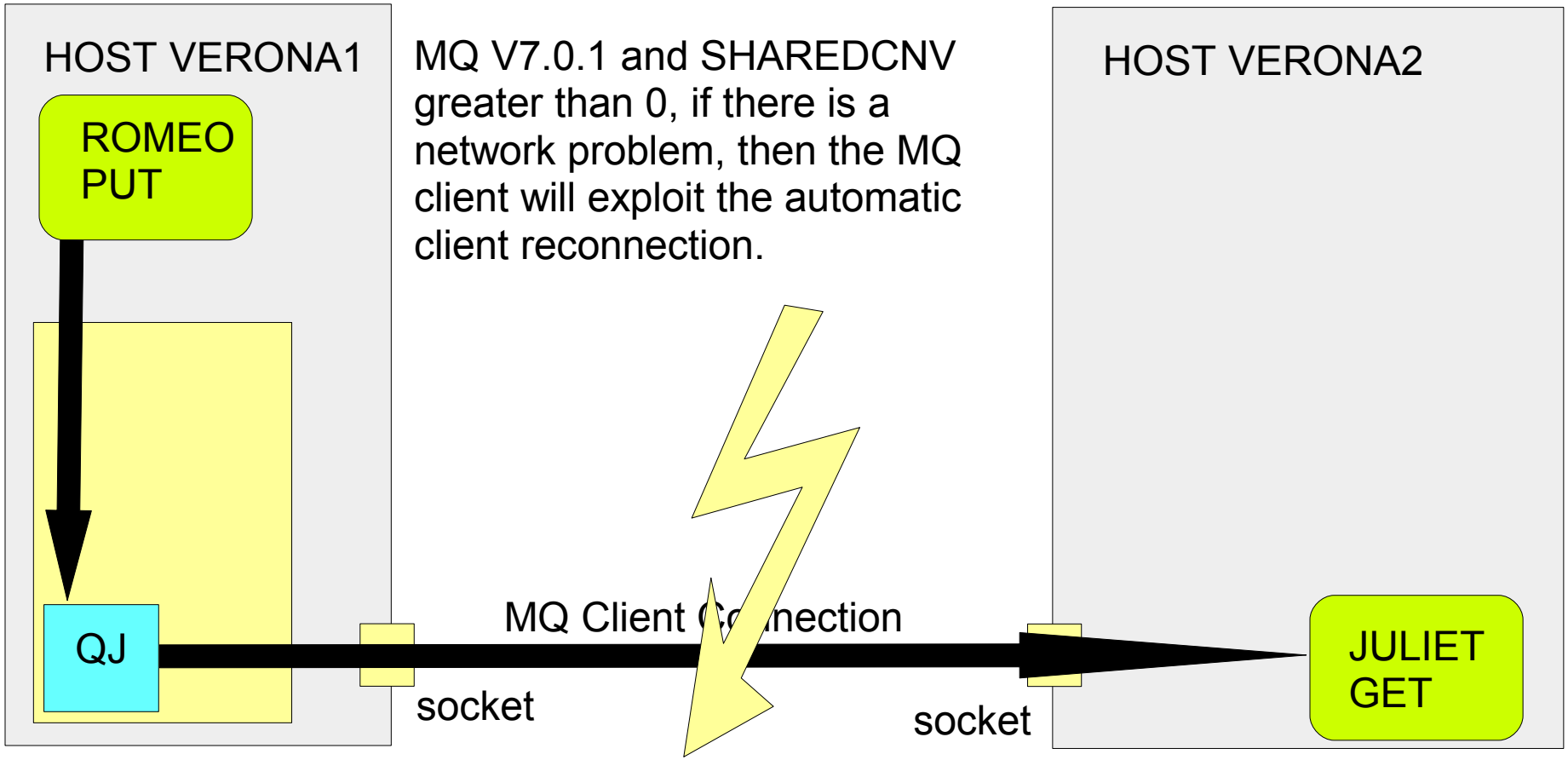
+ QMgr1: QM_ANGELITO

```
C:\> amqsbcg Q5 QM_ANGELITO
AMQSBCG0 - starts here
...
****   Message      ****
 length - 15 bytes
00000000:  5445 5354 2D46 524F 4D2D 514D 4D49 31       'TEST-FROM-QMVER '
```

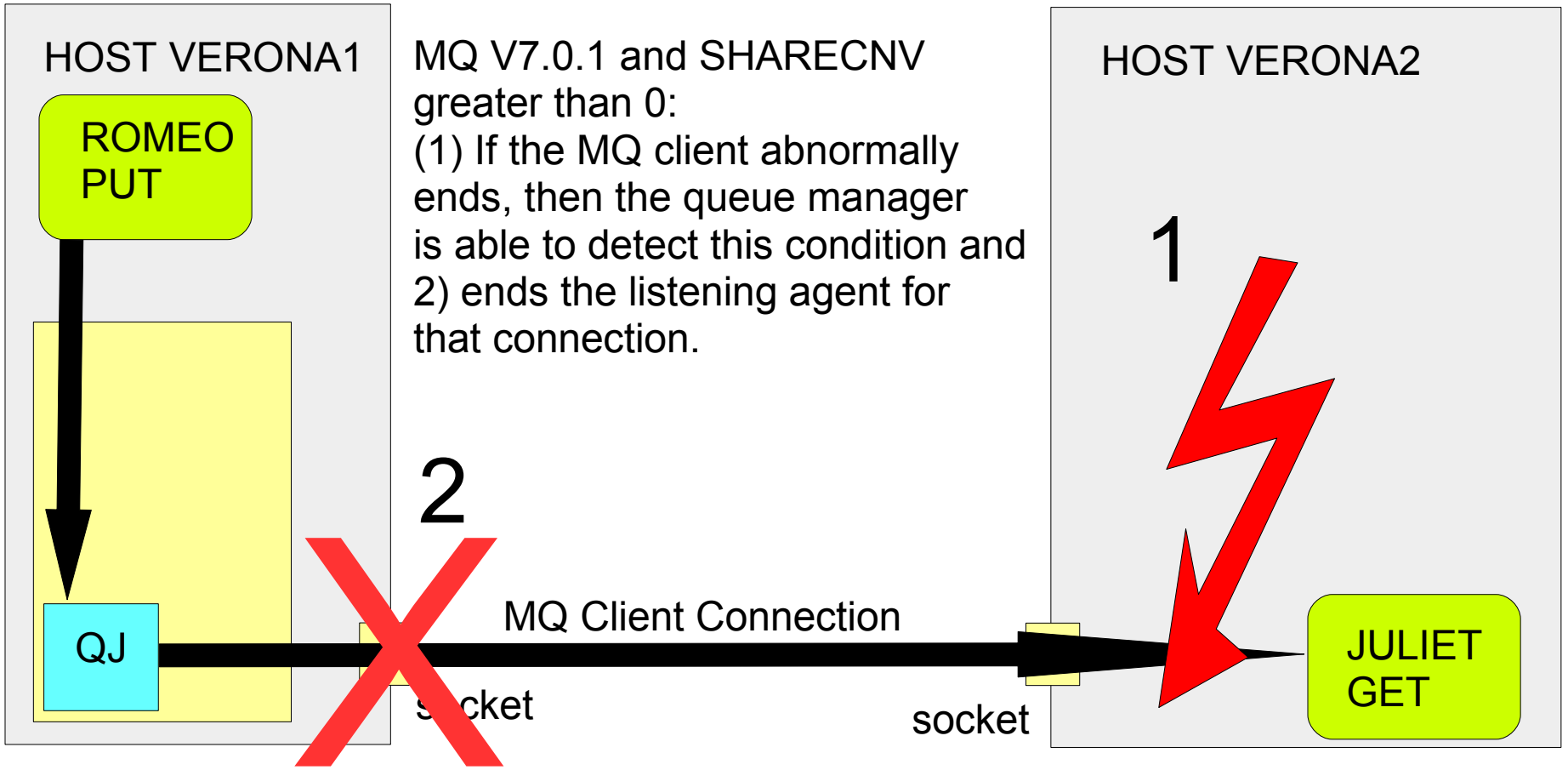
+ QMgr2: QM_VER

```
$ amqsbcg Q6 QM_VER
AMQSBCG0 - starts here
...
****   Message      ****
 length - 18 bytes
00000000:  5445 5354 2D46 524F 4D2D 414E 4745 4C49     'TEST-FROM-ANGELI'
00000010:  544F                                        'TO              '
```

