What's new in Messaging for WAS v8.5 and Complex SIBus topologies

Lohitashwa Thyagaraj
WebSphere Messaging – Product Architect
IBM Master Inventor

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

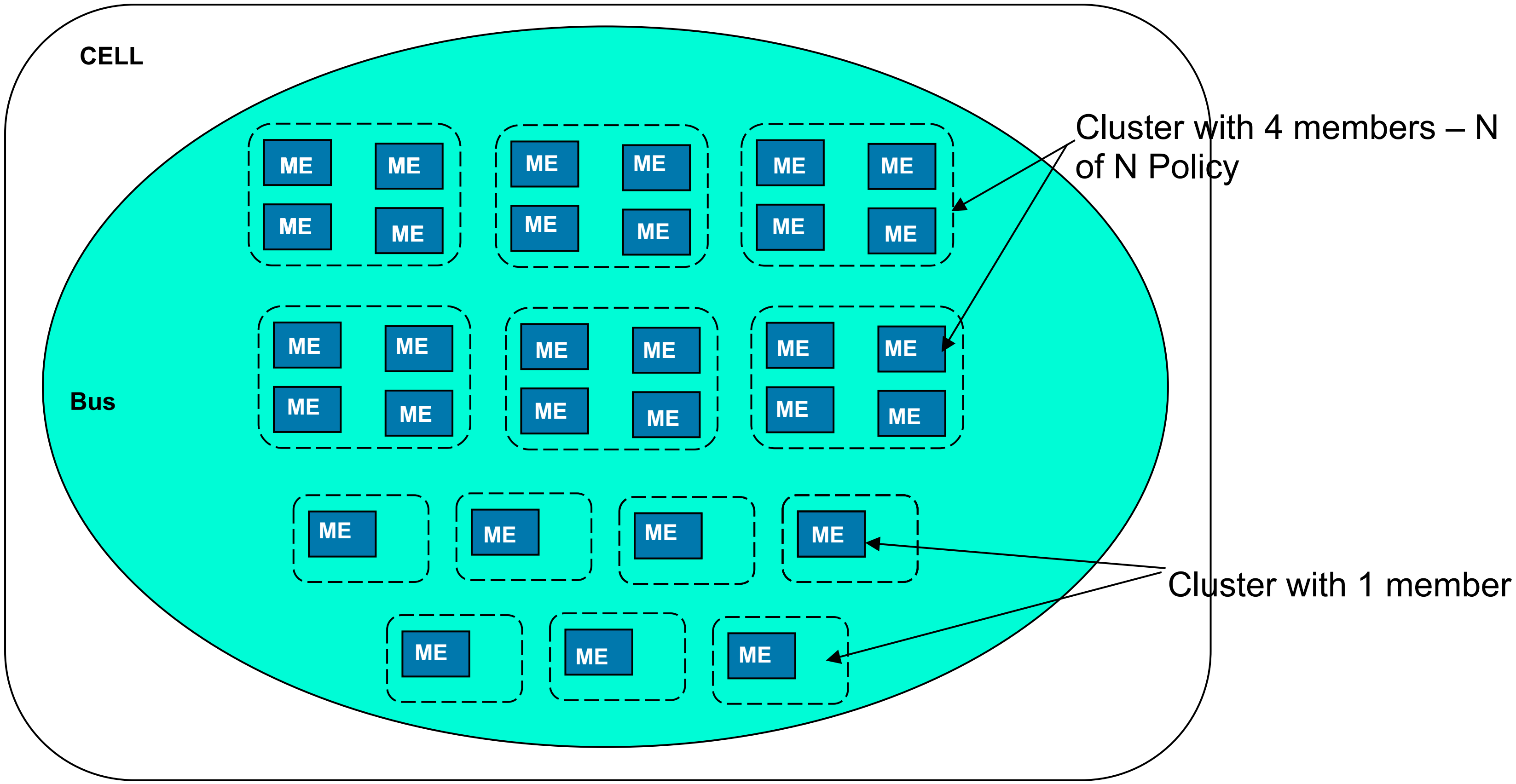
The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Objectives

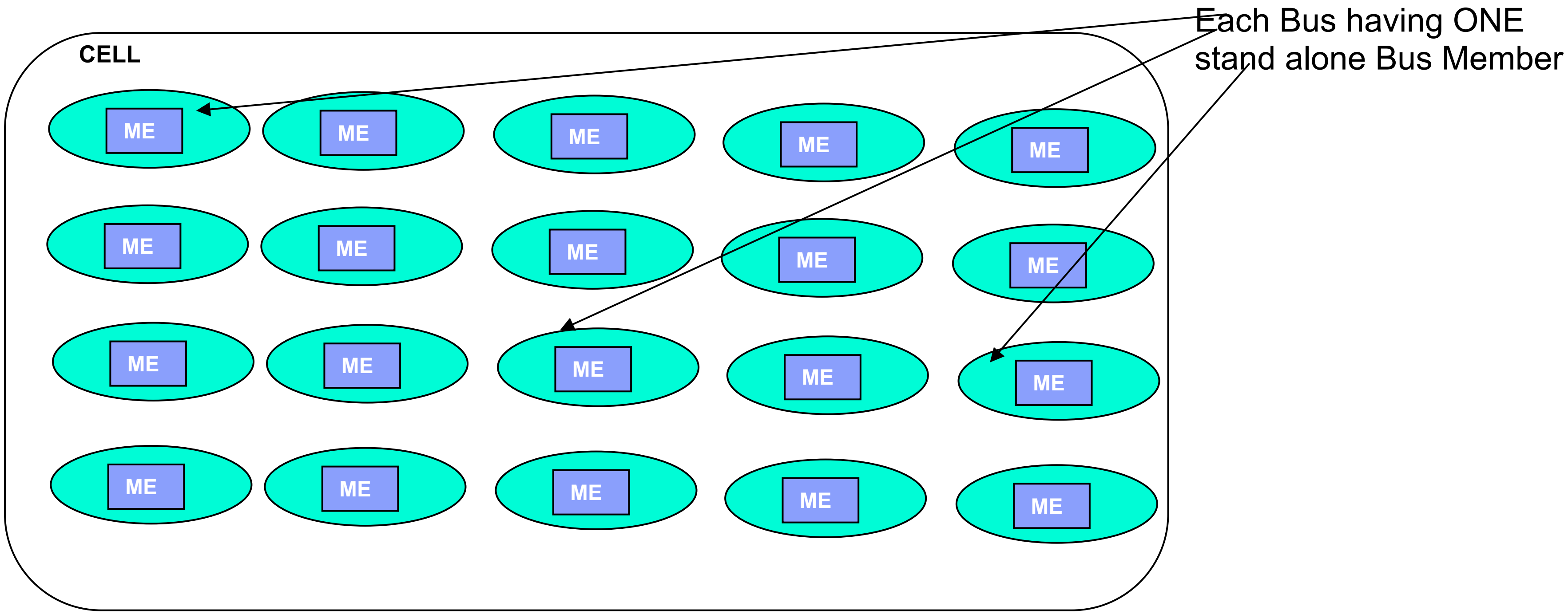
- Discuss the factors that lead to more complex SIBus topologies
- Explore some of the subtler aspects of SIBus topologies that ensure a correctly functioning, manageable and scalable system
- Examine various SIBus topologies and to know how to get the best out of them
- What's new in Messaging for WAS v8.5

Interesting Customer Topologies (1)



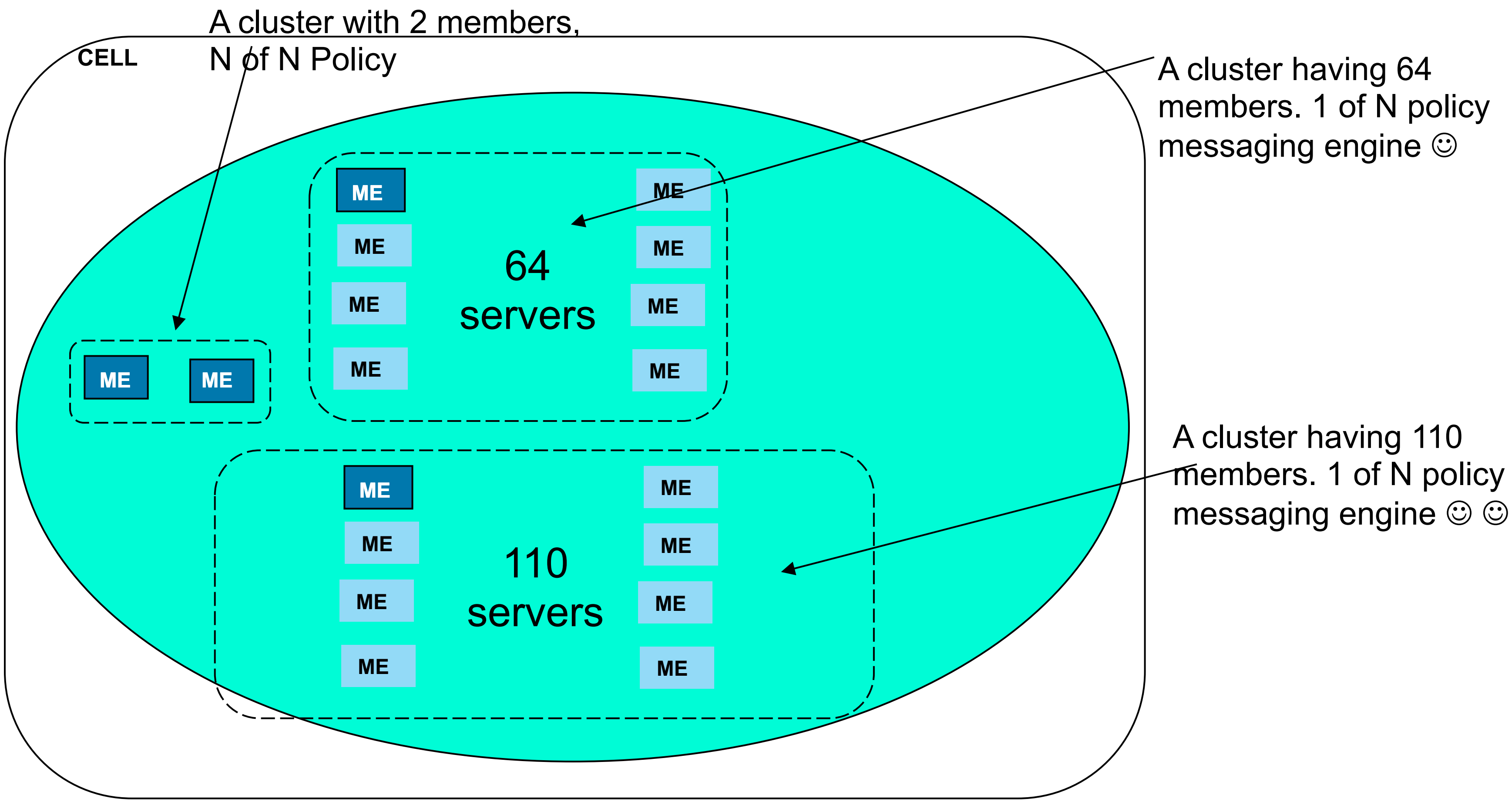
- 42 Destinations
- 42 Exception Destinations
- 25+ connection factory/Activation Spec definition

Interesting Customer Topologies (2)



