Where's my message?

Mark Taylor, IBM Hursley

marke_taylor@uk.ibm.com

Symptoms include:

- Timeouts for request/response messaging
- Inconsistent data between front-end and back-end systems
- Customer complaints about ‘missing’ transactions
- Developer frustration when testing an app / MQ environment
- NOTE: This presentation concentrates on features in the Distributed platforms, though many applicable to z/OS too

Agenda – the MQ Toolbox

- Who's connected?
- Are messages flowing?
- Where are messages going?
- What are the apps doing?
- How can I look back in time?

Presentation contains MQSC example commands and output. Your own admin tools will vary by task, situation and preference.

WHO'S CONNECTED?

DISPLAY CONN

```
DISPLAY CONN(*) TYPE(HANDLE) ALL      DIS CONN(*) ALL
```

```
AMQ8276: Display Connection details.
   CONN(577C425321295301)
   EXTCONN(414D514347415445574159312020202020)
   TYPE(HANDLE)

   OBJNAME(WLMMDB.REQUEST)                  OBJTYPE(QUEUE)
   ASTATE(NONE)                             HSTATE(INACTIVE)
   OPENOPTS(MQOO_OUTPUT,MQOO_FAIL_IF_QUIESCING)
   READA(NO)

   OBJNAME(SENDINGAPP.REPLY)                OBJTYPE(QUEUE)
   ASTATE(ACTIVE)                           HSTATE(ACTIVE)
   OPENOPTS(MQOO_INPUT_SHARED,MQOO_INQUIRE,MQOO_SAVE_ALL_CONTEXT,MQOO_FAIL_IF_QUIESCING)
   READA(NO)
```

Use CONN to match TYPE(CONN) and TYPE(HANDLE) records

TYPE(HANDLE) records let you find applications by the objects they access.
See all open handles for an app in one place, unlike DIS QSTATUS records

```
AMQ8276: Display Connection details.
   CONN(577C425321295301)
   EXTCONN(414D514347415445574159312020202020)
   TYPE(CONN)
   PID(9740)                                TID(185)
   APPLDESC(WebSphere MQ Channel)           APPLTAG(jms/GATEWAY1_CF)
   APPLTYPE(SYSTEM)                         ASTATE(NONE)
   CHANNEL(WAS.CLIENTS)                     CONNAME(127.0.0.1)
   CONNOPTS(MQCNO_SHARED_BINDING)           USERID(pbroad)
   UOWLOG( )                                UOWSTDA(2014-04-08
   UOWSTTI(13.24.00)                        UOWLOGDA( )
   UOWLOGTI( )                              URTYPE(XA)
   EXTURID(XA_FORMATID[DSAW]
XA_GTRID[00000145414B8AB40000000104DF48FC0001020304050607080900
XA_BQUAL[00000145414B8AB40000000104DF48FC0001020304050607080900
000000000001])
   QMURID(0.7940075)                        UOWSTATE(ACTIVE)
```

Channel name + IP help identify client apps.

MQ V7.5 and later JMS clients can supply an application name in the CF

Long running UOW information.
XID can be tied up with app server txn timeout

DISPLAY CHSTATUS

```
DISPLAY CHSTATUS(*)
```

```
AMQ8417: Display Channel Status details.
   CHANNEL(WAS.CLIENTS)                    CHLTYPE(SVRCONN)
   BUFSRCVD(17)                            BUFSSENT(13)
   BYTSRCVD(2296)                          BYTSSENT(2456)
   CHSTADA(2014-04-08)                     CHSTATI(15.26.59)
   COMPHDR(NONE,NONE)                      COMPMSG(NONE,NONE)
   COMPRATE(0,0)                           COMPTIME(0,0)
   CONNAME(127.0.0.1)                      CURRENT
   EXITTIME(0,0)                           HBINT(5)
   JOBNAME(0000260C000000B9)               LOCLADDR( )
   LSTMSGDA(2014-04-08)                    LSTMSGTI(15.26.59)
   MCASTAT(RUNNING)                        MCAUSER(pbroad)
   MONCHL(OFF)                             MSGS(6)
   RAPPLTAG(jar)                           SSLCERTI(CN=ExampleCA,O=Example)
   SSLKEYDA( )                             SSLKEYTI( )
   SSLPEER(SERIALNUMBER=53:43:FD:D6,CN=ExampleApp1,O=Example)
   SSLRKEYS(0)                             STATUS(RUNNING)
   STOPREQ(NO)                             SUBSTATE(RECEIVE)
   CURSHCNV(1)                             MAXSHCNV(1)
   RVERSION(00000000)                      RPRODUCT(MQJM)
```

Check suitable heartbeats are negotiated

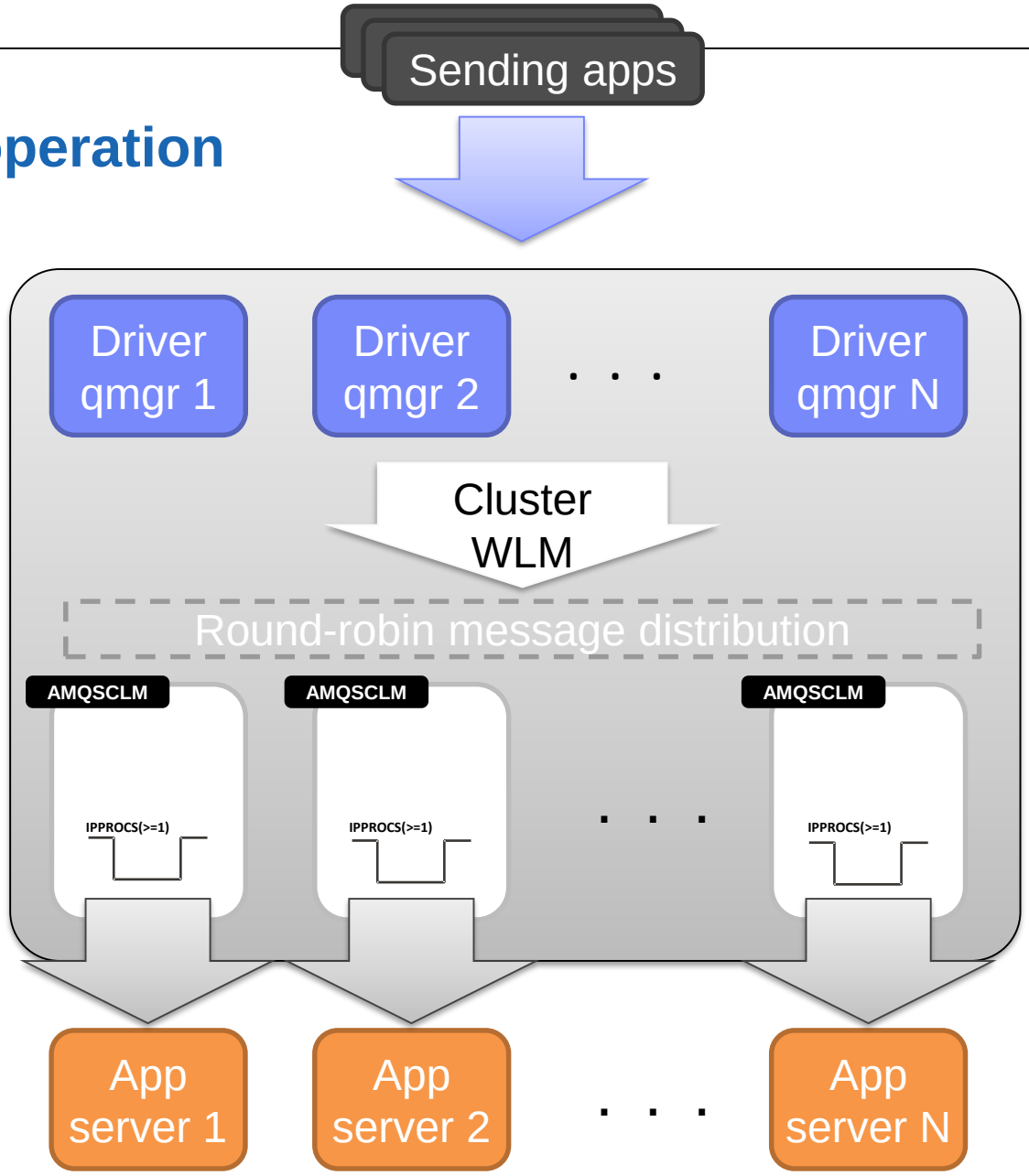
See SSLPEER information not in DIS CONN

JOBNAME contains PID (except z/OS): 0x260C = PID(9740)
On Linux/UNIX (not Win) TID matches CONN: 0xB9 = TID(185)

