NATIONAL IT SERVICES
FEDERAL RESERVE

WebSphere MQ & TCP Buffers – Size DOES Matter!

MQTC V2.0.1.3

Arthur C. Schanz
Distributed Computing Specialist
Federal Reserve Information Technology (FRIT)

Disclaimer :

The views expressed herein are my own and not necessarily those of Federal Reserve Information Technology, the Federal Reserve Bank of Richmond, or the Federal Reserve System.

About the Federal Reserve System

- The Federal Reserve System, the Central Bank of the United States – Created in 1913 – aka “The Fed”
- Consists of 12 Regional Reserve Banks (Districts) and the Board of Governors
- Roughly 7500 participant institutions
- Fed Services – Currency and Coin, Check Processing, Automated Clearinghouse (FedACH℠) and Fedwire®
- Fedwire – A real-time gross settlement funds transfer system – Immediate, Final and Irrevocable
- Fedwire® Funds Service, Fedwire® Securities Services and National Settlement Services
- Daily average Volume – Approximately 600,000 Originations
- Daily average Value – Approximately $4 Trillion
- Average transfer – $6,000,000
- Single-day peak:
    - Volume – 1.2 Million Originations
    - Value – $6.7 Trillion

Just for Fun: How much is $4 Trillion?

If you could spend $100,000,000 ($100 Million) per day, how long would it take you to spend $4 Trillion?

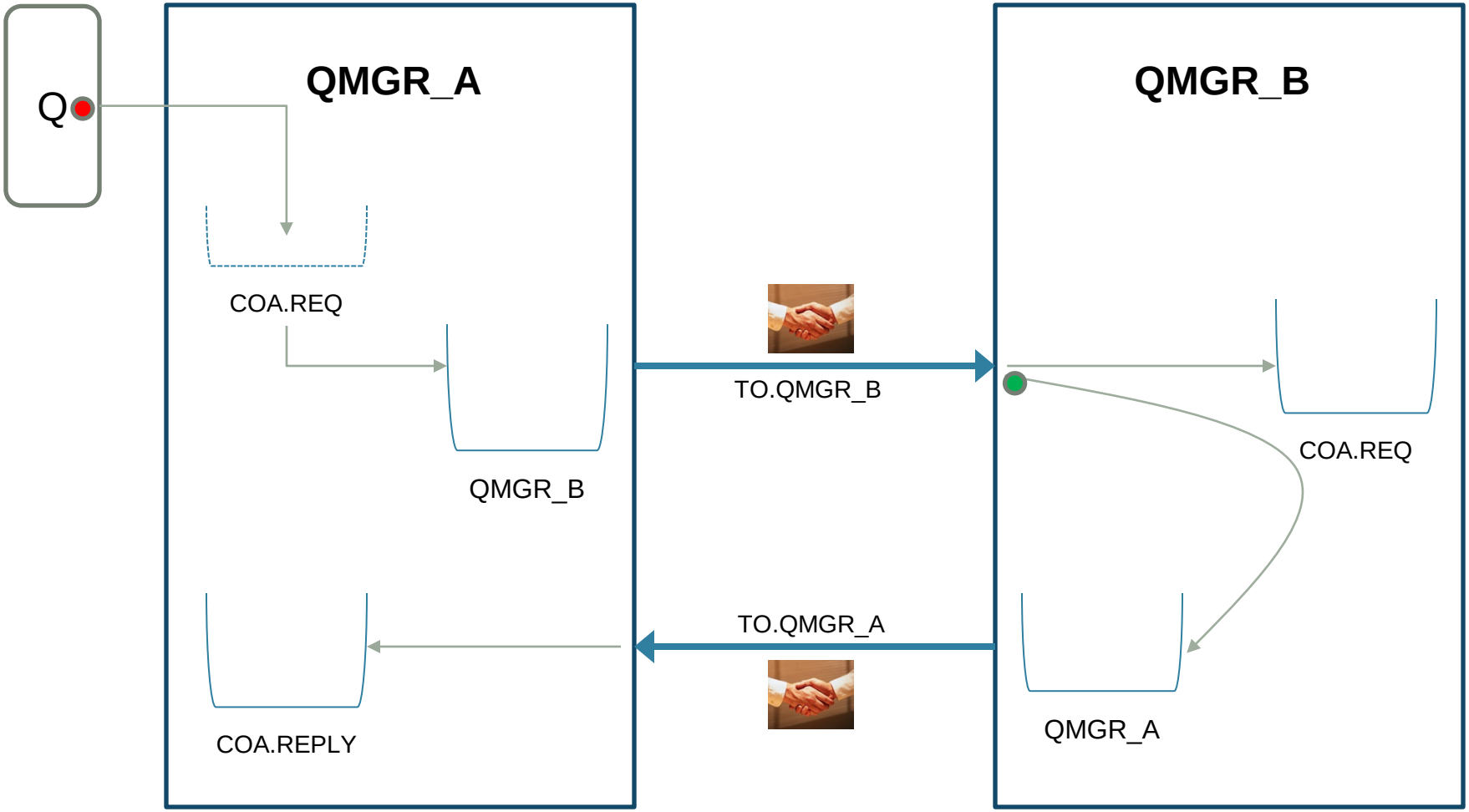
- 1 year?
- 5 years?
- 10 years?
- 50 years?
- 100 years?