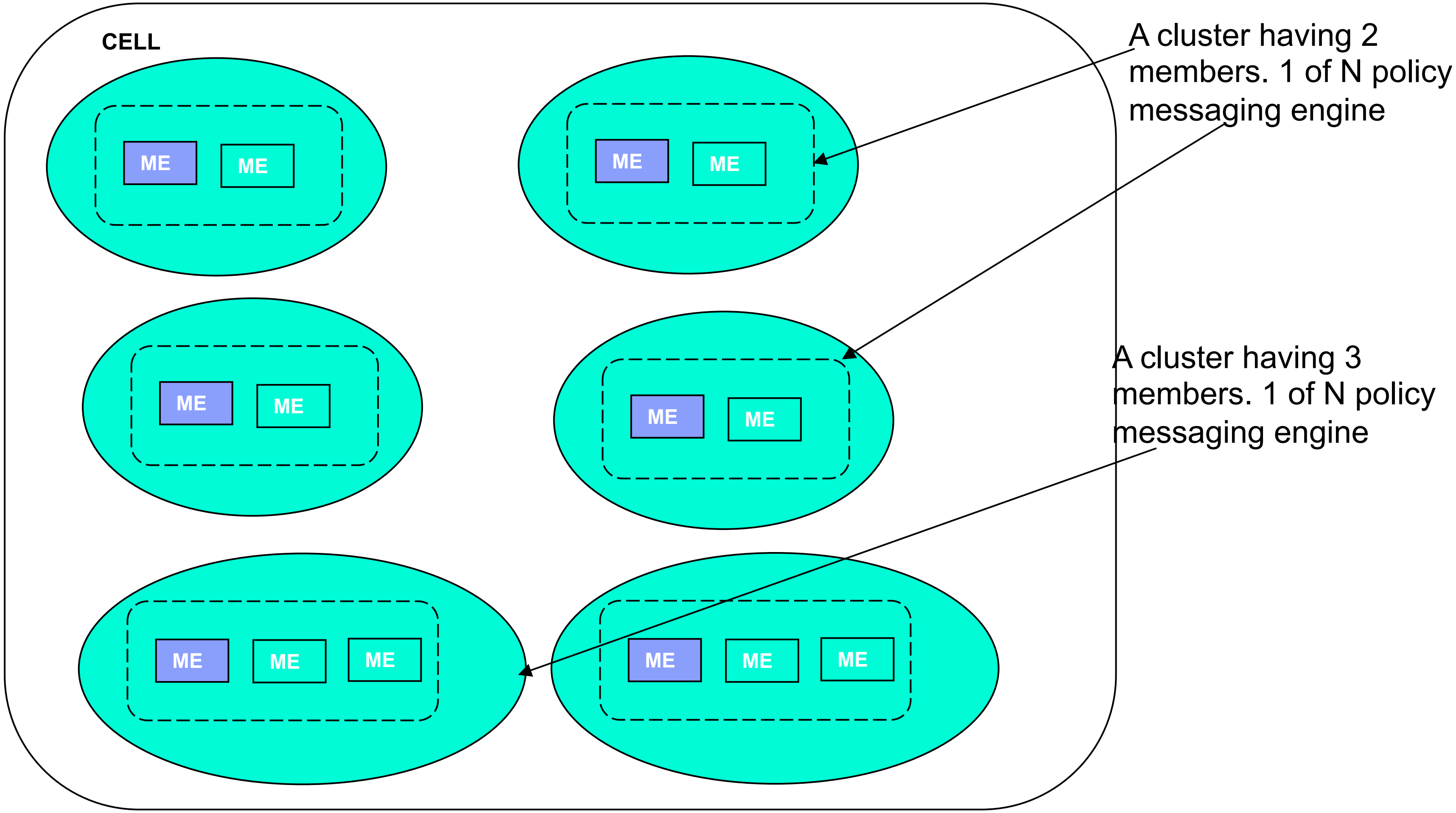
- 20 Buses in a cell
- Each bus having one Bus member (stand alone – no HA)
- Most of the buses having "bus-links"
- All buses having identical Queues
- Several JMS resources defined to manage individual bus connections

Interesting Customer Topologies (3)



- 34 Nodes
- 11 Clusters
- 30 Destinations

Interesting Customer Topologies (4)



High level concepts

•The SIBus is a ***location transparent***, asynchronous messaging system that allows applications to access any resource on the bus, regardless of location

•SIBus is an ***embedded component*** of WebSphere Application Server, automatically installed as part of an application server

•It is configured using the WAS admin model and administered using the same mechanisms as any other aspect of WAS

•The SIBus ***runs within the application servers***, requiring no separate runtime

•Simple SIBus topologies can run in standalone WAS servers

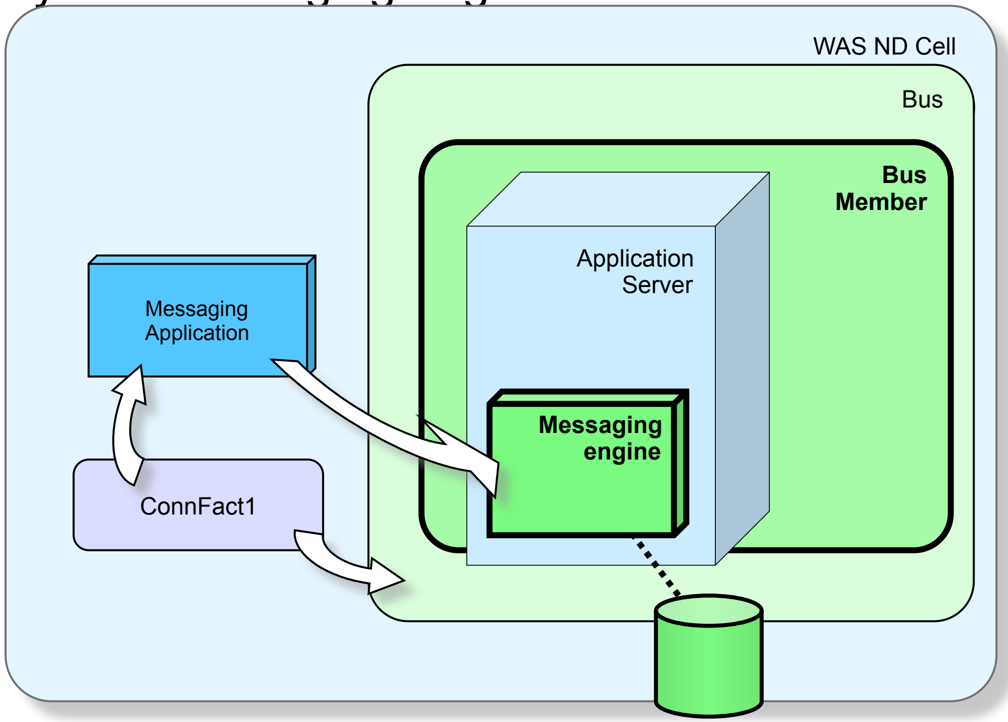
•Networked SIBus topologies sit on top of WAS ND cell topologies, benefiting from no additional network configuration

•SIBus exploits the WAS ND ***clustering*** model to provide ***scalability*** and ***high availability***

SIBus runtime model

- **Bus Members**
  - WAS **application servers** or **clusters** of servers that are enabled to *host* the SIBus runtime component
    - ➢ Each SIBus runtime is a messaging engine
    - ➢ Clusters enable high availability and scalability of the SIBus runtime
  - A bus requires *at least one bus member* to function
  - The application server that a JMS application runs in *does not need to* be a bus member, it can connect remotely to a messaging engine in another application server.

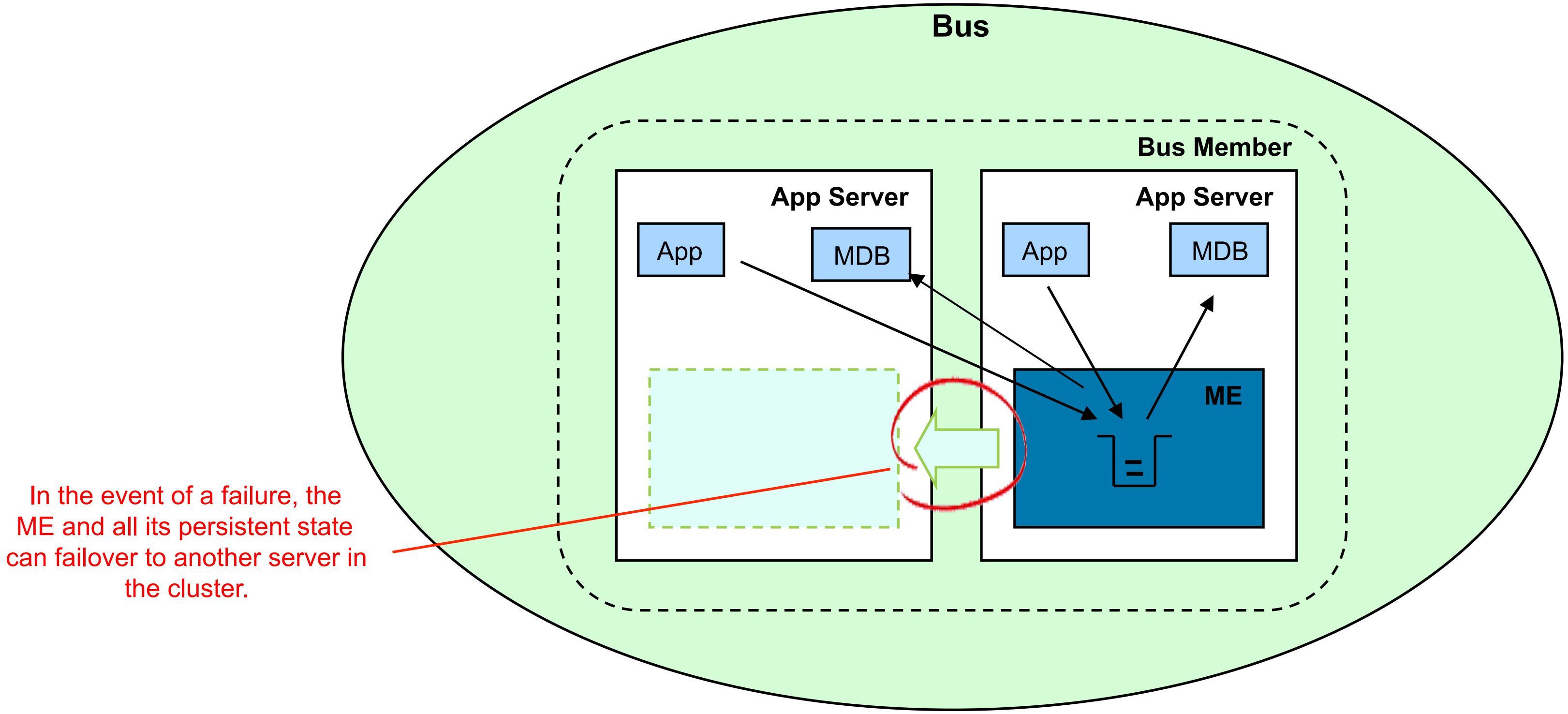
- **Messaging engines (MEs)**
  - Each ME owns specific bus resources
  - Each ME has its own set of persistent storage
  - Applications physically connect to an ME, the ME satisfies the application's messaging needs.