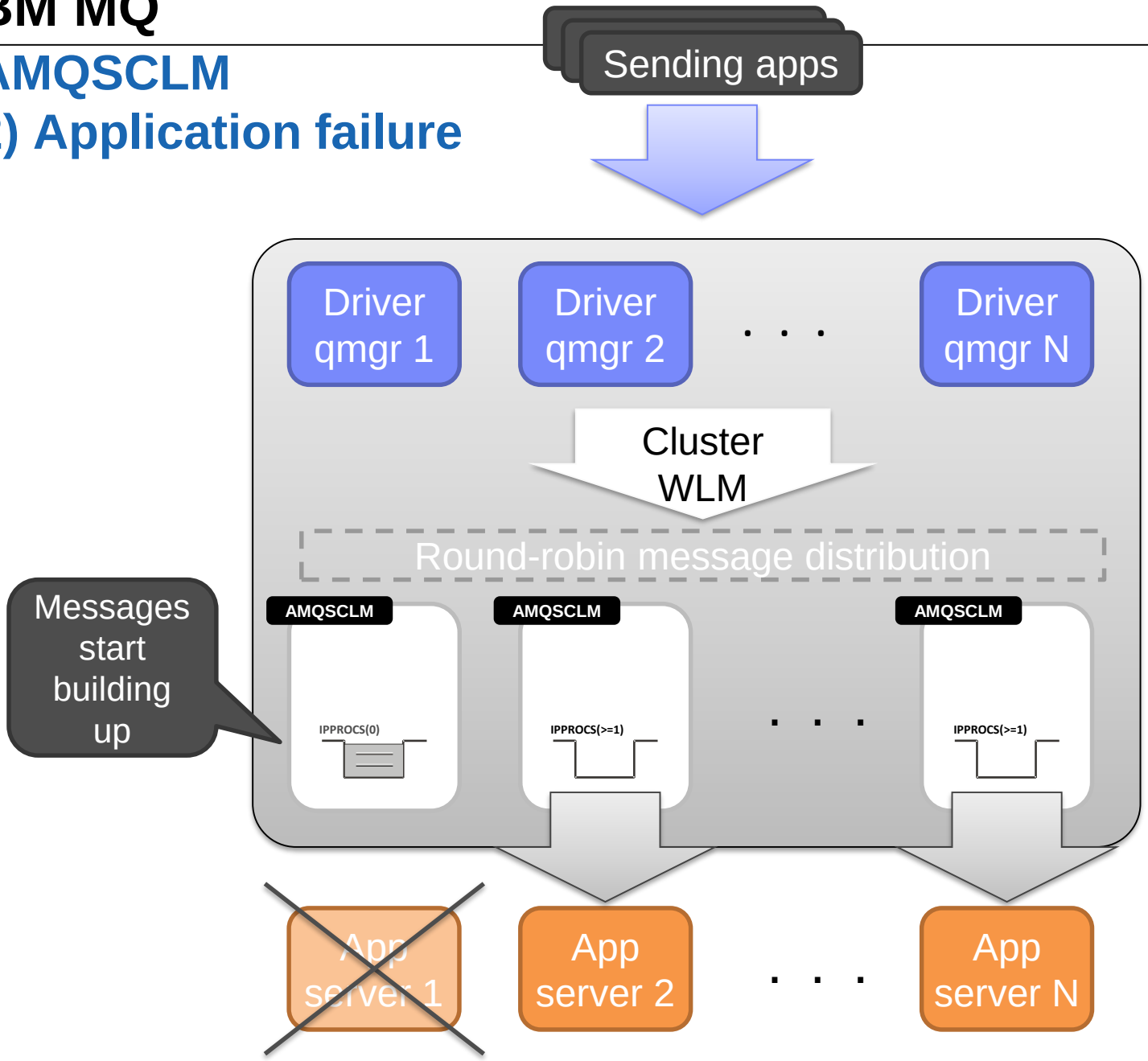
Note that multiple CONN might share one SVRCONN channel instance

