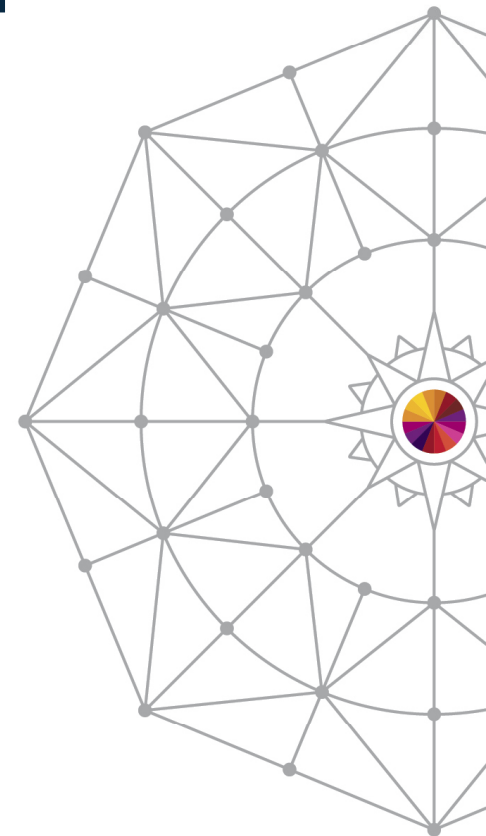


MQ Workload Balancing in a 'Plexed World

Session 15998

IBM WSC for WebSphere MQ
Lyn Elkins – elkinsc@us.ibm.com



#SHAREorg



Agenda

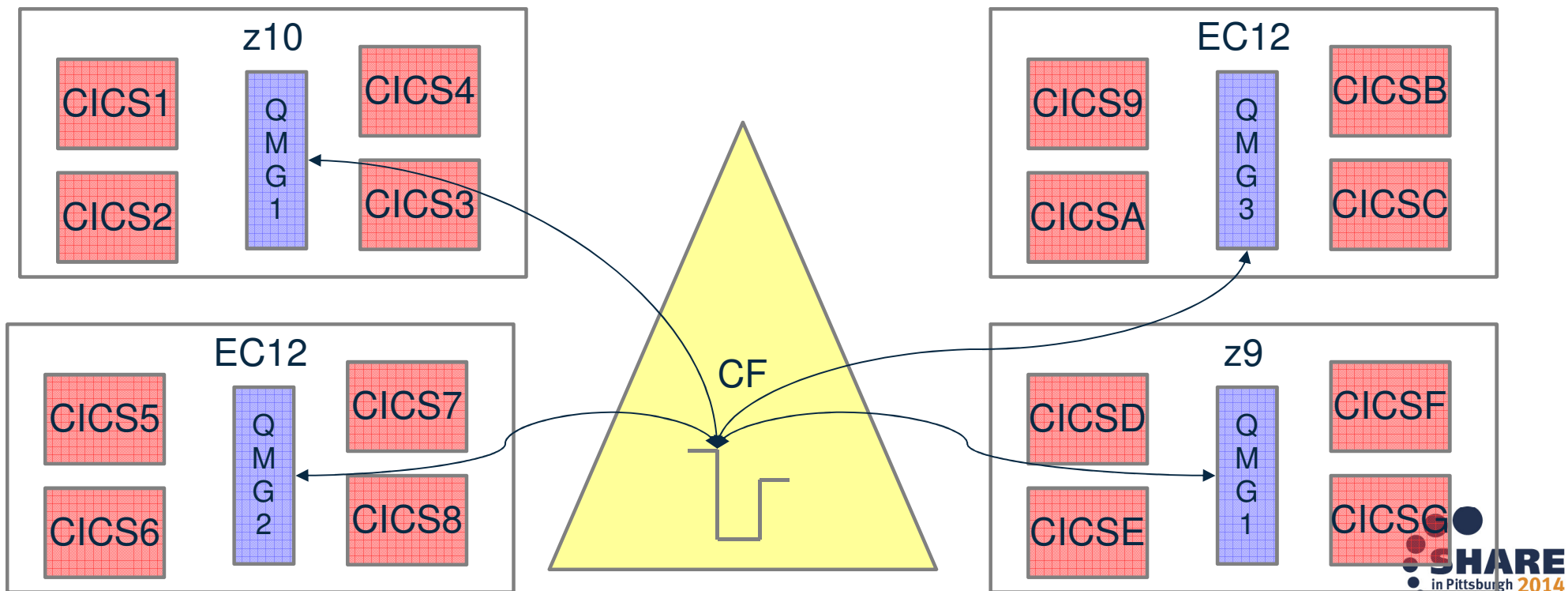
- What is workload skewing and why is it a problem?
- What can cause or contribute to workload skewing?
 - Asymmetrical Sysplex
 - Connection Skewing
 - Put to Waiting Getter
 - ‘Local’ favoritism
- Mitigation Techniques:
 - Queue Manager Clustering
 - Gateway queue managers
 - CICS CPSM options

What is MQ Workload Skewing?

- MQ workload skewing is detected when workload is not close to being evenly distributed across the queue managers.
 - MQ is a ‘data delivery’ system.
- Workload skewing in a QSG is often a result of the efficiencies of working locally
 - z/OS, and all subsystems try to process requests locally to take advantage of CPU efficiency
- This is often less a technical problem, more of a pricing problem
 - If the MLC ‘rolling average’ is taken from the LPAR that is heavily favored, usage pricing is not going to reflect reality
 - Technical solutions to this problem may prove to be less efficient overall - lower throughput, slower response
- Can cause increased capacity demands in downstream workload
 - Again this can contort MLC charges

MQ Workload Skewing Causes

- Asymmetric Sysplex
 - When the LPARs in the Sysplex are not equally weighted
 - Examples include:
 - One LPAR is on an EC12, the others on older hardware
 - Two LPARs have 12 dedicated engines, two have 12 shared
 - One LPAR is co-located with the primary coupling facility, the others are on different CPCs

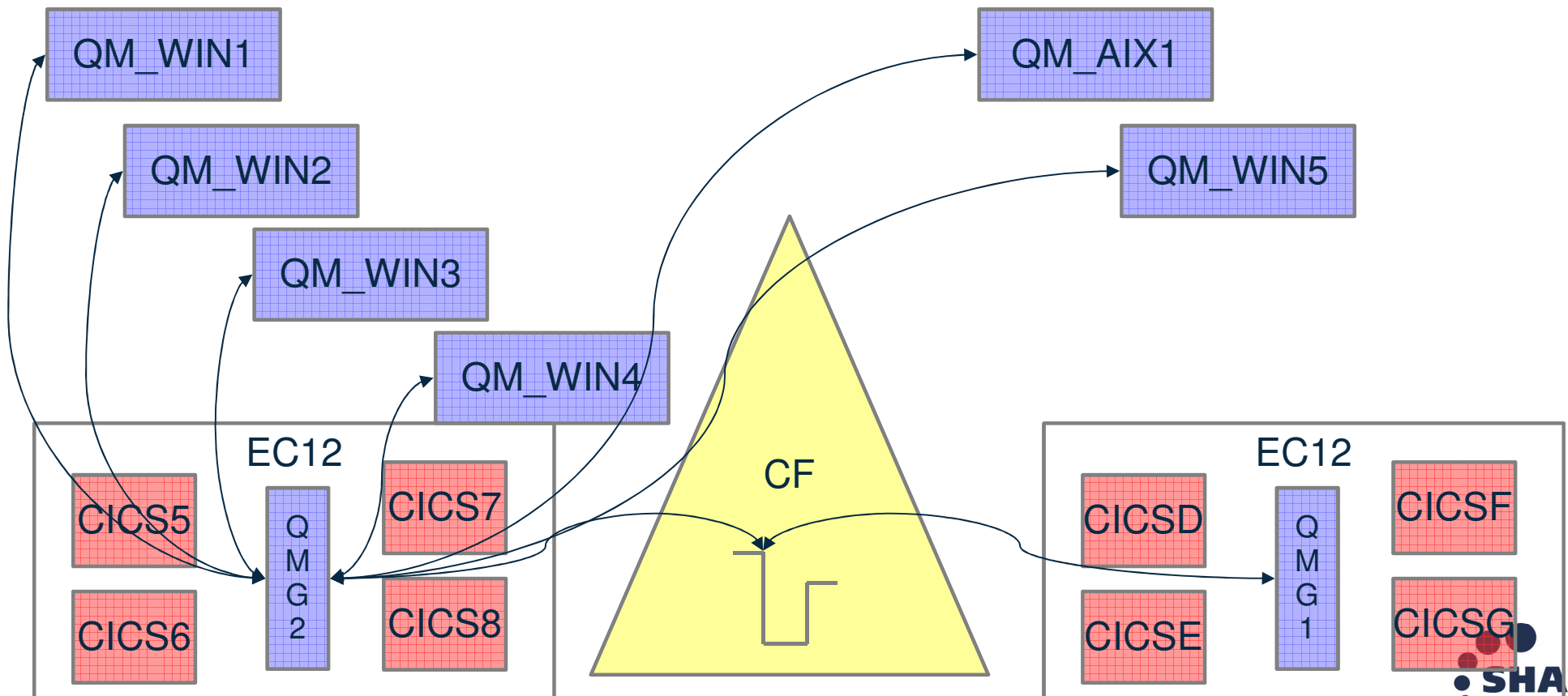


Notes

- **Slide shows an asymmetric sysplex.**
 - Each LPAR is configured the same, but two LPARs are on much faster, and newer, hardware.
 - Everything else being equal, work will tend to go to the LPARs on the EC12s, as they are faster engines.
 - This is the ‘natural’ tendency
- Other examples of asymmetry are listed, but not illustrated
 - One of the most dramatic examples of this was a situation where the LPAR acting as the CF was co-located on the CPC as a processing LPAR, and it was on the latest hardware. The queue manager in the application processing LPAR routinely handled more than 95% of the workload, because the response time for CF requests was 40% better on that LPAR than the others in the sysplex.