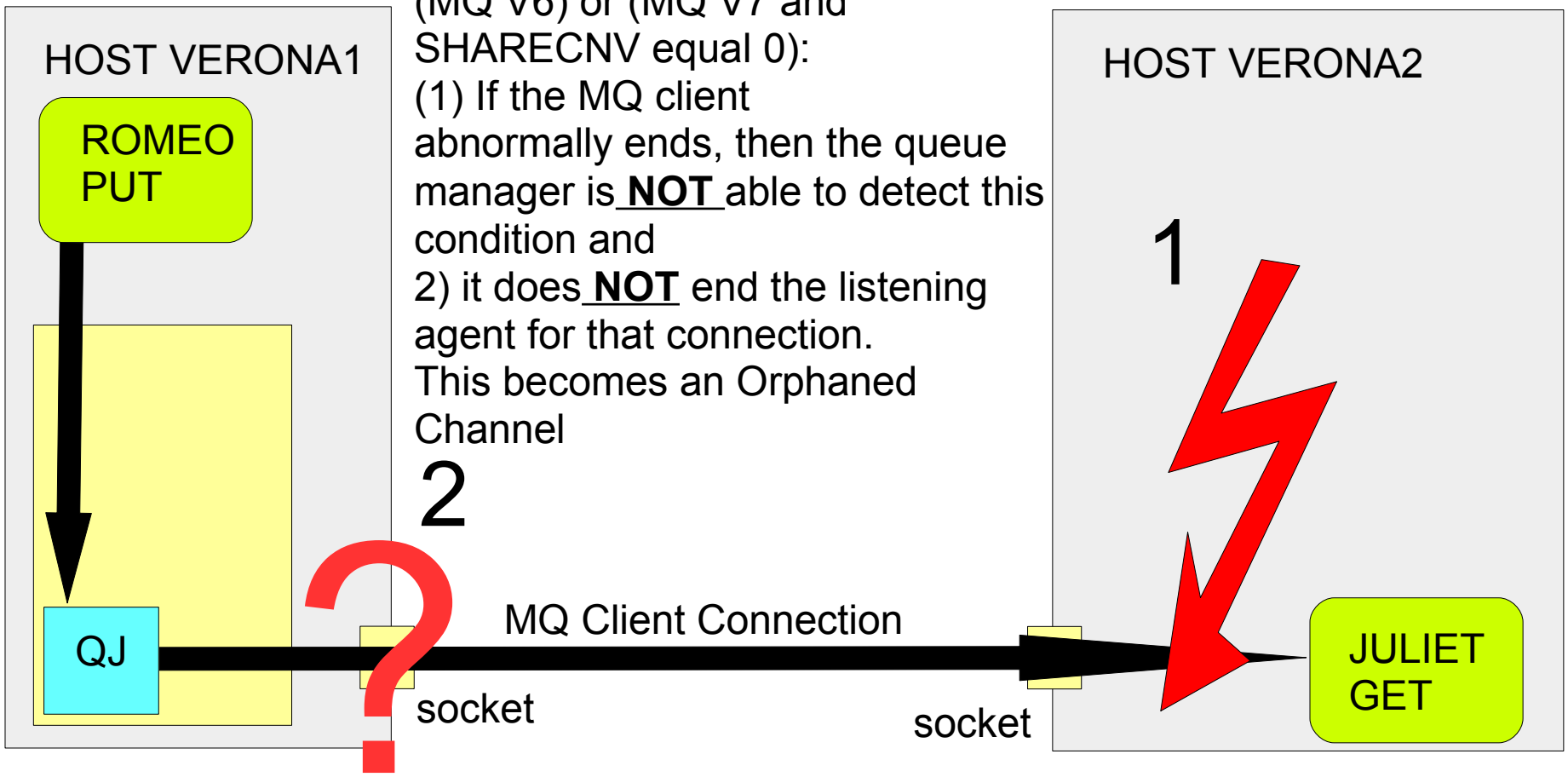
MQ V7.0.1 – Automatic Client Reconnection