AUTOMATIC DETECTION IF *NO-ONE* IS CONNECTED

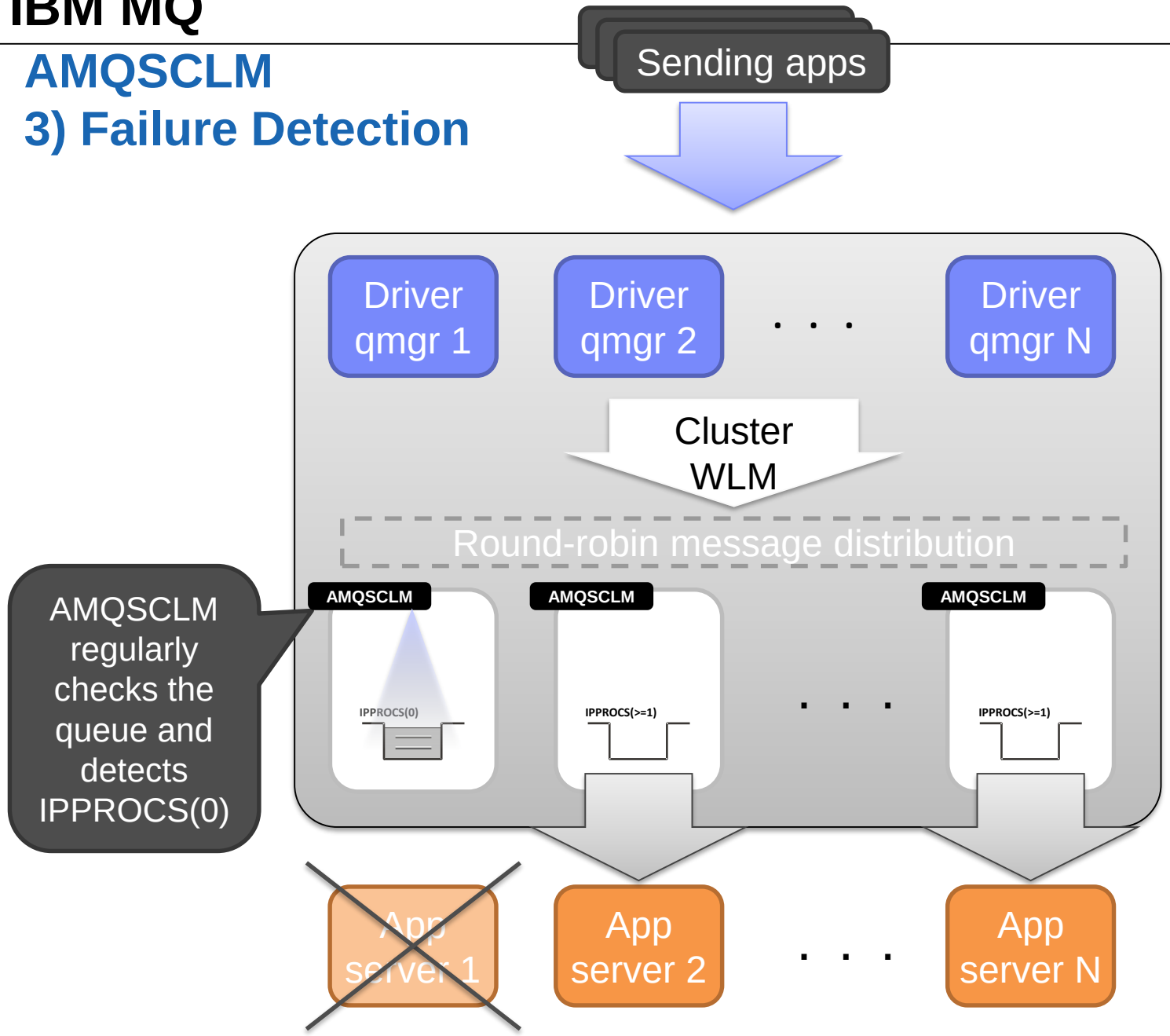
AMQSCLM
1) Normal operation



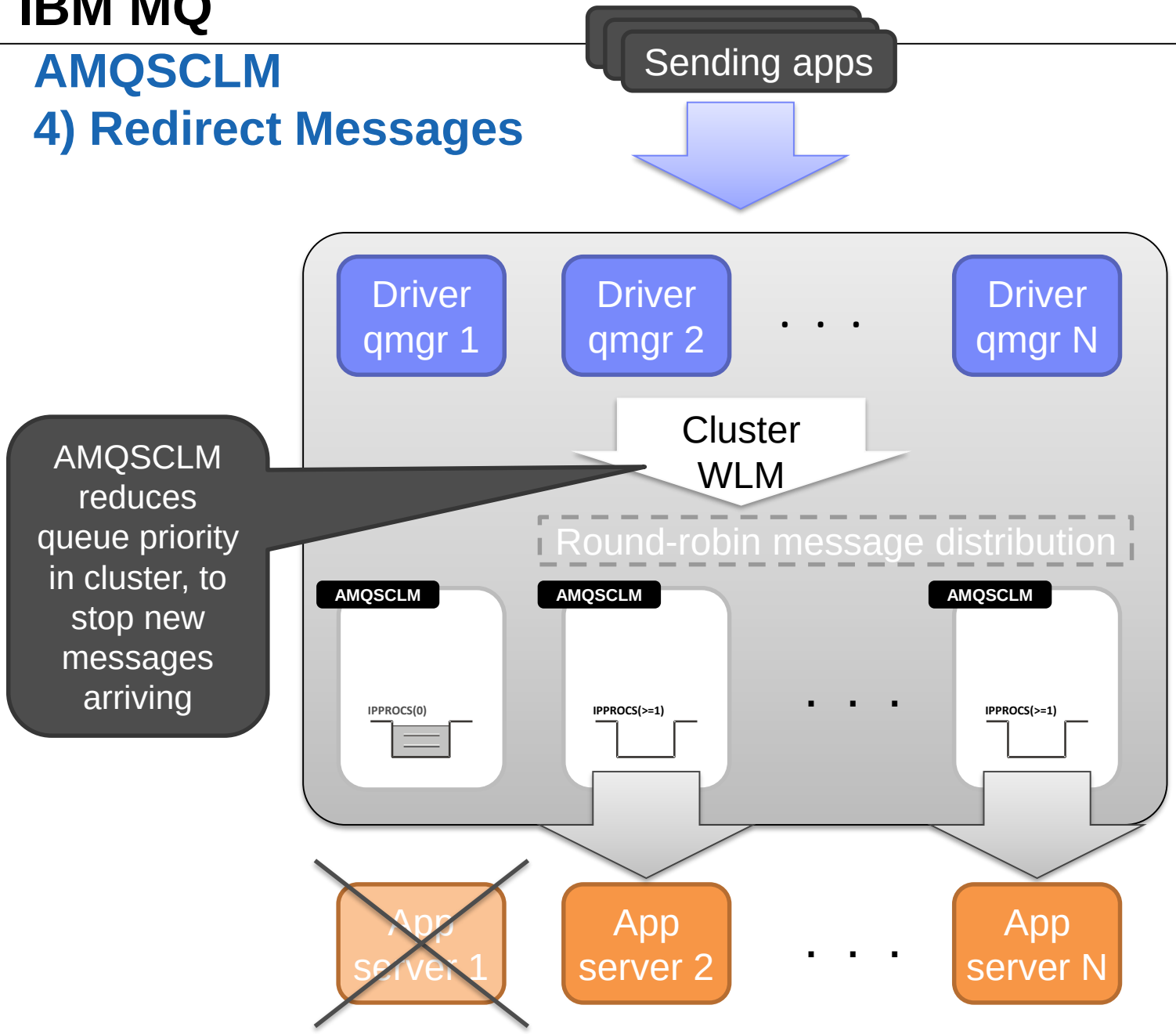
AMQSCLM
2) Application failure



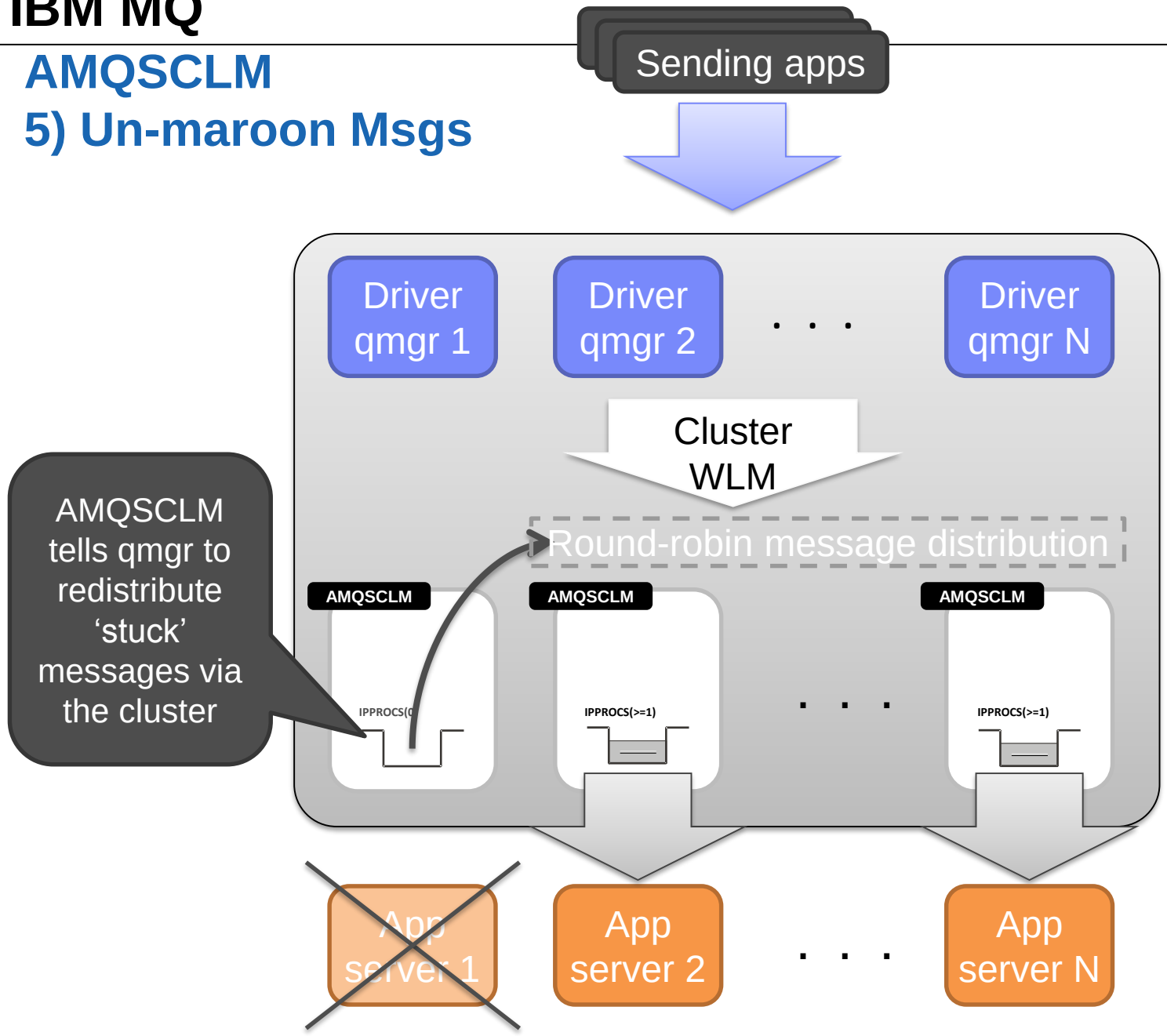
AMQSCLM
3) Failure Detection



AMQSCLM
4) Redirect Messages



AMQSCLM
5) Un-maroon Msgs



AMQSCLM summary

- The Cluster Queue Monitoring sample program (AMQSCLM)
- Shipped with the product as a sample
  - Precompiled
  - Source code
  - Platforms except on z/OS
- More information here: http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q024620_.htm

ARE MESSAGES FLOWING?