Actually, it would take **over 109 years** to spend the entire $4 Trillion.

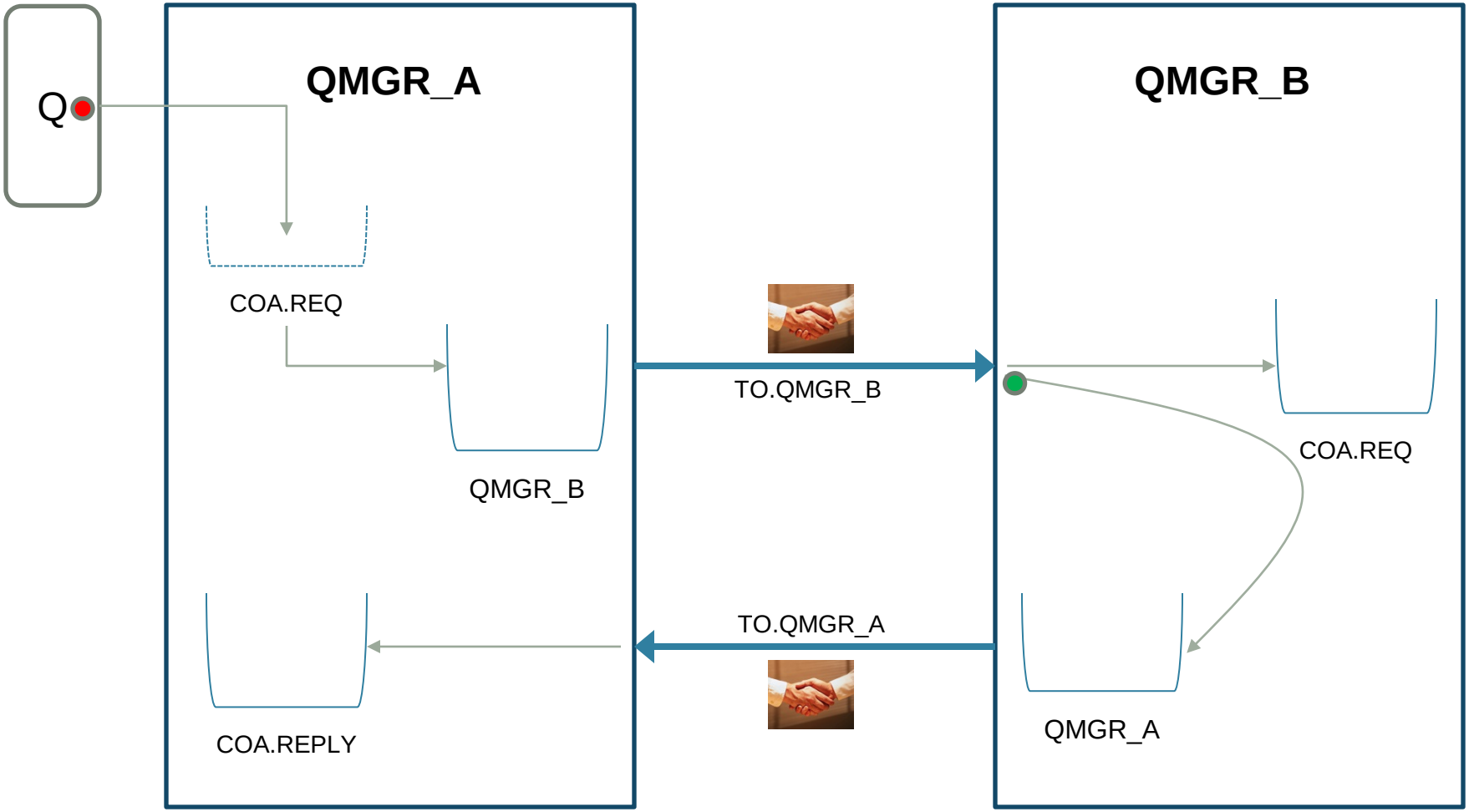
Scenario Details

- Major redesign of a critical application
- Platform change – moving from “non-IBM” to “IBM” (AIX)
- Initial use of Confirmation on Arrival (COA)
- QMGRs are located at data centers which are ~1500 Miles (2400 KM) apart
- 40ms latency for round-trip messages between data centers
- SSL in use for SDR/RCVR channels between the QMGRs
- WMQ V7.1 / GSKit V8
- Combination of JMeter and ‘Q’ program to generate test msg traffic
- Used 10K byte messages for application traffic simulation