MQ V7.0.1 – Client abnormally ends

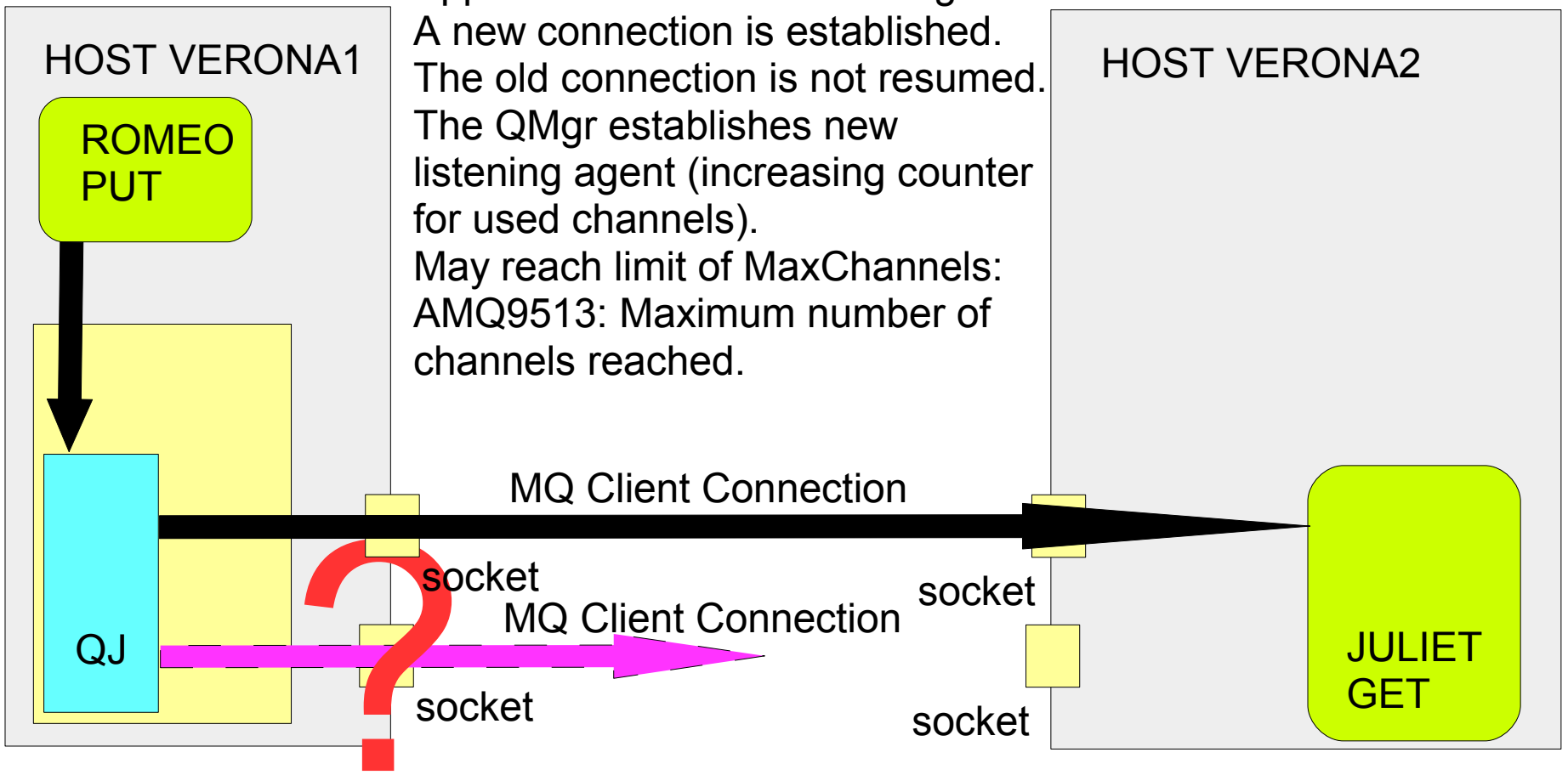


MQ V6 – Client abnormally ends

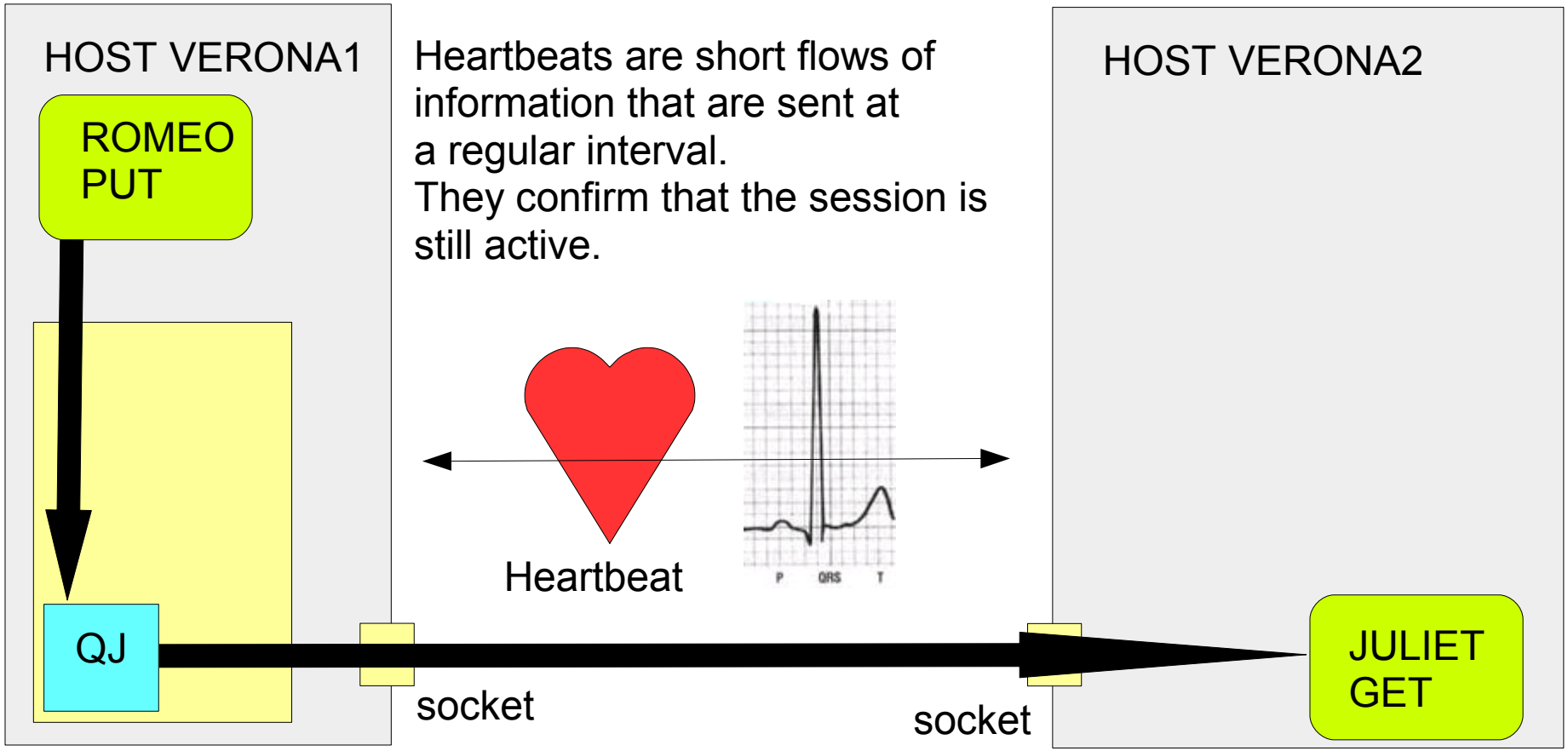


MQ V6 – Client creates new connection

Application tries to connect again.
A new connection is established.
The old connection is not resumed.
The QMgr establishes new listening agent (increasing counter for used channels).
May reach limit of MaxChannels:
AMQ9513: Maximum number of channels reached.