Realizing HA topology with SIBus

- Single Cluster
- Multiple Cluster
- Maximizing HA

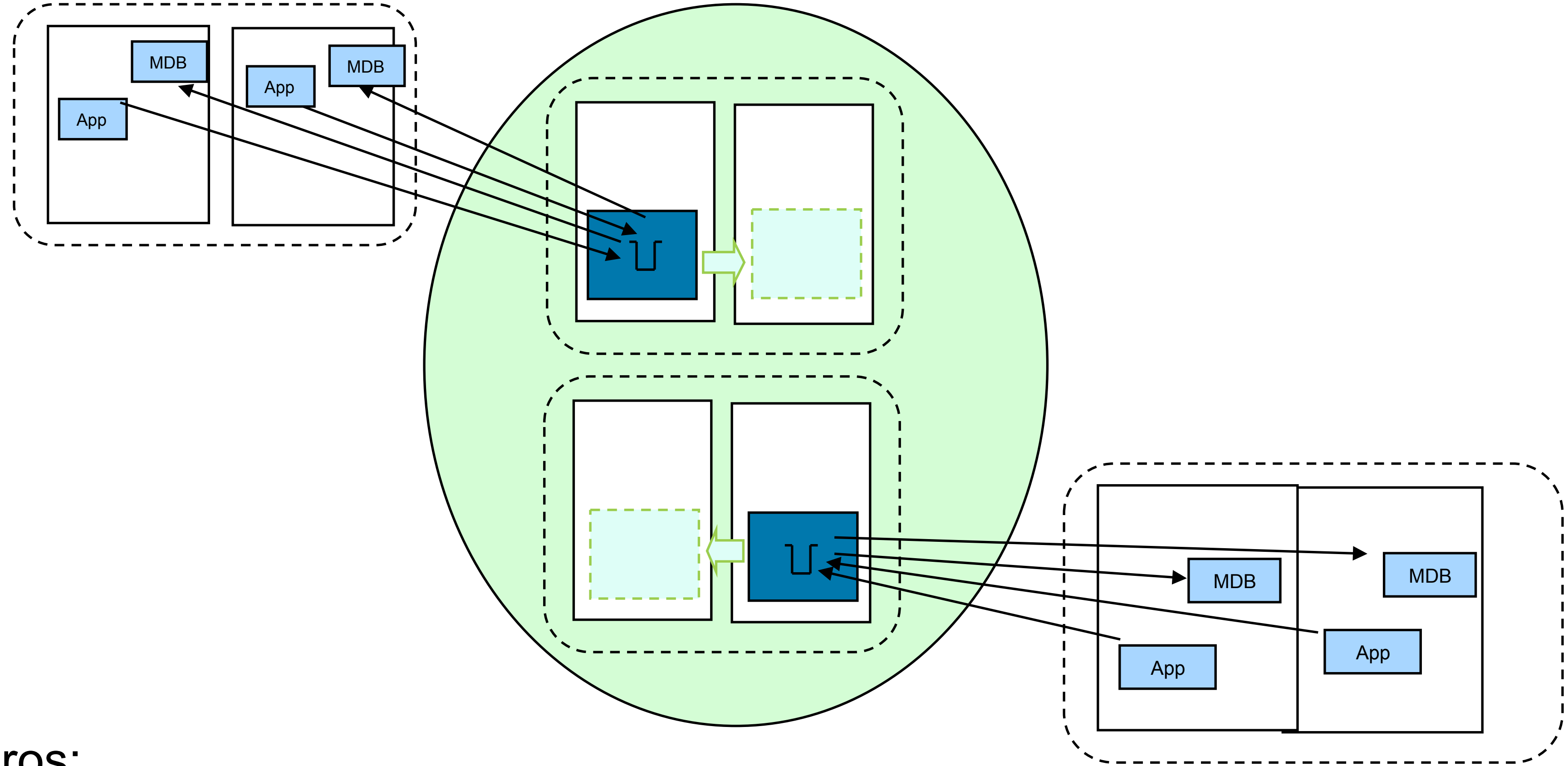
HA Option 1: Single Cluster



•Pros:
- All Messaging is still in one place
- Scalability of applications
- High availability of Applications
- High availability of Messaging

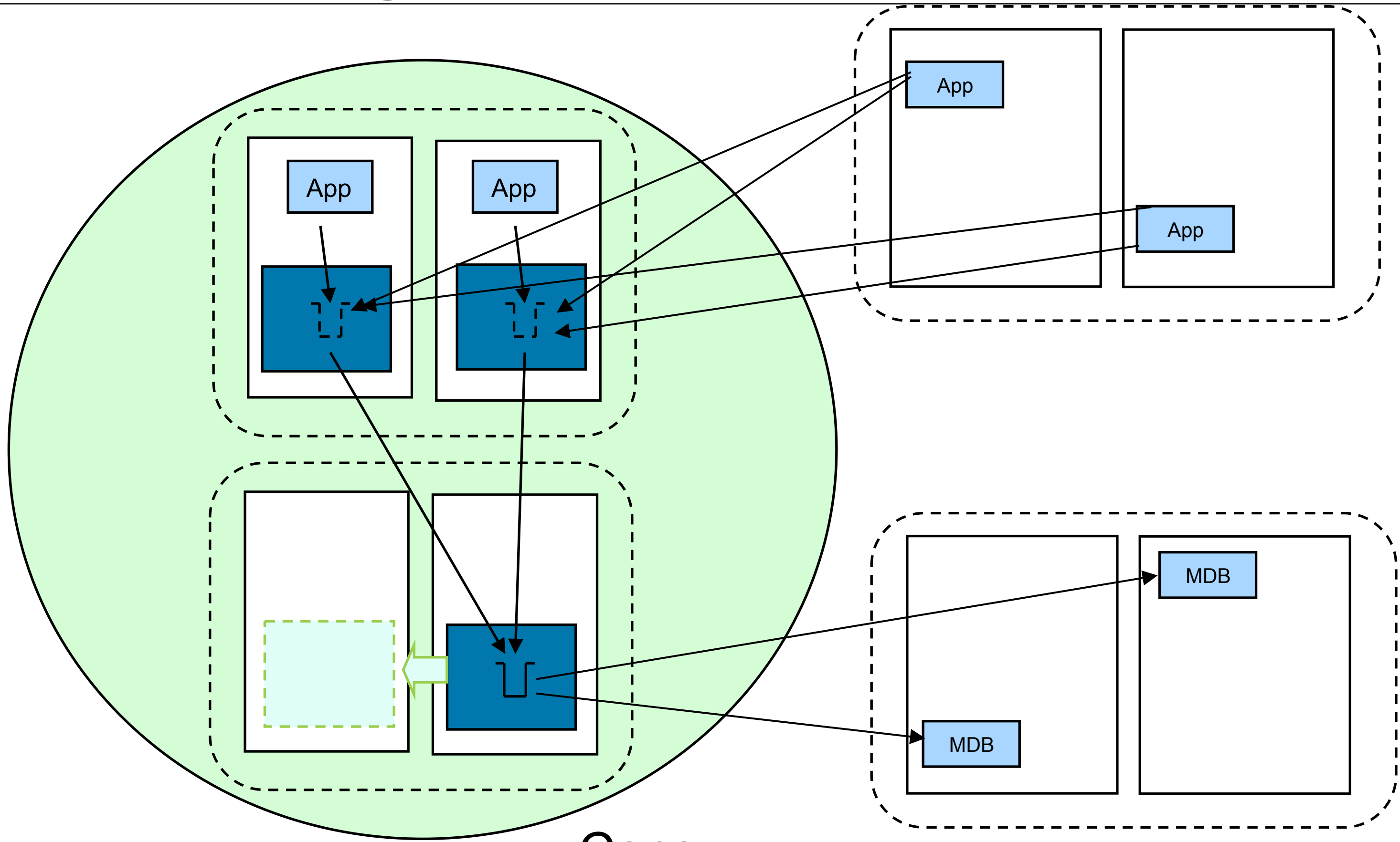
•Cons:
- Scalability of messaging
- Under load messaging engine could be in stress
- All destinations localized on a single messaging engine

HA Option 2: Multi Cluster



- Pros:
  - Scalability of applications
  - High availability of Applications
  - High availability of Messaging
  - Destinations can be localized across messaging engines
- Cons:
  - Scalability of messaging
  - Segregation of messages

HA Option 3: Maximizing HA



- Pros:
  - Scalability of applications
  - High availability of Applications
  - High availability of Messaging
  - Store and Forward
- Cons:
  - Scalability of messaging
  - Destinations localized on a single messaging engine

Managing the Messaging environment

- Core Groups and Policies
- Preferred Servers
- Fail back

Core Group and Policies

- A core group is a collection of resources managed by the High Availability manager

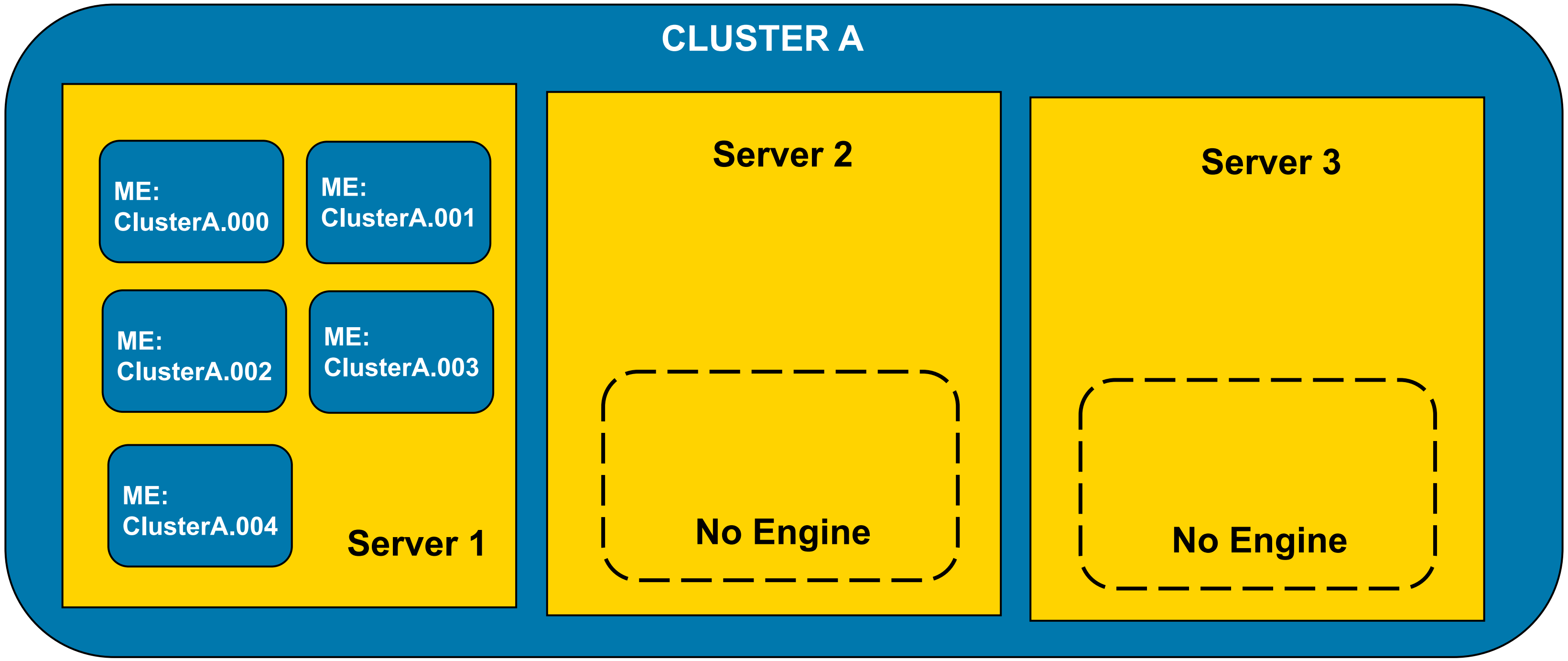
- Each resource is associated with a policy that defines how the HA Manager should control that resource

- ME's use the "Default SIBus Policy"
  - This is a "One of N policy" with no preferred servers

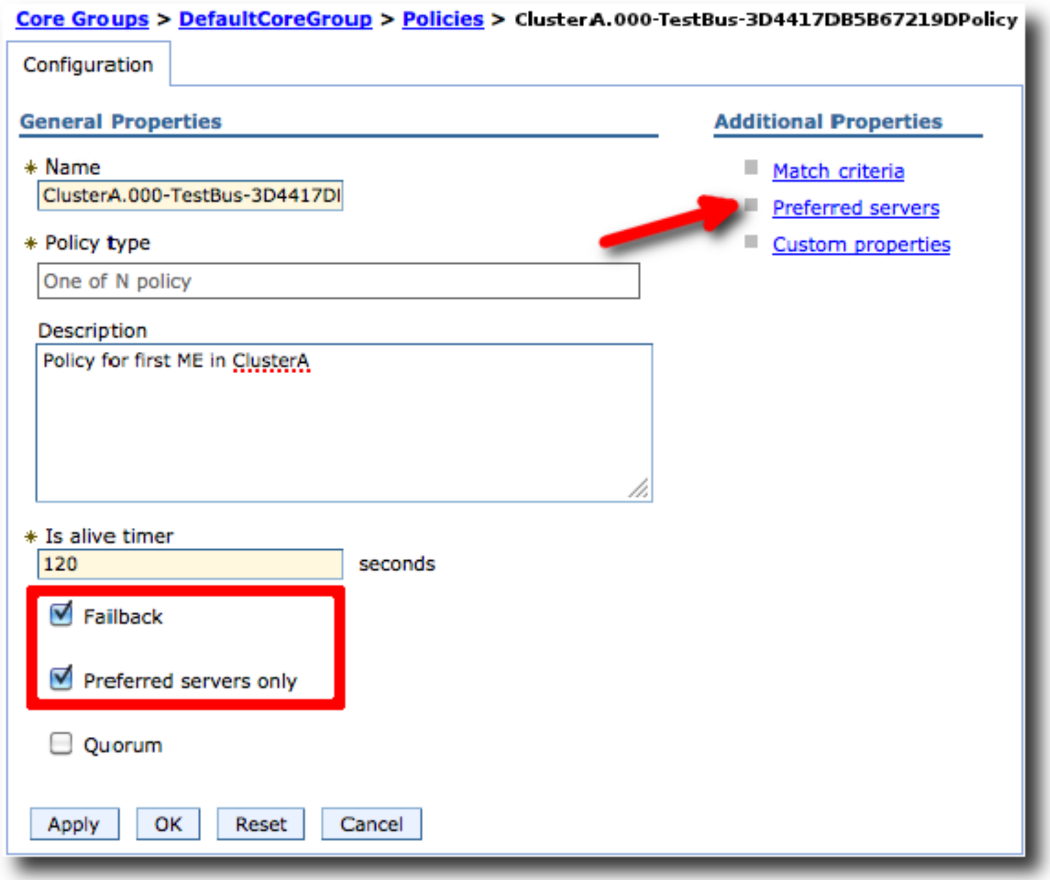
- Default ME behavior
  - The ME joins the HA core group
  - The HA Manager tells the ME to start in one server
  - ME state is visible in the SystemOut.log file (CWSID0016I)

