Integration Technical Conference 2019

M08: Monitoring and problem determination of an MQ z/OS system

Matt Leming
lemingma@uk.ibm.com

IBM **Cloud**

Agenda – the MQ Toolbox

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

NB: Capabilities differ in form between Distributed and z/OS, this presentation is z/OS focused

What’s connected?

DISPLAY CONN

```
!MQ21 DISPLAY CONN(*) TYPE(HANDLE) WHERE(OBJNAME EQ WLMMDB.REQUEST)
```

```
CSQM201I !MQ21 CSQMDRTC  DIS CONN DETAILS
   CONN(577C425321295301)
   EXTCONN(414D5143474154455741593120202020)
   TYPE(HANDLE)

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

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

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

```
!MQ21 DISPLAY CONN(577C425321295301) ALL
```

```
CSQM201I !MQ21 CSQMDRTC  DIS CONN DETAILS
   CONN(577C425321295301)
   EXTCONN(414D5143474154455741593120202020)
   TYPE(CONN) URDISP(QMGR) CONNOPTS(MQCNO_SHARED_BINDING)
   UOWLOGDA( ) UOWLOGTI( )
   UOWSTDA(2014-04-08) UOWSTTI(13.24.00) UOWSTATE(ACTIVE)
   NID( )
   EXTURID(XA_FORMATID[DSAW]
XA_GTRID[00000145414B8AB40000000104DF48FC00010203040506070809000102030405060708O9]
XA_BQUAL[00000145414B8AB40000000104DF48FC0001020304050607080900010203040506070809000
000000000001])
   QMURID(00000000036A70C4)
   URTYPE(XA) ASTATE(NONE)
   USERID(MLEMING) APPLTAG(jms/GATEWAY1_CF)
   ASID(00D5) APPLTYPE(CHINIT) APPLDESC(IBM MQ Channel Initiator)
   CHANNEL(WAS.CLIENTS) CONNAME(127.0.0.1)
```

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

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

ASID for bindings based applications

Channel name + IP help identify client apps.

DISPLAY CHSTATUS