Connection Skewing

- Connection skewing may be historical
 - Hard-coded connections to specific queue managers
- Connection skewing may be the result of a queue manager outage
 - Connections to a QSG are routed to available queue managers



- **Slide shows connection skewing.**

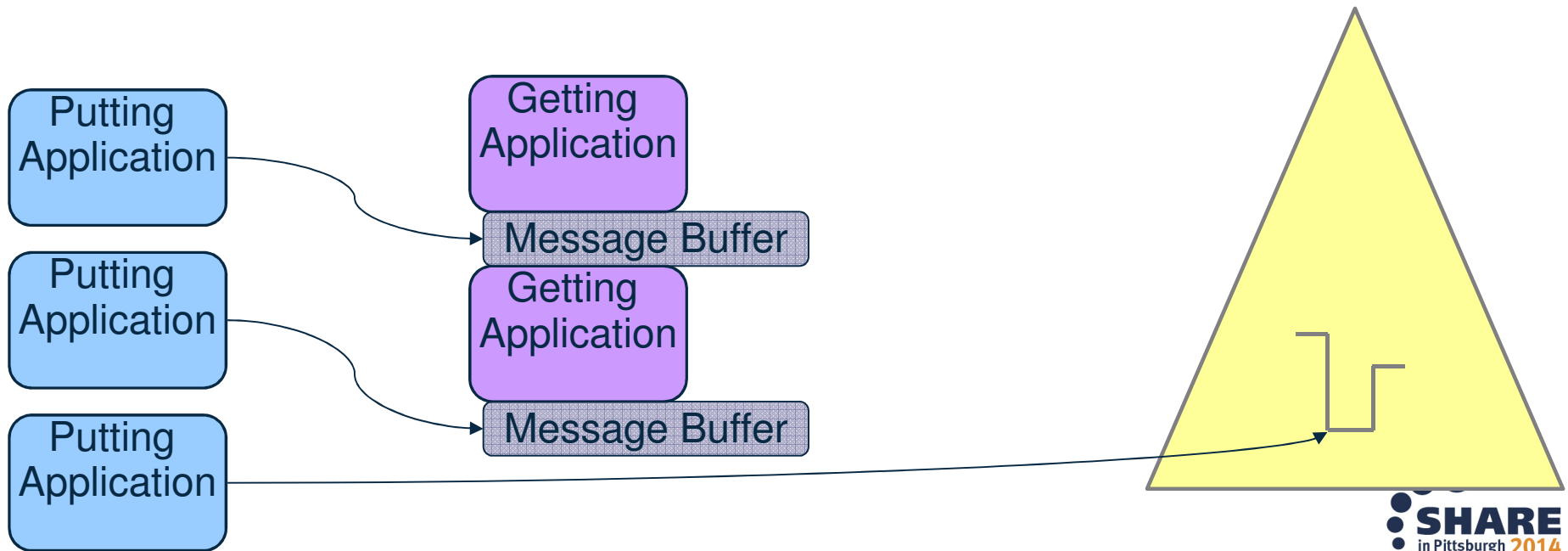
- In some cases this can be due to historical connection definitions. If clients and queue manager connections are to an individual queue manager and have not been updated to connect to the QSG, workload skewing can be a result of connection skewing.
 - This can also be the result of copying channel definitions that point to specific queue managers.
- Connection skewing can also be the result of clients and queue managers connecting (or reconnecting) during outages.
 - In the slide shown if one of the queue managers (or the LPAR) is unavailable due to any type of outage, all connections are made to the available queue manager. Connections once made are not re-driven unless the connection is stopped and restarted. We (the WSC) have seen substantial workload skewing of both connections and workload as a result of this availability feature.
- In this example, 99% of the workload was processed on the LPAR hosting the connection queue manager QMG1. The other LPAR essentially sat idle. During peak periods SLA were not being met due to the overuse of one 'side' of the 'plex.

‘Downstream’ consequences

- We’ve talked about the MLC impact
- Resource use
 - Not every queue manager is sized to absorb the entire workload
 - Log impact of skewing has been seen
 - Rapid Log switches due to heavier workload – increasing I/O and CPU costs
 - Bufferpool/Pageset impact
 - Filling the bufferpool, forced into I/O
 - SMDS impact
 - One queue manager in QSG gets all offloaded messages

MQ Workload Skewing Causes

- Put to waiting getter
 - In V6 a performance feature was added called ‘put to waiting getter’
 - If a local put, from an application or message channel agent, is done and there is a getting application waiting the message is moved directly to the getting applications buffer
 - There is no posting to a shared queue
 - There is no notification to other available waiting applications
 - The CPU savings can be substantial
 - This works with connection skewing, and can maximize the effect



Notes

- **Putting to Waiting Getter**

- This was a performance/CPU consumption reduction feature added to WMQ for z/OS V6
- It can be turned off, at the queue manager level
- As the illustration shows, the messages put by the first two applications go directly to the message buffer available from the getting applications; the third put actually goes to the shared queue because there is no message buffer available.

MQ Workload Skewing Causes

- Local Favoritism
 - When a message is posted to a shared queue, the queue manager where the message is put is typically notified **FIRST** about the availability.
 - Normal processing by XCF, taking advantage of the efficiency of local processing.

Notes



- **Local favouritism**

- Without application intervention (see the CICS CPSM mitigation) there's not much of a way to circumvent this.

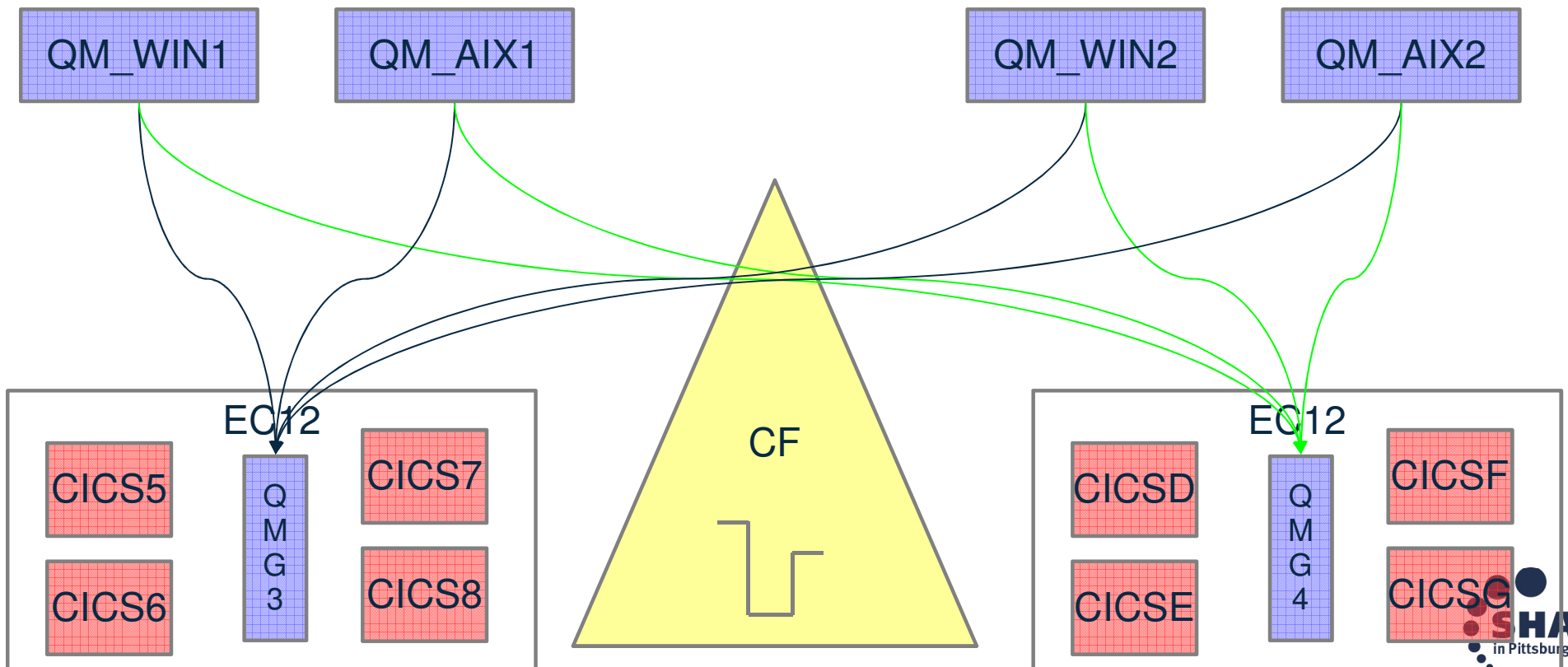


Skewing Mitigation Techniques

- Queue Manager Clusters
 - Clusters provide workload balancing across queue managers
 - Works with shared queues to distribute message ‘puts’ across queue managers in the QSG
- Connection skewing mitigation
 - Gateway queue managers
 - Re-driving connections
- CPSM mitigation

Queue Manager Clustering

- When messages are not bound to a specific queue manager ('bind not fixed'), the messages are routed evenly across the receiving queue managers
 - Black arrows show the first message put to the clustered queue
 - Green arrows show the second message