Default SIBus Policy

- In a cluster no server is better (or worse) than any other
- Server load does not figure in the choice at all
- If you have multiple MEs in a cluster, HA Manager may attempt to start them all in the first server itself



Custom Policies



- Preferred servers
  - HA will use the highest available preference
  - Engines can be restricted to preferred servers
- Failback
  - Engines will restart on a preferred server when one becomes available
  - State StoppingMember indicates failback

•Best Practice: Configure a custom One of N policy with different preferred servers for each ME and failback

Challenges with Multi ME environment

- Remote Queue Point
- Store and Forward
- Remote GET

Challenges with Multi-ME

Remote Queue Point

• When bus contains multiple bus members, applications can connect to any available ME in the bus (location transparency)

• When an application accesses a destination which is not localized on the ME to which it is connected, the ME determines the location of the queue point from the Work Load Manager (WLM)

•Once the location of the destination queue point is known, the ME creates a remote queue point

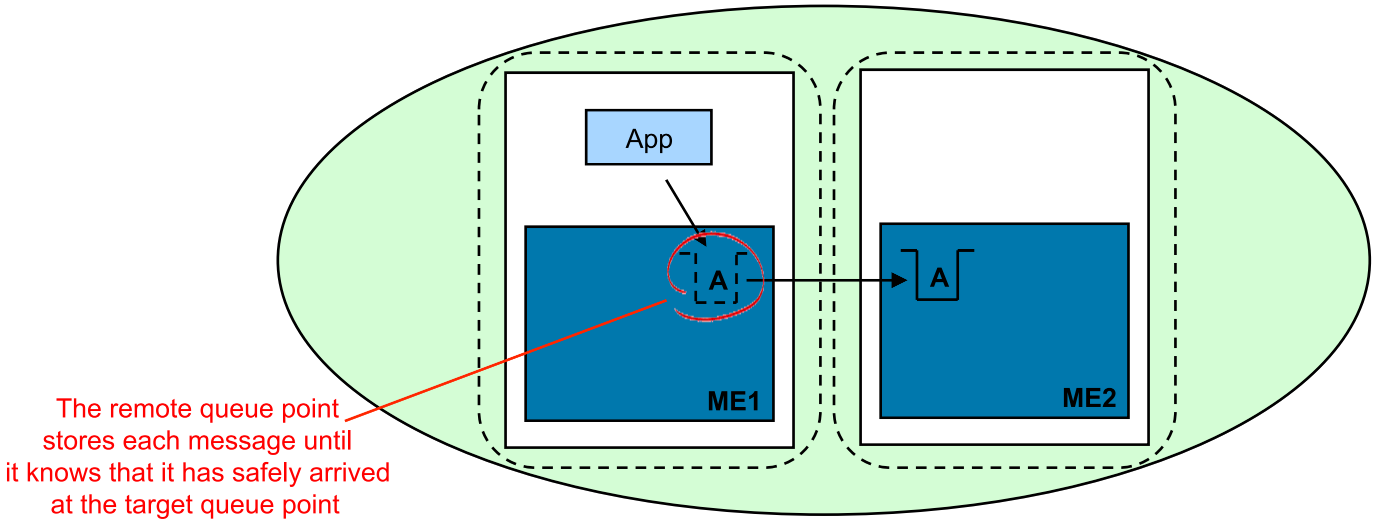
•The ME uses this remote queue point to send and receive messages to the queue point for the application

Challenges with Multi-ME (Store & Forward)

- The producing application, connected to ME 1, produces messages to a queue localized on ME 2

- ME 1 creates a remote queue point to forward the messages

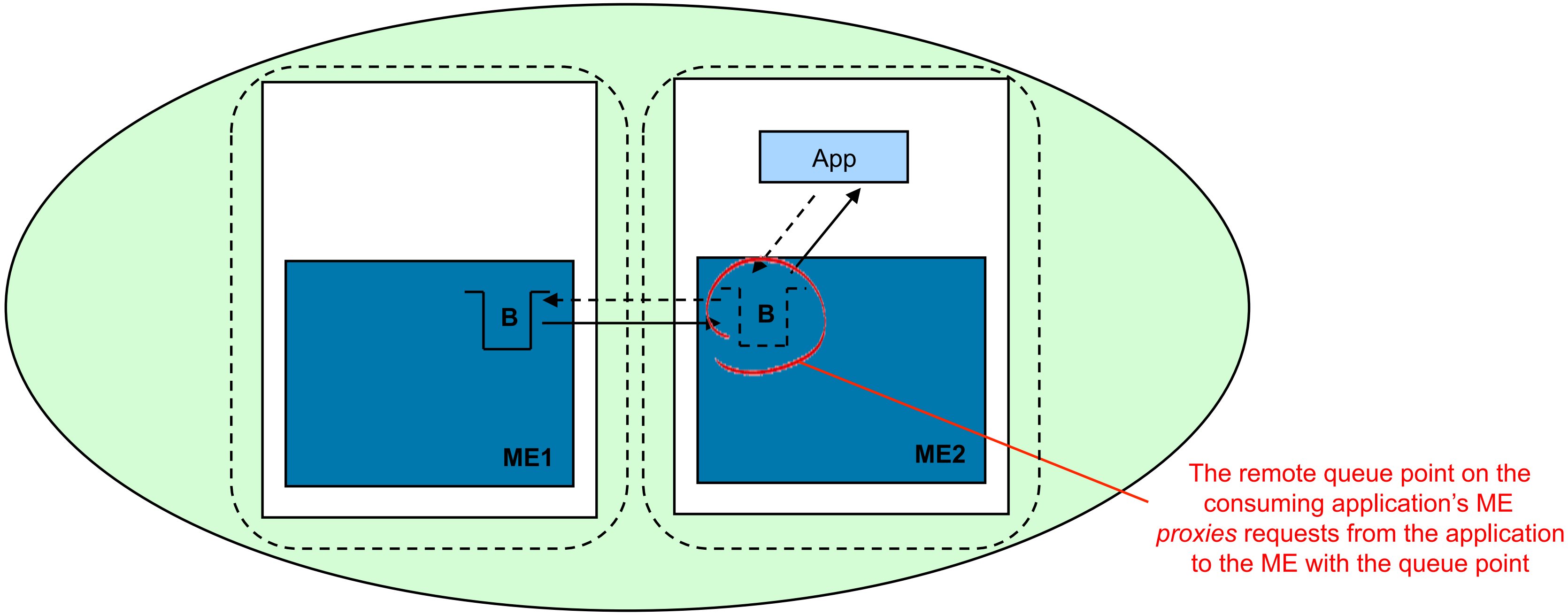
- Store and forward can be beneficial as it allows message production to continue while a Queue Point's ME is unavailable



- However, this results in additional potential places where messages may become blocked for various reasons, multiple persistence, performance

Challenges with Multi-ME (Remote GET)

- The consuming application, connected to ME 2, consumes messages from a queue localized on ME 1
- Unlike store and forward, remote get requires the Queue Point's ME to be available for messages to be consumed



Challenges with Multi-ME

Pros and Cons

- Store and Forward
  - 👍 Transparency of the queue point location
  - 👍 Does not require the localizing ME to be available because message production is asynchronous
  - 👍 Takes advantage of WLM for partitioned queues
  - 👎 Additional latency in synchronous request/reply
- Remote GET
  - 👍 Transparency of the queue point location
  - 👎 Consumer latency due to multiple message hops
  - 👎 Multiple points of failure, complex diagnosis
  - 👎 Requires the localizing ME to be available

Best Practice: Configure your system to avoid remote get

Bus topology having several bus members

- Application connectivity
- Targeting Connections

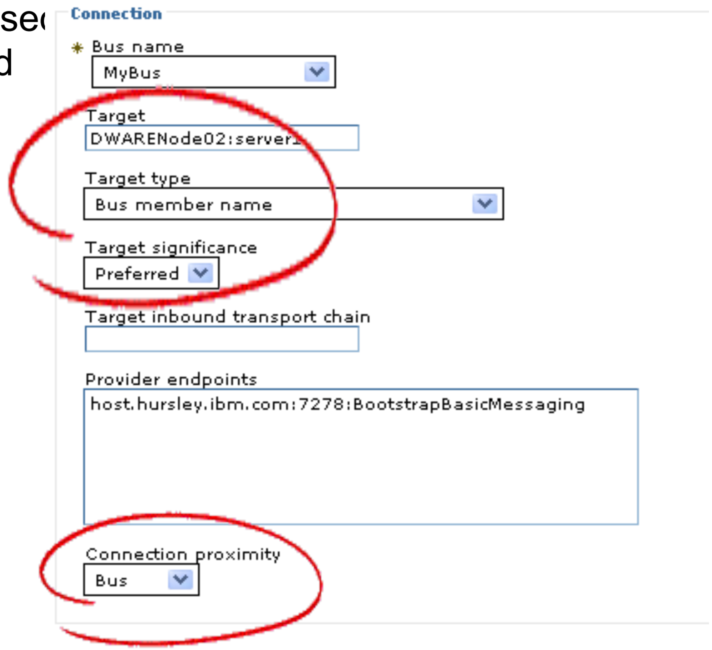
Bus having multiple bus members

Connecting to the Bus

- When JMS applications connect to a bus the minimum information required is the name of the ***bus***
- Plan your configuration carefully to achieve maximum reliability while minimizing complexity
- By default, applications will connect to any available ME and will prefer an ME in the same server, if one is available
- Quite often this will lead to problems in terms of
  - Store and Forward
  - Remote GET

Bus having multiple bus members

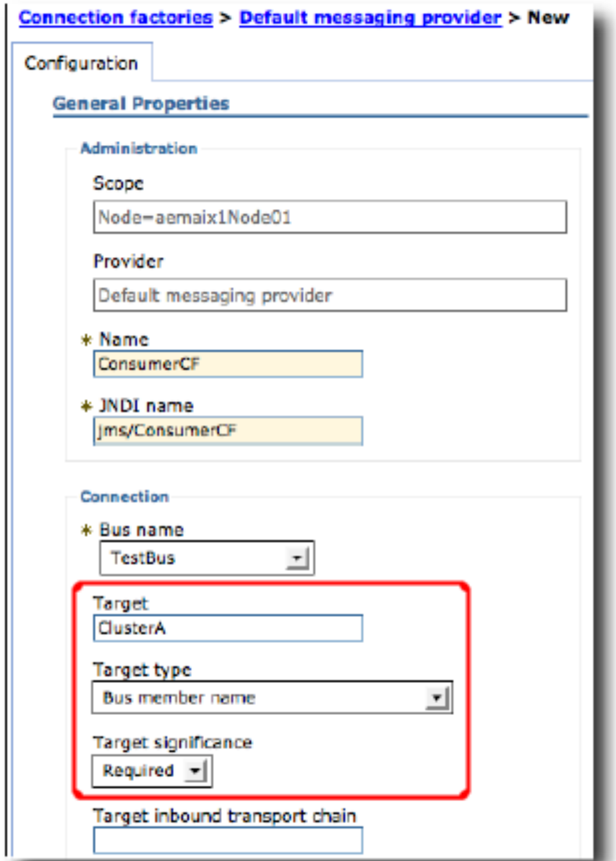
- By ***targeting*** the connection, varying levels of control over the actual connection made into the bus can be defined.
  - **Target type**
    - The type of entity that is to be targeted
    - ***Bus member*** - any available ME in that bus member.
    - ***Messaging engine*** - a specific ME.
    - ***Custom messaging engine group*** - a manually configured set of MEs.
  - **Target significance**
    - ***Required*** - The connection will ***only*** be made to the specified target.
    - ***Preferred*** - If the specified target is available it will be used otherwise any other available ME in the bus will be used
  - **Target**
    - The name of the above target