MQ V7.0 – Heartbeat (with SHARECNV > 0)

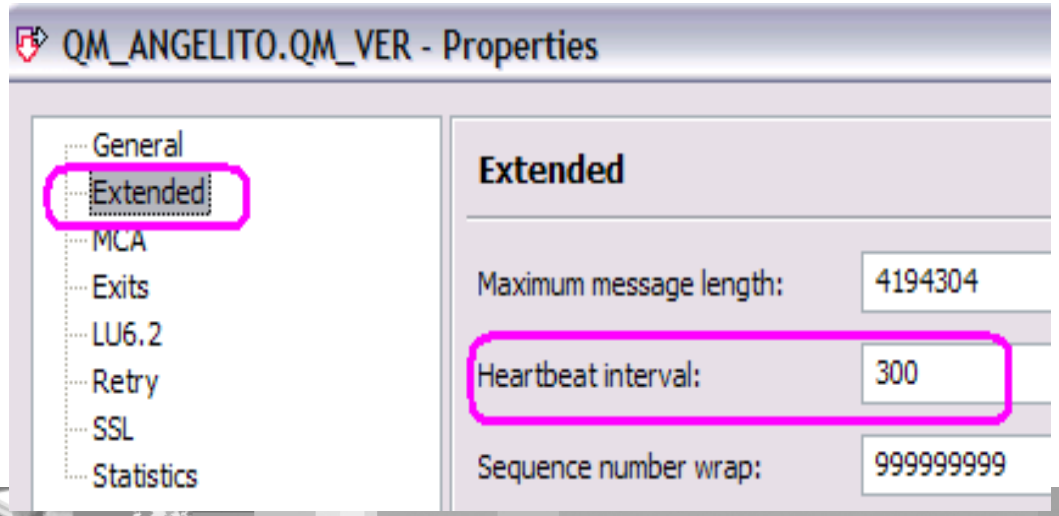


Notes: MQ V7 – Heartbeats - 1

notes

- WebSphere MQ V7.0 Features and Enhancements (Redbook SG24-7583)
- http://www.redbooks.ibm.com/abstracts/SG247583.html?Open
- 5.2 Full duplex channels, heartbeat, and quiesce

- The only purpose for a heartbeat is to confirm that the session is still active.
- Heartbeats can now be performed from both the client end and the queue manager end at a negotiated rate that is based on the "HeartBeat INTerval" (HBINT) parameter of both channels.
- The default value is 300 seconds or 5 minutes.
- In MQ Explorer, this parameter can be specified in the Properties for a channel, in the Extended tab:



Notes: MQ V7 – Heartbeats - 2

notes

- Runmqsc:
- display channel(SYSTEM.DEF.SVRCONN) HBINT
- CHANNEL(SYSTEM.DEF.SVRCONN) CHLTYPE(SVRCONN)
- HBINT(300)

- display CHANNEL(SYSTEM.DEF.CLNTCONN) HBINT
- CHANNEL(SYSTEM.DEF.CLNTCONN) CHLTYPE(CLNTCONN)
- HBINT(300)

- The use of Heartbeats leads to earlier detection of communications network failures and other channel problems.

- The client program and the queue manager can now carry out recovery and reconnecting functions in a more timely manner, resulting in an overall improvement to the quality of service.

- Previous versions of MQ use a half duplex protocol, where a heartbeat could only be performed from the queue manager end during a MQGET operation with a WaitInterval specified. This limited its usefulness for detecting problems.

System Cluster Transmission Queue SCTQ

- Each cluster queue manager has a cluster transmission queue called SCTQ: SYSTEM.CLUSTER.TRANSMIT.QUEUE
- A definition for this queue (and others required for clustering) is created by default on every queue manager except on z/OS®.
- A queue manager that is part of a cluster can send messages on the cluster transmission queue to any other queue manager that is in the same cluster.