Queue Manager Clustering - Notes

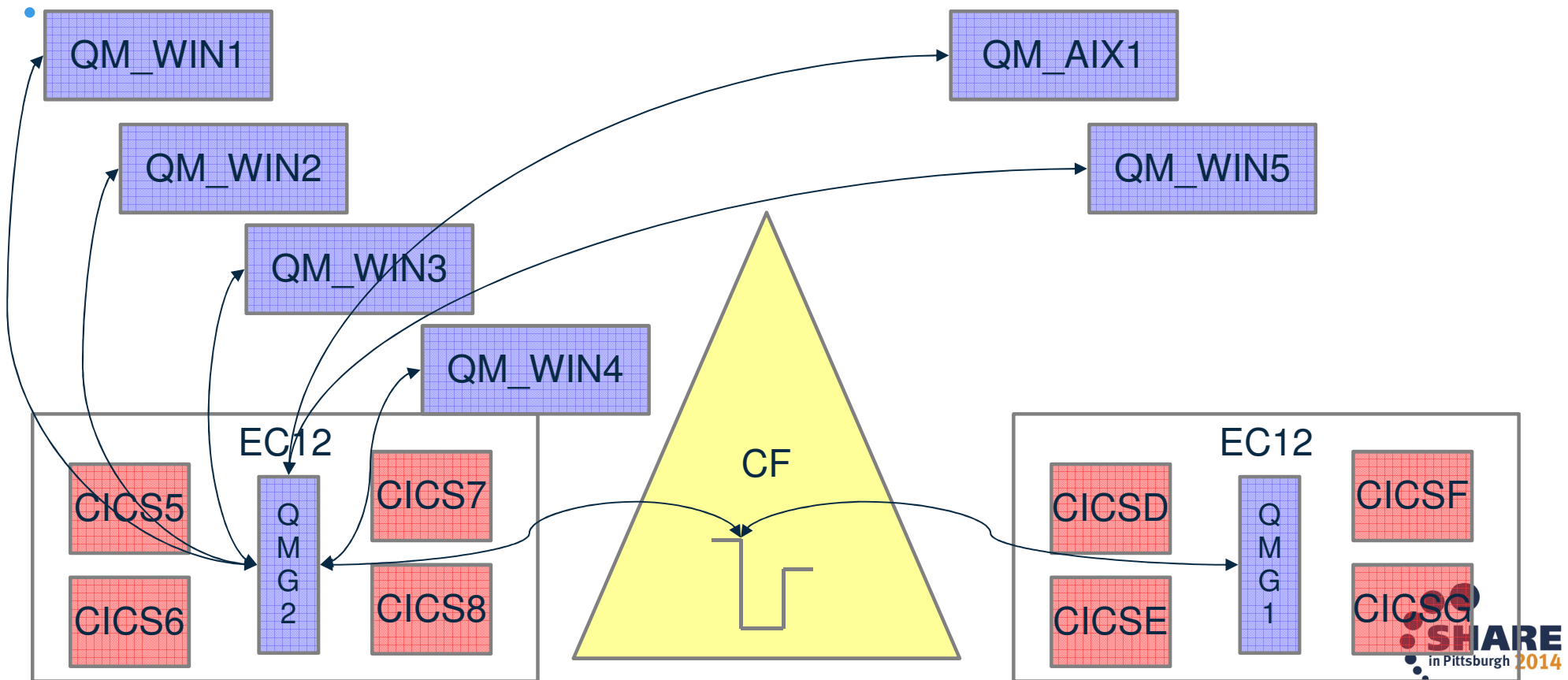
- Queue manager clusters is often the simplest (and least expensive) mitigation technique for workload skewing.
 - Most clusters are use round robin distribution of messages
 - This can be changed via definitions
 - *Messages can be bound to a specific queue manager instance using the open options:*
 - » *Bind not fixed – each message is routed independently*
 - » *Bind on open – all messages put will go to the same target queue manager/queue combination until the queue is closed*
 - » *Bind on group – messages that are part of a group will be delivered to the same target queue manager/queue combination*
 - *Channels may have different weights*
 - Illustration shows messages being delivered evenly. The first message is the black arrow, second is in green.
 - *Even delivery of messages, in most cases, will reduce skewing and allow MQ and other subsystems to take advantage of local processing!*
 - Workload imbalance can still occur under some conditions:
 - *If one queue manager or processing regions is under heavier load*
 - *If queue managers are in an asymmetric sysplex*
 - *If the applications requires binding (as described above)*

Connection Skewing Mitigation

- The slides that follow outline two mitigation techniques for connection skewing:
 - Gateway queue managers
 - Re-driving connections

Connection Skewing – No Gateway queue managers

- When external queue managers or clients are passing work directly to application hosting queue managers, every attempt is made to process the work locally
- Environments that use gateway queue managers into the Queue Sharing group often eliminate connection skewing.

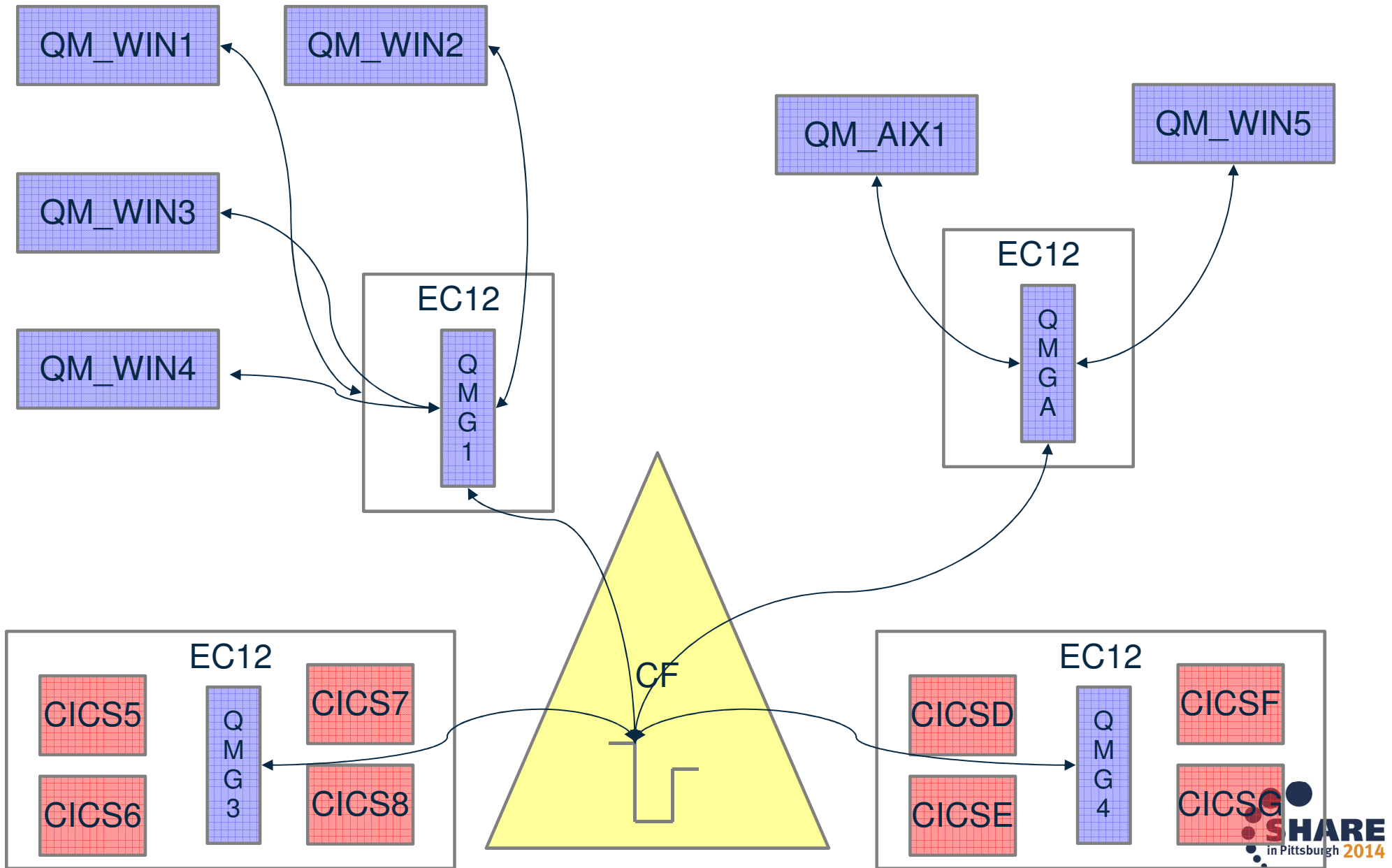


Notes

- **Gateway queue managers**

- The picture illustrates the connection imbalance that has been shown before.
- This mitigation technique was discovered almost accidentally, when we were comparing some large customer environments.
- One high volume customer was experience a great deal of ‘LPAR favouritism’, while another customer with similar volumes was not. Both customers had very similar hardware and system software configurations, we first looked to the topology for differences. It became clear very quickly that the customer that was having problems had about 95% of their connections ‘hard wired’ to two of their four queue managers.