Real-time/online Monitoring – Queues

- Set detail level for QMGR. Override for individual Queues

```
ALTER QMGR MONQ(MEDIUM)
ALTER QLOCAL(QUEUE1) MONQ(HIGH)
ALTER QLOCAL(QUEUE2) MONQ(OFF)
```

- Gives live view of application responsiveness

```
DIS QSTATUS(QUEUE1) ALL
```

```
AMQ8450: Display queue status details.
   QUEUE(QUEUE1)                   TYPE(QUEUE)
   CURDEPTH(16)                    IPPROCS(3)
   LGETDATE(2014-04-08)            LGETTIME(17.05.59)
   LPUTDATE(2014-04-08)            LPUTTIME(17.12.16)
   MEDIALOG( )                     MONQ(HIGH)
   MSGAGE(112)                     OPPROCS(5)
   QTIME(10101414, 10101414)       UNCOM(NO)
```

Without MONQ you only get the depth and how many handles are open

Timestamps of last PUT/GET to check for recent activity

Age in Seconds of the oldest message on the queue

Estimations of the time in Microseconds that messages are waiting on the queue for processing.
First value: Calculated from recent activity
Second value: Calculated from longer term activity

Real-time/online Monitoring – Channels

```
ALTER QMGR MONCHL(MEDIUM)
ALTER CHANNEL(CLUSTER1.QM1) CHLTYPE(CLUSRCVR) MONCHL(HIGH)
```

- Gives live view of channel throughput

```
DIS CHSTATUS(MQHUB.GATEWAY2) ALL
```

```
AMQ8417: Display Channel Status details.
   CHANNEL(MQHUB.GATEWAY2)                 CHLTYPE(CLUSSDR)
   BATCHES(52)                             BATCHSZ(50)
   BUFSRCVD(55)                            BUFSSENT(1616)
   BYTSRCVD(1748)                          BYTSSENT(1192330)
   CHSTADA(2014-04-08)                     CHSTATI(17.49.03)
   COMPHDR(NONE,NONE)                      COMPMSG(NONE,NONE)
   COMPRATE(0,0)                           COMPTIME(0,0)
   CONNAME(127.0.0.1(1422))                CURLUWID(0107445310001B34)
   CURMSGS(50)                             CURRENT
   CURSEQNO(11823)                         EXITTIME(0,0)
   HBINT(5)                                INDOUBT(YES)
   JOBNAME(00002DA4000047D0)               LOCLADDR(127.0.0.1(53557))
   LONGRTS(999999999)                      LSTLUWID(0107445310001B33)
   LSTMSGDA(2014-04-08)                    LSTMSGTI(17.49.51)
   LSTSEQNO(11773)                         MCASTAT(RUNNING)
   MONCHL(MEDIUM)                          MSGS(1580)
   NETTIME(137538,29555)                   NPMSPEED(FAST)
   RQMNAME(GATEWAY2)                       SHORTRTS(180)
   SSLCERTI( )                             SSLKEYDA( )
   SSLKEYTI( )                             SSLPEER( )
   SSLRKEYS(0)                             STATUS(RUNNING)
   STOPREQ(NO)                             SUBSTATE(RECEIVE)
   XBATCHSZ(20,17)                         XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
   XQMSGSA(112)                            XQTIME(545784,3929968)
   RVERSION(07050002)                      RPRODUCT(MQMM)
```

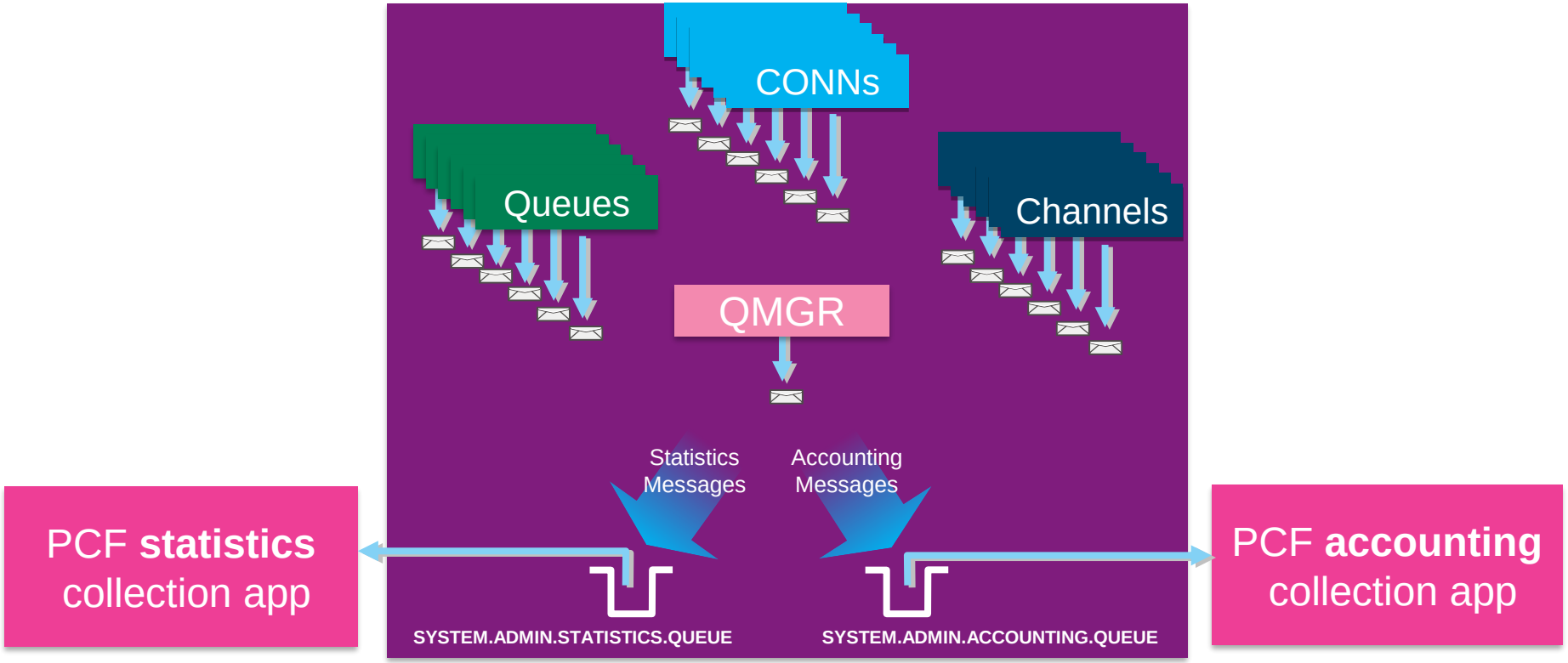
Short/long term calculations of how full your batches are getting, to help you tune BATCHSZ/BATCHINT

Last time a message was sent over the channel

Depth of messages on XMITQ for this channel (capped at 999)