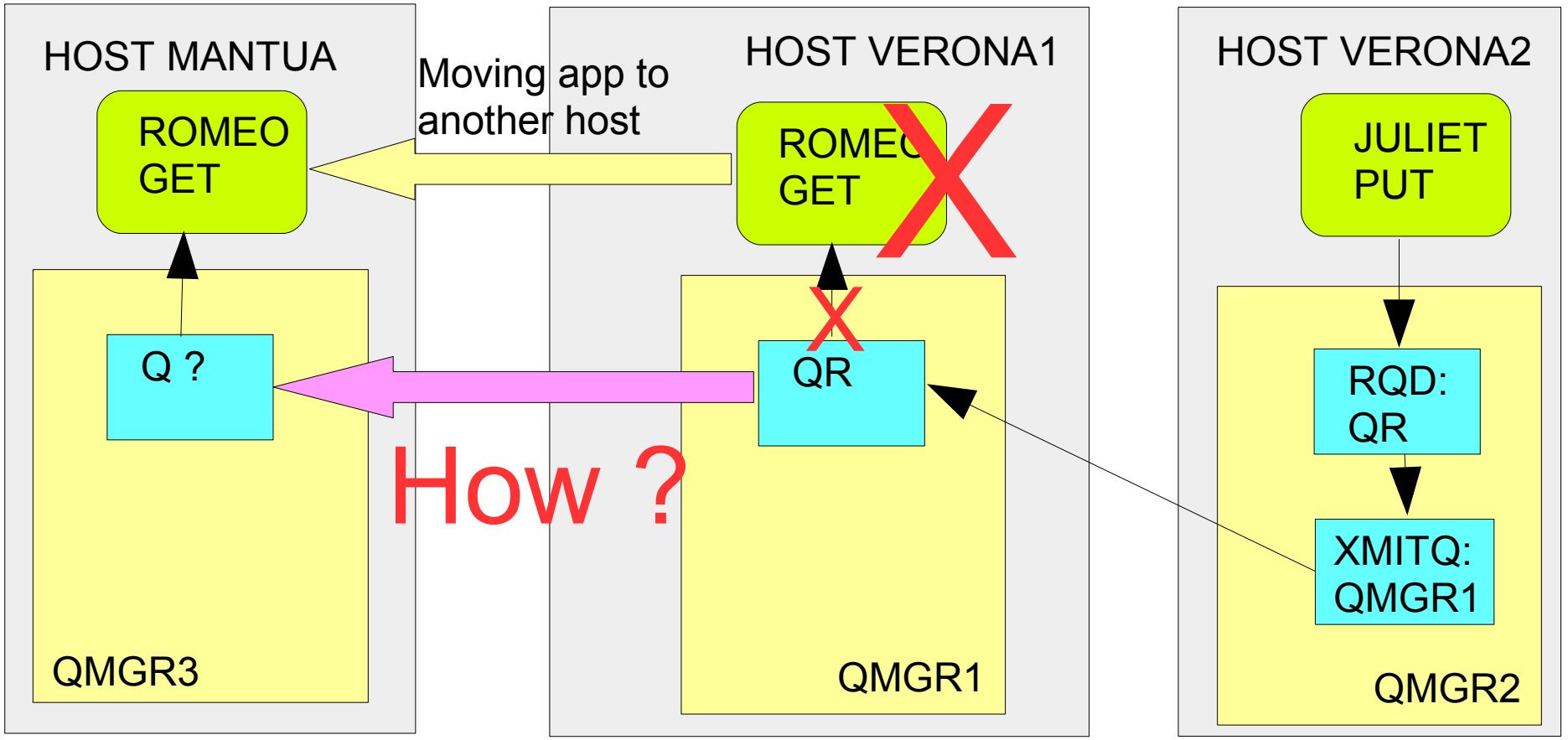
Notes: Resolving problems with SCTQ

notes

- http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzax.doc/mo13960_.htm
- Solving problems with cluster channels
- If you have a build up of messages on the SYSTEM.CLUSTER.TRANSMIT.QUEUE queue, check each channel in the cluster.
- See above page for more details

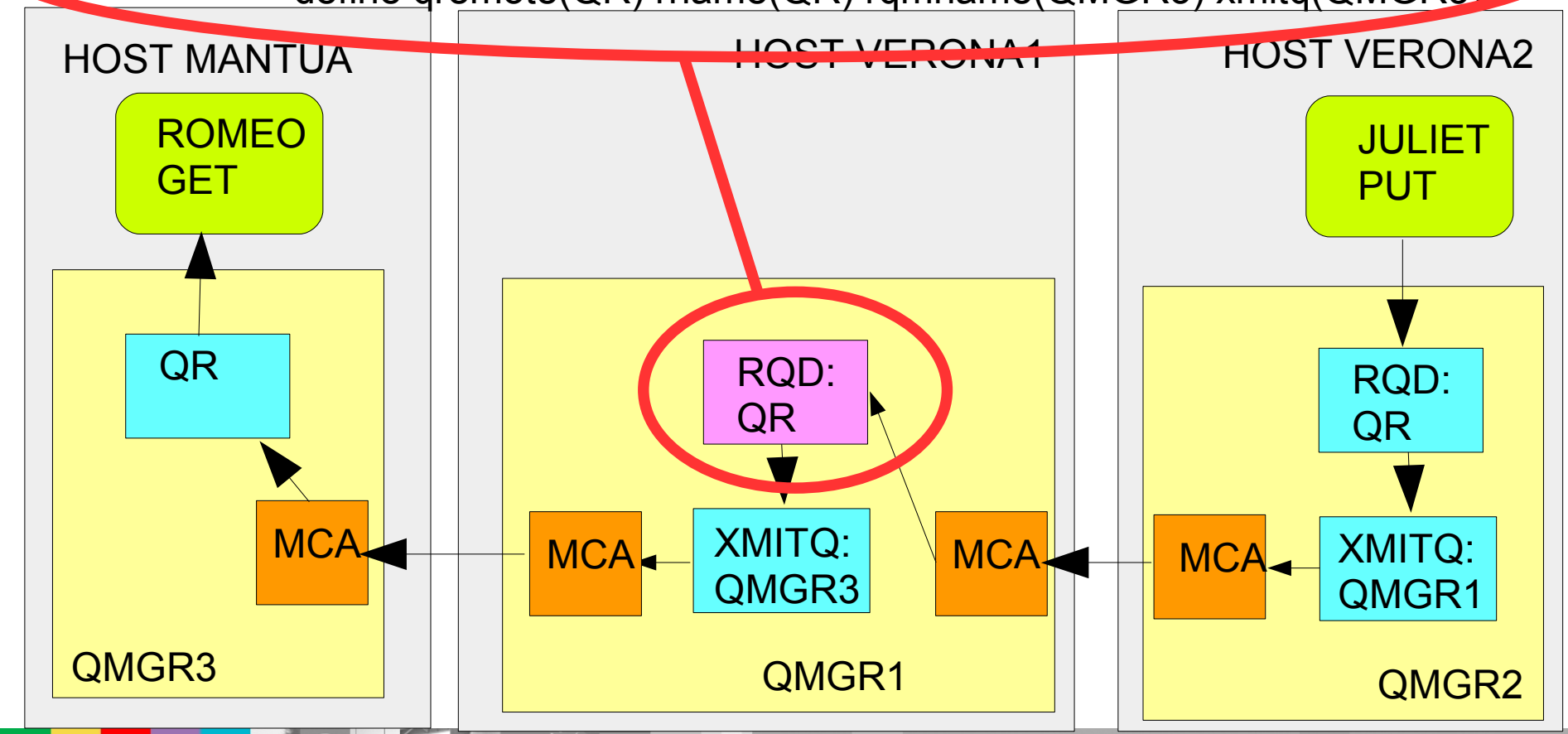
- http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzah.doc/qc13180_.htm
- Messages are not appearing on the destination queues.
- Cause
- The messages may be stuck at their origin queue manager. Make sure that the SYSTEM.CLUSTER.TRANSMIT.QUEUE is empty and also that the channel to the destination queue manager is running.
- See above page for more details