Gateway queue managers – the mitigation



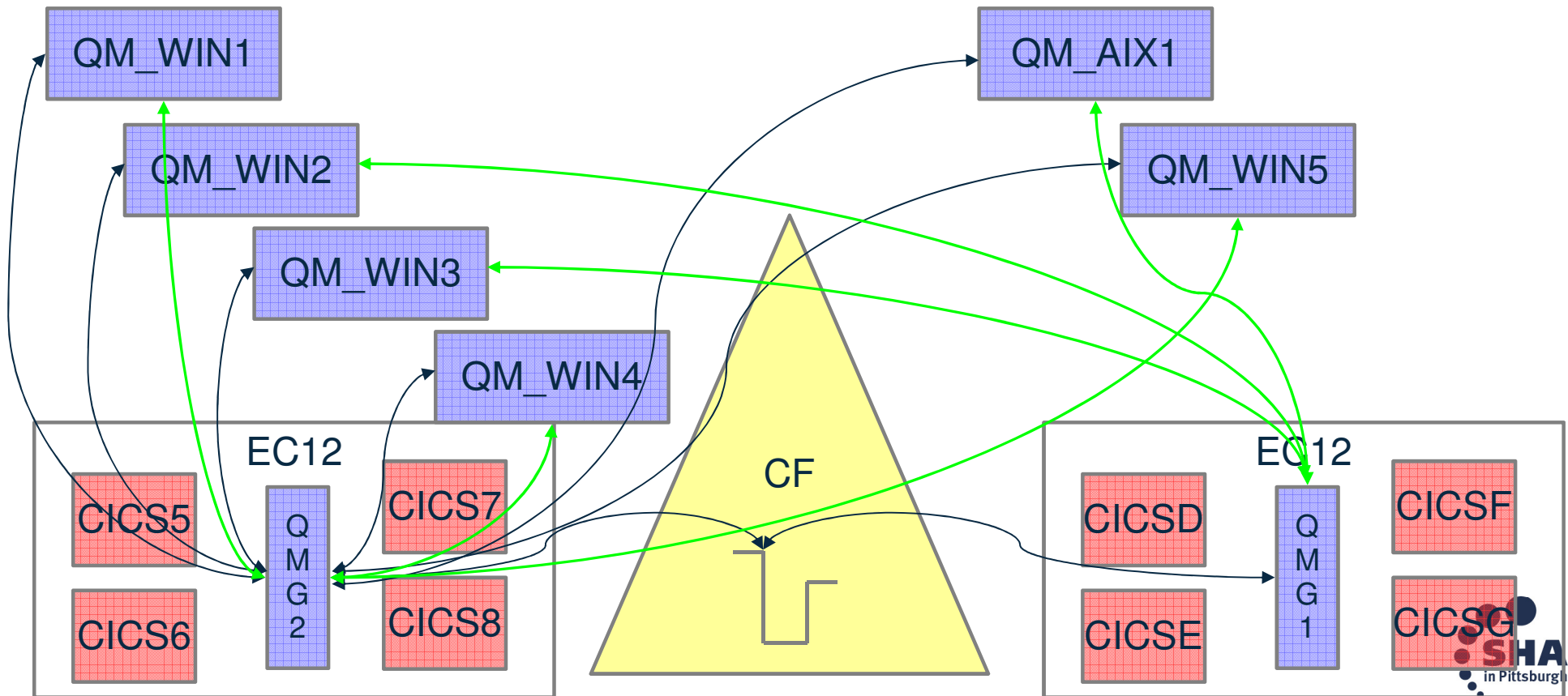
Notes

- **Gateway queue managers**

- The picture illustrates the mitigation technique.
- All connections from external systems were made to the gateway queue managers. The gateway queue managers did not have any application workload running. The customer had originally set up their topology this way for the following reasons:
 - The gateway queue managers were more available (stable) because they were not running any application work.
 - The logging for message PUTs coming in from their distributed queue managers and clients was confined to these queue managers, reducing the logging load on their application queue managers.
- As the messages are put to the shared queues, each ‘application owning’ queue manager has about the same chance of processing the requests.
- Even if connections get skewed, the back end workload is distributed more evenly.
- Note that they have experienced some workload skewing when upgrading the hardware underlying one LPAR, but that is now a known situation.

Re-driving Connections

- When a queue manager is unavailable, inbound connections can get skewed to the other queue manager(s) in the group.
 - This is normal availability processing!
 - Once a connection is live and active, no attempt is made to balance the connections once all the queue managers are available.



Notes

- **Re-driving connections**

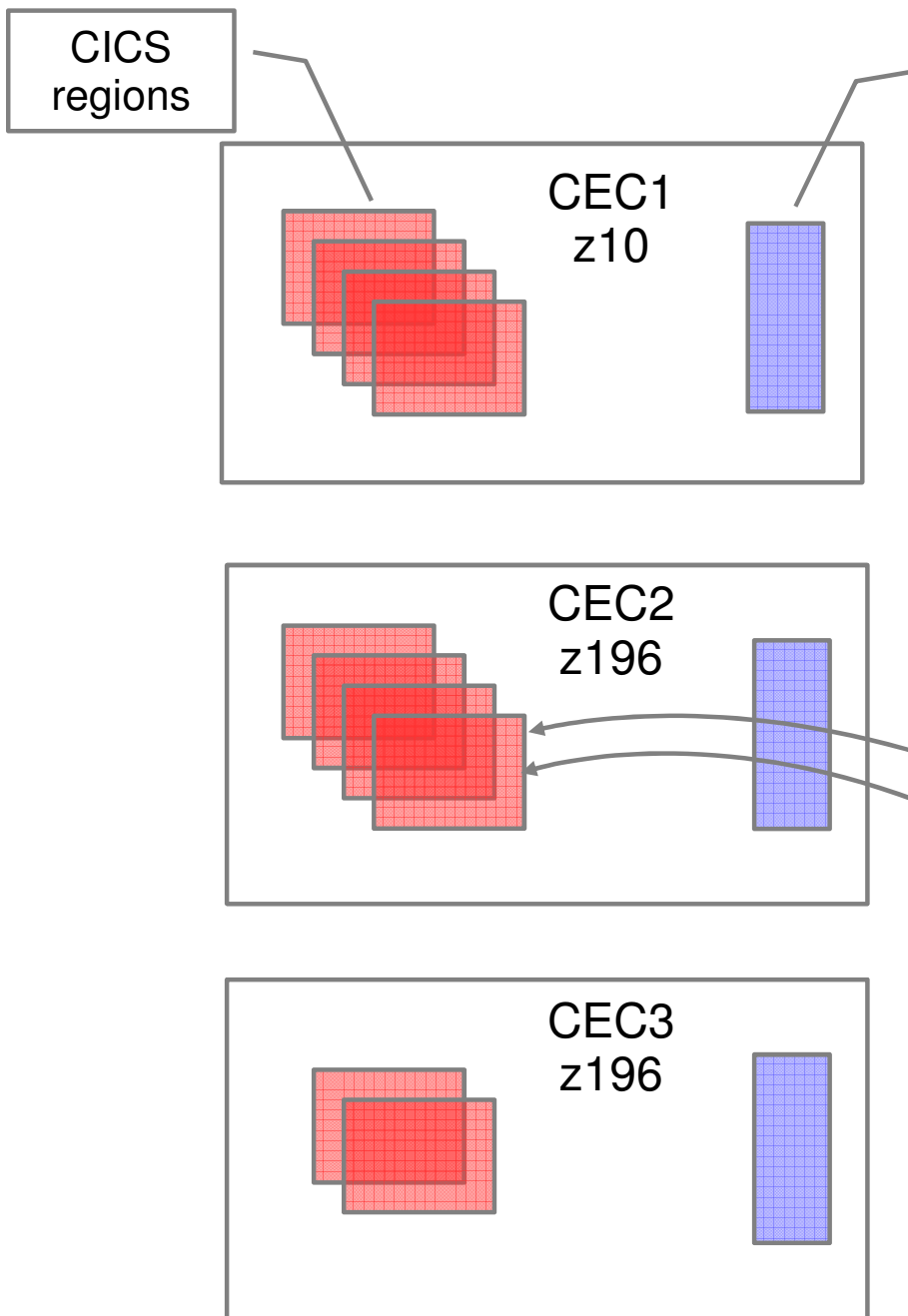
- The picture illustrates what can happen when connections are made to the QSG and one or more queue managers are unavailable at connection time.
 - This is working as designed, and is a major availability feature
 - BUT this can create a workload imbalance because of efforts to process work efficiently (locally)
- Once connections are made, they are not rebalanced.
 - The connections are not sent back thru the sysplex distributor as long as the channels are open and running.
- Channels should not be stopped from the receiving side, as that can lead to synchronization problems.
- Some customers have experienced real problems during peak processing periods
 - Often after a planned outage they find that the first queue manager brought back up has become 'connection concentrator' without realizing it
 - They want to redistribute the connections without sync problems, if possible
- One customer has created timed jobs for their distributed queue managers to stop and restart all channels to the queue sharing group.
 - They run this periodically to force the connections to be rebalanced.
 - This is a simple mitigation technique

CICS – CPSM Mitigation



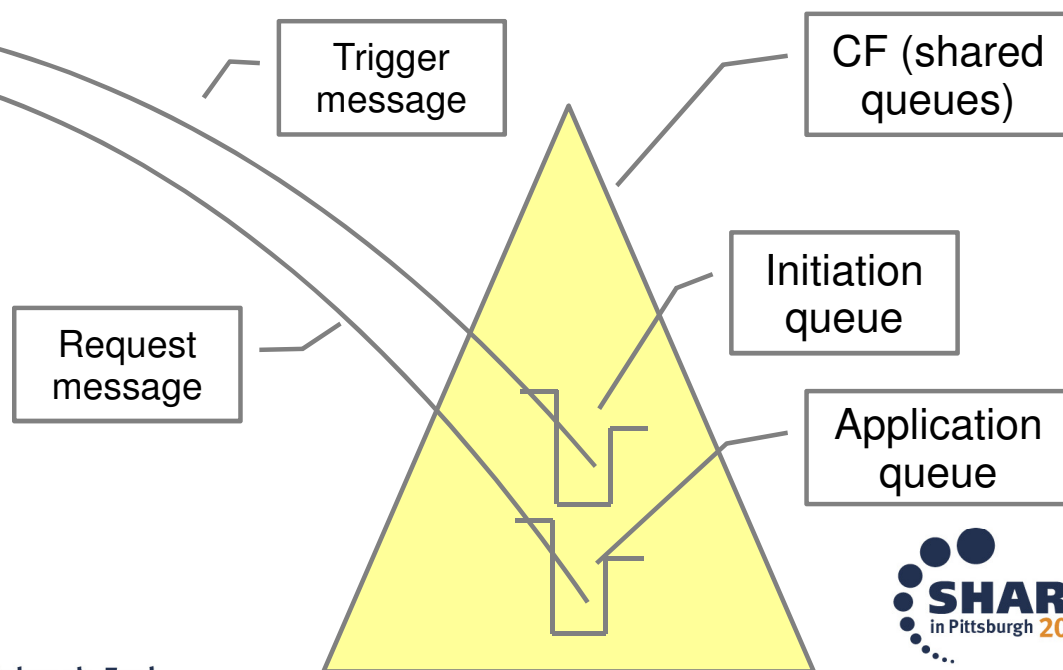
- The slides that follow outline a CPSM solution to the skewing problem based on the interaction between MQ triggering (CKTI) and CICS