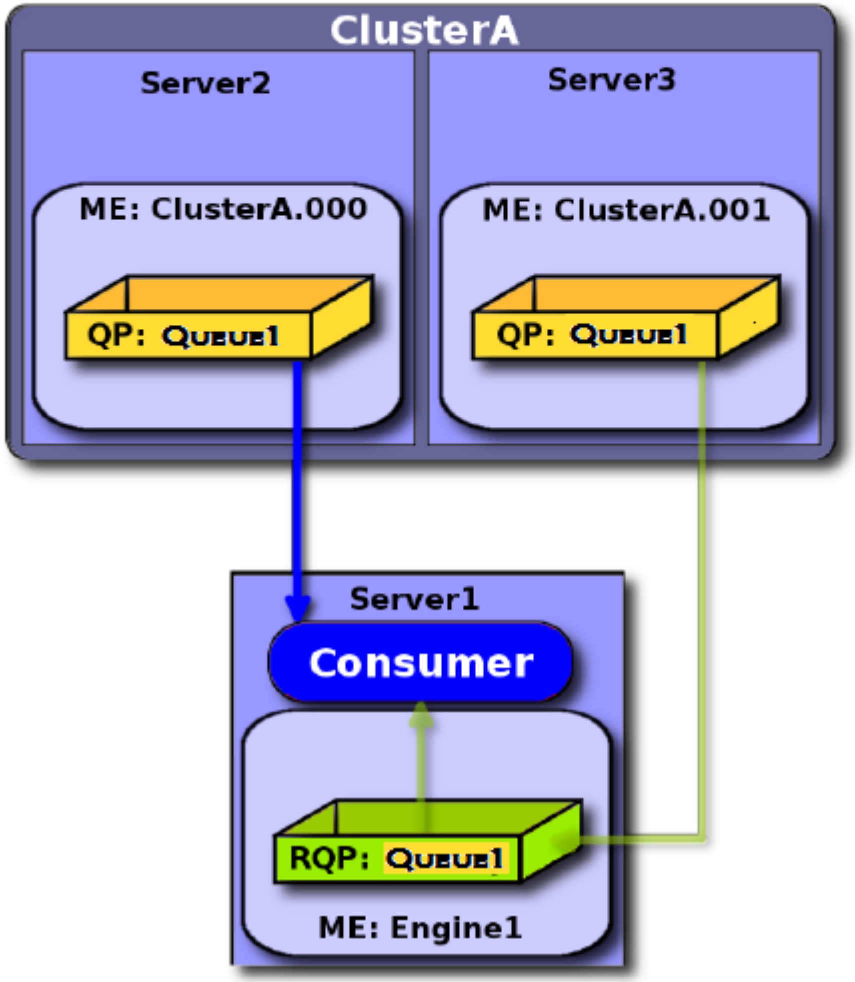


- ***Connection proximity***
  - This provides an additional level of control, based on the physical location relative to the application or bootstrap server
    - E.g. within the same server or cluster, or on the same host

Bus having multiple bus members

Connection Example





✔ Best Practice: Activation specifications and consumers should target MEs localizing the destination holding their messages

Scaling Producers and Consumers

- Scaling Producers
- Balancing Message Workload
- Scaling Consumers
- Message Visibility

Scaling Producers

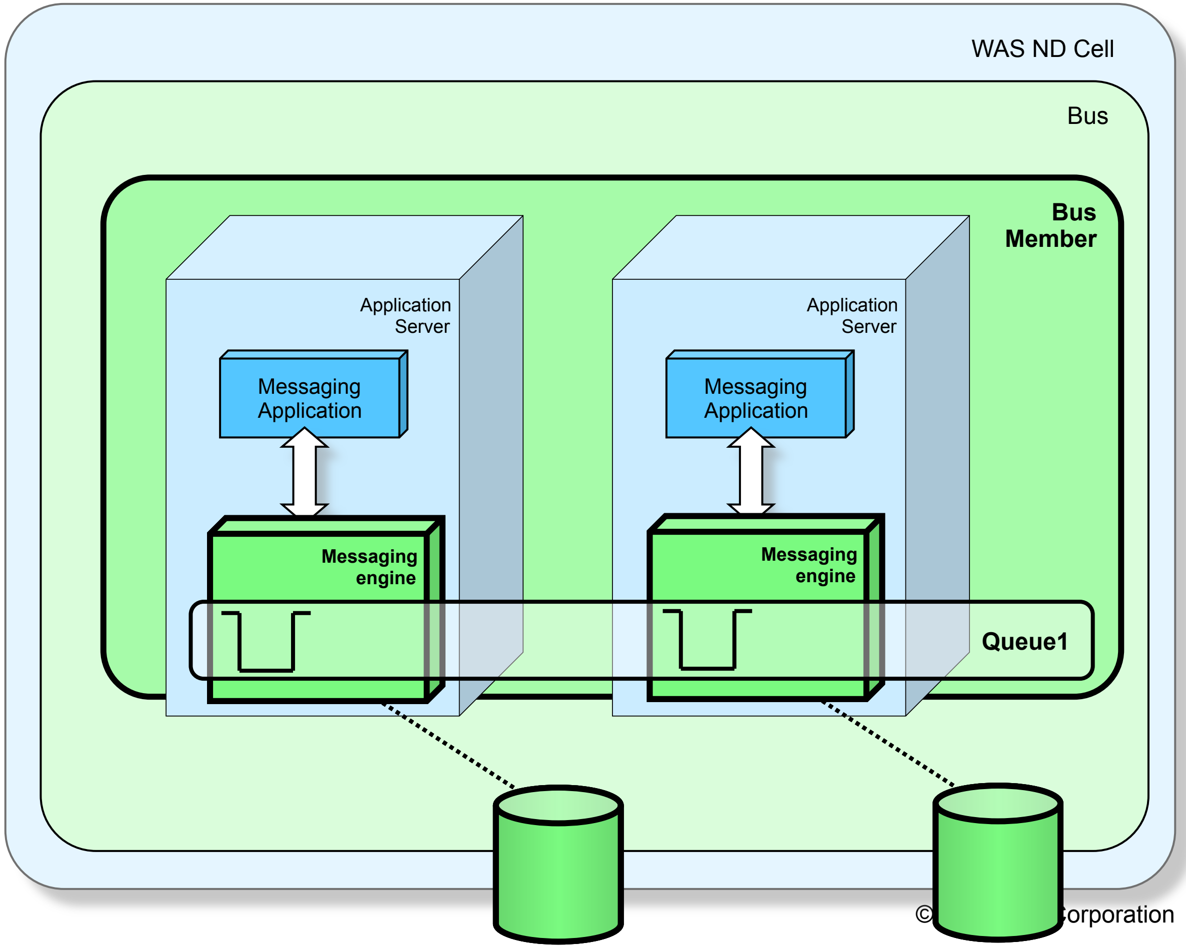
- Give latitude when connecting
  - Producers can connect to any ME, not just the one which localizes a particular destination
  - Do not restrict the choice too much with target properties

- Add more MEs for high connection loads
  - Set up a cluster of MEs to handle connections
  - Use target properties to connect producers to this cluster
  - Connections will be balanced across available ME

- Best Practice: Add MEs to handle very large numbers of connections and rely on remote put to forward messages to their destination

Scaling Producers : Partitioned Queues

- Multiple producers may exceed the capacity of a destination
- Each ME can run **concurrently** and they each have their **own set of persistent data**
- As you add MEs to a cluster, you add queue points



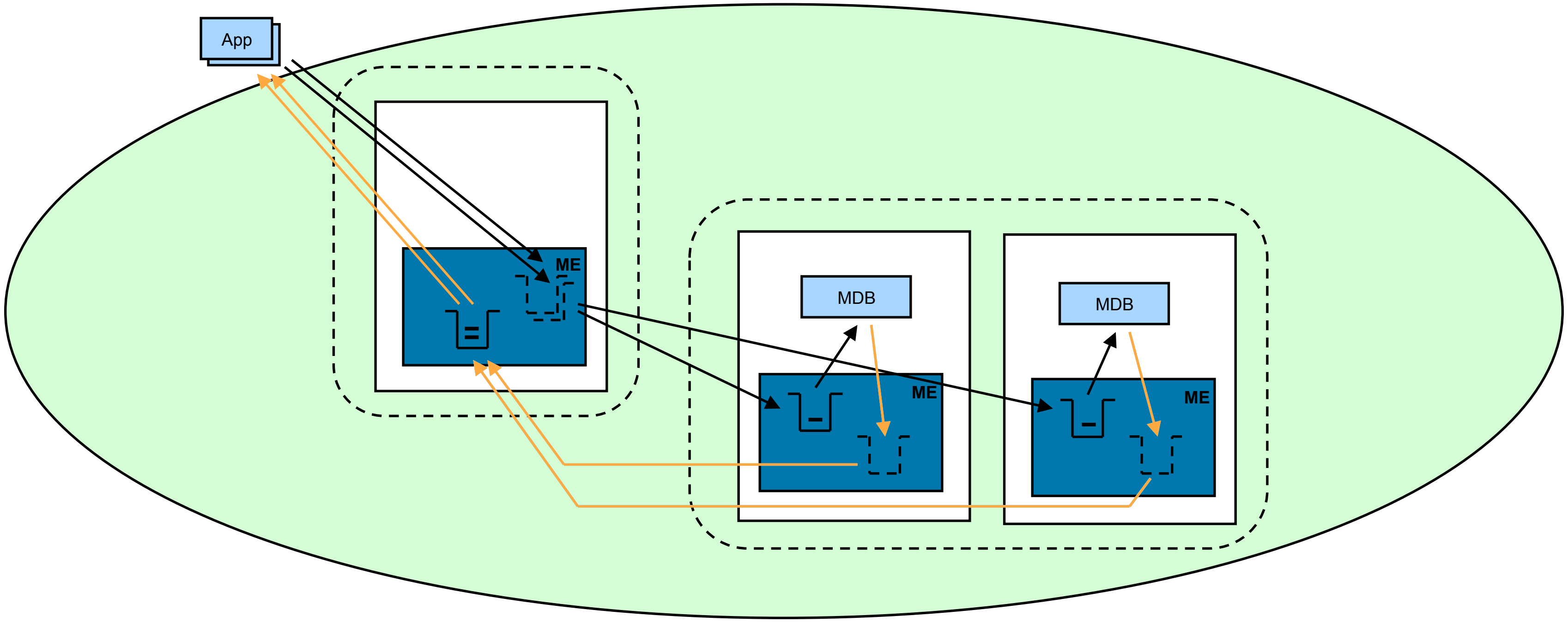
- Queues assigned to such a bus member will be ***partitioned*** across each ME
- Each partition has the capacity of a normal queue point
- The queue capacity scales linearly with the cluster