Romeo is exiled to Mantua



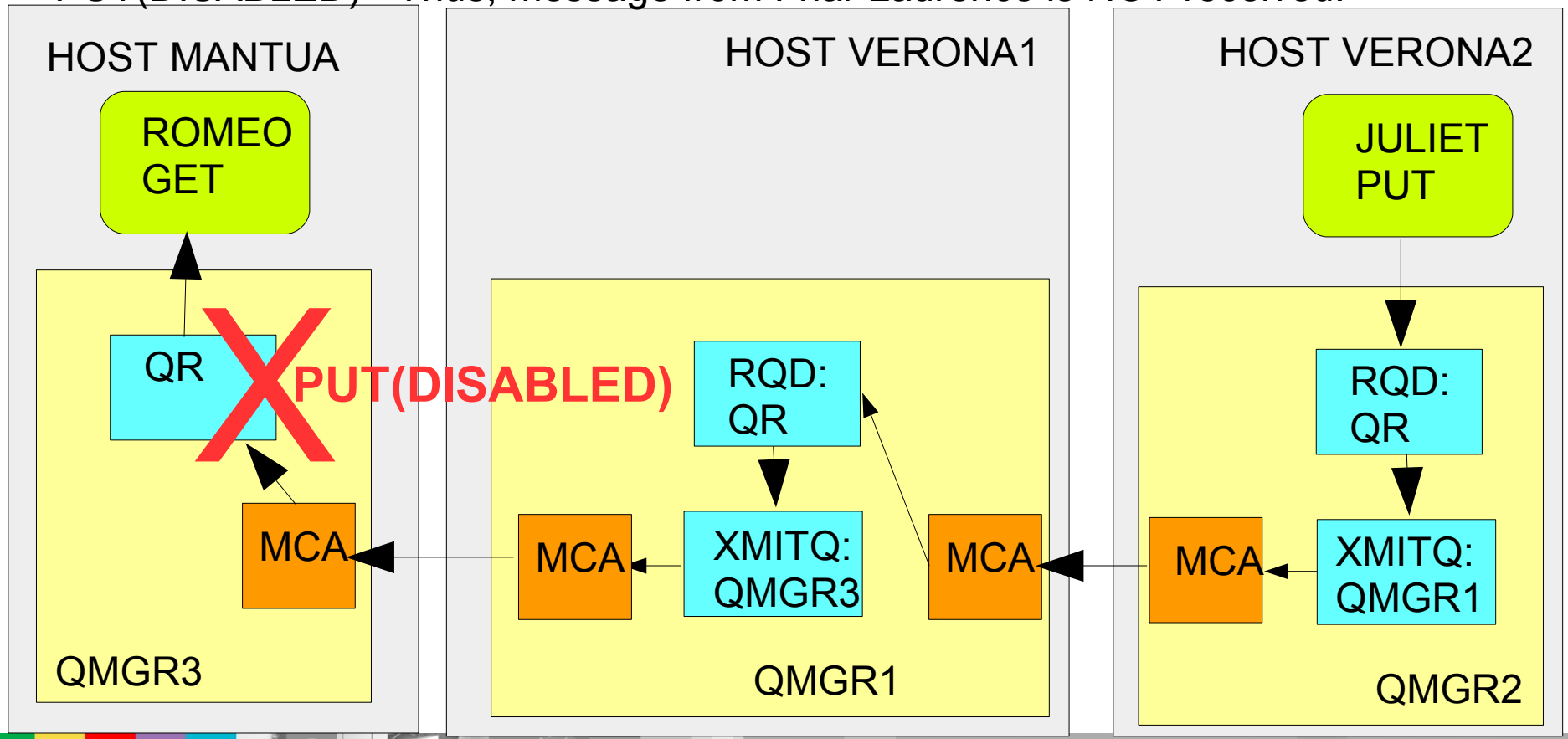
Forwarding messages to Mantua

In QMGR1: delete QL(QR)
define qremote(QR) rname(QR) rqmname(QMGR3) xmitq(QMGR3)



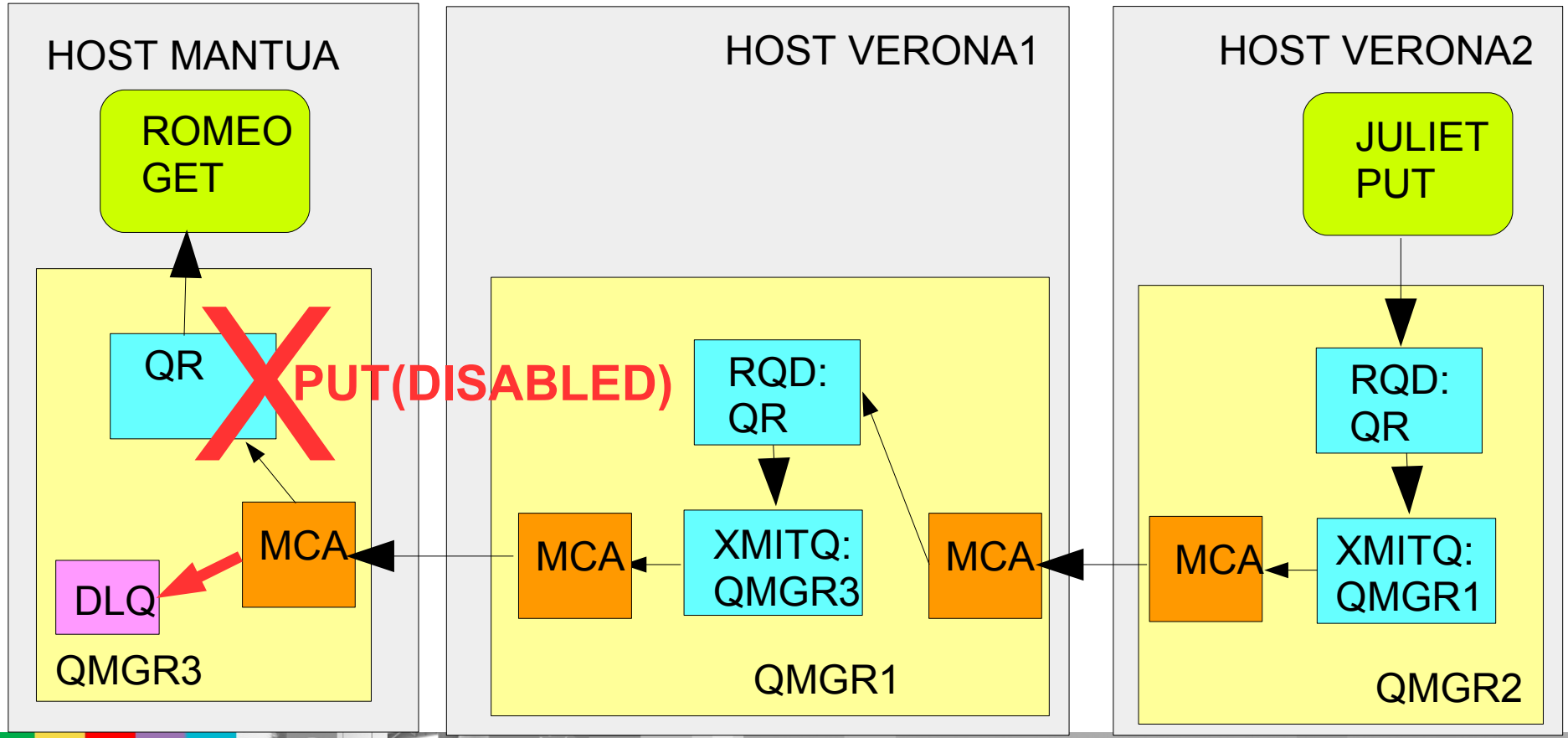
Error causes message NOT to be delivered