```
!MQ21 DISPLAY CHSTATUS(*) ALL
```

```
CSQM201I !MQ21 CSQMDRTC  DIS CHSTATUS DETAILS
   CHANNEL(WAS.CLIENTS)
   CHLDISP(PRIVATE) CONNAME(127.0.0.1) CURRENT
   CHLTYPE(SVRCONN) STATUS(RUNNING) SUBSTATE(RECEIVE)
   CURSHCNV(1) MAXSHCNV(1)
   LSTMSGTI(15.26.59) LSTMSGDA(2014-04-08) MSGS(6)
   BYTSSENT(2456) BYTSRCVD(2296)
   CHSTATI(15.26.59) CHSTADA(2014-04-08)
   BUFSSENT(13) BUFSRCVD(17)
   MONCHL(OFF) STOPREQ(NO) KAINT(360) QMNAME(MQ21)
   RAPPLTAG(jms/GATEWAY1_CF)
   SECPROT(TLSV12) SSLCERTI(CN=ExampleCA,O=Example)
   SSLCERTU(MQSX00) SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
   SSLRKEYS(0) SSLKEYTI( ) SSLKEYDA( )
   SSLPEER(SERIALNUMBER=53:43:FD:D6,CN=ExampleApp1,O=Example)
   RPRODUCT(MQJM) RVERSION(09010200)
   MCAUSER(MLEMING) LOCLADDR( )
   MAXMSGL(4194304)