Configuration for inbound WMQ work using triggering (schematic)

- Each CICS region acts as an independent consumer from the shared queues
- Unbalanced workload distribution



Notes

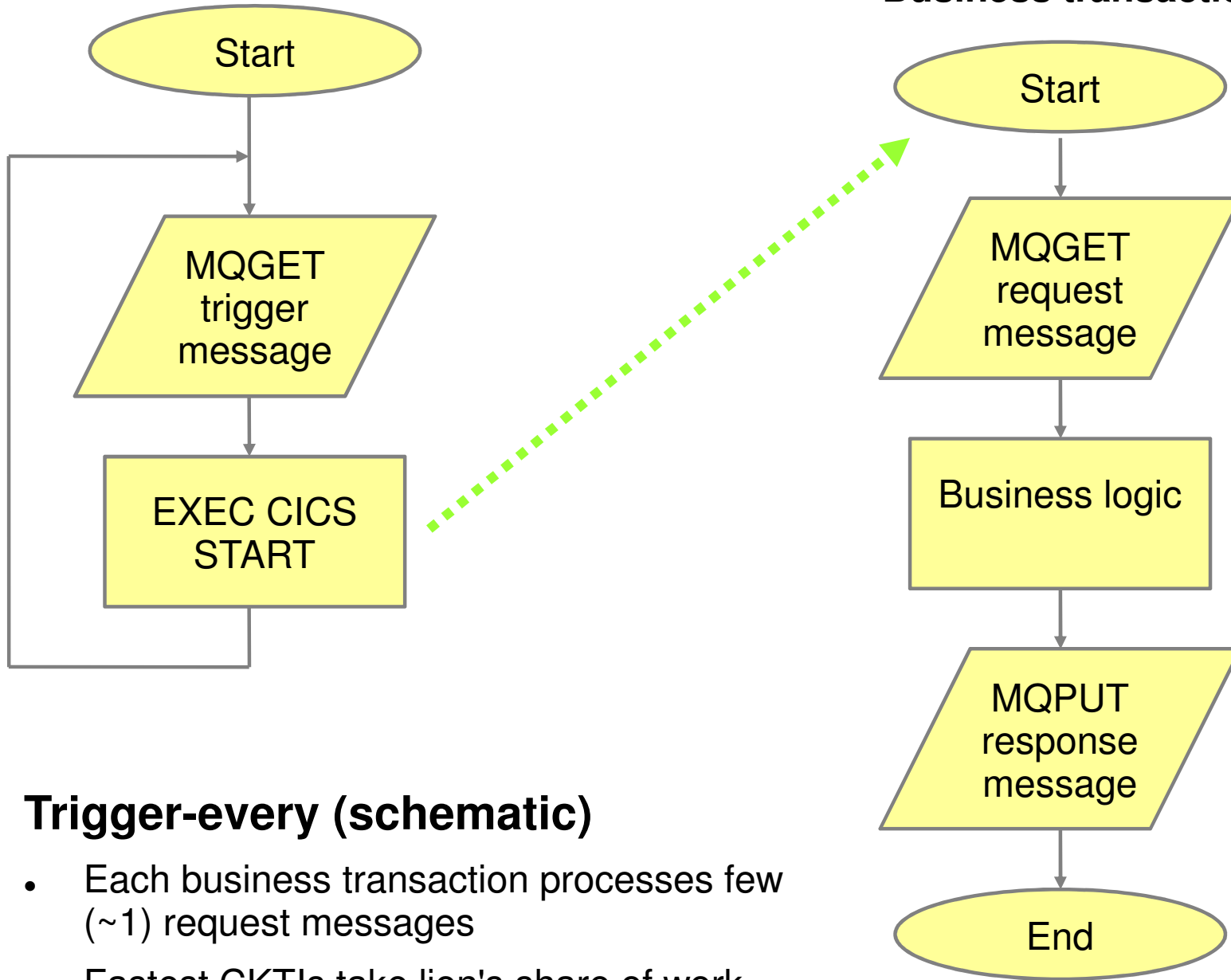
- **Slide illustrates basic elements of triggered WMQ message processing:**
 - Intention is to show how workload distribution issues can arise
 - Assumes multiple CICS regions in multiple CECs
 - Assumes use of WMQ shared queue – all CECs in same sysplex
 - Assumes trigger (CKTI) starts transactions in the same CICS region
 - Many different configurations are used – this is only an example
- **Features of the example configuration:**
 - Heterogeneous CICS regions
 - Different CECs have different power, memory, etc
 - Different CECs have different numbers of CICS regions
 - “Pull” workload distribution from shared WMQ queues
 - There is no routing of WMQ messages to CICS regions
 - Most aggressive (fastest) consumer gets lion's share of work – some machines over-utilised
 - Less aggressive consumers are starved – some machines under-utilised

Many customers want “balanced” or “even” workload distribution:

- Pricing considerations – On/Off Capacity on Demand (CoD)
- Why am I not using equipment that I could be using?
- ... and other reasons

Trigger monitor (CKTI)

Business transaction



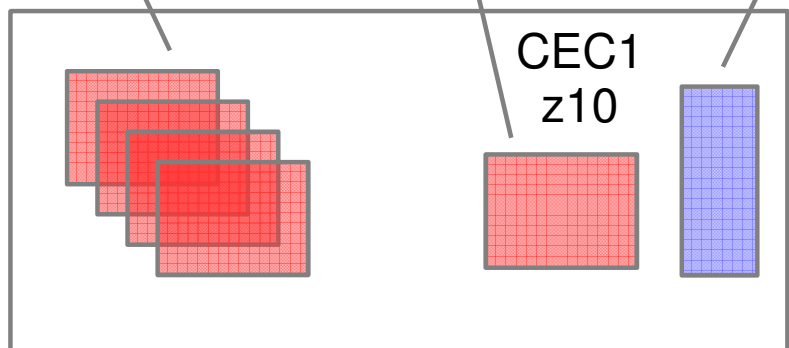
Trigger-every (schematic)

- Each business transaction processes few (~1) request messages
- Fastest CKTIs take lion's share of work

Notes

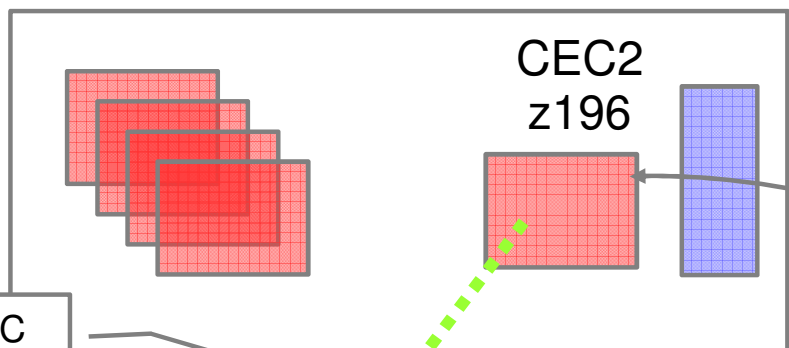
- **Slide illustrates typical CICS processing of WMQ messages using trigger-every:**
 - Trigger monitor (CKTI or similar) is an MQGET-WAIT loop on the initiation queue
 - When a request message arrives on shared application queue, trigger-every generates one trigger message¹ (*not*, for example, one for each queue manager or CKTI instance)
 - CKTI starts a business transaction for each trigger message (\approx one for each request *message*)
 - Business transaction processes one message from the application queue
 - MQGETs a request message, executes business logic, and MQPUTs a response message
- " **Other characteristics of trigger-every:**
 - Each WMQ message is processed by a separate CICS transaction
 - Lightweight communication between monitor and business transaction (no message payload)
 - CICS region running business logic needs a connection to WMQ
 - MQOPEN and MQCLOSE for each message
- ▶ **Pull, not push, distribution of trigger messages:**
 - There is no routing of trigger messages to CKTI instances (first come, first served)
 - Next MQGET-WAIT in CKTI does not wait for business logic to complete
 - CKTI on fastest machine consumes trigger messages fastest (all other things being equal)
- **Push distribution of business transaction (EXEC CICS START):**
 - EXEC CICS START can start the transaction in any CICS region in the sysplex
 - Indicated by the green arrow
- ▶ **Opportunity to distribute work (business logic) across regions in the sysplex**

¹ This is a simplification. Refer to the WMQ InfoCenter for a definitive description of triggering with shared queues.