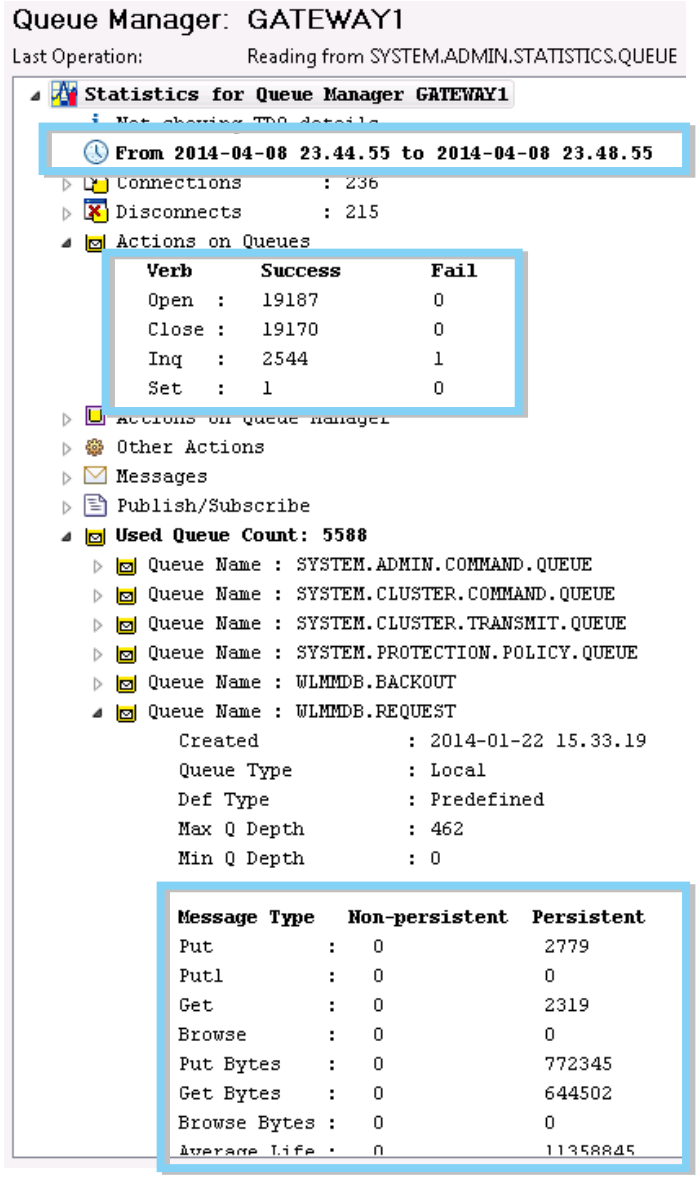
Short/long term calculations of how long messages are waiting on the XMITQ for transmission

Accounting and Statistics Overview



- Monitoring data sent as a PCF message at a configured interval
  - Statistics – scoped to a Queue / Channel / QMGR
  - Accounting – scoped to an individual CONN and Queue / QMGR
- Applications must consume the messages
  - Enterprise monitoring – Tivoli ITCAM / Tivoli OMEGAMON XE for Messaging
  - Sample applications and SupportPacs
  - Custom applications – `com.ibm.mq.headers` Java package

Taking a look with SupportPac MS0P



- Time period
- MQ Statistics at QMGR level
- Detailed queue statistics

Simple practical example using amqsmon

```
ALTER QMGR STATMQI(ON)
Wait a bit
RESET QMGR TYPE(STATISTICS)
amqsmon -m GATEWAY1 -t statistics -a -w 0
```

```
MonitoringType: MQIStatistics
QueueManager: 'GATEWAY1'
IntervalStartDate: '2014-04-09'
IntervalStartTime: '00.00.35'
IntervalEndDate: '2014-04-09'
IntervalEndTime: '00.01.13'
CommandLevel: 700
ConnCount: 35

PutCount: [271, 0]
PutFailCount: 0
Put1Count: [2, 0]
Put1FailCount: 0
PutBytes: [273976, 0]
GetCount: [270, 0]
GetBytes: [269468, 0]
GetFailCount: 19

DurableSubscriptionHighWater: [0, 0, 0, 0]
DurableSubscriptionLowWater: [0, 0, 0, 0]
NonDurableSubscriptionHighWater: [0, 0, 0, 0]
NonDurableSubscriptionLowWater: [0, 0, 0, 0]
PutTopicCount: [0, 0]
PutTopicFailCount: 0
Put1TopicCount: [0, 0]
Put1TopicFailCount: 0
PutTopicBytes: [0, 0]
PublishMsgCount: [0, 0]
PublishMsgBytes: [0, 0]
```

- Overall QMGR busy-ness
- Simple data format
  - Multiple values are `[Persistent, NonPersistent]`
- One message every X seconds
  - Use amqsmon directly (perl/cron)
- Low/high water marks for
  - Connections
  - Subscriptions

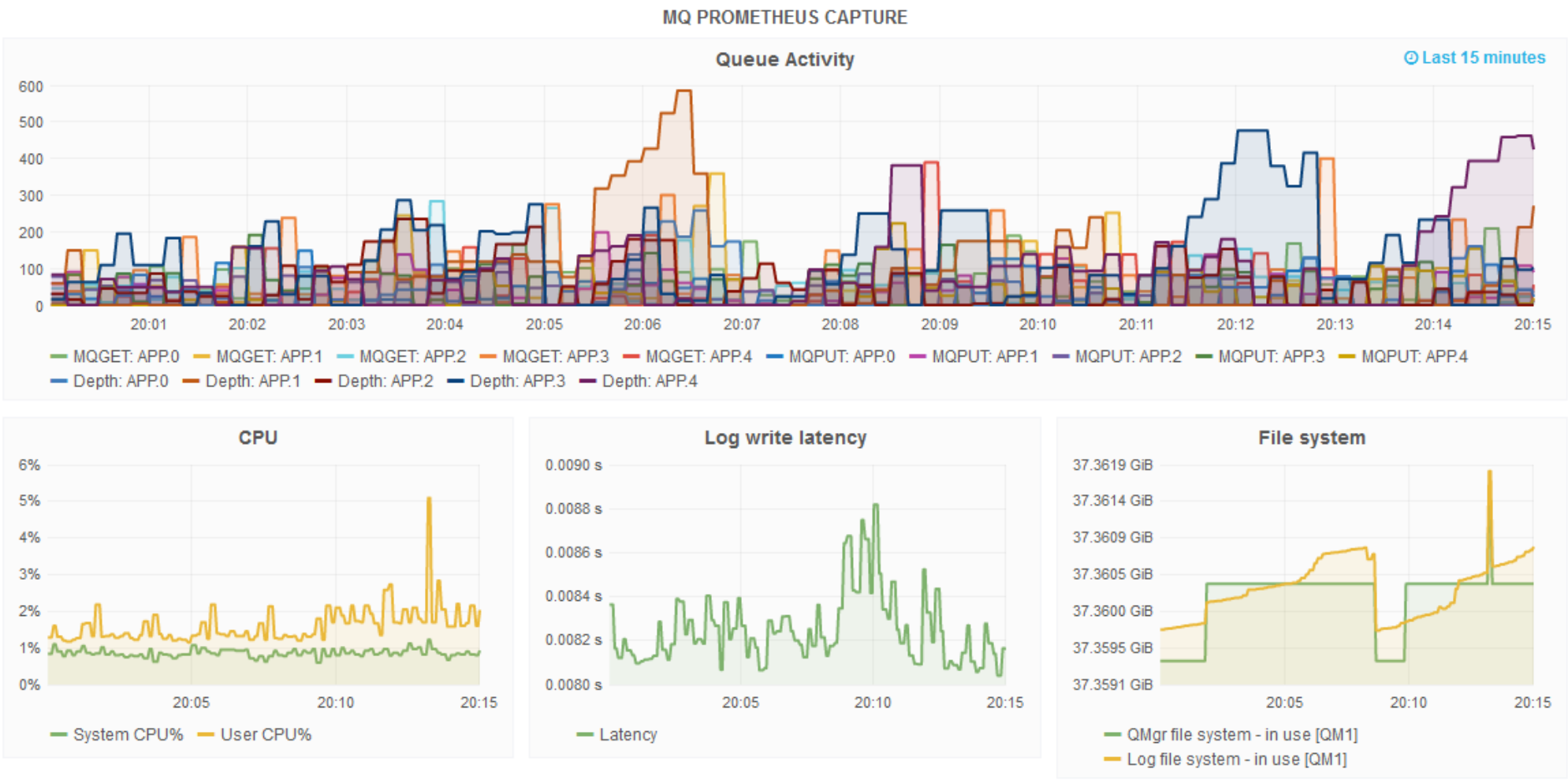
MQ Appliance and MQ V9 – System and qmgr monitors

- The MQ Appliance introduced a new style of event generation
  - Also available in MQ V9
  - Allows multiple consumers of the same information
  - Always non-persistent to avoid capability changes causing migration issues

- A monitoring application subscribes to a well-known (meta-)topic
  - MQSUB("$SYS/MQ/INFO/QMGR/<qmgrname>/Monitor/METADATA")
  - Publications then respond with which real information is available
  - And the monitor can then choose to subscribe to specific topics
  - Possible topics include CPU, Disk and MQI statistics