   COMPHDR(NONE NONE) COMPMSG(NONE NONE)
   COMPRATE(0,0)                         COMPTIME(0,0)
   HBINT(5)
```

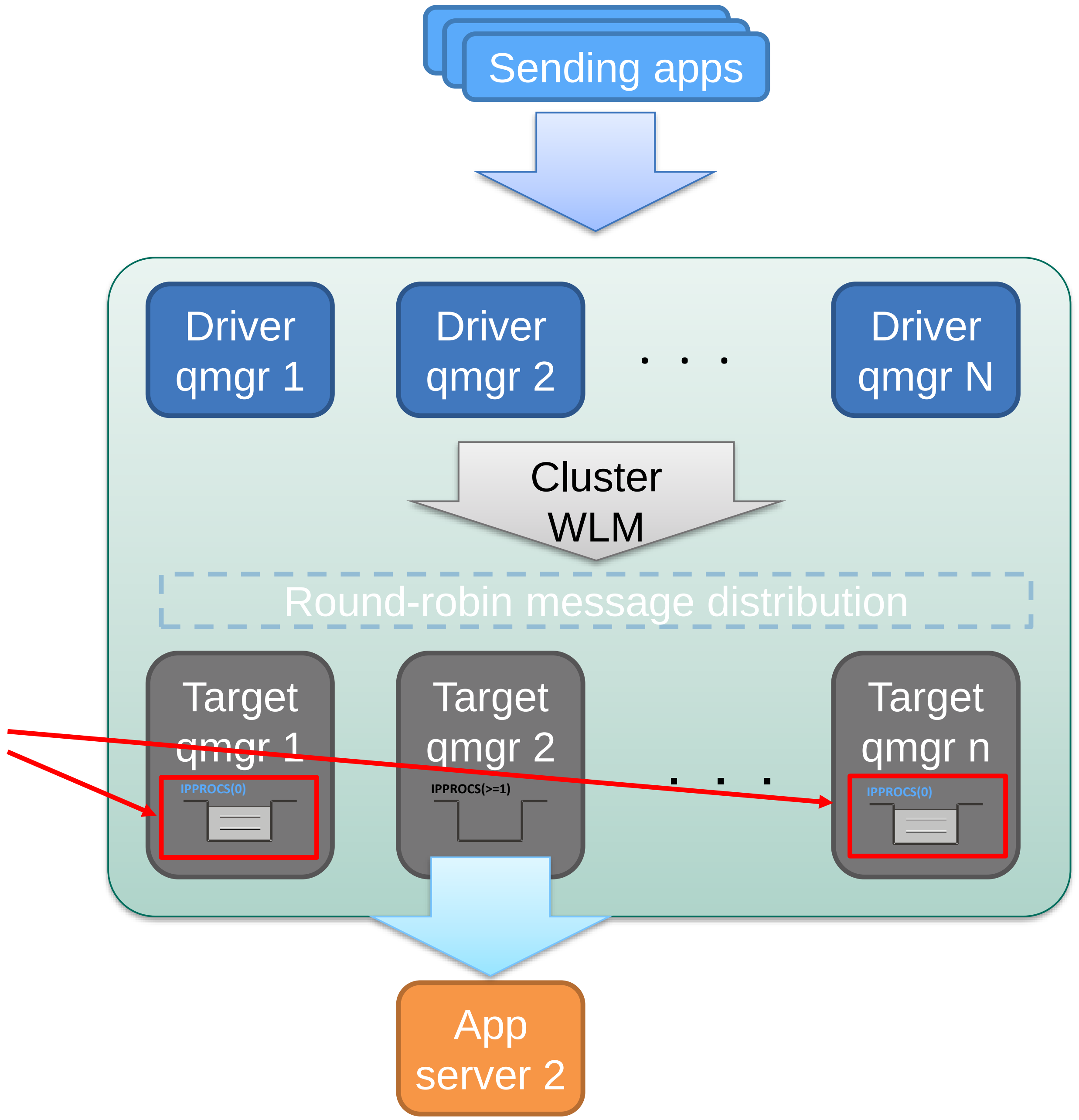
Note that multiple CONN might share one SVRCONN channel instance

See SSLPEER information not in DIS CONN

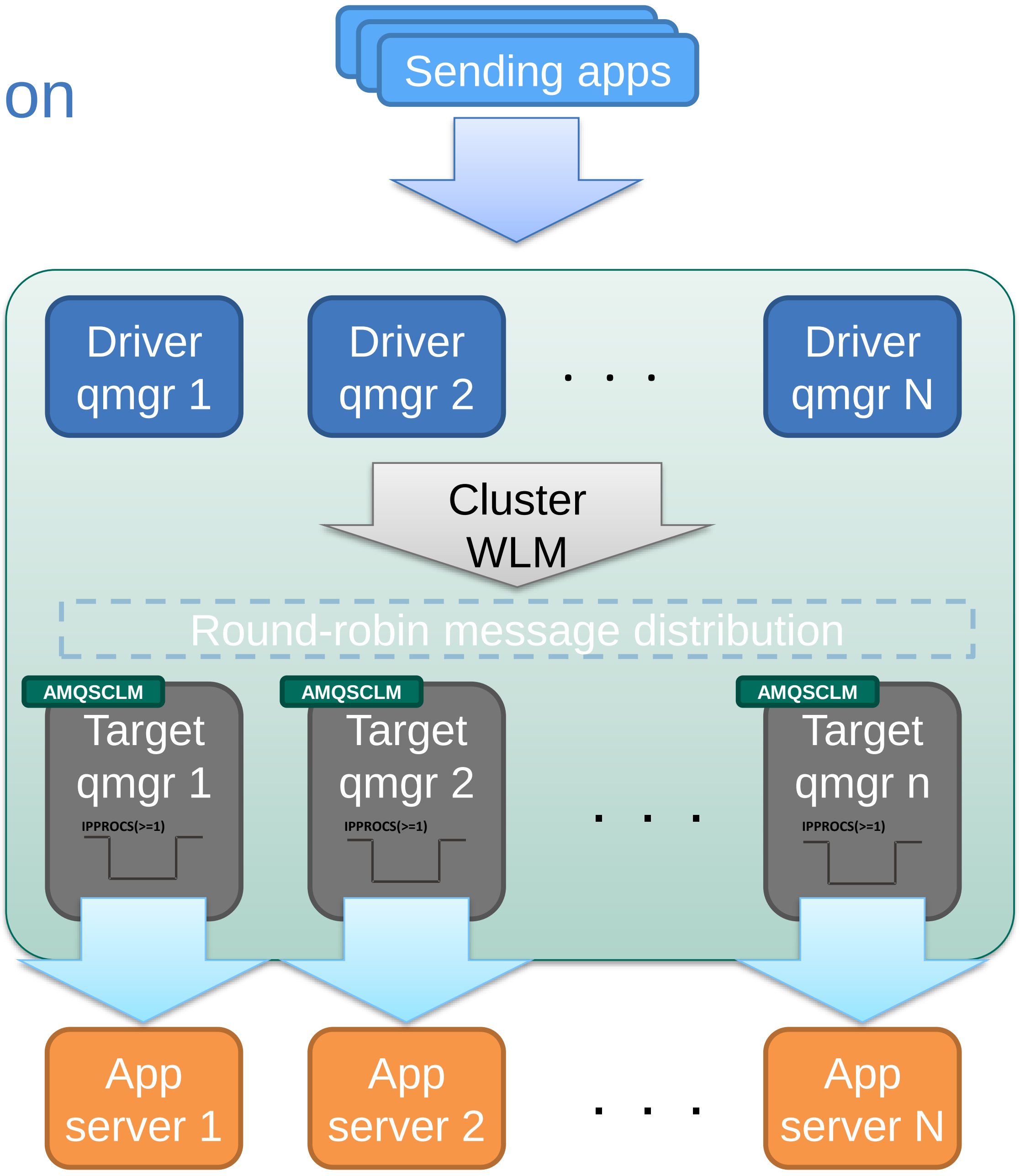
Rerouting messages if **no-one** is connected

Cluster WLM

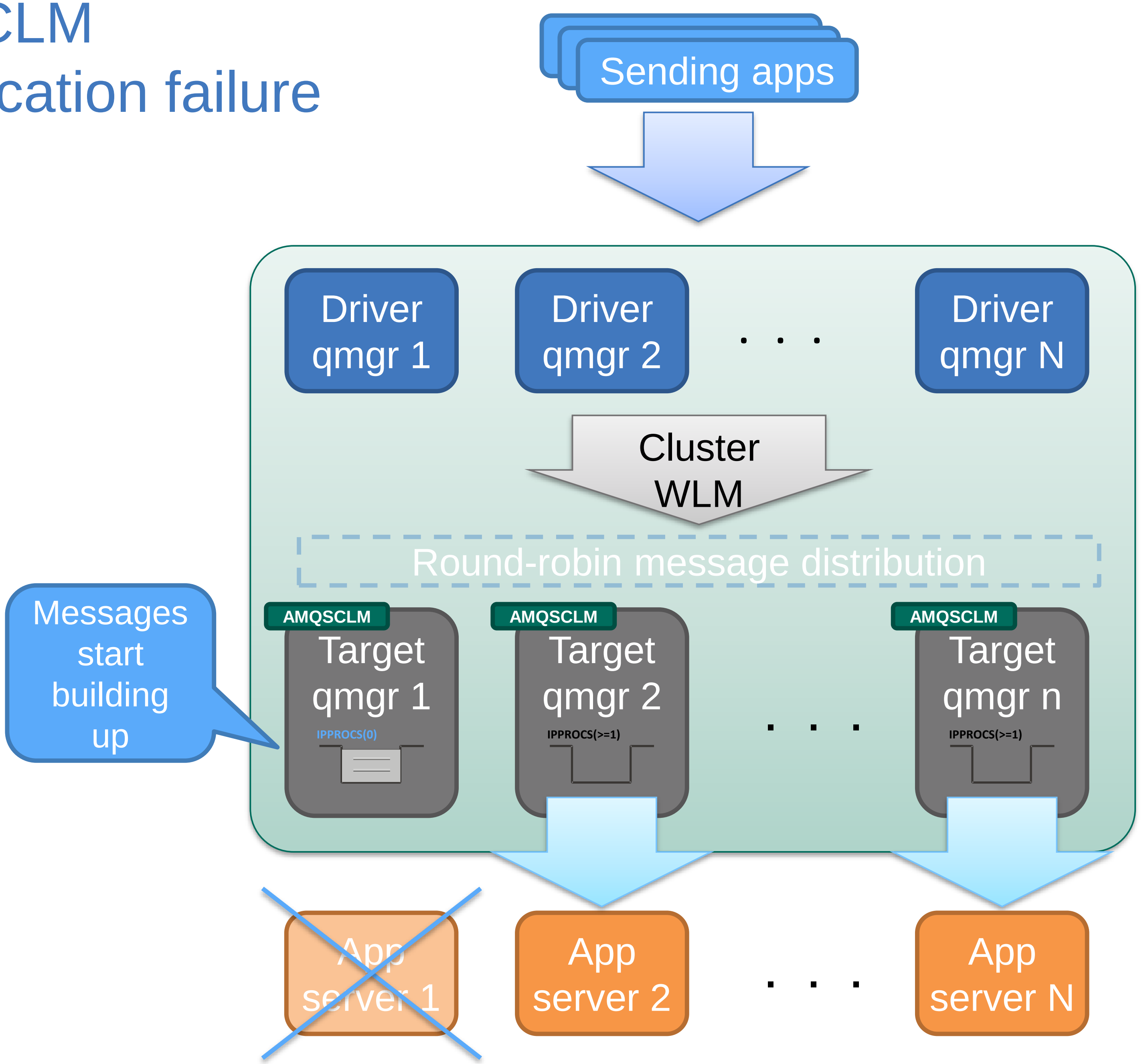
- Cluster WLM algorithm doesn't check whether consuming applications are connected to cluster queues
  - I.e. whether something is getting messages from them
- You need to ensure that applications are consuming from all instances of a clustered queue to prevent messages building up
- However, there is an alternative: AMQSCLM



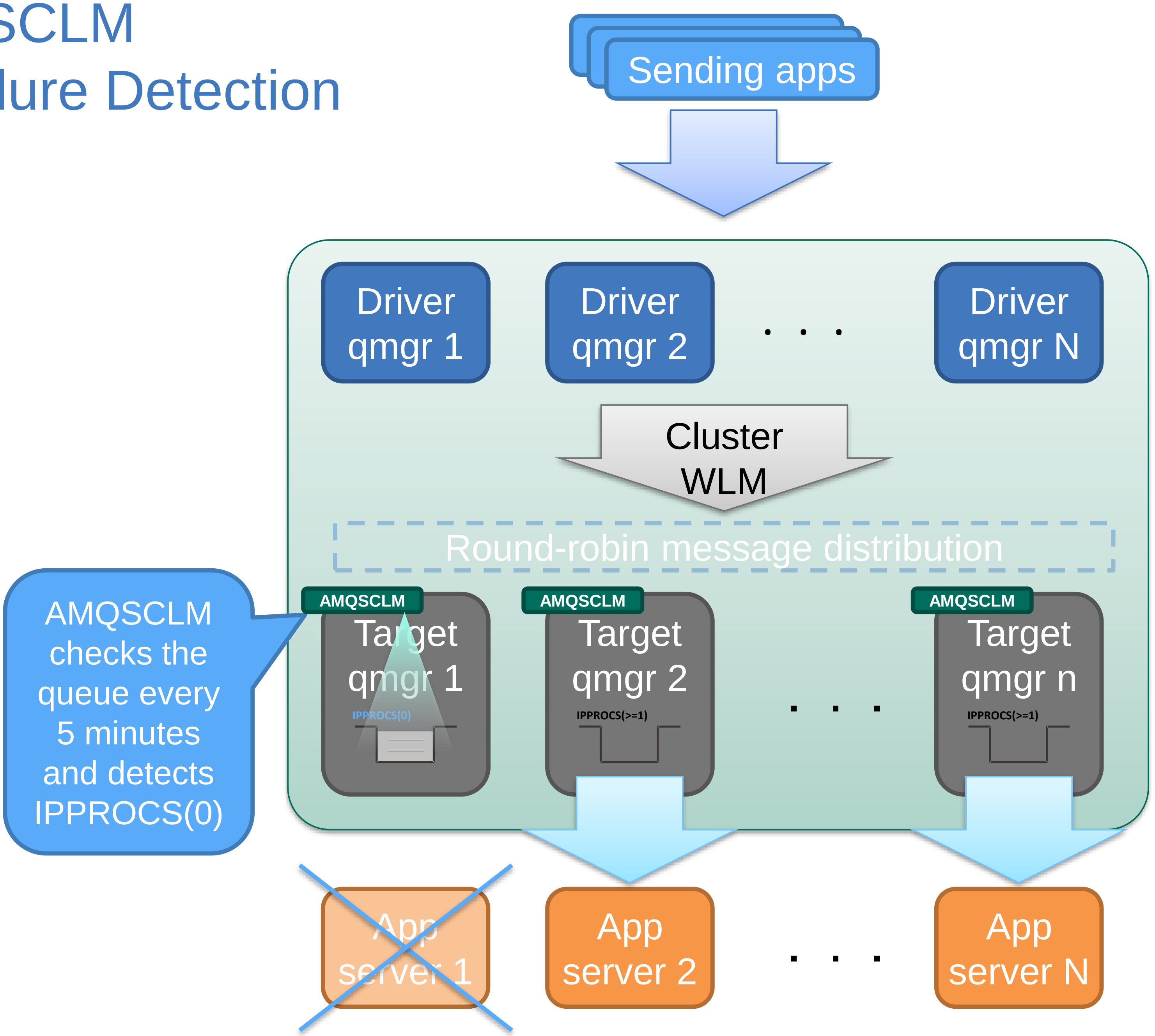
AMQSCLM
1) Normal operation