Environment and Message Flow – Msg + COA



Environment and Message Flow – Msg + COA: MQPUTting it Together



Initial Observations…Which Led to WMQ PMR

- Functionally, everything is working as expected
- “Q” program reading file and loading 1000 10K byte messages
- No delay seen in generation of COA msgs
- IBM platform shows linear progression in time differential between MQPUT of original msg by Q and MQPUT of the COA
- “non-IBM” platform does not exhibit similar behavior
- Network core infrastructure is common for both platforms
- Evidence points to a potential issue in WMQ’s use of the resources in the communications/network layer (buffers?)
- Issue seen as a ‘show-stopper’ for platform migration by Appl

Week #1 – IBM Initial Analysis

- Several sets of traces, RAS output and test result spreadsheets were posted to the PMR
- WMQ L2 indicated that they are "…*not finding any unexpected processing relating to product defects…*", and that "…*the differences between the coa.req and coa.reply times are due to the time taken to transmit the messages to the remote qmgr.*"
- Received links to tuning documents and info on Perf & Tuning Engagements ($$$)
- IBM: OK to close this PMR?
- Our Answer: Open a PMR w/ AIX

Week #2 – Let's Get a 2nd Opinion (When in Doubt, Look at the Trace)

- AIX TCP/IP and Perf L2 request various trace/perfpmr data be captured on QMGR lpars and VIOS, while executing tests
- Network interfaces showing enormous amount of errors
- SSL Handshake taking an inordinate amount of time, as the channel PING or START is not immediate. This developed into another PMR ☹ – However, this is not contributing to the original problem
- Additional traces requested, using different criteria
- AIX PMR escalated to S1P1 – Executive conference call
- Internal review of WMQ traces reveals several interesting details…

Week #2 (con'd) – …and When in Doubt, Look at the Trace

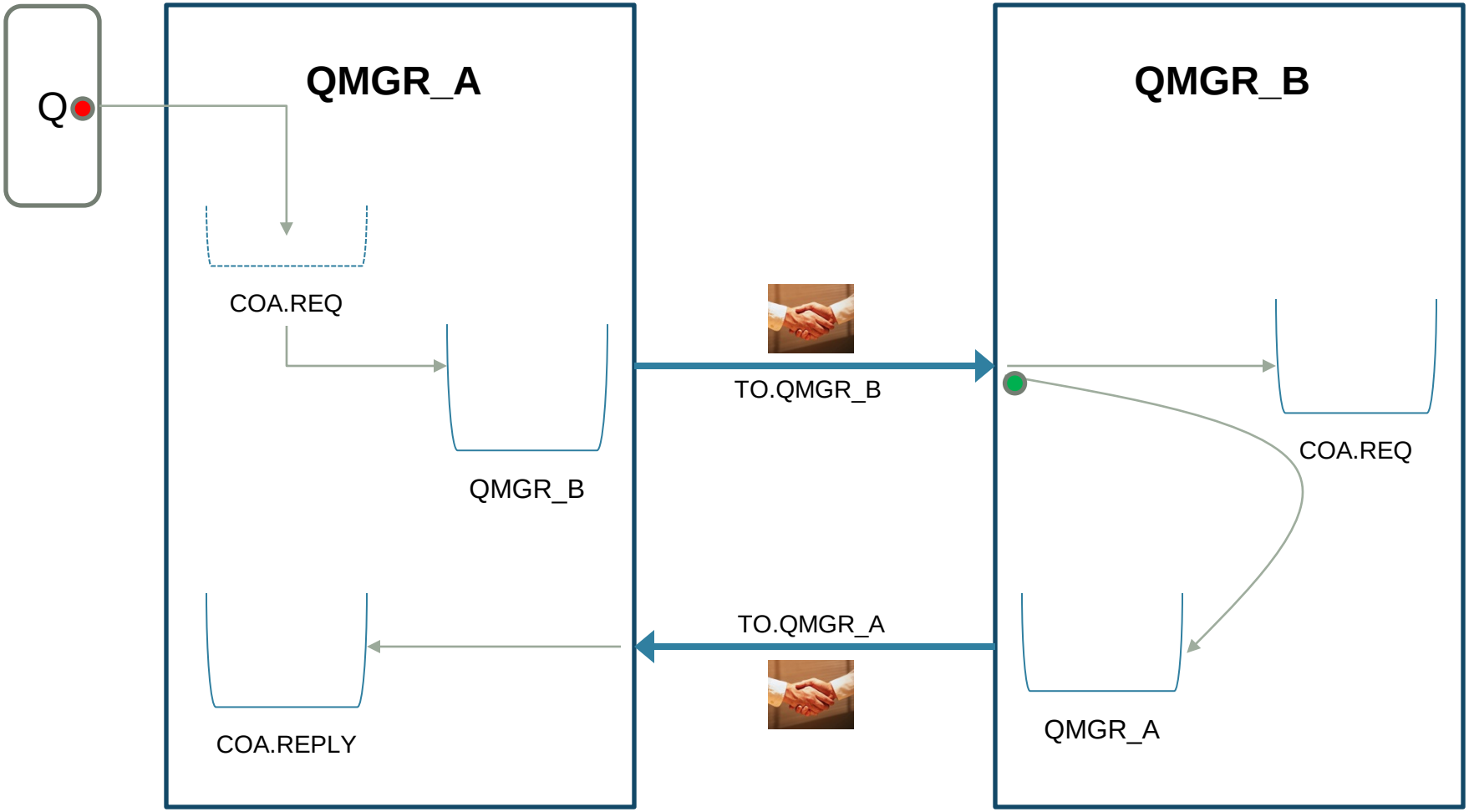
- SSL Handshake - GSKit V8 delay issue - 18 secs spent in this call:

12:19:44.621675 10289324.1 RSESS:000001 **gsk_environment_init**: input: gsk_env_handle=0x1100938b0
***12:20:02.527643** 10289324.1 RSESS:000001 **gsk_environment_init**: output: gsk_env_handle=0x1100938b0

- Qtime of msgs on XMITQ validates the linear progression of MQPUT t/s:

11:42:45.012377 9306238.433 RSESS:00011e Empty(TRUE) Inherited(FALSE) QTime(5829)
11:42:45.167286 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(153205)
11:42:45.169208 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(148920)
11:42:45.170910 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(145578)
11:42:45.172451 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(143312)
11:42:45.256783 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(223727)
11:42:45.297237 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(259181)
11:42:45.299316 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(256659)
11:42:45.379582 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(330958)
…
11:43:02.309739 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(11701686)
11:43:02.311993 9306238.433 RSESS:00011e Empty(FALSE) Inherited(FALSE) QTime(11699020)
11:43:02.314702 9306238.433 RSESS:00011e Empty(TRUE) Inherited(FALSE) QTime(11696105)

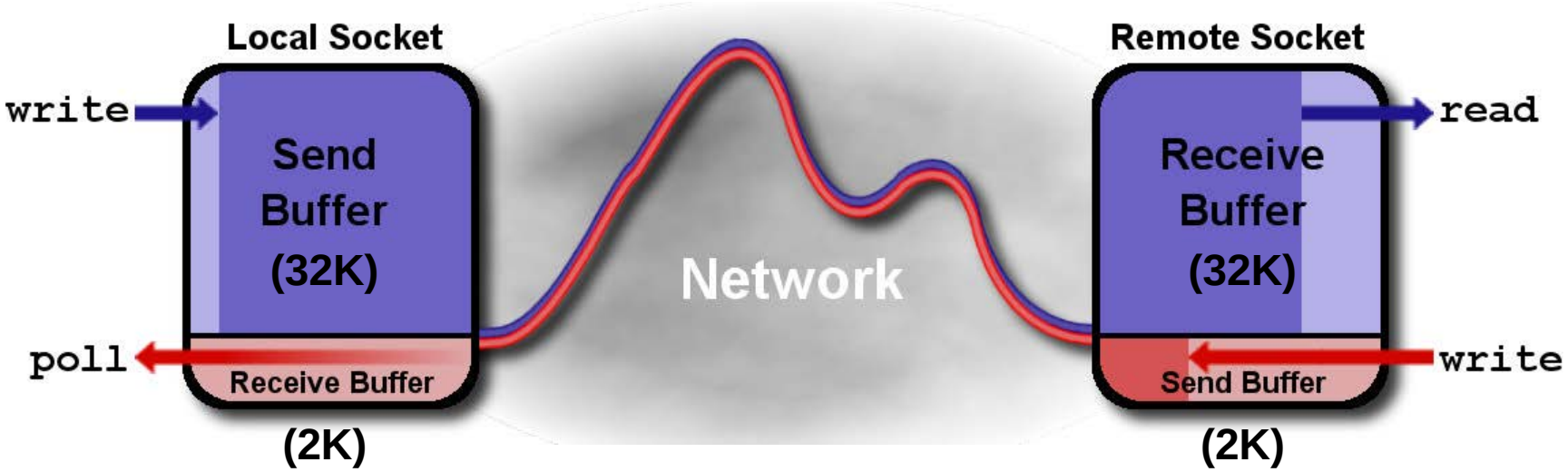
Environment and Message Flow – What We Were Experiencing



Week #2 (con'd) – …and When in Doubt, Look at the Trace

- Negotiated and Set Channel Options/Parameters:
  - KeepAlive = 'YES'
  - Conversation Type: SSL
  - Timeout set to 360 (360) (30)
  - GSKit version check '8.0.14.14' >= '8.0.14.12' OK
  - TCP/IP TCP_NODELAY active
  - TCP/IP KEEPALIVE active
  - Current socket receive buffer size: 262088
  - Current socket send buffer size: 262088
  - Socket send buffer size: 32766
  - Socket receive buffer size: 2048
  - Final socket receive buffer size: 2048
  - Final socket send buffer size: 32766
- PMR update: "This may be at the root of the problem, but I will leave that determination to the experts."