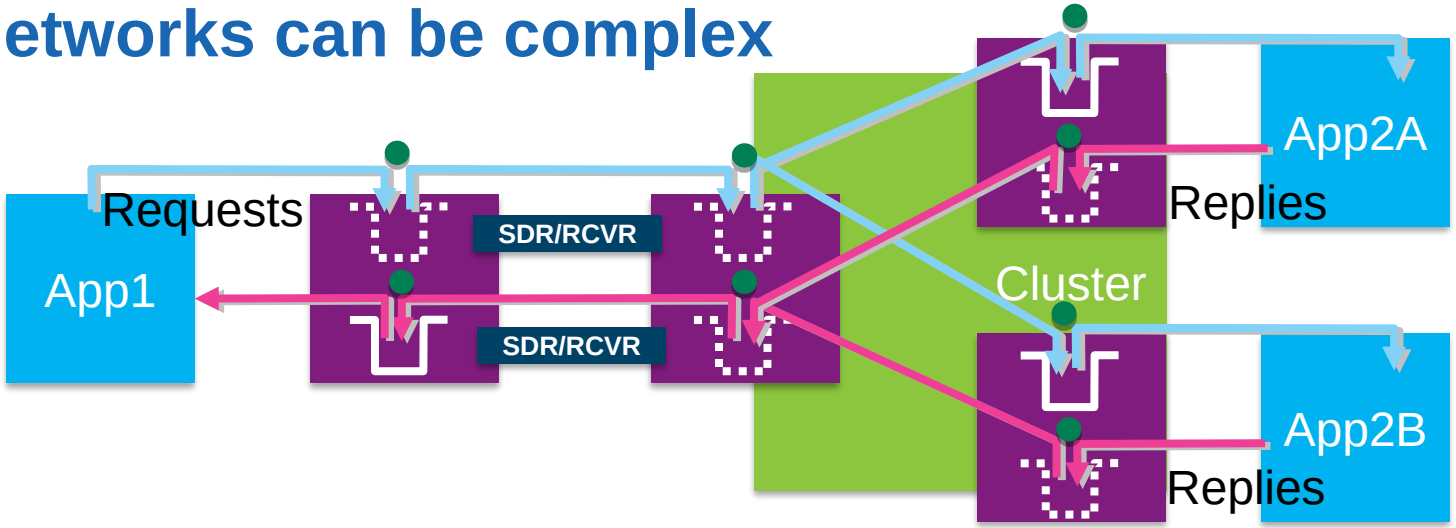
- Each publication is still in PCF format

Feeding a dashboard



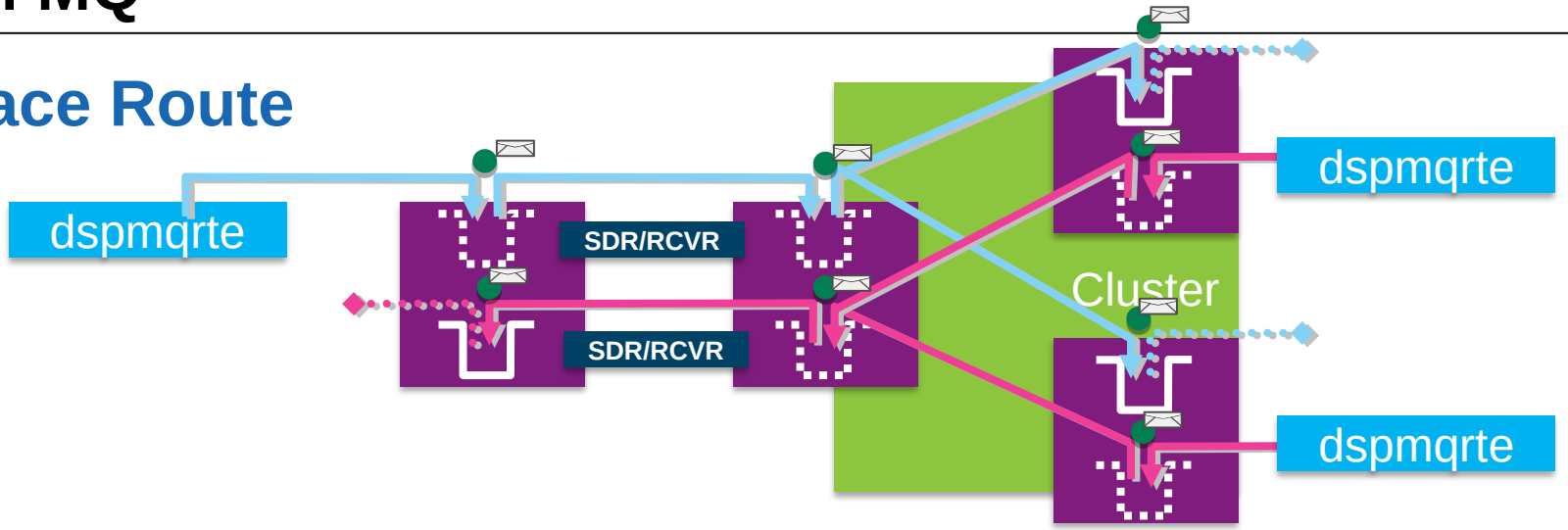
WHERE ARE MESSAGES GOING?

MQ networks can be complex



- At each of the ● dots stuck / mis-sent messages are possible
  - MQOPENs of the wrong Queue / Queue Manager by apps
  - Full queues
  - Stopped channels
  - Stopped apps
  - Incorrectly configured QREMOTE/QALIAS routing objects
  - Cluster membership problems
- The standard problem diagnosis approach
  - Methodically checking channels/queues/DLQs at each point
- Is there anything to speed up this process?

Trace Route



- MQ has the ability to inject **tracer** messages
  - Hidden from applications
  - Generate **activity reports** as they pass through
- Tools are available to trace routes using these reports
  - dspmqrte – command line tool supplied with the product
  - MS0P – Cat 2 SupportPac extension to MQ Explorer
  - *Report messages need to get back to your ReplyQ of your tool*
- Lets you see the path messages ***could*** have taken
  - Test connectivity through the MQ network
  - Test cluster workload balancing
- Can quickly jump you close to the problem
  - The point your trace message veers off in the wrong direction
  - The point the trail goes cold

WHAT ARE THE APPS DOING?

Application Activity Tracing

- Similar infrastructure to Accounting & Statistics
  - PCF messages on SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE
  - Configurable via ini file
    - Can be changed without QMGR restart
    - Configurable detail level can include partial/full message payload
    - Frequency options for tuning
  - Can be enabled on a per-application basis
    - Via MQCONNX flags
    - Via application name
      - MQ V7.5 JMS clients can supply an application name in the CF
- Enables scenarios such as
  - Application audit trail
  - Message duplication
  - Resource usage
    - Which queues or topics are actually being used
  - Problem Determination
    - Which Queue / Queue Manager is the application actually opening
  - Application Coding Standards
    - Does everyone use the MQI in the recommended way
  - And more …

Health warning: Performance impact
http://ow.ly/vA8wB

Application Activity Trace in MQ V9

- Application activity trace enabled through subscriptions rather than queue manager configuration

- Subscribe to meta topics
  - e.g. **$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ApplName/amqsput**
  - Filter by application name, channel or connection id

- When a subscription is created, PCF messages start to flow to the subscriber's queue. When subscription is deleted, messages stop.