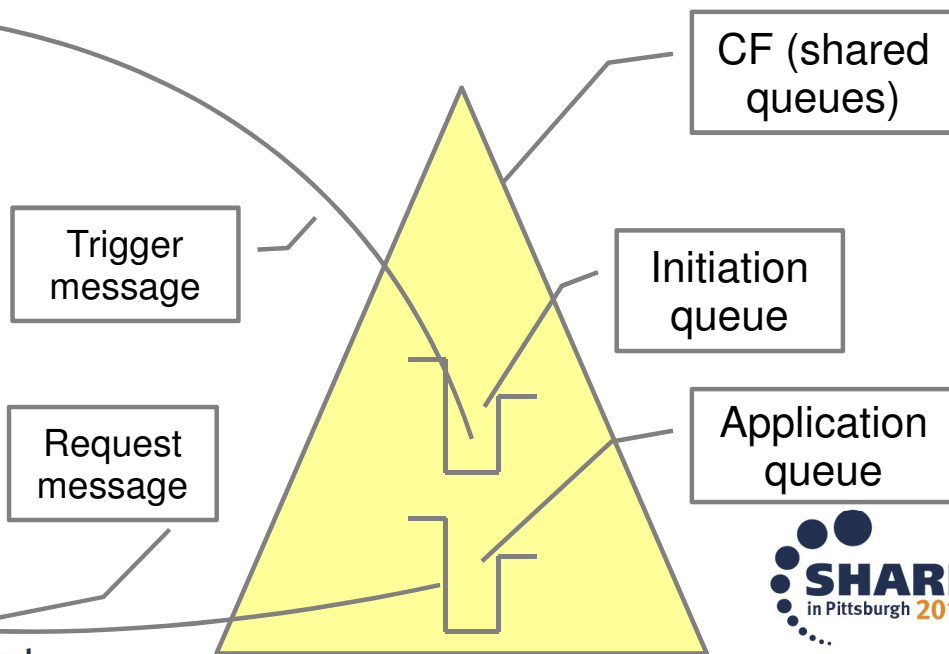
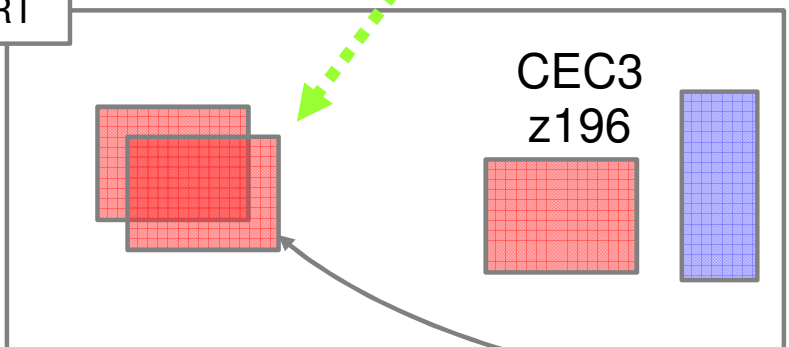


Preferred configuration for trigger-every (schematic)

- “TORs” run CKTI
- Each acts as an independent consumer from the shared init queue
- Each distributes STARTs across all AORs
- “Balanced” workload distribution



EXEC
CICS
START

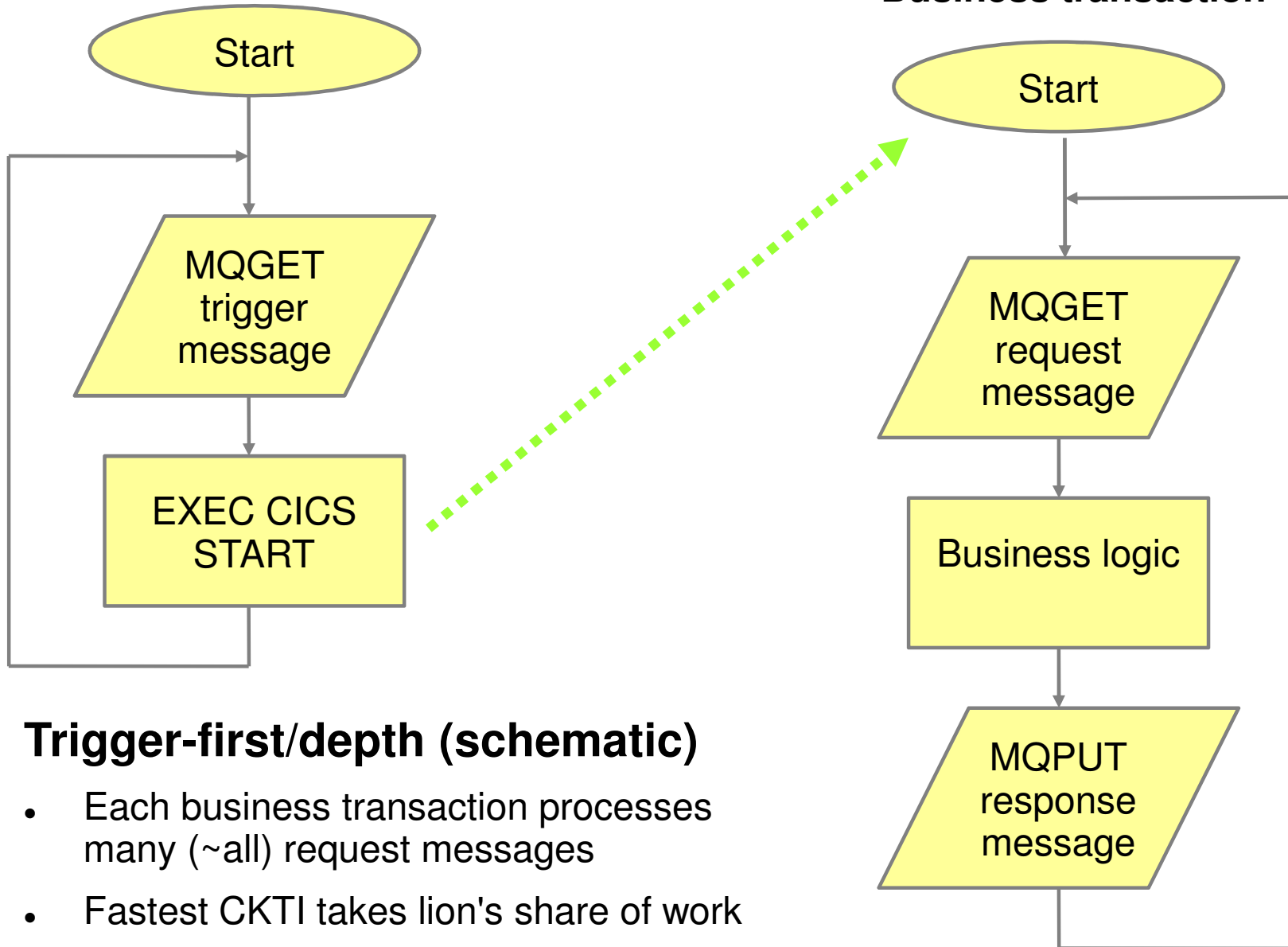


Notes

- **Slide illustrates a preferred CICS configuration for trigger-every:**
 - Uses front-end routing regions (“TORs”) that run the trigger monitor (CKTI)
 - CKTI starts (EXEC CICS START) a business transaction for each trigger message
 - CPSM routing directs the start to a suitable back-end region (AOR)
 - Indicated by the green arrow
 - Business transaction MQGETs a request message, executes business logic, and MQPUTs a response message – all in the AOR
 - Next MQGET-WAIT in CKTI does not wait for business logic to complete
- **Preferred technology for CPSM routing:**
 - Link-neutral goal algorithm for “remote” starts
 - Selects target AOR based on AOR load and health
 - Does not “prefer” local (= same LPAR) AORs
 - Even distribution across AORs, but ...
 - ... responds to transient load/health variation
 - XCF MRO for “remote” starts
 - High-performance System z sysplex technology
 - Uses coupling facility (CF) instead of TCP/IP stack
 - Sysplex-optimised workload routing
 - Highly responsive to transient variations
 - Uses CF to maintain current status for AORs

Trigger monitor (CKTI)

Business transaction



Trigger-first/depth (schematic)

- Each business transaction processes many (~all) request messages
- Fastest CKTI takes lion's share of work
- Corresponding business transaction takes lion's share of the work

Notes

- **Slide illustrates typical CICS processing of WMQ messages using trigger-first/depth:**
 - Trigger monitor (CKTI or similar) is an MQGET-WAIT loop on the initiation queue
 - When a new request message arrives on an empty shared-queue (trigger-first) or increases the queue depth to the specified value (trigger-depth), WMQ generates one trigger message for each queue manager¹ (*not*, for example, one for each CKTI instance).
 - CKTI starts a business transaction for each trigger message (\approx one for each request *queue*)
 - Business transaction is an MQGET-WAIT loop on the application queue
 - MQGETs a request message, executes business logic, and MQPUTs a response message; then loops back to process the next message.

Other characteristics of trigger-first/depth:

- Many WMQ messages from the same queue processed by the same CICS transaction
- No per-message communication between monitor and business transaction
- One MQOPEN and MQCLOSE for many messages
- CICS region running business logic needs a connection to WMQ

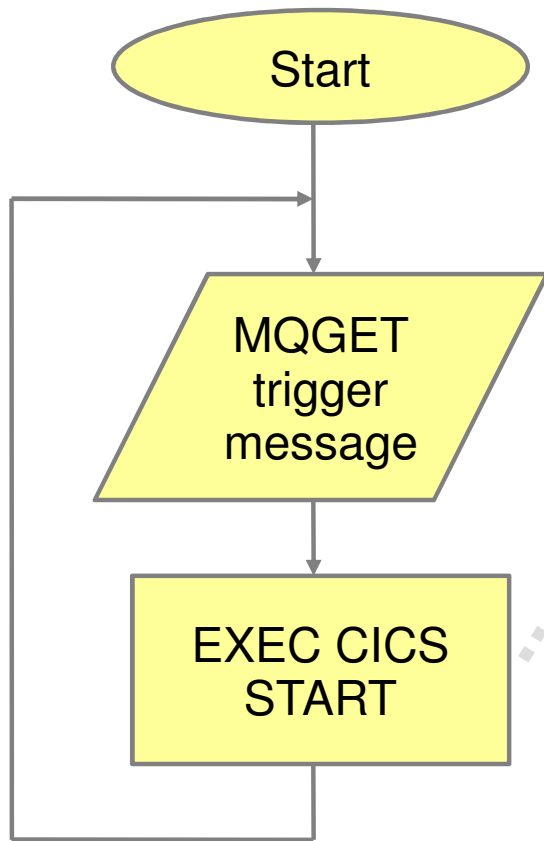
No distribution of messages:

- There is no routing of trigger messages or request messages (first come, first served)
- CKTI can start the business transaction in any CICS region (indicated by the green arrow), but...
- Business transaction on fastest machine consumes request messages fastest (all other things being equal)