WebSphere MQ – Established TCP Channel Connection



Both sockets have a send buffer and a receive buffer for data

- **write()** adds data to the send buffer for transfer
- **read()** removes data arrived on the receive buffer
- **poll()** and **select()** wait for a buffer to be ready

Week #3 – AIX/TCP Tuning (Lather…Rinse…Repeat)

- More traces
- AIX and TCP L2 suggest tuning changes: ‘Largesend=1’ and ‘tcp_sendspace=524288’ & ‘tcp_recvspace=524288’ for network interfaces
- Wireshark shows that ‘Largesend=1’ made performance worse. TCP window size updates still constantly occurring during test
- Network interface errors were ‘bogus’ – known AIX bug
- VIOS level could be contributing to the problem
- High fork rate
- Tcp_nodelayack
- WMQ PMR escalated to S1
- IBM changes this entire issue to Crit Sit – we now have a dedicated team to work the issue

Week #3 (con'd) – WMQ Tuning – Back to (Buffer) Basics

- More traces
- WMQ L2/L3 suggest increasing both WMQ TCP Send/Recv buffers to 32K, then 64K, via qm.ini - (But why isn't this needed on current platform?)

TCP:

SndBuffSize=65536

RcvBuffSize=65536

- Change had no positive effect
- We see (and L3 confirms) that trace shows SDR-side and RCVR-side sockets are in many wait/poll calls, some of which are simultaneous

```
11:42:45.381709  9306238.433   RSESS:00011e ---{  cciTcpSend
11:42:45.381713  9306238.433   RSESS:00011e ----{  send
11:42:45.381723  9306238.433   RSESS:00011e ----}! send rc=Unknown(B)
```

- rc=Unknown(B), where B=11 (EAGAIN) - No buffer available

Week #3 (con'd) – More WMQ and AIX Tuning

- More traces
- AIX Perf team suggests 'hstcp=1'
- WMQ L3: "Delays are outside of MQ's control and requires further investigation by OS and Network experts"
- AIX L3: "We are 95% sure that an upgrade to VIOS will correct the issue"
- We determined that AIX L3 had made several tuning recommendations by looking at the wrong socket pair…none of which had any positive effect
- AIX L3: "It appears that TCP buffers need to be increased." Value chosen: 5MB.
- Pushed back on making that change, but IBM WMQ L2/L3 agreed with that recommendation
- Tests also requested w/ Batchsize of 100, then 200

Week #3 (con'd) – More WMQ and AIX Tuning

- 5MB buffers showed some improvement (How could they not?!)
- Batchsize changes did not show significant improvement
- Removed trace hooks (30D/30E), set 'tcp_pmtu_discover=1' & 'tcp_init_window=8', as they were said to be "pretty significant" to increased performance. They weren't. ☹
- Back to the $64,000 question – Why do we need changes on this new platform, but not on the current platform, all other things being equal?
- While IBM contemplates their next recommendation, we decided to take a closer look at the two environments, more specifically the WMQ TCP buffers. What is actually happening 'under the covers' during channel negotiation.
- Guess what we found?

Week #3 (con'd) – All Flavors of Unix are NOT Created Equal

- SDR channel formatted trace file for new (AIX) platform:
  - Current socket receive buffer size: 262088
  - Current socket send buffer size: 262088
  - Socket send buffer size: 32766
  - Socket receive buffer size: 2048
  - Final socket receive buffer size: 2048
  - Final socket send buffer size: 32766

- SDR channel formatted trace file for current (non-IBM) platform:
  - Current socket receive buffer size: 263536
  - Current socket send buffer size: 262144
  - Socket send buffer size: 32766
  - Socket receive buffer size: 2048
  - Final socket receive buffer size: 263536
  - Final socket send buffer size: 32766

➢ IBM platform uses setsockopt() values – non-IBM uses hybrid

Week #3 (con'd) – All Flavors of Unix are NOT Created Equal

- RCVR channel formatted trace file for new (AIX) platform:
  - Current socket receive buffer size: 262088
  - Current socket send buffer size: 262088
  - Socket send buffer size: 2048
  - Socket receive buffer size: 32766
  - Final socket receive buffer size: 32766
  - Final socket send buffer size: 2048

- RCVR channel formatted trace file for current (non-IBM) platform:
  - Current socket receive buffer size: 263536
  - Current socket send buffer size: 262144
  - Socket send buffer size: 2048
  - Socket receive buffer size: 32766
  - Final socket receive buffer size: 263536
  - Final socket send buffer size: 2048

➢ IBM platform uses setsockopt() values – non-IBM uses hybrid

Week #3 (con'd) – If Only it Were That Easy

- This should be simple – hard code the Send/Recv buffer values that are being used in the current environment into the qm.ini for the QMGR in the new environment, retest and we should be done:

TCP:

SndBuffSize=32766

RcvBuffSize=263536

- Success? Not so much – In fact, that test showed no difference than the 'out of the box' values?!? Now what?
- Of course – more AIX tuning – 'tcp_init_window=16', increase CPU entitlements to lpars and increase VEA buffers…no help
- Wireshark shows 256K max 'Bytes in Flight', even with 5MB buffers
- AIX L3 supplies TCP Congestion Window ifix – "Silver Bullet"
- WMQ L3 requests a fresh set of traces, for both platforms
- This is their analysis:

Week #4 – WMQ L3 Analysis of Latest Traces

	Non-IBM	**AIX**
	WMQ 7.0.1.9	WMQ 7.1.0.1
	non-SSL	SSL
Log write avg:	2359us	1562us
ccxSend avg:	165us	3234us
Send avg:	55us	122us
Send max:	187us	10529us
ccxReceive avg:	91ms	177ms
ccxReceive min:	74ms	100ms
Batch size:	50	50
MQGET avg:	6708us	2784us
Msg size:	10K	10K
Overall time:	10s	10s

Notes:

ccxSend:	This is the time to execute the simple MQ wrapper function ccxSend which calls the TCP send function.
Send:	A long send might indicate that the TCP send buffer is full.
ccxReceive:	This is the time to execute the simple MQ wrapper function ccxReceive which calls the TCP recv function.

Week #4 (con'd) – WMQ L3 Analysis of Latest Traces

- We temporarily disabled SSL on AIX and saw a definite improvement – it was comparable w/ the current platform. (The fallacy here is that it should be better!)
- In addition, we are still using 5MB Send/Recv buffers, so this is not a viable solution.
- With all of the recommended changes that have been made, we requested IBM compose a list of everything that is still 'in play', for both AIX and WMQ. What is really needed vs. what was an educated guess.
- IBM believes they have met/exceeded the original objective of matching/beating current platform performance.
- Exec. conf call held to discuss current status, review recommended changes and determine next steps.
- Most importantly – We are still seeing the increasing Qtime.

Week #4 (con'd) – You Can't (Won't) Always Get What You Want

- There had to be some reason why, when hard-coding the values being used on the non-IBM system, we did not see the same results

TCP:

SndBuffSize=32766

RcvBuffSize=263536

- Looking at the traces, something just didn't seem right
- Were we actually getting the values we were specifying in qm.ini? It did not appear so.
- There had to be an explanation
- What were we missing?
- As it turns out, a very important, yet extremely difficult to find set of parameters was the key…

Week #4 (con'd) – Let's Go to the Undocumented Parameters

- Sometimes, all it takes is the 'right' search, using your favorite search engine.
- We found some undocumented ('under-documented') TCP buffer parameters, that complement their more well-known brothers:

TCP:

SndBuffSize=262144

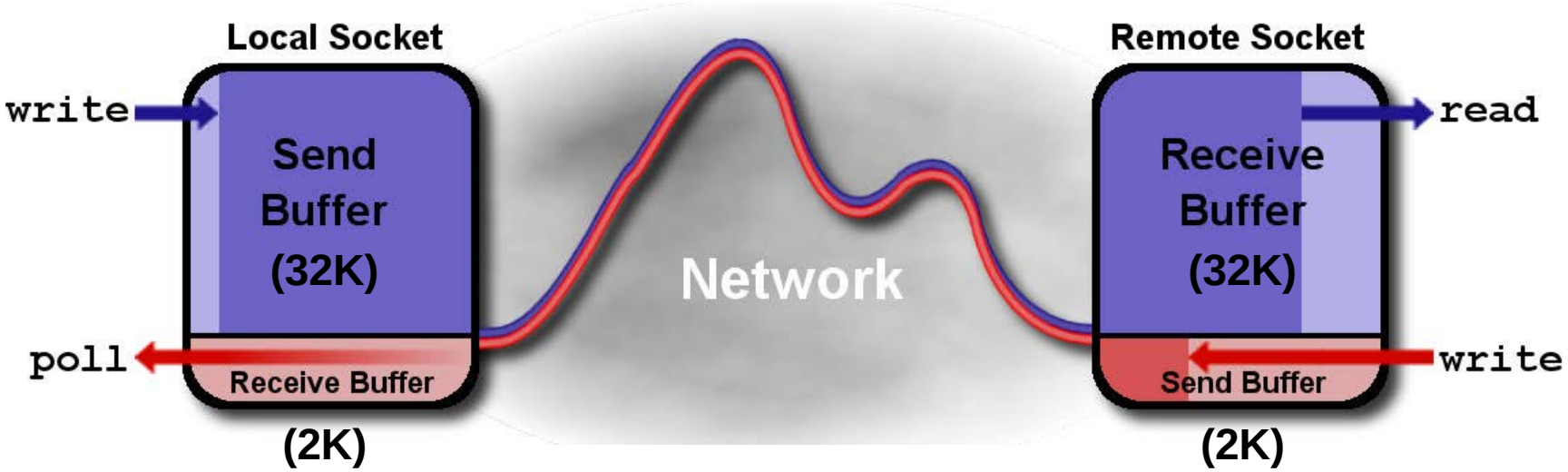
RcvBuffSize=2048

RcvSendBuffSize=2048

RcvRcvBuffSize=262144

- Now it's starting to make sense.
- Now we see why the traces were not showing our values being accepted.
- Now we know how to correctly specify the buffers for both SDR and RCVR