Balancing Message Workload

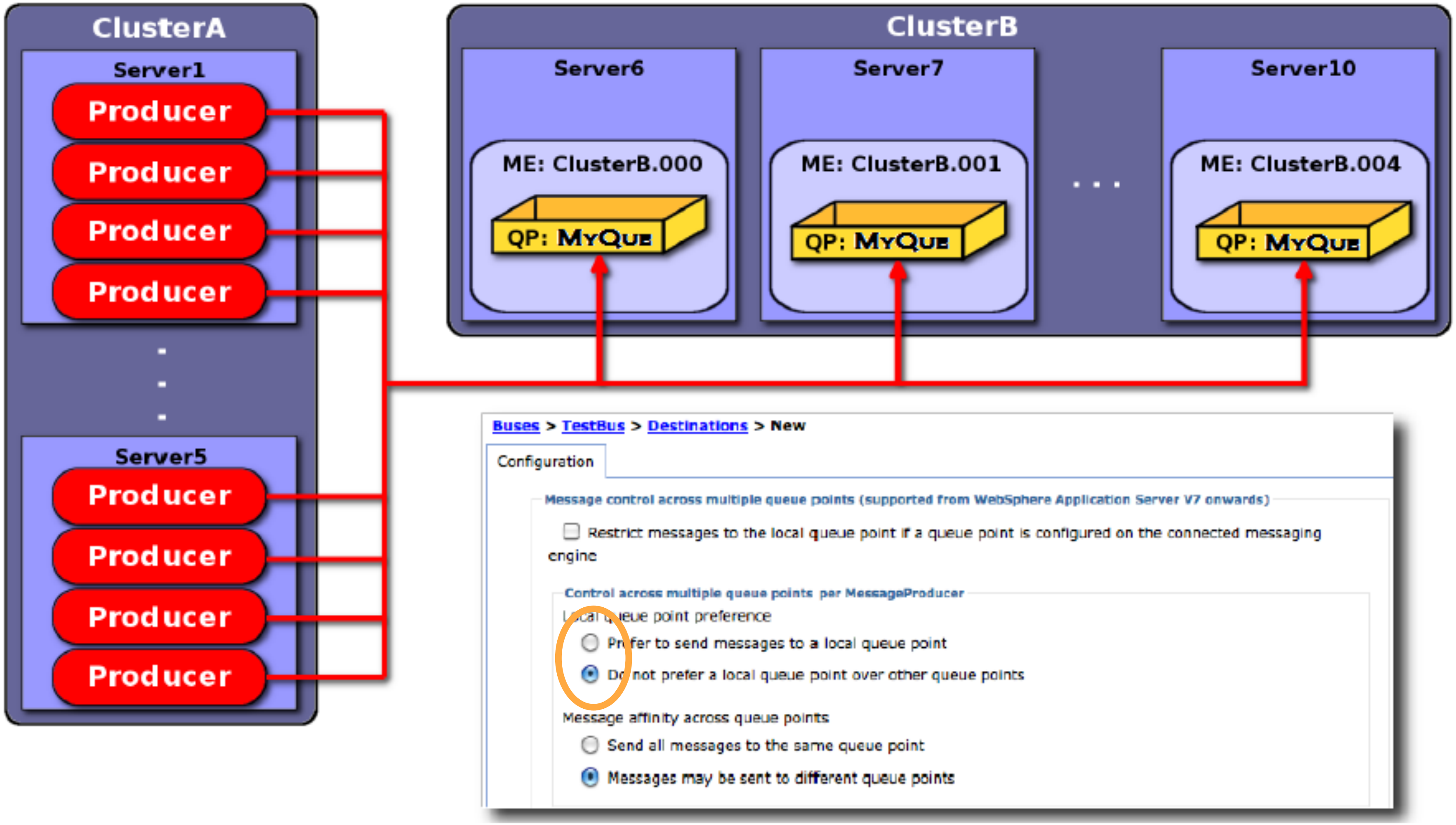
• Messages will be distributed to all available partitions if the producing application relies on remote put

• By default, If the producing application is connected directly to an ME in the cluster, all its messages will go to the local partition

• A JMS Queue option can be enabled to distribute messages to all available partitions even when there is a local partition



Producer Example

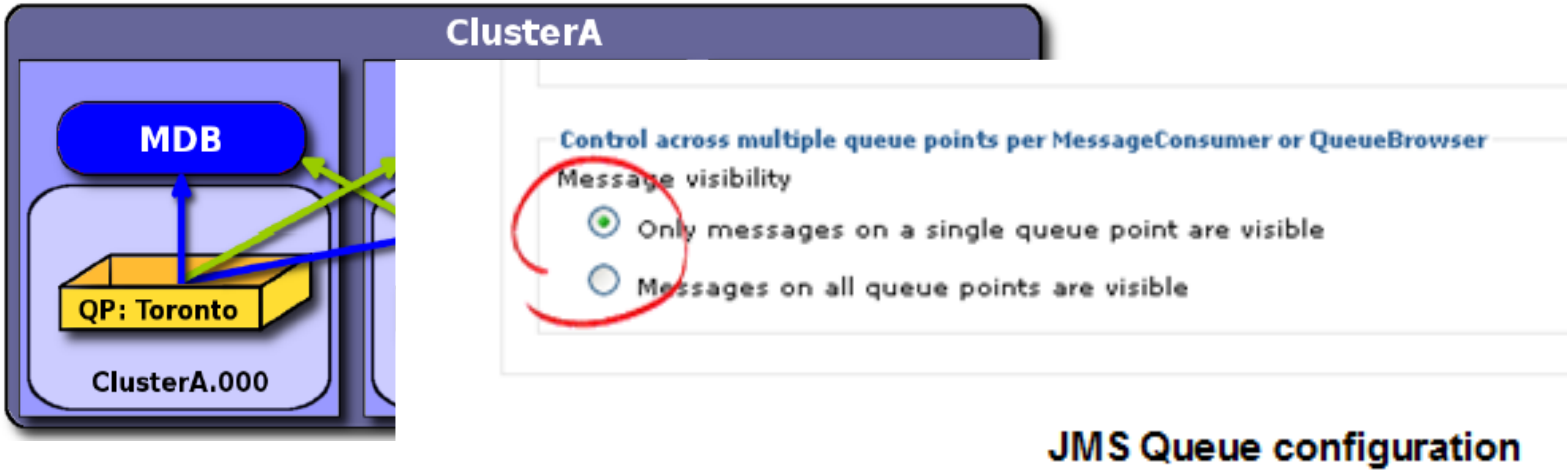


Scaling Consumers

- Deploy consumer applications and MDBs to a cluster
  - Each member of the cluster will consume messages
  - Throughput will scale will the size of the cluster

•By default a JMS consumer is only able to consume messages stored on the queue point that it is connected to

•This can be changed by enabling the "**Message visibility**" option



Multi-Bus Topologies

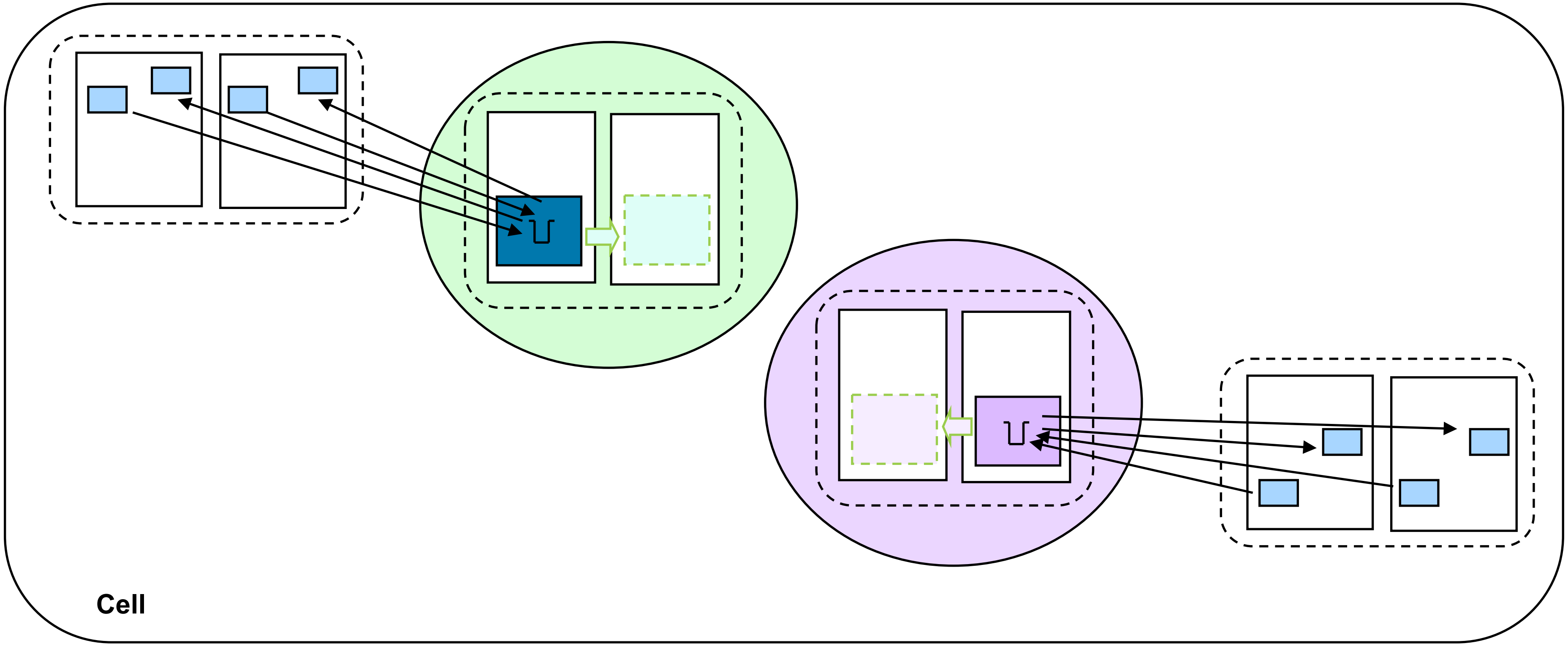
Multi Bus topologies

There are 2 main reasons for having a multi-bus SIB topology:

- Because, for simplicity or isolation, you want to scope unrelated messaging applications within a cell
- Because related messaging applications span multiple WAS cells and a bus is restricted to a single cell.

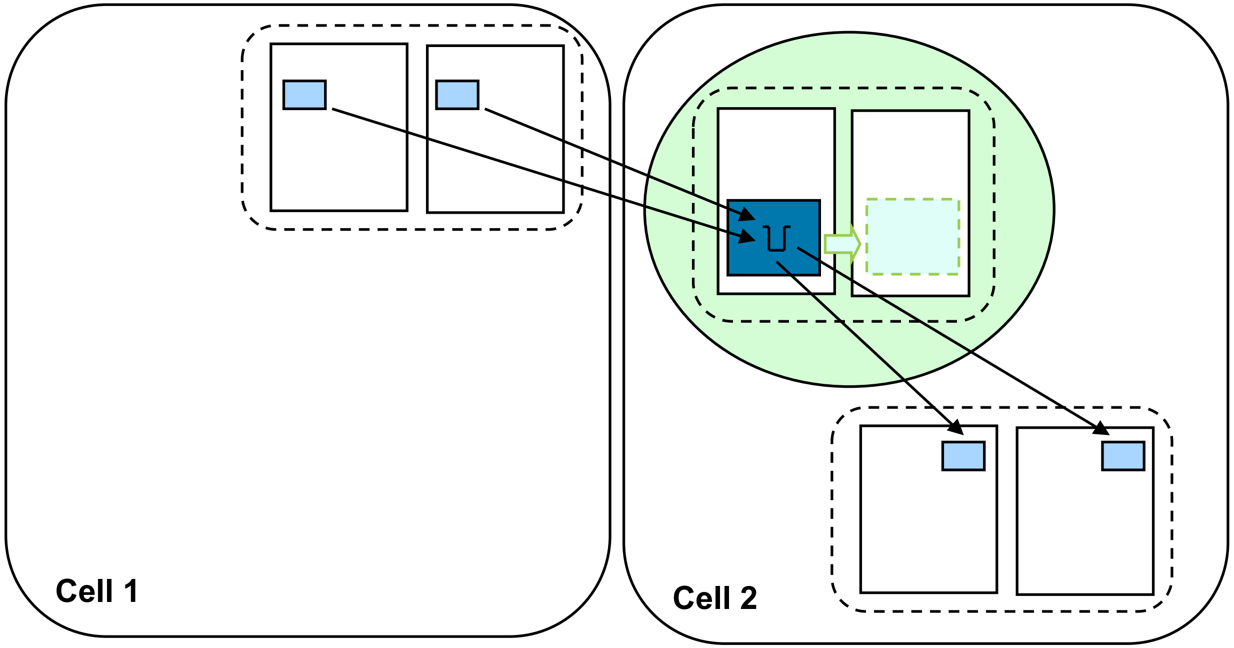
Multi Bus topologies (1)

- If you're using multiple buses to scope unrelated messaging applications, no inter-bus linkage will be required.
- This doesn't change any configuration or runtime aspects of the multiple buses, they exist in simply ignorance of each other, despite potentially running in the same cell.