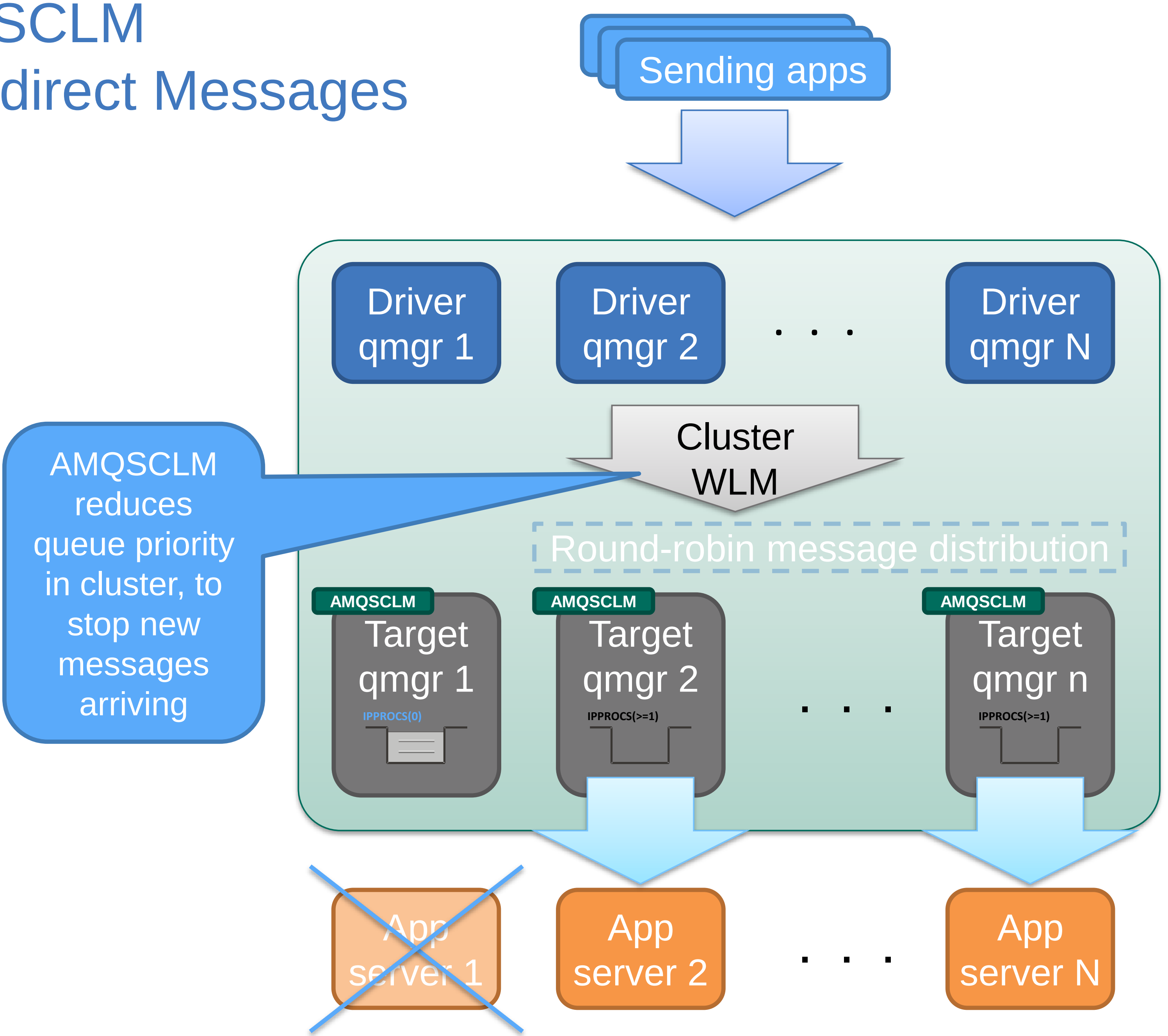
AMQSCLM
2) Application failure



AMQSCLM
3) Failure Detection

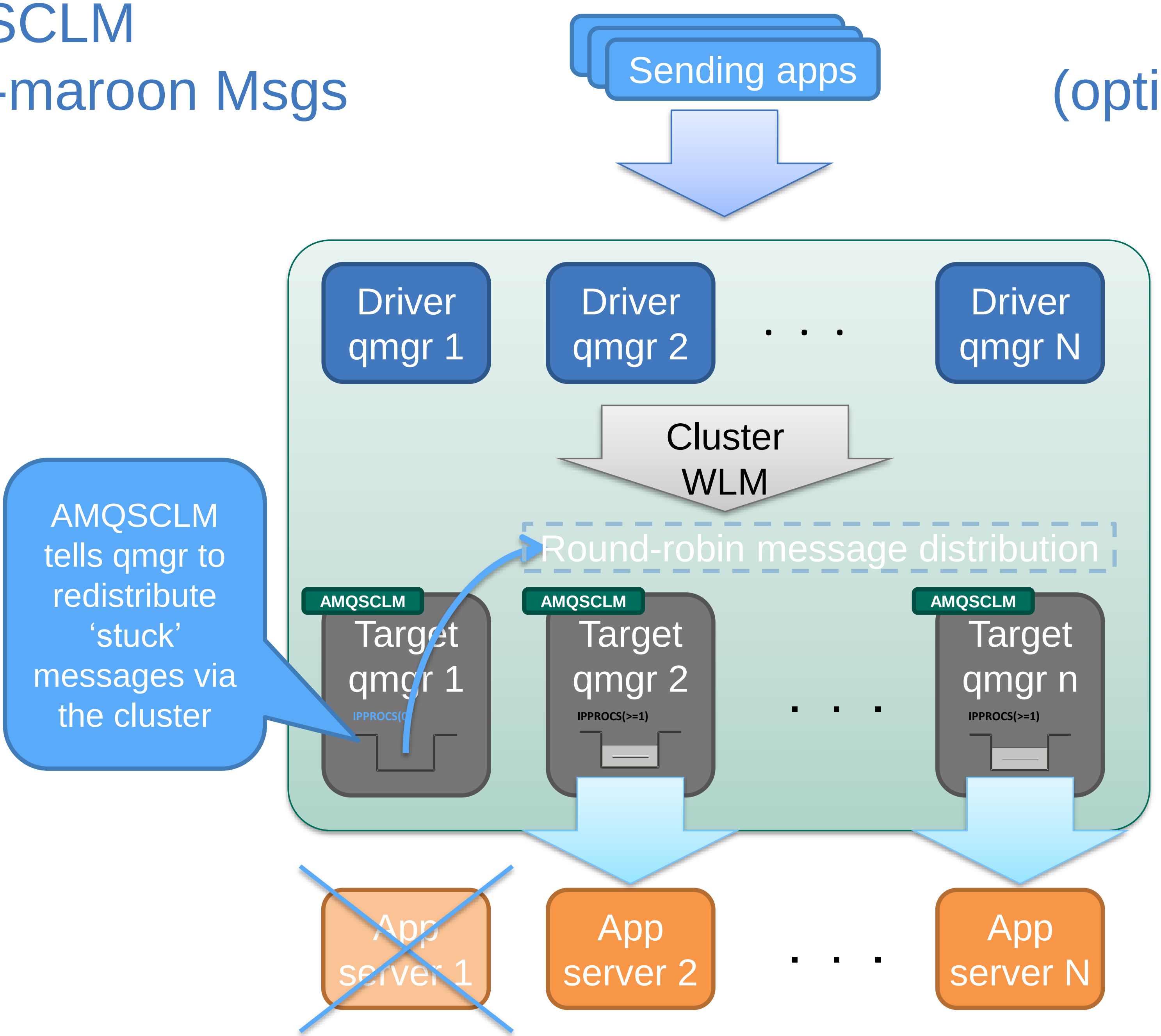


AMQSCLM
4) Redirect Messages



AMQSCLM
5) Un-maroon Msgs

(optional)



AMQSCLM summary

- The cluster queue monitoring sample program (AMQSCLM)
- Shipped with the distributed product as a sample
  - Precompiled
  - Source code
  - Not shipped on z/OS, but distributed source code can be compiled and used on z/OS
- More information here: http://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.dev.doc/q024620_.htm

Are messages flowing?

Real-time/online monitoring – queues

NB: Off by default, so consider switching it on at the queue manager level!

NB: HIGH/MEDIUM/LOW doesn't matter for queues

- Set detail level for queue manager. Override for individual queues