Application Activity Trace Sample

Sample provided to demonstrate usage and format output

```
$ amqsput QUEUE1 QMGR1
Sample AMQSPUT0 start
target queue is Q1
Hello
World

Sample AMQSPUT0 end
```

```
$ amqsact -m QMGR1 -a amqsput -w 60
Subscribing to the activity trace topic:
  '$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ApplName/amqsput'

MonitoringType: MQI Activity Trace
…
QueueManager: 'QMGR1'
ApplicationName: 'amqsput'
Application Type: MQAT_UNIX
…
================================================================================
  Tid Date       Time      Operation       CompCode      MQRC  HObj  (ObjName)

  001 2016-04-14 09:56:53  MQXF_CONNX      MQCC_OK       0000  -
  001 2016-04-14 09:56:53  MQXF_OPEN       MQCC_OK       0000  2  (QUEUE1)
  001 2016-04-14 09:56:53  MQXF_PUT        MQCC_OK       0000  2  (QUEUE1)
  001 2016-04-14 09:56:53  MQXF_PUT        MQCC_OK       0000  2  (QUEUE1)
  001 2016-04-14 09:56:53  MQXF_CLOSE      MQCC_OK       0000  2  (QUEUE1)
  001 2016-04-14 09:56:53  MQXF_DISC       MQCC_OK       0000  -
================================================================================
```

Looking at the data with the amqsact sample

```
ALTER QMGR ACTVTRC(ON)              <-  should be tuned via mqat.ini
amqsact -m GATEWAY1 -v
```

```
MQI Operation: 6
  Operation Id: MQXF_PUT
  ApplicationTid: 12451
  OperationDate: '2014-04-09'
  OperationTime: '01:39:48'
  High Res Time: 1397003988665548
  Completion Code: MQCC_OK
  Reason Code: 0
  Hobj: 18225032
  Put Options: 139330
  Msg length: 460
  Recs_present: 0
  Known_dest_count: 1
  Unknown_dest_count: 0
  Invalid_dest_count: 0
  Object_type: MQOT_Q
  Object_name: 'SENDINGAPP.REPLY'
  Object_Q_mgr_name: 'GATEWAY1'
  Resolved_Q_Name: 'SENDINGAPP.REPLY'
  Resolved_Q_mgr: 'GATEWAY1'
  Resolved_local_Q_name: 'SENDINGAPP.REPLY'
  Resolved_local_Q_mgr: 'GATEWAY1'
  Resolved_type: MQOT_Q
  Report Options: 0
  Msg_type: MQMT_DATAGRAM
  Expiry: -1
  Format_name: 'MQHRF2'
  Priority: 4
  Persistence: 0
  Msg_id:
  00000000:  414D 5120 4741 5445 5741 5931 2020 2020  'AMQ GATEWAY1    '
  00000010:  0207 4453 2007 2603                      '..DS .&.        '
  Correl_id:
  00000000:  414D 5120 4741 5445 5741 5931 2020 2020  'AMQ GATEWAY1    '
  00000010:  0207 4453 2007 2203                      '..DS ."."       '
  Reply_to_Q : '                                                   ^D'
  Reply_to_Q_Mgr: '                                                  ^C'
  Coded_char_set_id: 1208
  Encoding: 273
  Put_date: '20140409'
  Put_time: '00394866'
```

Check the options used for coding standards

Check queue name resolution, to find out why messages are going to the wrong place

Track individual messages and request/reply scenarios with Msg_id and Correl_id

HOW CAN I LOOK BACK IN TIME?

What happened to my messages at 2am today?

- Enterprise monitoring solution
  - DLQ alerts, queue depth alerts, channel status alerts
  - Unresolved running units of work
  - Historical MQ monitoring, accounting and stats data
- Application logs from the time of the problem
  - Exceptions, MQ error codes, timeouts
- MQ error logs for all qmgrs that could have been involved
  - Channel errors
  - Authentication issues
- ??? – what else is there

What about the MQ Recovery Log?

- For persistent messages inside transactions
  - MQ logs each operation performed
  - May or may not be logged for operations outside transaction

- Why can't we use this to
  - Look back in time to 2am and see what happened
  - Recover the original payload if the app lost the message
  - See what happened inside long-running units of work
  - Provide a list of operations within the failed business transaction
- MQ documents how you can… *if*
  - You use the text formatting tool provided with MQ (dmpmqlog)
  - The logging is linear so the historical data is available in the tool
  - You follow the right steps to extract data from running qmgrs
  - You do the work to follow through the logs

Dmpmqlog output is readable, but analysis is tedious

```
LOG RECORD - LSN <0:0:954:44817>
**********

HLG Header: lrecsize 873, version 1, rmid 0, eyecatcher HLRH

LogRecdType . . : AQM Put Message (257)
Eyecatcher  . . : ALRH                  Version . . . . : 1
LogRecdLen  . . : 853                   LogRecdOwnr . . : 256    (AQM)
XTranid . . . . : TranType: XA
   XID: formatID 1463898948, gtrid_length 36, bqual_length 54
        gtrid [000001430B3C84EF0000001000000002734721FAD52A950DDA913D08C5C13719A34E164F2]
        bqual
[000001430B3C84EF0000001000000002734721FAD52A950DDA913D08C5C13719A34E164F2000000010000000000
00000000000000001]
QueueName . . . : Not known
Qid . . . . . : {Hash 2147211283, Counter: 5}
ThisLSN . . . . : <0:0:0:0>
PrevLSN . . . . : <0:0:954:43944>

Version . . . . : 4
MapIndex  . . . : 199
PrevLink.Locn . : 102408                PrevLink.Length : 8
PrevDataLink  . : {High 0, Low 103424}
Data.Locn . . . : 103424                Data.Length . . : 613
Data  . . . . . :
00000:  41 51 52 48 04 00 00 00 FF FF FF FF FF FF FF FF    AQRH....ÿÿÿÿÿÿÿÿ
00016:  00 00 00 00 00 00 00 00 C7 00 00 00 02 00 C0 01    ........Ç.....À.
00032:  00 00 00 00 04 00 01 00 A5 00 00 00 00 00 00 00    ........¥.......
00048:  63 00 00 00 41 4D 51 20 49 49 42 30 31 5F 51 4D    c...AMQ IIB01_QM
00064:  20 20 20 20 D9 26 B0 52 20 08 53 F5 30 30 30 30        Ù&°R .Sõ0000
00080:  30 30 39 39 00 00 00 00 00 00 00 00 00 00 00 00    0099............
00096:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00112:  00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00    ................
00128:  00 00 00 00 FF FF FF FF 00 00 00 00 04 00 00 09    ....ÿÿÿÿ........
00144:  00 00 00 00 E3 ED 04 80 10 FD 67 E4 FF FF FF FF    ....ãí.€.ýgäÿÿÿÿ
00160:  4D 44 20 20 01 00 00 00 00 00 00 00 08 00 00 00    MD  ............
00176:  00 00 00 00 11 01 00 00 B8 04 00 00 4D 51 48 52    ........¸...MQHR
00192:  46 32 20 20 04 00 00 00 01 00 00 00 20 20 20 20    F2  ........
00208:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00224:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00240:  20 20 20 20 20 20 20 20 20 20 20 20 49 49 42 30                IIB0
00256:  31 5F 51 4D 20 20 20 20 20 20 20 20 20 20 20 20    1_QM
00272:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00288:  20 20 20 20 20 20 20 20 20 20 20 20 4D 55 53 52                MUSR
00304:  5F 4D 51 41 44 4D 49 4E 16 01 05 15 00 00 00 64    _MQADMIN.......d
00320:  20 3E AC 57 48 B3 09 B8 71 B0 4C F2 03 00 00 00     >¬WH..¸q.Lò....
00336:  00 00 00 00 00 00 00 0B 20 20 20 20 20 20 20 20    ........
00352:  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00368:  20 20 20 20 20 20 20 20 1C 00 00 00 57 65 62 53            ....WebS
00384:  70 68 65 72 65 20 4D 51 20 43 6C 69 65 6E 74 20    phere MQ Client
00400:  66 6F 72 20 4A 61 76 61 32 30 31 33 31 32 31 39    for Java20131219
00416:  31 34 32 32 33 32 32 32 20 20 20 20 00 00 00 00    14223222    ....
00432:  A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ¤...............
00448:  52 46 48 20 00 00 00 02 00 00 00 A4 00 00 01 11    RFH .......¤....
00464:  00 00 04 B8 4D 51 53 54 52 20 20 20 00 00 00 00    ...¸MQSTR   ....
00480:  00 00 04 B8 00 00 00 20 3C 6D 63 64 3E 3C 4D 73    ...¸... <mcd><Ms
00496:  64 3E 6A 6D 73 5F 74 65 78 74 3C 2F 4D 73 64 3E    d>jms_text</Msd>
00512:  3C 2F 6D 63 64 3E 20 20 00 00 00 58 3C 6A 6D 73    </mcd>  ...X<jms
00528:  3E 3C 44 73 74 3E 71 75 65 75 65 3A 2F 2F 2F 51    ><Dst>queue:///Q
00544:  31 3C 2F 44 73 74 3E 3C 54 6D 73 3E 31 33 38 37    1</Dst><Tms>1387
00560:  34 36 32 39 35 32 32 32 36 3C 2F 54 6D 73 3E 3C    462952226</Tms><
00576:  43 69 64 3E 30 30 30 30 30 30 39 39 3C 2F 43 69    Cid>00000099</Ci
00592:  64 3E 3C 44 6C 76 3E 32 3C 2F 44 6C 76 3E 3C 2F    d><Dlv>2</Dlv></
00608:  6A 6D 73 3E 61                                     jms>a
```

Ordered unique IDs for each record (LSN)

A set of documented record types

Transaction information with XIDs, or re-used MQ transaction IDs

MQMD header data at discoverable offsets in the hex of a message Put

The message payload itself

- Wouldn't it be easier to let the computer do the tedious bit?