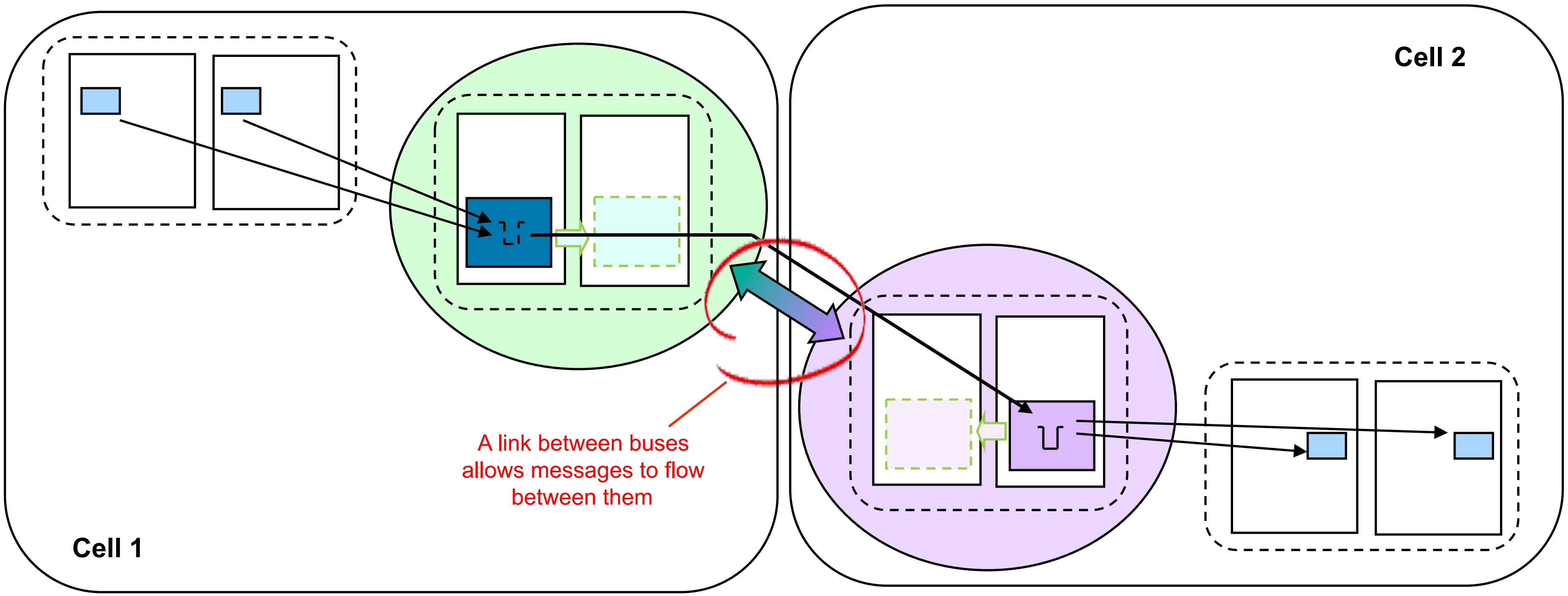
Multi Bus topologies (2)

- If messaging spans cells, the first option to consider is to have a single bus in one of the cells and for the messaging application to connect across-cells to access the messages.
  - Originally MDB's required use of a Core Group Bridge between cells to allow this.
  - V7 makes it possible for an MDB's Activation Specification to specify a provider endpoint in a different cell from the MDB application



Multi Bus topologies (3)

- If a single bus is not appropriate, multiple linked buses will be required.
  - This introduces more complexity, both for the initial configuration and the manageability of the resultant system.
  - Messages are *stored and forwarded* between buses
  - Messages cannot be consumed remotely from destinations in a different linked bus



Addressing ME resiliency

- Long running DB locks
- Split-Brain scenarios
- JVM Hang
- Gracefully stop ME from DB failures
- Configure ME to recover data from a orphaned persistence store

Restrict Long running locks

•ME can be configured to use any Database as its persistence layer

•ME maintains a set of tables to store both runtime information and also the message data

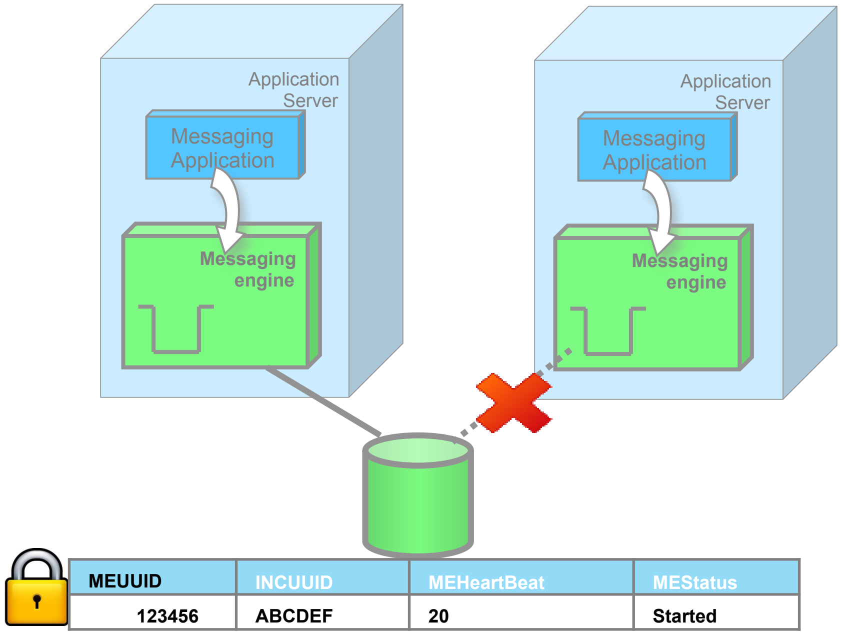
•At any given point of time, only one ME can access the persistence store (to maintain data integrity)

How does it work ?

•Active ME maintains a long running shared lock on the ***SIBOWNER*** table

•The lock on the SIBOWNER table is acquired during ME start-up and is held by the ME all throughout its life-cycle

•This poses several challenges to the DBA

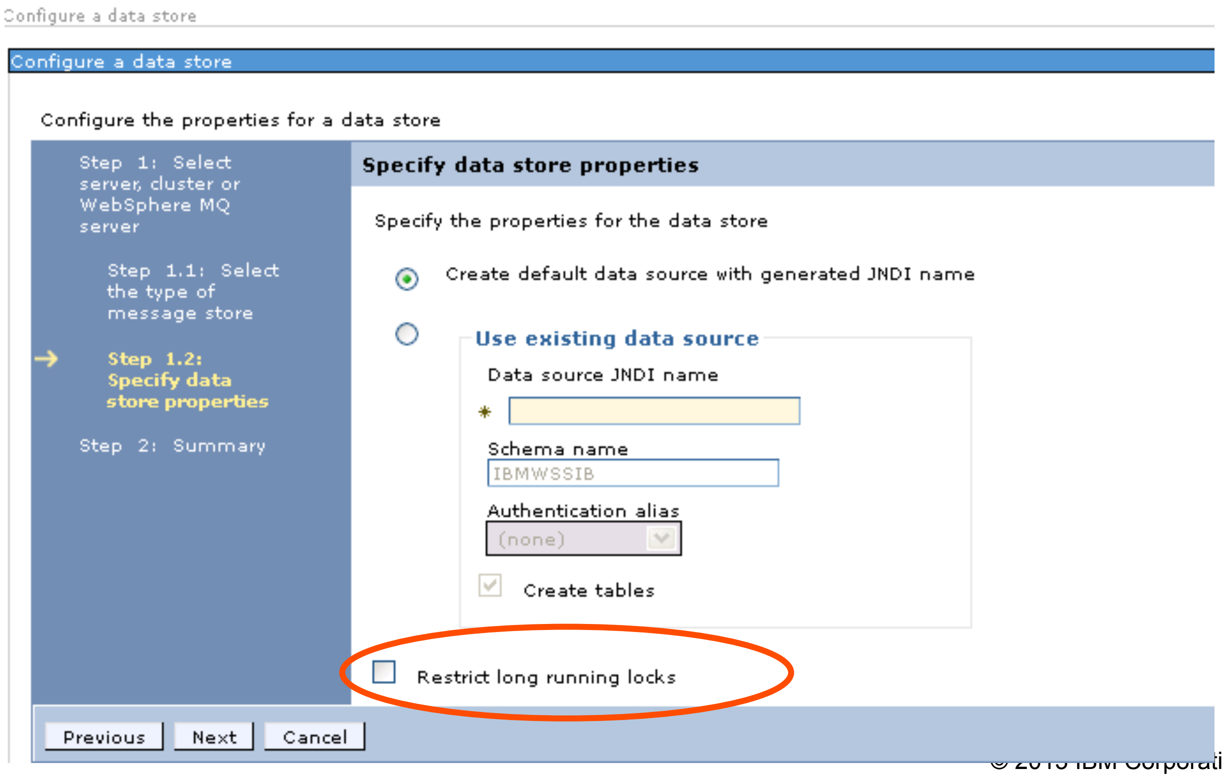


Restrict Long running locks

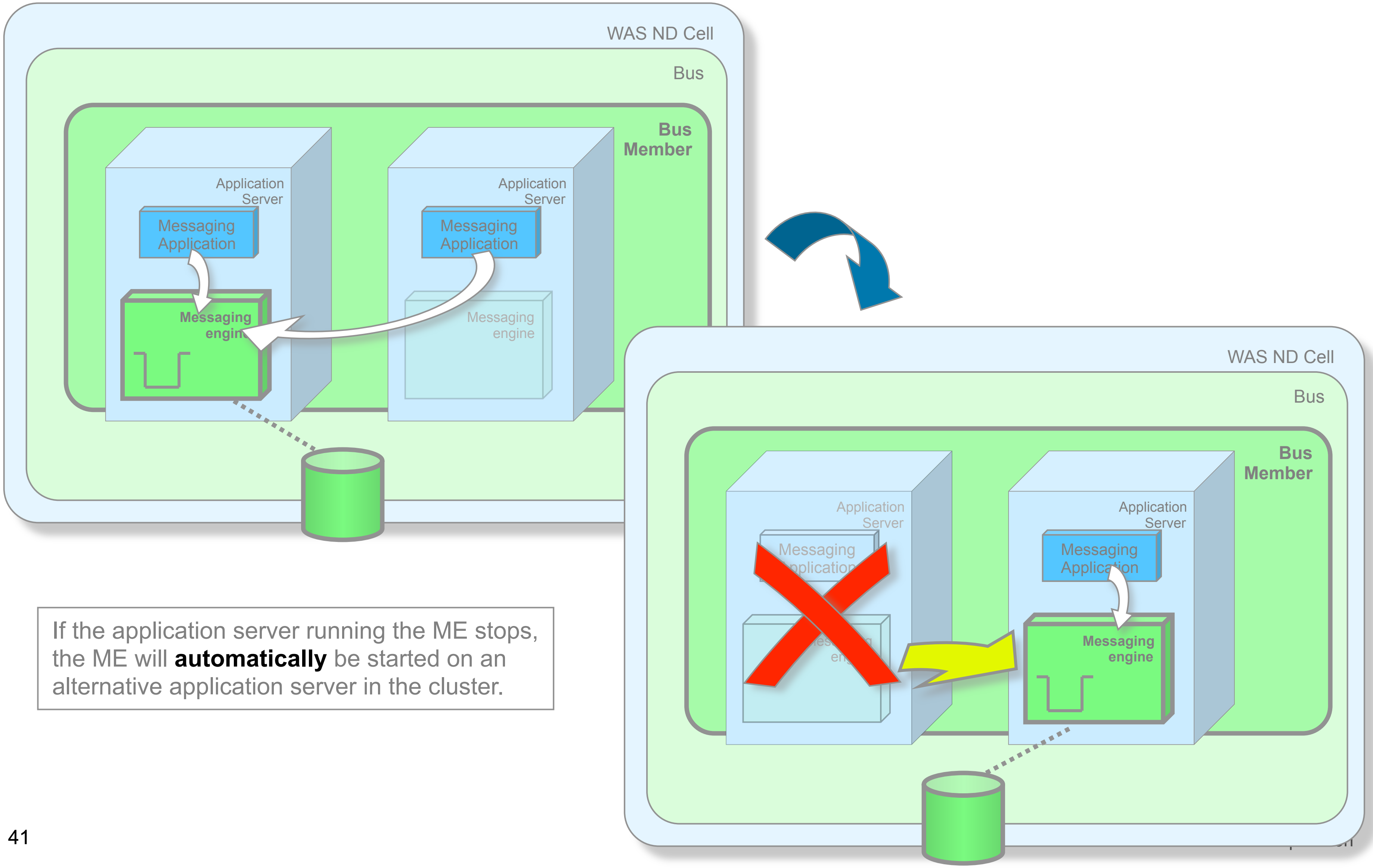
•A new algorithm that ensures ME no longer holds long running locks on the SIBOWNER table