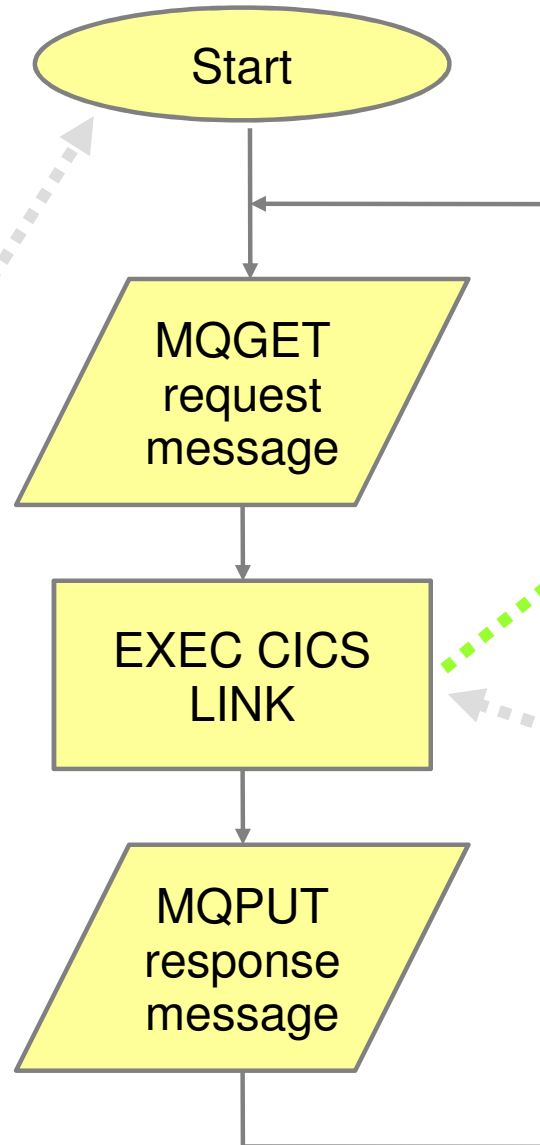
- **No opportunity to control distribution of work (business logic) across regions in the sysplex**

¹ This is a simplification. Refer to the WMQ InfoCenter for a definitive description of triggering with shared queues.

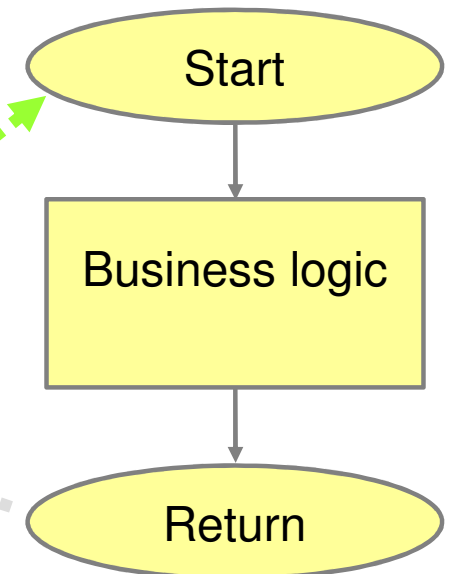
Trigger monitor (CKTI)



Staging transaction



Business logic



Trigger-first/depth staged (schematic)

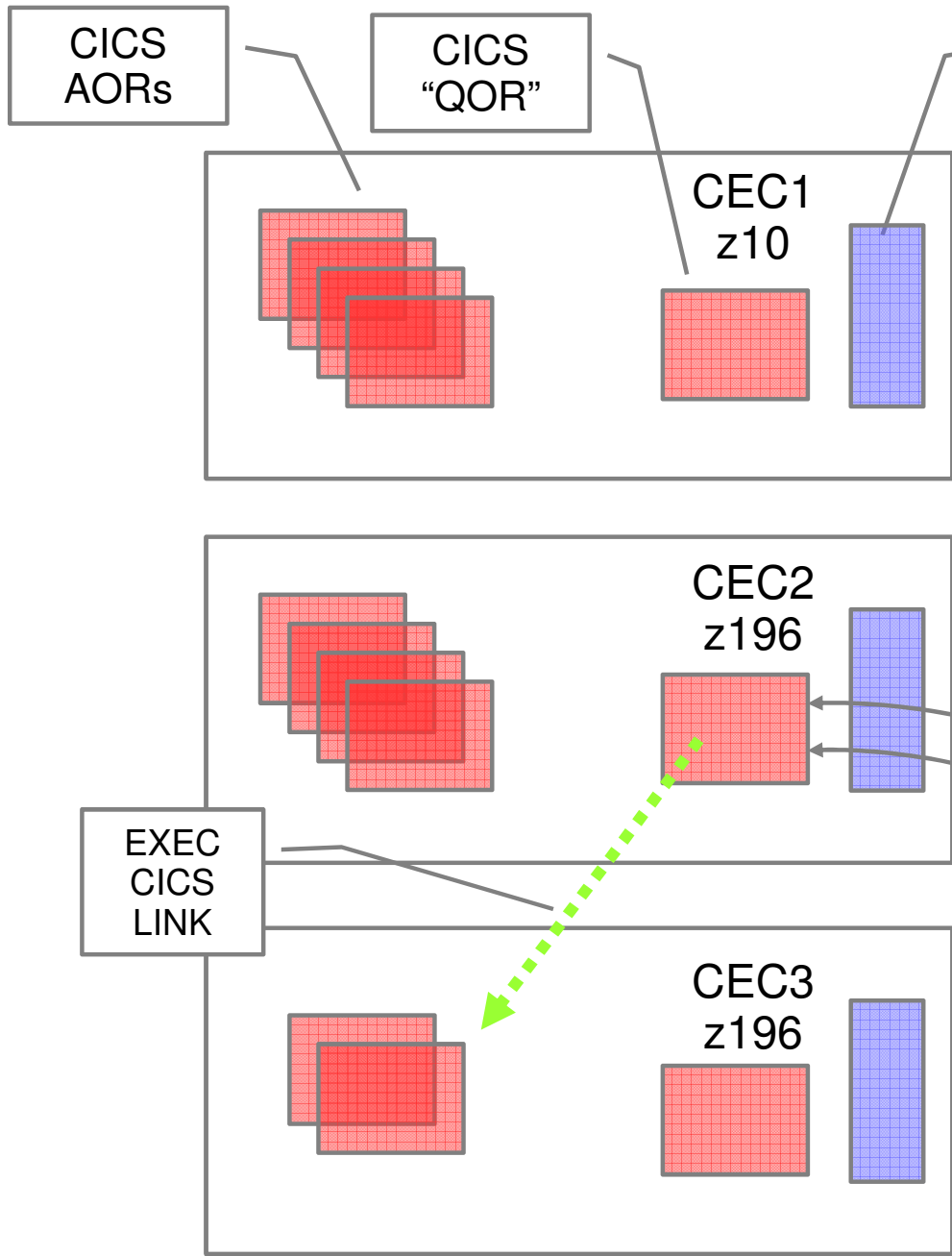
- Staging transaction processes all request messages

- Business transaction processes one request message

Notes

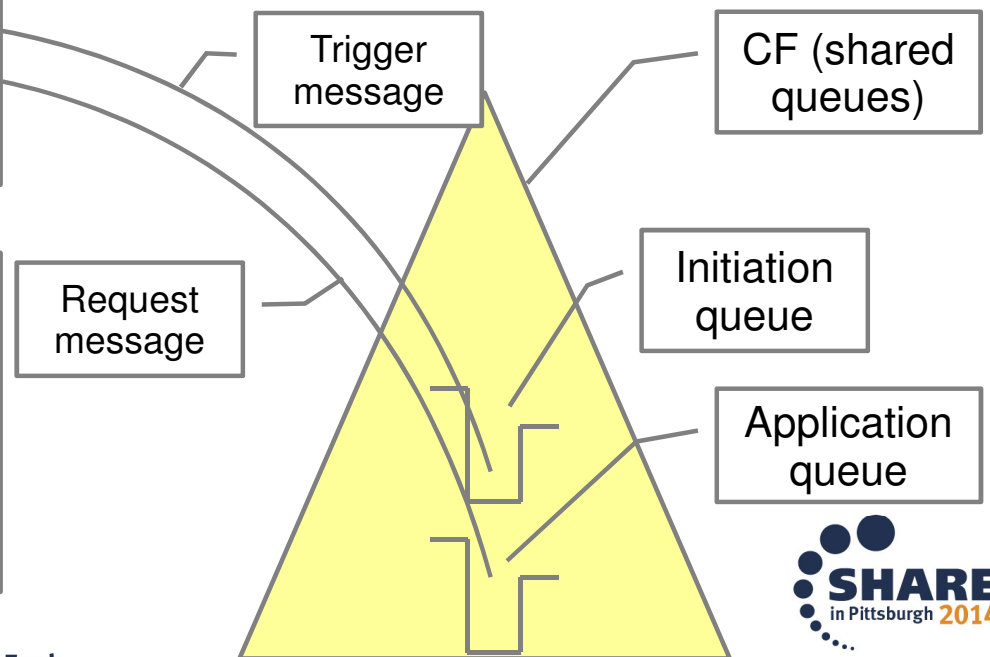
- **Slide illustrates typical CICS processing of WMQ messages using trigger-first/depth with staging:**
 - Trigger monitor (CKTI or similar) is an MQGET-WAIT loop on the initiation queue
 - When a new request message arrives on an empty shared-queue (trigger-first) or increases the queue depth to the specified value (trigger-depth), WMQ generates one trigger message for each queue manager¹ (*not*, for example, one for each CKTI instance).
 - CKTI starts a staging transaction for each trigger message (\approx one for each request *queue*)
 - Staging transaction is an MQGET-WAIT loop on the application queue
 - MQGETs a request message, links to business logic (passes payload), and MQPUTs a response message (payload returned by business logic); then loops back to process the next message.
- **Other characteristics of trigger-first/depth with staging:**
 - Many WMQ messages from the same queue processed by the same CICS transaction
 - Communication between staging transaction and business logic includes message payloads
 - One MQOPEN and MQCLOSE for many messages
 - CICS region running business logic does not need a connection to WMQ
- ☞ **No distribution of messages, push distribution of work (business logic):**
 - There is no routing of trigger messages or request messages (first come, first served)
 - Staging transaction can link to any CICS region in the sysplex (indicated by the green arrow),
- **Opportunity to control distribution of work (business logic) across regions in the sysplex**

¹ This is a simplification. Refer to the WMQ InfoCenter for a definitive description of triggering with shared queues.