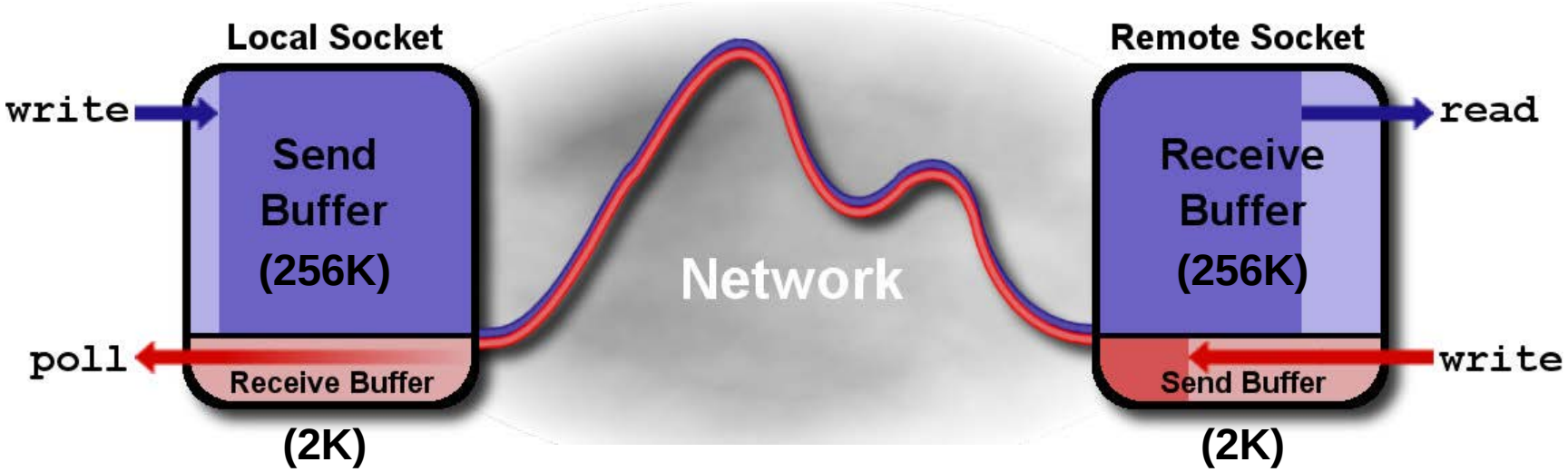
WebSphere MQ – Established TCP Channel Connection



Both sockets have a send buffer and a receive buffer for data

- **write()** adds data to the send buffer for transfer
- **read()** removes data arrived on the receive buffer
- **poll()** and **select()** wait for a buffer to be ready

WebSphere MQ – Established TCP Channel Connection



SndBuffSize=**256K** (Default is 32K)

RcvBuffSize=2K (Default is 2K)

'Under-documented' tuning parameters

RcvRcvBuffSize=**256K** (Default is 32K)

RcvSndBuffSize=2K (Default is 2K)

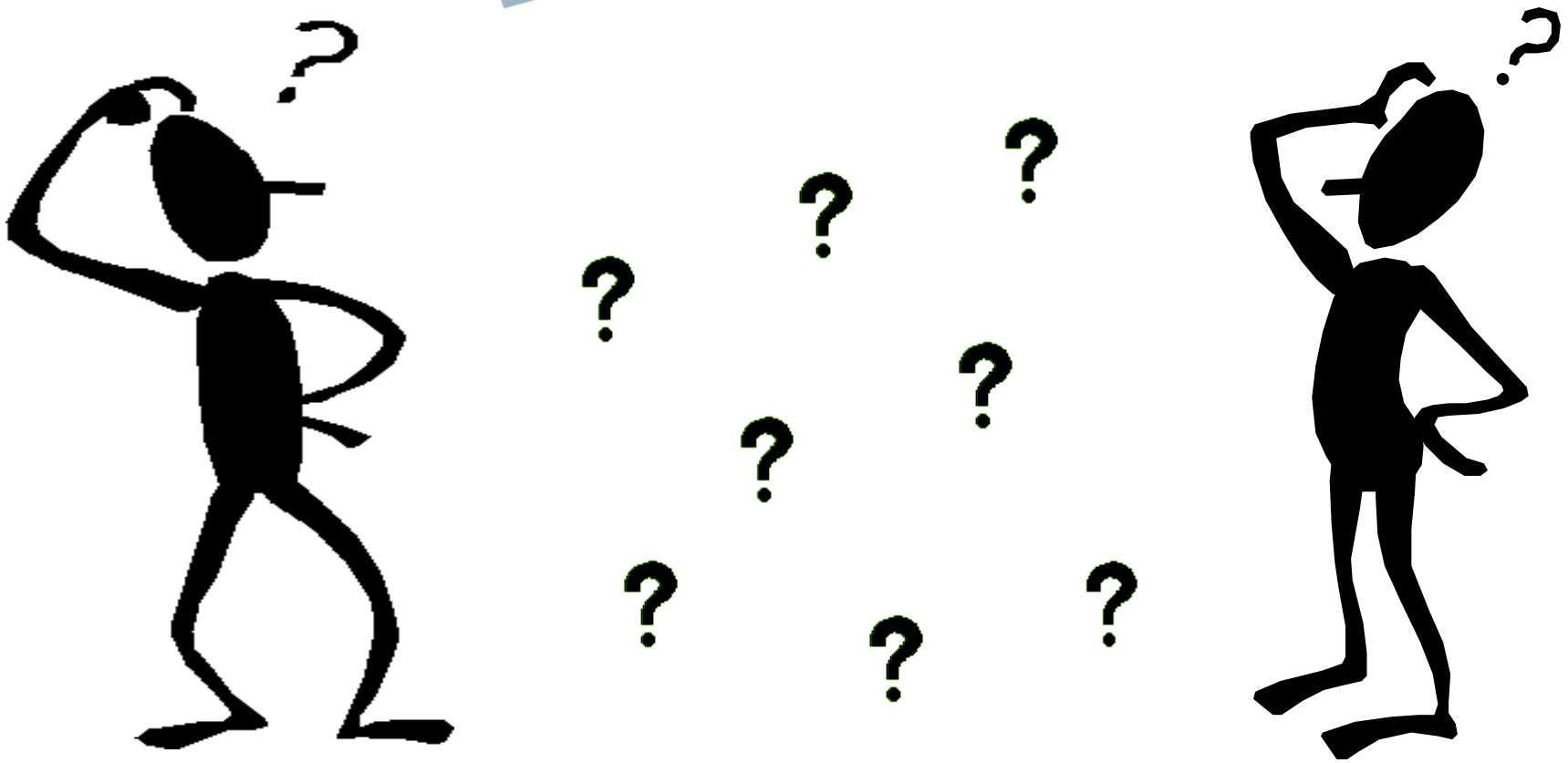
Both sockets have a send buffer and a receive buffer for data