For testing purposes, the QR in QMGR3 Mantua is accidentally made: PUT(DISABLED) - Thus, message from Friar Laurence is NOT received.

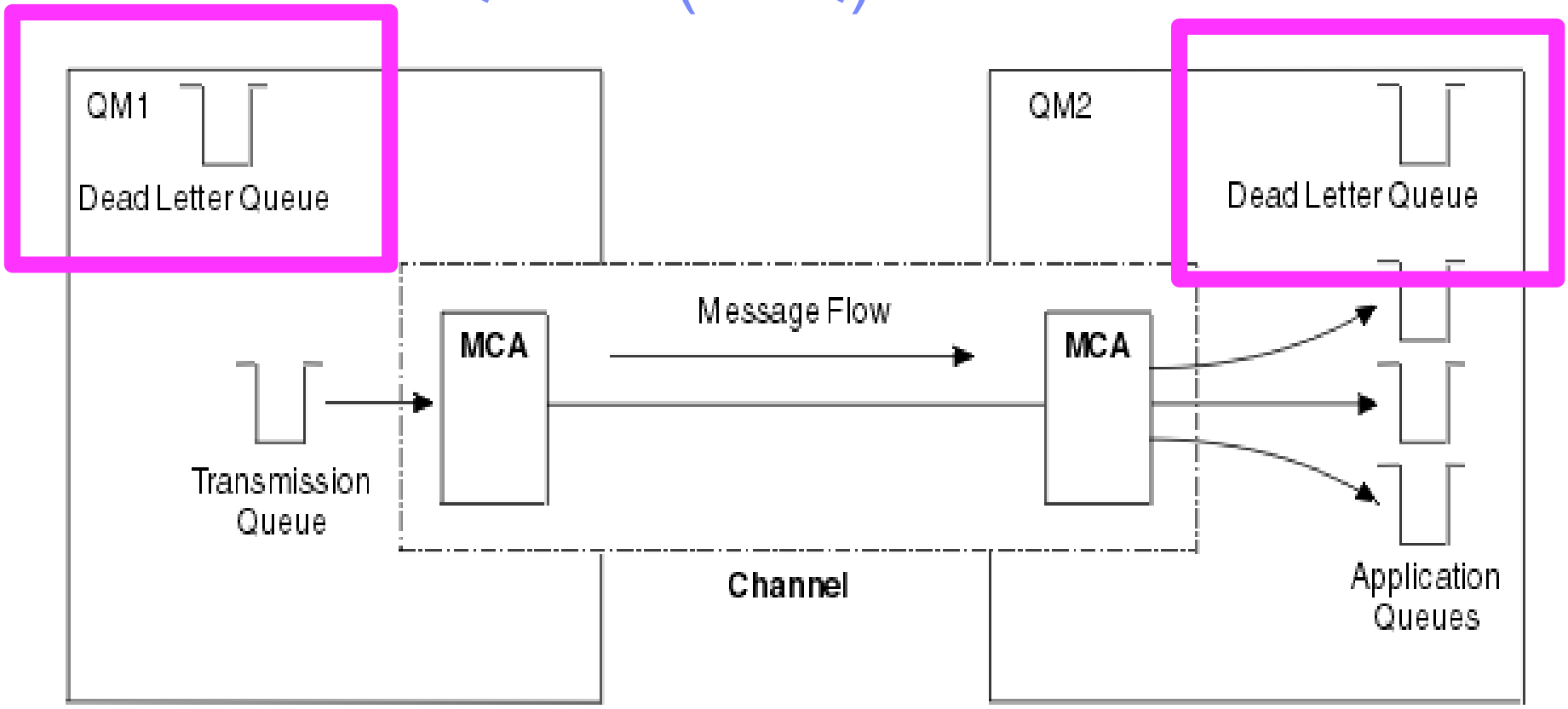


Undelivered message sent to DLQ

If there is a DLQ, the MCA will move undelivered messages to the DLQ



Dead Letter Queue (DLQ)



- Best practice:
  Always define a DLQ in each queue manager

Notes: DLQ example - small inside big

notes

+ Another example of small envelope inside a bigger one (for DLQ)

BIGGER ENVELOPE (Notice the MQDEAD format and DLH "eye catcher")

```
****Message descriptor****
  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 437
   Format : 'MQDEAD  '  <== Notice the FORMAT
  Priority : 0  Persistence : 0
  MsgId : X'414D5120514D5F4652312020202020202020D2B8A54D20001F02'
  CorrelId : X'00000000000000000000000000000000000000000000000000000000'
…
****   Message      ****
length - 434 bytes
00000000:  444C 4820 0100 0000 0308 0000 5152 2020 'DLH ........QR  '
00000010:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000020:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
```

SMALLER ENVELOPE

```
00000020:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000030:  2020 2020 2020 2020 2020 2020 514D 5F50 '            QM_P'
00000040:  5231 2020 2020 2020 2020 2020 2020 2020 'R1              '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020 2020 2020 2202 0000 '            "...'
00000070:  B804 0000 4D51 5354 5220 2020 0B00 0000 '....MQSTR   ....'
00000080:  6562 5370 6865 7265 204D 515C 6269 6E5C 'ebSphere MQ\bin\'
00000090:  616D 7172 6D70 7061 2E65 7865 3230 3131 'amqrmppa.exe2011'
000000A0:  3034 3133 3134 3534 3231 3832 6672 6F6D '041314542182from'
000000B0:  2046 5231                               ' FR1            '
***   END of Message ***
```

Notes: DLQ considerations - 1

notes

- The Dead-Letter Queue (DLQ) is also known as the undelivered-message queue.