```
!MQ21 ALTER QMGR MONQ(MEDIUM)
!MQ21 ALTER QLOCAL(QUEUE1) MONQ(HIGH)
!MQ21 ALTER QLOCAL(QUEUE2) MONQ(OFF)
```

- Gives live view of application responsiveness

```
!MQ21 DIS QSTATUS(QUEUE1) ALL
```

```
CSQM201I !MQ21 CSQMDRTC  DIS QSTATUS DETAILS
   QSTATUS(QUEUE1) TYPE(QUEUE)
   OPPROCS(5) IPPROCS(3) CURDEPTH(16)
   UNCOM(YES) MONQ(HIGH)
   QTIME(10101414, 10101414)
   MSGAGE(112)
   LPUTTIME(17.12.16) LPUTDATE(2019-04-08)
   LGETTIME(17.05.59) LGETDATE(2019-04-08)
   QSGDISP(QMGR)
```

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

Bear in mind that CURDEPTH includes committed and uncommitted messages. Look at UNCOM!

Age in **seconds** of the oldest message on the queue

Timestamps of last PUT/GET to check for recent activity

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

NB: for shared queues these values are only this queue manager's view, apart from CURDEPTH & MSGAGE

Real-time/online monitoring – channels

```
!MQ21 ALTER QMGR MONCHL(MEDIUM) MONACLS(MEDIUM)
!MQ21 ALTER CHANNEL(CLUSTER1.QM1) CHLTYPE(CLUSRCVR) MONCHL(HIGH)
```

- Gives live view of channel throughput

```
!MQ21 DIS CHSTATUS(TO.GWY2) ALL
```

```
CSQM201I !MQ21 CSQMDRTC  DIS CHSTATUS DETAILS
   CHSTATUS(TO.GWY2) CHLDISP(PRIVATE) XMITQ(GATEWAY2) CONNAME(127.0.0.1(1522))
   CURRENT CHLTYPE(SDR) STATUS(RUNNING) SUBSTATE(RECEIVE) INDOUBT(YES)
   LSTSEQNO(11773) LSTLUWID(0107445310001B33)
   CURMSGS(50) CURSEQNO(11823) CURLUWID(D5F6358CE52A358A)
   LSTMSGTI(17.49.51) LSTMSGDA(2014-04-08)
   MSGS(1580) BYTSSENT(1192330) BYTSRCVD(1748) BATCHES(52)
   CHSTATI(17.49.03) CHSTADA(2019-04-08)
   BUFSSENT(1616) BUFSRCVD(55)
   LONGRTS(999999999) SHORTRTS(180)
   MONCHL(MEDIUM)
   XQTIME(545784,3929968)
   NETTIME(137538,29555)
   EXITTIME(0,0)
   XBATCHSZ(20,17)
   COMPTIME(0,0) COMPRATE(0,0)
   STOPREQ(NO) KAINT(360)
   QMNAME(MQ21) RQMNAME(QWY2)
   SECPROT(NONE) SSLCERTI( ) SSLCERTU(MQSX00) SSLCIPH( )
   SSLRKEYS(0) SSLKEYTI( ) SSLKEYDA( )  SSLPEER( )
   RPRODUCT(MQMV) RVERSION(09010200)
   STATCHL(OFF) LOCLADDR(127.0.0.1(53557))
   BATCHSZ(50) MAXMSGL(4194304)
   COMPHDR(NONE NONE) COMPMSG(NONE NONE) HBINT(5) NPMSPEED(FAST))
```

Last time a message was sent over the channel

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

Short/long term view of the network time to send a request and receive a response. Calculated during batch completion

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

NB: Off by default, so consider switching it on at the queue manager level!

NB: HIGH/MEDIUM/LOW doesn't matter for channels on z/OS

NB: Can be expensive for large numbers of channels due to impact on BP 0

Accounting and statistics overview

- Realtime information isn't always sufficient. It is useful to be able to spot trends over time, and see how the system is currently deviating from the average
- Queue manager can periodically output monitoring data to SMF
- SMF 115: Statistics
  - High level queue manager wide information. E.g. memory usage, log information, locking information, etc
  - Most useful from perspective of this presentation is likely to be message manager information which gives information about counts of MQI operations

- SMF 116: Accounting
  - More detailed…
  - Per task information: log usage, CF usage, CPU time, some MQI information
  - Per queue information: gives information on MQI usage, open/close time, more detailed log information
  - Per channel information: similar to output from DIS CHSTATUS
  - Can generate a lot of data on busy queue managers

- All this information is in binary format. Assembler and C mapping are provided

Example SMF 116 header

```
000000 01A40000 5E740035 61240100 223FD4E5  *....;.../.....MV*
000000 F4F1D4D8 F2F10000 F9F1F200 00000134  *41MQ21..912.....*
000000 00700001 00000054 00B00001 00000104  *................*
000000 00300001 00000000 00000000 00000000  *................*
000000 00000000 00000000 00000000 00000000
```

Accounting and statistics overview

- To collect:
  - Enabled SMF to collect 115/116 data
  - Enable the queue manager to generate it from start up either via CSQ6SYSP
  - Or via MQ's START TRACE command
  - Fine grained control available on per queue/channel basis

- Then dump out records from SMF

- Tooling is available to work with records
  - CSQ4SMFD: basic sample C program
  - MP1B: a SupportPac, see later slides
  - MQSMFCSV: simple formatter to convert SMF records into csv. Can then be imported into spreadsheets and databases

    https://github.com/ibm-messaging/mq-smf-csv
  - IBM OMEGAMON for messaging
  - Others…

```
CSQY103I !MQ21 SMFACCT= YES (F0000000),
                     SMFSTAT=YES (F0000000), STATIME=1
```

```
!MQ21 START TRACE(ACCT) CLASS(*)
```

```
!MQ21 ALTER QMGR  ACCTQ(ON)
!MQ21 ALTER QLOCAL(QUEUE1)  ACCTQ(QMGR)

!MQ21 ALTER CHANNEL(TO.GWY2) STATCHL(HIGH)
        CHLTYPE(SDR)
```

A gotcha: for chinit stats and accounting you have to issue:
START TRACE(STAT) CLASS(4)
START TRACE(ACCTG) CLASS(4)

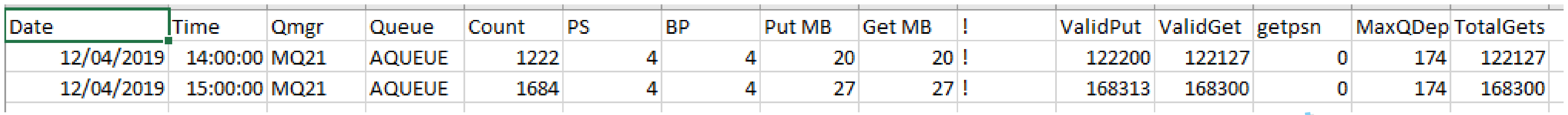
MP1B

- Unsupported SupportPac: https://www-01.ibm.com/support/docview.wss?uid=swg24005907
- Kept reasonably up to date by MQ development team
- Uses:
  - Summary of transactions, jobs and channels
  - Summary of queues being used by application
  - List of “high use” queues
  - …
- Will detect potential issues (based on our past experience) and output them as a summary
- Can generate csv files for import into spreadsheets which allows for graphs of activity over time to be generated
- Lots of options to customize the level of output

- Much more useful than CSQ4SMFD
- Used by lots of customers

MP1B example

- QSUML report gives a summary of activity against local queues
- Useful for understanding your expected workload against a set of queues. You can then spot anomalies



Date	Time	Qmgr	Queue	Count	PS	BP	Put MB	Get MB	!	ValidPut	ValidGet	getpsn	MaxQDep	TotalGets
12/04/2019	14:00:00	MQ21	AQUEUE	1222	4	4	20	20	!	122200	122127	0	174	122127
12/04/2019	15:00:00	MQ21	AQUEUE	1684	4	4	27	27	!	168313	168300	0	174	168300

Number of times message data wasn't in buffer pool

MP1B Example

To dig further you can use the TASK report which digs further into accounting data