Preferred configuration for trigger-first/depth staged (schematic)

- “QORs” run CKTI and staging transaction
- Each acts as an independent consumer from the shared init queue and shared application queue
- Each distributes LINKs across all AORs
- “Balanced” workload distribution



Notes

- **Slide illustrates a preferred CICS configuration for trigger-first/depth with staging:**
 - Uses front-end routing queue-owning regions (QORs) to perform WMQ interactions:
 - Trigger monitor (CKTI) runs in QOR. Starts a staging transaction for each trigger message
 - Staging transaction runs in QOR. Links to business logic for each request message (passes message payload in container or COMMAREA)
 - CPSM routing directs the link to a suitable back-end region (AOR)
 - Indicated by the green arrow
 - Business logic processes request message payload, executes business logic, and returns response message payload in container or COMMAREA
 - Staging transaction MQPUTs the response, MQGETs the next request, and loops

Preferred technology for CPSM routing:

- Link-neutral goal algorithm for “remote” starts
 - Selects target AOR based on AOR load and health
 - Does not “prefer” local (= same LPAR) AORs
 - Even distribution across AORs, but ...
 - ... responds to transient load/health variation
- XCF MRO for “remote” starts
 - High-performance System z sysplex technology
 - Uses coupling facility (CF) instead of TCP/IP stack
- Sysplex-optimised workload routing
 - Highly responsive to transient variations
 - Uses CF to maintain current status for AORs

Highlights

- **Solution uses proven technology for CPSM routing:**
 - Each TOR/QOR uses link-neutral goal algorithm
 - Selects target AOR based on AOR load and health
 - Does not “prefer” local (= same LPAR) AORs
 - Even distribution across AORs, but ...
 - ... responds to transient load/health variation
 - XCF MRO for “remote” STARTs or LINKs
 - High-performance System z sysplex technology
 - Uses coupling facility (CF) instead of TCP/IP stack
 - Sysplex-optimised workload routing
 - Highly responsive to transient variations
 - Uses CF to maintain current status for AORs
- **Continuous operation and high availability through WMQ shared queues:**
 - “Glitchless” recovery from region/LPAR/CEC outage
 - “Instant” redistribution of workload
 - In-flight messages backed-out, restart in another CICS region
- **High throughput:**
 - Exploits all available capacity
 - Highly responsive to transient spare capacity

MQ Workload Balance Summary

- MQ is a message delivery system, it does not try to balance workload
- Balancing the workload is attempting a technical solution for what is often a pricing problem
 - Beware spending a lot of effort for a solution to a temporary problem as well!
 - Turning off performance improvements like put to waiting getter will impact all applications, not just the skewed ones
- There are some mitigation techniques that can help the overall environment
 - Clustering
 - Gateway queue managers
 - Using CPSM to make appropriate routing decisions

Additional Resources

- The following links are to additional information about WMQ
 - Queue Sharing Groups:
http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.explorer.doc/e_qsg.htm
 - Clustering:
<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/qc11220.htm>
 - Intercommunication
<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/zx00011.htm>
 - Redbooks:
 - IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements
<http://www.redbooks.ibm.com/abstracts/sg248087.html?Open>
 - High Availability in WebSphere Messaging Solutions
<http://www.redbooks.ibm.com/abstracts/sg247839.html?Open>
 - WebSphere MQ Queue Sharing Group in a Parallel Sysplex environment (dated, but still good basic information)
<http://www.redbooks.ibm.com/redpieces/abstracts/redp3636.html?Open>
 - My first YouTube video:
<http://www.youtube.com/playlist?list=PL9N7JP2yU3T8JycrCOvEPM8c-0UdE97VT>

This was session 15998 - The rest of the week

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30			16203 Application programming with MQ verbs (Chris Leonard)	16202 The Dark Side of Monitoring MQ - SMF 115 and 116 Record Reading and Interpretation (Lyn Elkins)	15998 CICS and MQ - Workloads Unbalanced! (Lyn Elkins)
10:00					
11:15	16194 Introduction to MQ (Chris Leonard)	16199 What's New in IBM Integration Bus & WebSphere Message Broker (David Coles)	15844 MQ – Take Your Pick Lab (Ralph Bateman, Lyn Elkins)	16197 Using IBM WebSphere Application Server and IBM WebSphere MQ Together (Chris Leonard)	 <p>You are HERE!</p>
12:15					
01:30		16195 All about the new MQ v8 (Mark Taylor)	16192 MQ Security: New v8 features deep dive (Neil Johnston)	16201 New MQ Chinit monitoring via SMF (Mayur Raja)	
03:00	16205 MQ Beyond the Basics (Neil Johnston)	16204 MQ & DB2 – MQ Verbs in DB2 & InfoSphere Data Replication (Q Replication) Performance (Lyn Elkins)	15503 What's wrong with MQ? (Lee E. Wheaton)	16200 IIIB - Internals of IBM Integration Bus (David Coles)	
04:15	16198 First Steps with IBM Integration Bus: Application Integration in the new world (David Coles)	16193 MQ for z/OS v8 new features deep dive (Mayur Raja)	16196 MQ Clustering - The Basics, Advances and What's New in v8 (Neil Johnston)		

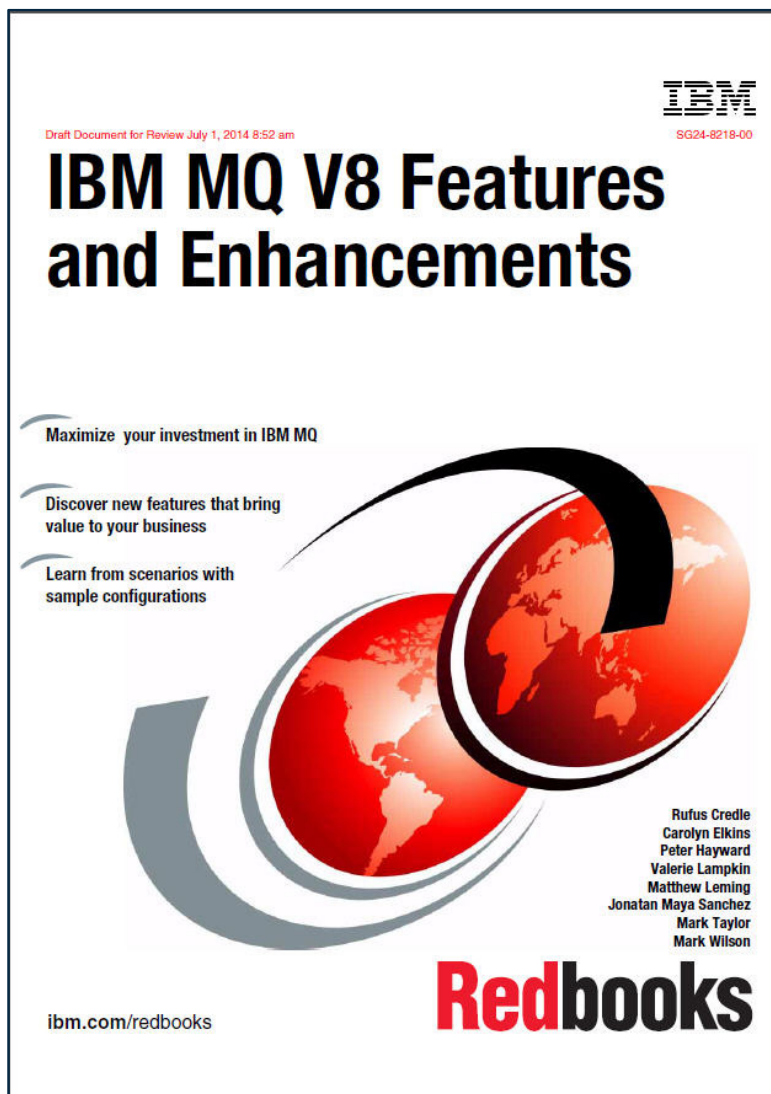
Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Further information in real books



Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

And ... already available (draft)



IBM

Draft Document for Review July 1, 2014 8:52 am

IBM MQ V8 Features and Enhancements

SG24-8218-00

Maximize your investment in IBM MQ

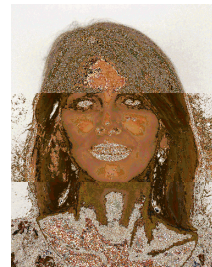
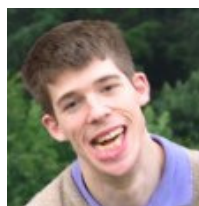
Discover new features that bring value to your business

Learn from scenarios with sample configurations

Rufus Credle
Carolyn Elkins
Peter Hayward
Valerie Lampkin
Matthew Leming
Jonatan Maya Sanchez
Mark Taylor
Mark Wilson

ibm.com/redbooks

Redbooks



<https://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248218.html>

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

MQ Workload Balance - thanks



- Many thanks to
 - Steve Hobson for the CICS/CPSM expertise and the wonderful graphics
 - Mark Taylor and Gene Kuelhthau for their patience and guidance on the rest of the foils
 - Mark Taylor for providing the excellent editing and recording studio



Thank you!

- Remember this was session 15998
- And MQ has left the convention center!