- To find out the reason code for a message being sent to the DLQ, see the presentation (PDF file) from the Webcast:
- Browsing Message Fields, Properties and Contents in WebSphere MQ
- http://www-01.ibm.com/support/docview.wss?uid=swg27018059
- Pages 30 thru 37

- http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzae.doc/ic19430_.htm
- Dead-letter queue (DLQ) considerations
- If a channel ceases to run for any reason, applications will probably continue to place messages on transmission queues, creating a potential overflow situation.
- When this occurs in a message-originating node, and the local transmission queue is full, the application's PUT fails.
- When this occurs in a staging or destination node, there are three ways that the MCA copes with the situation:

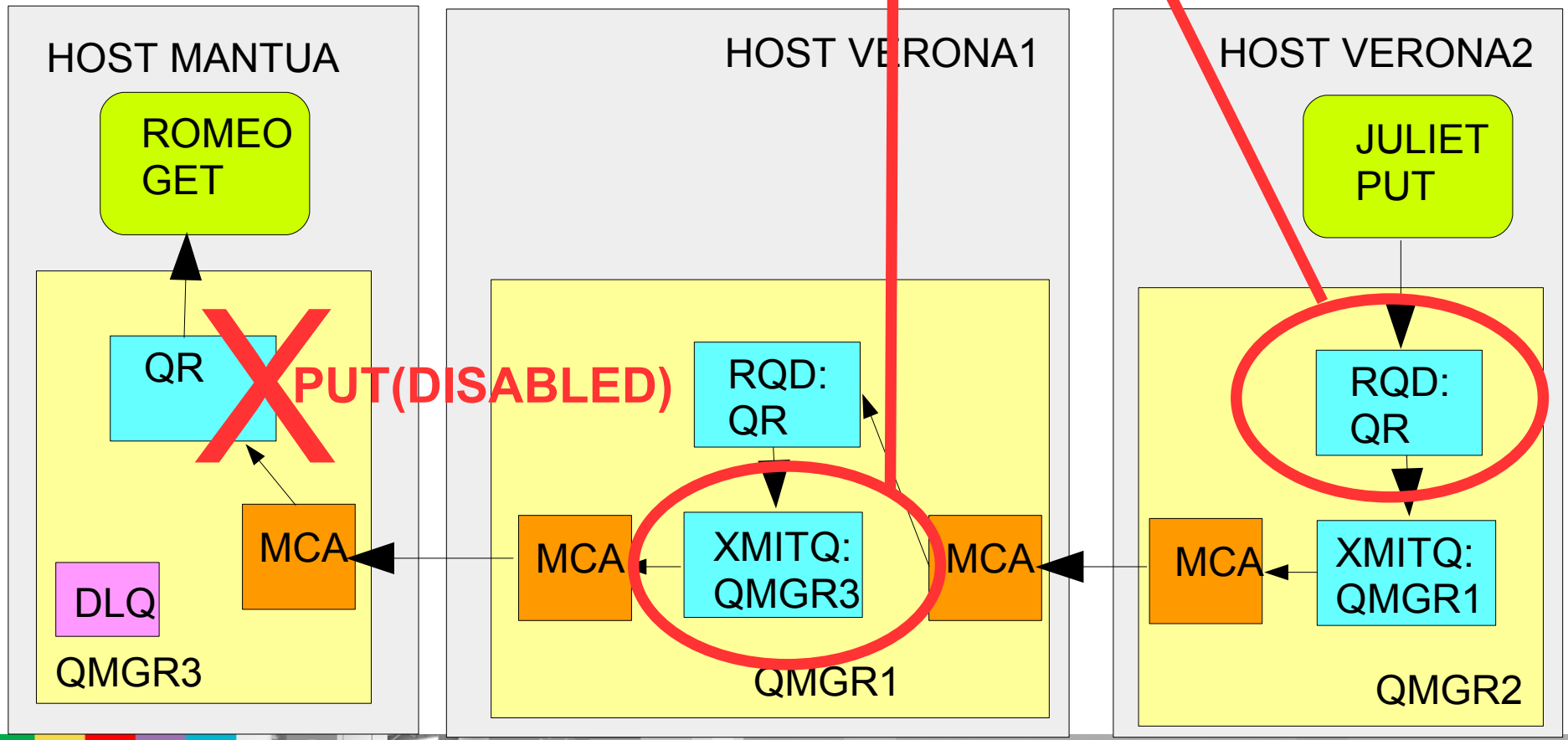
Notes: DLQ considerations - 2

notes

- 1. By calling the message-retry exit, if one is defined.
- 2. By directing all overflow messages to a dead-letter queue (DLQ), returning an exception report to applications that requested these reports.
- Note: In distributed-queuing management, if the message is too big for the DLQ, the DLQ is full, or the DLQ is not available, the channel stops and the message remains on the transmission queue. Ensure your DLQ is defined, available, and sized for the largest messages you handle.
- 3. By closing down the channel, if neither of the previous options succeeded.
- 4. By returning the undelivered messages back to the sending end and returning a full report to the reply-to queue (MQRC_EXCEPTION_WITH_FULL_DATA and MQRO_DISCARD_MSG).
- .
- If an MCA is unable to put a message on the DLQ:
- • The channel stops
- • Appropriate error messages are issued at the system consoles at both ends of the message channel
- • The unit of work is backed out, and the messages reappear on the transmission queue at the sending channel end of the channel
- • Triggering is disabled for the transmission queue

Using Persistent Message

If RQD QR in QMGR2 is made persistent: DEFPSIST(YES)
… then XMITQ QMGR3 in QMGR1 will retain the message if MCA in QMGR3 could not deliver the message to QR in QMGR3.



Sad fate for Romeo and Julie

Summary

- Overview of distributed messaging
- Components:
  - Remote Queue Definition
  - Transmission Queue
  - MCA
  - Sender/Receiver Channels
  - Heartbeat
- System Cluster Transmit Queue
- Dead Letter Queue

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:
  http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:
  http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community:
  http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant:
  http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:
  http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails:
  http://www.ibm.com/software/support/einfo.html

We Want to Hear From You!

Tell us about what you want to learn

Suggestions for future topics
Improvements and comments about our webcasts
We want to hear everything you have to say!

Please send your suggestions and comments to:
wsehelp@us.ibm.com

Questions and Answers