```
16305 MV41,MQ21,2019/04/12,15:19:05,VRM:913,
16305 MQ21 MOVER Jobname:MQ21CHIN Userid:MQSX00
16305 Channel SYSTEM.DEF.SVRCONN   9.174.27.35
16305 Start time Apr 12 15:18:39 2019 Started in a different time interval
16305 Interval   Apr 12 15:18:39 2019 - Apr 12 15:18:41 2019 : 1.549716 seconds
16305 Other reqs : Total ET           0.000024 Seconds
16305 Other reqs : Total CPU           0.000024 Seconds
16305 Commit count                   101
16305 Commit avg elapsed time        628 uS
16305 Backout count                    1
16305 Backout avg elapsed time        15 uS
16305 Backout avg CPU time            15 uS
16305 MQCTL count                    301
16305 MQCTL avg elapsed time           0 uS
16305 MQCTL avg CPU time               0 uS
16305 Total suspend time              0.062778 Seconds
16305 Open name                       AQUEUE
16305 Queue type:QLocal               AQUEUE
16305 Page set ID                     4        AQUEUE
16305 Buffer pool                     4        AQUEUE
16305 Get count                       100      AQUEUE
16305 Get avg elapsed time             42 uS   AQUEUE
16305 Get avg CPU time                 41 uS   AQUEUE
16305 Get valid destructive           100      AQUEUE
16305 Put count                       100      AQUEUE
16305 Put avg elapsed time            519 uS   AQUEUE
16305 Put avg CPU time                 75 uS   AQUEUE
….
```

Lots of backouts is normally a bad sign

Start of per queue information

Discrepancy between these two is often a sign of messages not being in buffer pool so being got from pageset

Discrepancy between these two for persistent messages is likely to be because of logging. For non-persistent messages it could be because the buffer pool is full

MP1B Example

Then go look at channel accounting data too.