Check out dmpmqlog scraper tool

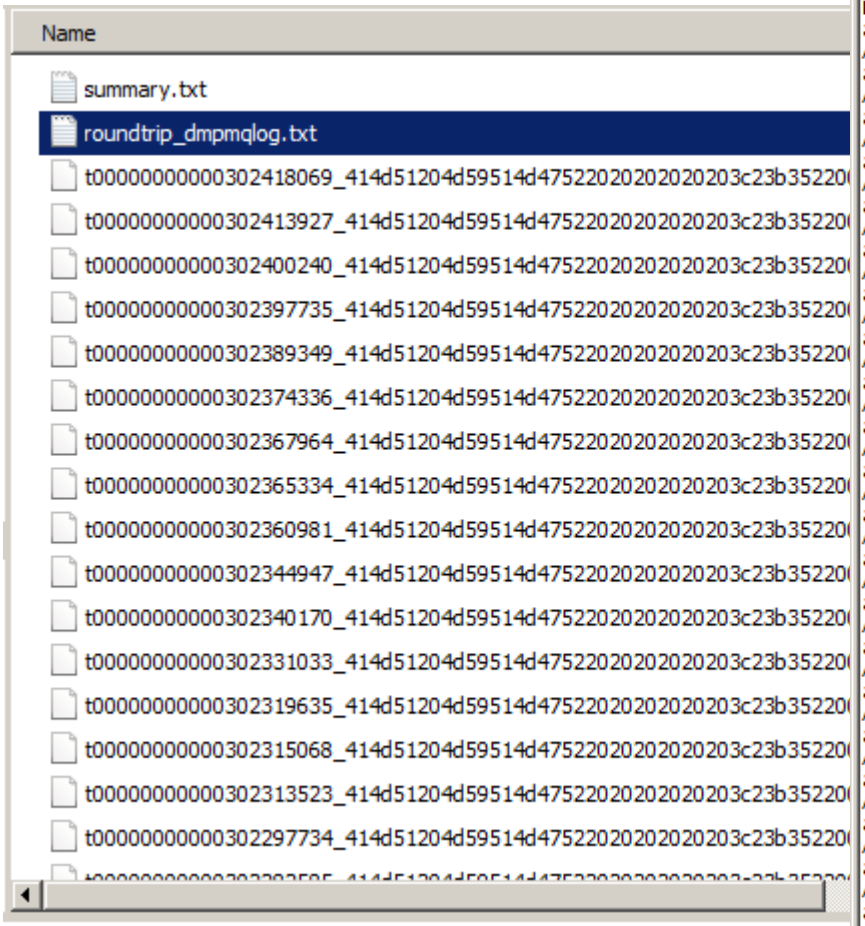
- Takes the tedium out of analysing the output from dmpmqlog
- Created by Peter Broadhurst
- Download from http://www.ibm.com/support/docview.wss?uid=swg21660642

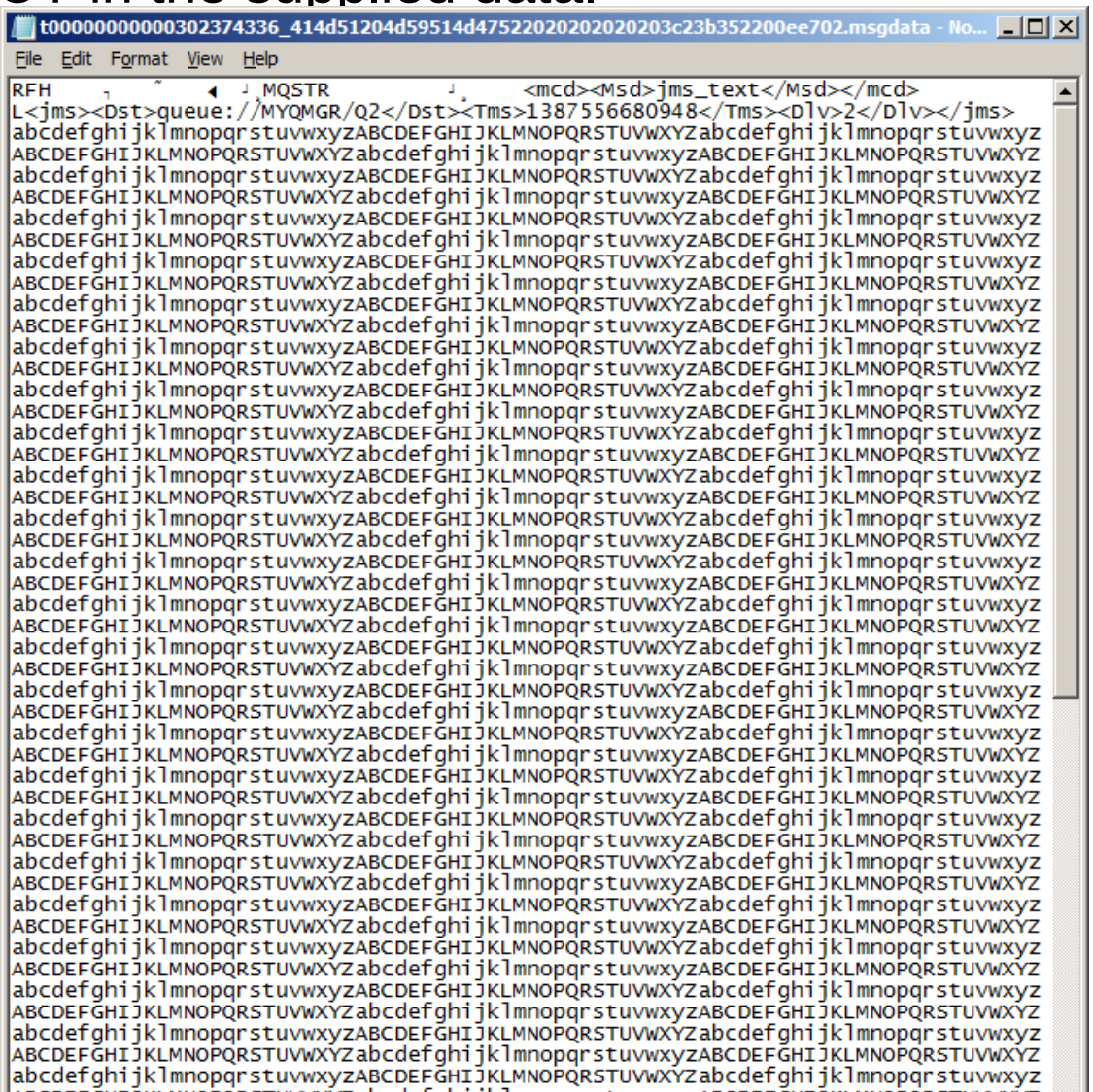
```
java -jar dmpmqlog.scraper-20151201.jar -b little-endian -i dmpmqlog.txt -o .
```

- Generates file per message PUT in the supplied data.
- Summary file

dmpmqlog scraper tool output

- Generates file per message PUT in the supplied data.
- Summary file





Summary

- Lots of tools in your MQ toolbox!
- On-line Status Commands
  - DISPLAY CONN
  - DISPLAY QSTATUS
  - DISPLAY CHSTATUS
- Cluster monitoring – AMQSCLM
- Off-line Statistics and Accounting
  - amqsmon and MS0P to view
- Tracking
  - Trace-route
  - Application Activity Trace
- MQ Recovery Logs
  - dmpmqlog scraper

Any questions?