- **write()** adds data to the send buffer for transfer
- **read()** removes data arrived on the receive buffer
- **poll()** and **select()** wait for a buffer to be ready

Conclusions – What We Learned

- Desired values for message rate and (lack of) latency were achieved by tuning **only** WMQ TCP Send/Recv buffers. ('tcp_init_window=16' was also left in place, as it produced a noticeable, albeit small performance gain)
- The use of the 'under-documented' TCP parameters was **the** key in resolving this issue.
- Continue doing in-house problem determination, even after opening a PMR.
- Do not be afraid to push back on suggestions/recommendations that do not feel right. No one knows your environment better than you.
- Take some time to learn how to read and interpret traces. (Handy ksh cmd we used: **find . –name '*.FMT' | xargs sort –k1.2,1 > System.FMTALL**, which creates a timestamp-sorted formatted trace file from all indiv files)
- TCP is complicated – Thankfully, WMQ insulates you from almost all of it.
- Keep searching the Web, using different combinations of search criteria – you never know what you might find. ☺

NATIONAL IT SERVICES
FEDERAL RESERVE



NATIONAL IT SERVICES
FEDERAL RESERVE

Chad Little

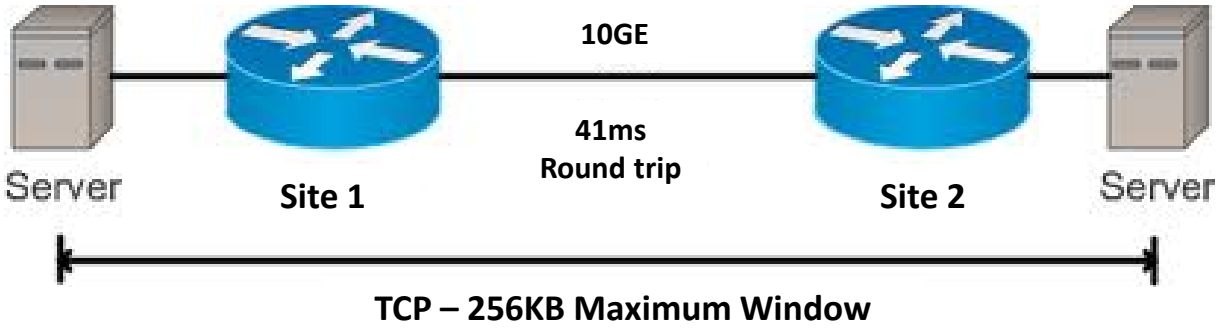
Quazi Ahmed

Josh Heinze

Larry Halley

Arthur C. Schanz
Distributed Computing Specialist
Federal Reserve Information Technology
Arthur.Schanz@frit.frb.org

Appendix: TCP Init Window Size



- AIX Initial TCP Window Size is 0, which causes TCP to perform dynamic window size increases (‘ramp-up’) as packets begin to flow across the connection.
    - 1K -> 2K -> 4K -> 8K -> 16K ->…
- Increasing the ‘tcp_init_window’ parameter to 16K allows TCP to start at a larger window size, therefore allowing a quicker ramp-up of data flow.