```
Jobname: MV41,MQ21,2019/04/12,16:44:14,VRM:913
SMF interval start  local time   2019/04/12,16:43:24
SMF interval end    local time   2019/04/12,16:44:14
SMF interval start  GMT          2019/04/12,15:43:24
SMF interval end    GMT          2019/04/12,15:44:14
SMF interval duration        49.332448 seconds
SYSTEM.DEF.SVRCONN   9.174.27.35    Connection name                        9.174.27.35
SYSTEM.DEF.SVRCONN   9.174.27.35    Channel disp                           PRIVATE
SYSTEM.DEF.SVRCONN   9.174.27.35    Channel type                           SVRCONN
SYSTEM.DEF.SVRCONN   9.174.27.35    Channel status                         RUNNING
SYSTEM.DEF.SVRCONN   9.174.27.35    Remote qmgr/app                        LoadQueueLoopJMSWIthDifferen
SYSTEM.DEF.SVRCONN   9.174.27.35    Channel started date & time            2019/04/12,15:43:26
SYSTEM.DEF.SVRCONN   9.174.27.35    Channel status collect time            2019/04/12,15:44:14
SYSTEM.DEF.SVRCONN   9.174.27.35    Active for                             47 seconds
SYSTEM.DEF.SVRCONN   9.174.27.35    Last MQI request time                  2019/04/12,15:44:13
SYSTEM.DEF.SVRCONN   9.174.27.35    Dispatcher number                      1
SYSTEM.DEF.SVRCONN   9.174.27.35    Number of MQI requests                 20,853
SYSTEM.DEF.SVRCONN   9.174.27.35    Number of persistent messages          0
SYSTEM.DEF.SVRCONN   9.174.27.35    Buffers sent                           20,855
SYSTEM.DEF.SVRCONN   9.174.27.35    Buffers received                       21,259
SYSTEM.DEF.SVRCONN   9.174.27.35    Current shared connections             4
SYSTEM.DEF.SVRCONN   9.174.27.35    Message data                           0     0  B
SYSTEM.DEF.SVRCONN   9.174.27.35    Persistent message data                0     0  B
SYSTEM.DEF.SVRCONN   9.174.27.35    Non persistent message data            0     0  B
SYSTEM.DEF.SVRCONN   9.174.27.35    Total bytes sent                       7,988,504  7801 KB
SYSTEM.DEF.SVRCONN   9.174.27.35    Total bytes received                   5,611,764  5480 KB
SYSTEM.DEF.SVRCONN   9.174.27.35    Bytes received/message                 269   269  B
SYSTEM.DEF.SVRCONN   9.174.27.35    Bytes sent/message                     383   383  B
….
```

Performance considerations

- Capturing SMF data has a cost: CPU and storage of the records

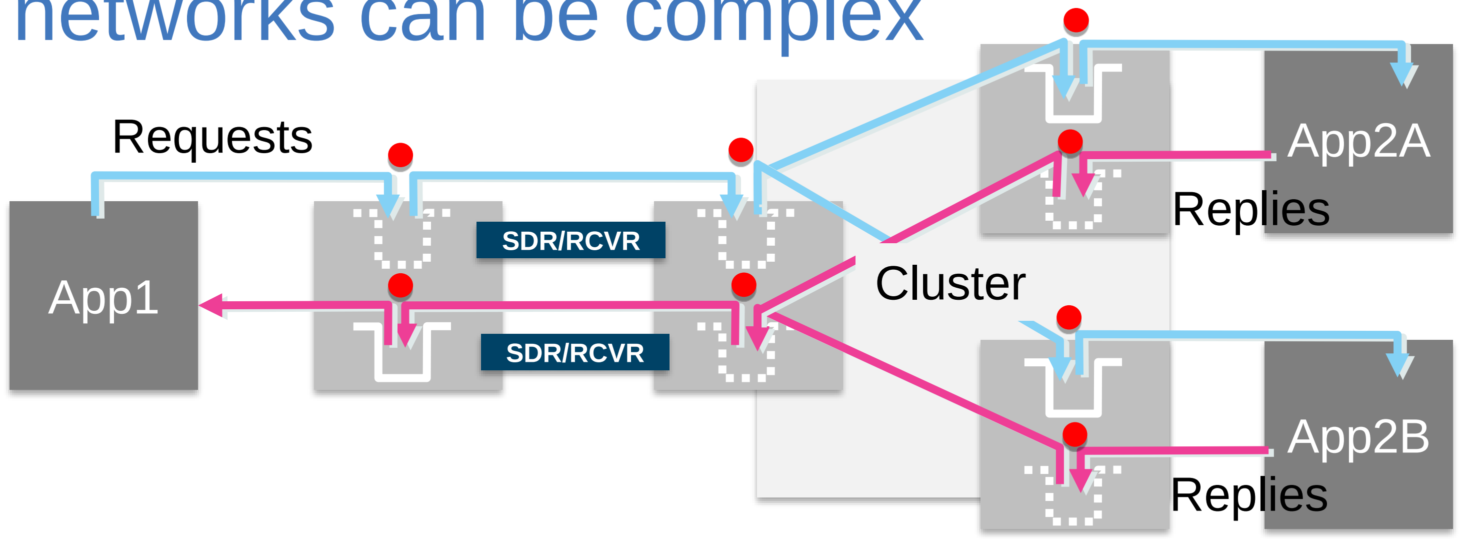
- SMF 115 (queue manager stats): **negligible** -> always have it on
- SMF 116 (queue manager accounting class 1): **0.5%** increase in transaction cost in our workloads
  - High level API count, and CPU usage per thread
- SMF 116 (queue manager accounting, class 1 & 4): **3.3%** increase in transaction cost in our workloads
  - Per thread and queue usage information
- SMF 115 + 116 (chinit): **1-2%** increase in transaction cost in our sample workloads

- This cost needs to be compared to the benefit of having the data
- Consider sizing your environment to assume that MQ SMF data of all types will be needed at some point, or always running with it on
- We have had many instances of customers raising PMRs for performance problems and not having enough headroom to enable accounting trace, which is how we diagnose performance problems!
- Plus when investigating performance issues it is always useful to have a baseline

- See capacity and planning guide: MP16 for more information
  - https://ibm-messaging.github.io/mqperf/mp16.pdf

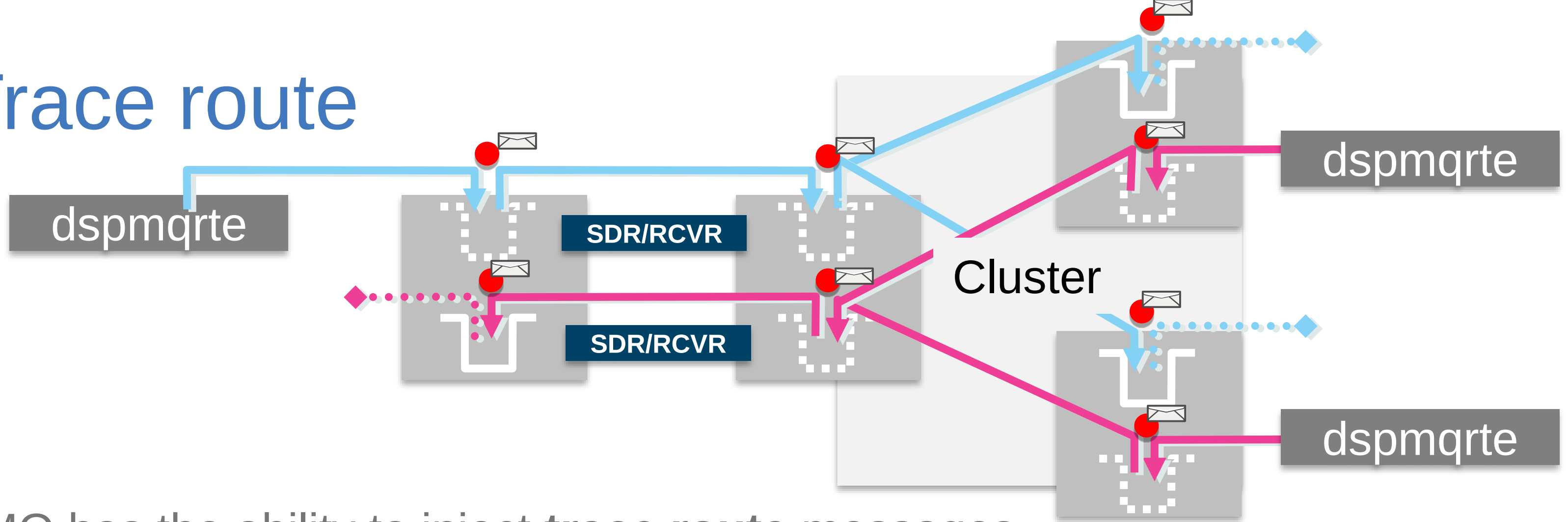
Where are the messages going?

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 **trace route** messages
  - (Can be) hidden from applications
  - Generate **activity reports** as they pass through, potentially accumulated in the message
- Tools are available to trace routes using these reports
  - dspmqrte – command line tool supplied with Distributed. Can be run against a z/OS queue manager as a client
  - MS0P – cat 2 SupportPac extension to MQ Explorer
- 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
- NB: there is also a related technology: activity recording which generates activity reports from **real** messages

What are the apps doing?

What are the apps doing?

- Distributed have application activity trace which can give information about all the MQI operations an application does in the order that they do it
- Useful for:
  1. Application audit trails
  2. Message duplication
  3. Detailed problem determination
  4. Enforcing application coding standards
- Should be selectively enabled to reduce performance impact

- MQ for z/OS doesn’t have any equivalent to this, but accounting stats can start to help with #1, and we will later see a tool for #2

- There is an RFE for adding application activity trace to MQ for z/OS
  – https://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=56321

How can I look back in time?

What happened to my messages at 2am this morning?

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

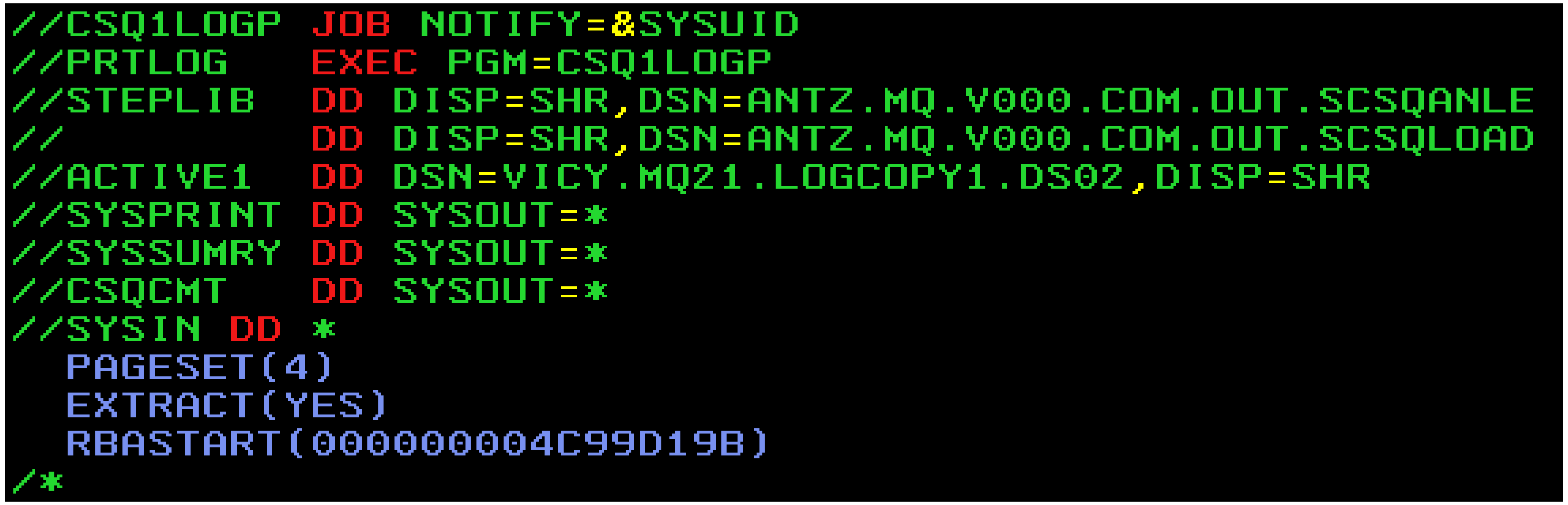
What about the MQ recovery log?

- For persistent messages regardless of whether in a transaction or not
  - MQ logs each operation performed

- 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 provides a utility to do so: CSQ1LOGP
  - Allows you to look at both active and archive logs even if queue manager is running
  - Can extract information on messages put / got in committed, rolled-back and in-flight units of work
  - Can also be used to get information on alterations to object definitions
  - Can be combined with DFSORT to do things like give the total amount of persistent message data sent to a queue

Running CSQ1LOGP



```
//CSQ1LOGP JOB NOTIFY=&SYSUID
//PRTLOG   EXEC PGM=CSQ1LOGP
//STEPLIB  DD DISP=SHR,DSN=ANTZ.MQ.V000.COM.OUT.SCSQANLE
//         DD DISP=SHR,DSN=ANTZ.MQ.V000.COM.OUT.SCSQLOAD
//ACTIVE1  DD DSN=VICY.MQ21.LOGCOPY1.DS02,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//CSQCMT   DD SYSOUT=*
//SYSIN DD *
  PAGESET(4)
  EXTRACT(YES)
  RBASTART(000000004C99D19B)
/*
```

- ACTIVE1 DD card: name of active log to search
- CSQCMT DD card: message data from committed units of work go here
  CSQBACK, CSQBOTH, CSQINFLT and CSQOBJS are also available
- PAGESET(4): only look at data in pageset 4
- EXTRACT(YES): extract message data to specified datasets
- RBASTART(): start from this RBA to the end of the log
  RBAEND and LRSN variants are also available

Example output

Output is one line for each MQPUT, MQGET and each UR related operation
MQPUT example shown below. Payload is in bold

2019.108 7:31:42.730   0N m   < w i 8XX** Ö \MQSX00 N m   l   CHIN   MQ21CHINBUR
MQ21AQUEUE   y MQPUT   CN   < x   A**MD**
CSQ MQ21   N m   MQ21
**MQSX00   MQ21CHIN1BEC3AE0 Ö **
LoadQueueLoopJMSWIthDifferen2019041807314273   This is a message

While this output format can be useful its likely that you will want to run tooling against it to further refine the data.

We provide sample C code (CSQ4LOGS) which will parse the output from CSQ1LOGP and then either:

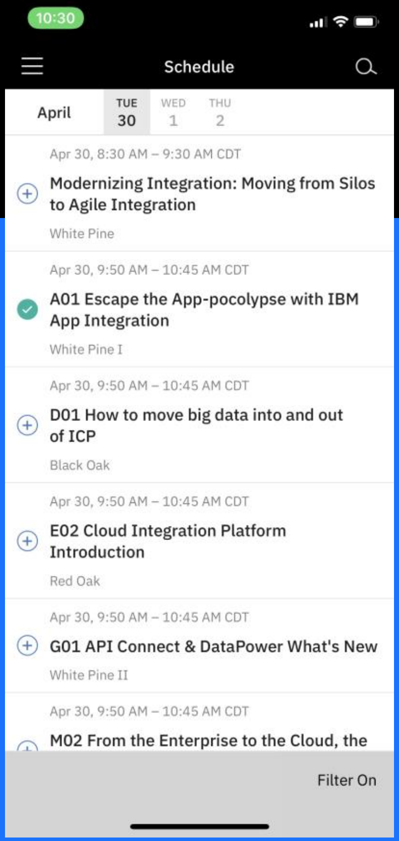
– Display summary information about each unit of work (no message data)

– Replay the messages to the target queue, possibly to a different queue manager, where it can then be browsed

– An obvious extension to this would be to dump just the message data. This is left as an action for the user…

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
  - View using MP1B
- Tracking
  - Trace-route
- MQ recovery logs
  - CSQ1LOGP


10:30

Schedule

April | TUE 30 | WED 1 | THU 2

Apr 30, 8:30 AM – 9:30 AM CDT
Modernizing Integration: Moving from Silos to Agile Integration
White Pine

Apr 30, 9:50 AM – 10:45 AM CDT
A01 Escape the App-pocolypse with IBM App Integration
White Pine I

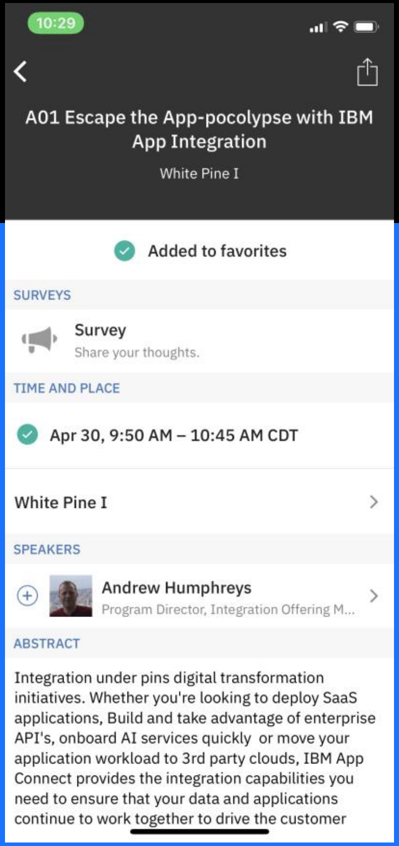
Apr 30, 9:50 AM – 10:45 AM CDT
D01 How to move big data into and out of ICP
Black Oak

Apr 30, 9:50 AM – 10:45 AM CDT
E02 Cloud Integration Platform Introduction
Red Oak

Apr 30, 9:50 AM – 10:45 AM CDT
G01 API Connect & DataPower What's New
White Pine II

Apr 30, 9:50 AM – 10:45 AM CDT
M02 From the Enterprise to the Cloud, the

Filter On


10:29

A01 Escape the App-pocolypse with IBM App Integration
White Pine I

Added to favorites

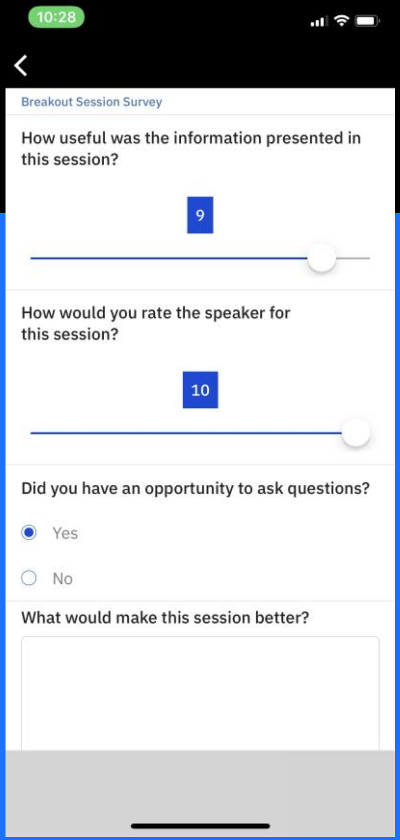
SURVEYS
Survey
Share your thoughts.

TIME AND PLACE
Apr 30, 9:50 AM – 10:45 AM CDT

White Pine I

SPEAKERS
Andrew Humphreys
Program Director, Integration Offering M...

ABSTRACT
Integration under pins digital transformation initiatives. Whether you're looking to deploy SaaS applications, Build and take advantage of enterprise API's, onboard AI services quickly or move your application workload to 3rd party clouds, IBM App Connect provides the integration capabilities you need to ensure that your data and applications continue to work together to drive the customer


10:28

Breakout Session Survey

How useful was the information presented in this session?
9

How would you rate the speaker for this session?
10

Did you have an opportunity to ask questions?
[✓ Yes] No

What would make this session better?

Don’t forget to fill out the survey!

Select your session, select survey, rate the session and submit!

Thank You

Matt Leming
lemingma@uk.ibm.com