•Active ME now holds only *short locks* on the SIBOWNER table while revalidating its ownership at regular intervals

•The right to own the SIB tables is always performed during ME start-up



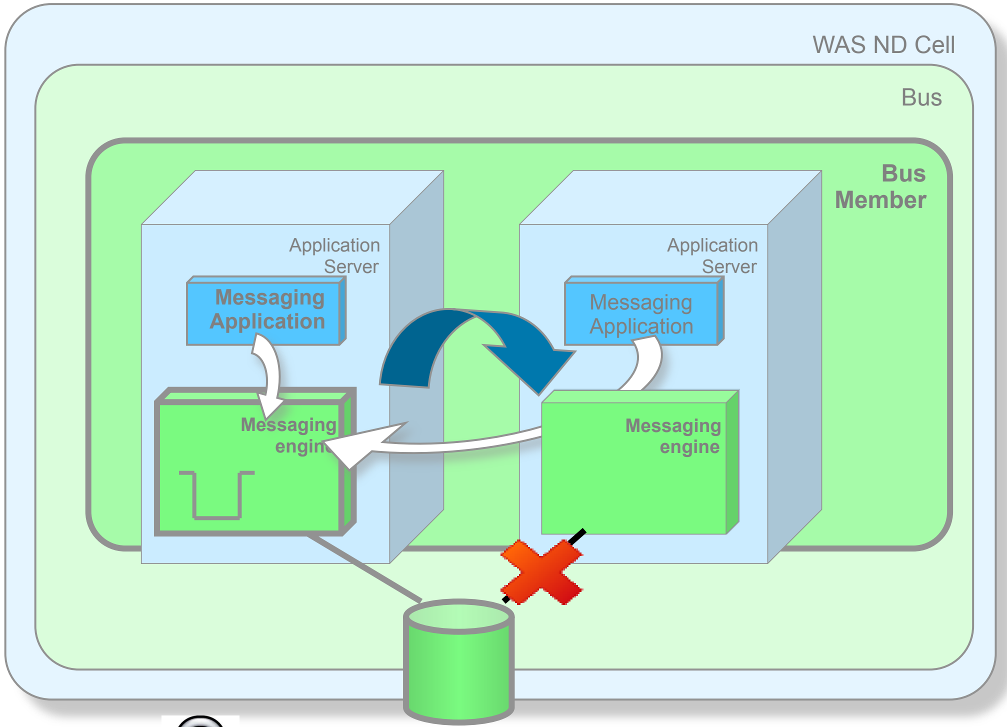
Detect JVM Hang and failover



If the application server running the ME stops, the ME will **automatically** be started on an alternative application server in the cluster.

Detect JVM Hang and failover

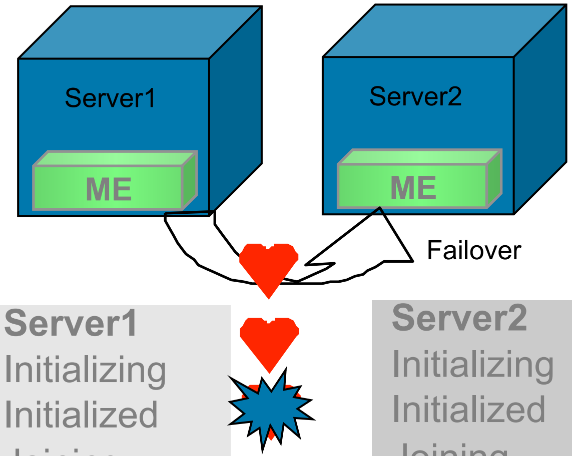
What happens when the active server does not crash but hangs ?



MEUUID	INCUUID	MELastUpdate	MEStatus
123456	ABCDEF	29/04/12 17:45:30	Started

ME cannot get ownership on SIBOWNER table since the active ME on server1 still holds the active lock on SIBOWNER table

What's happening under the covers?



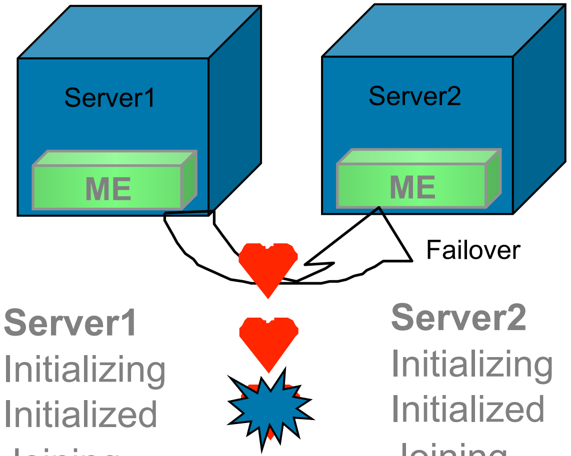
Server1	Server2
Initializing	Initializing
Initialized	Initialized
Joining	Joining
Joined	Joined
Starting	Starting
Acquire DB ownership	Trying to acquire DB ownership
Started	Cannot get Lock on SIBOWNER table. **ME Disabled**

Detect JVM Hang and failover

What happens when the active server does not crash but hangs ?

SIBOWNER Table

MEUUID	INCUUID	MELastUpdate	MEStatus
123456	ABCDEF	29/04/12 17:45:30	Started



Server1
Initializing
Initialized
Joining
Joined

Starting

Acquire DB ownership

Started

ME refreshes MELUTS at regular interval

Server2
Initializing
Initialized
Joining
Joined

Starting

Tries to acquire DB ownership/Lock

ME gets the lock on SIBOWNER table, but notices another ME owning the Database

Check ME1 Heartbeat time interval

Detect JVM Hang and failover

This leads to 2 different scenarios:

What happens if the MELastUpdate value is updated regularly?

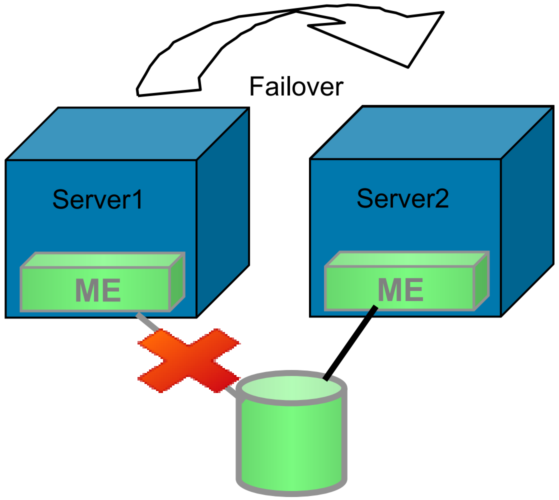
- This situation can occur when there is a **network failure** between the 2 servers
- Typical **Split-brain** scenario
- ME on server2 will go back into **standby mode** (joined state)

What happens if the MELastUpdate value is NOT updated regularly?

- This situation can happen when ME on server1 is ***hung*** or ***crashed*** or becomes ***unresponsive***
- This situation can occur when ME on server1 *loses network communication with the Database*
- ME on server2 takes ownership of the SIB tables and becomes ***Active***

Grace full stopping ME when connectivity to DB fails

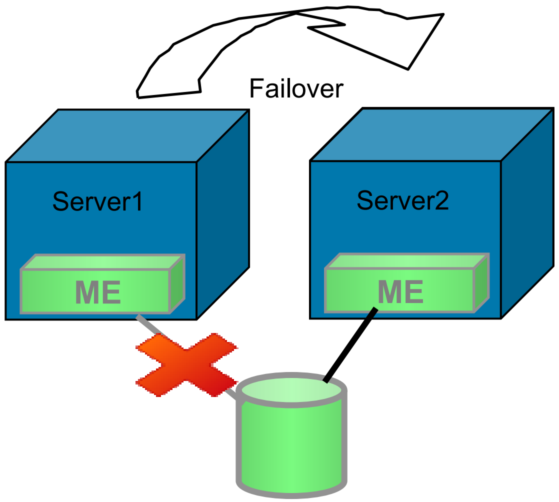
What happens when the active ME loses connectivity to database?



- When the active ME loses connectivity to DB, ME raises a local error and notifies the HAManager it cannot continue
- This causes the sever to panic and eventually the entire JVM is killed
- HAManager triggers the standby ME to start and take ownership of the DB
- Eventually all the applications running on Server1 are also terminated causing operational issues within the environment

Grace full stopping ME when connectivity to DB fails

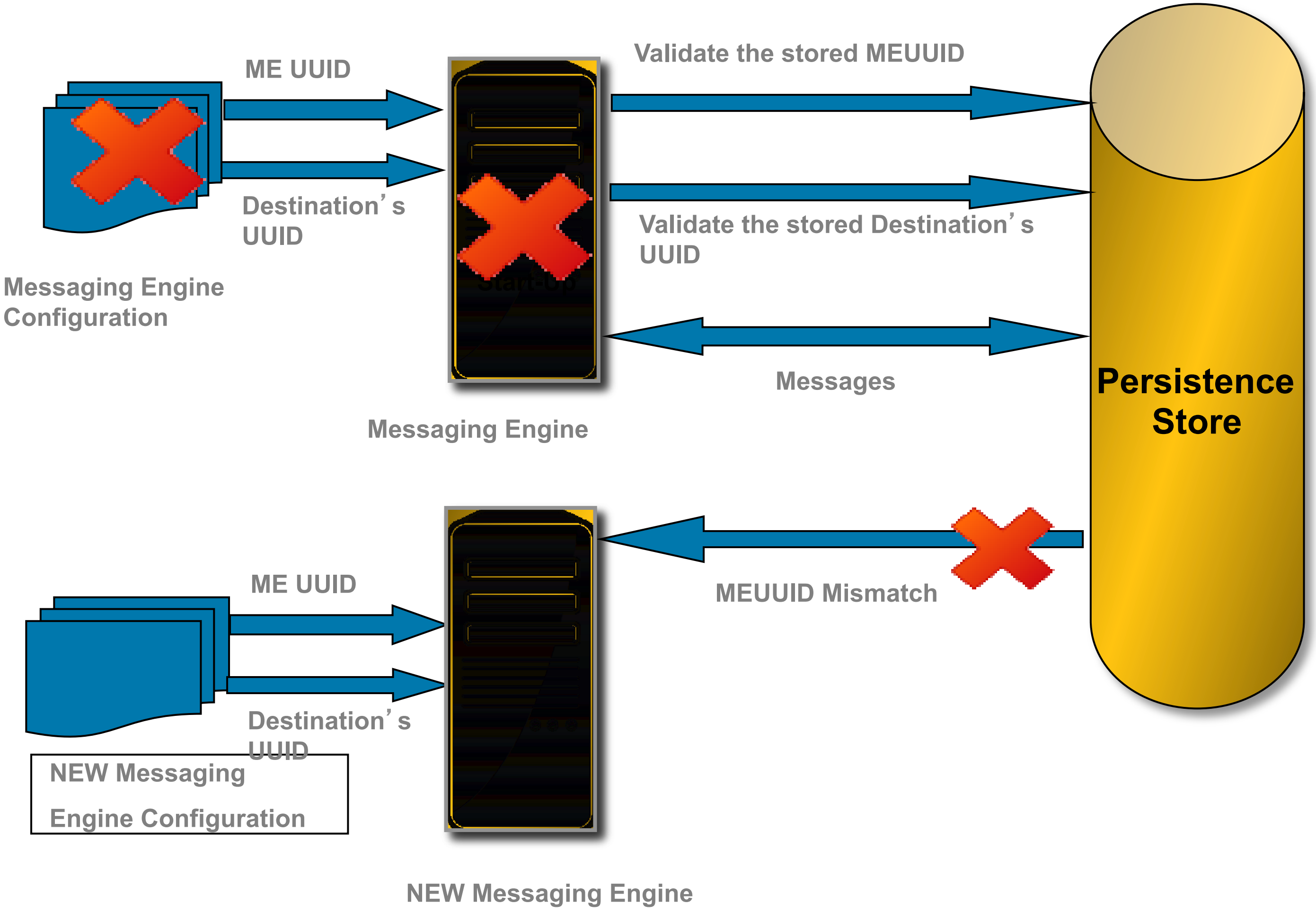
What happens when the active ME loses connectivity to database?



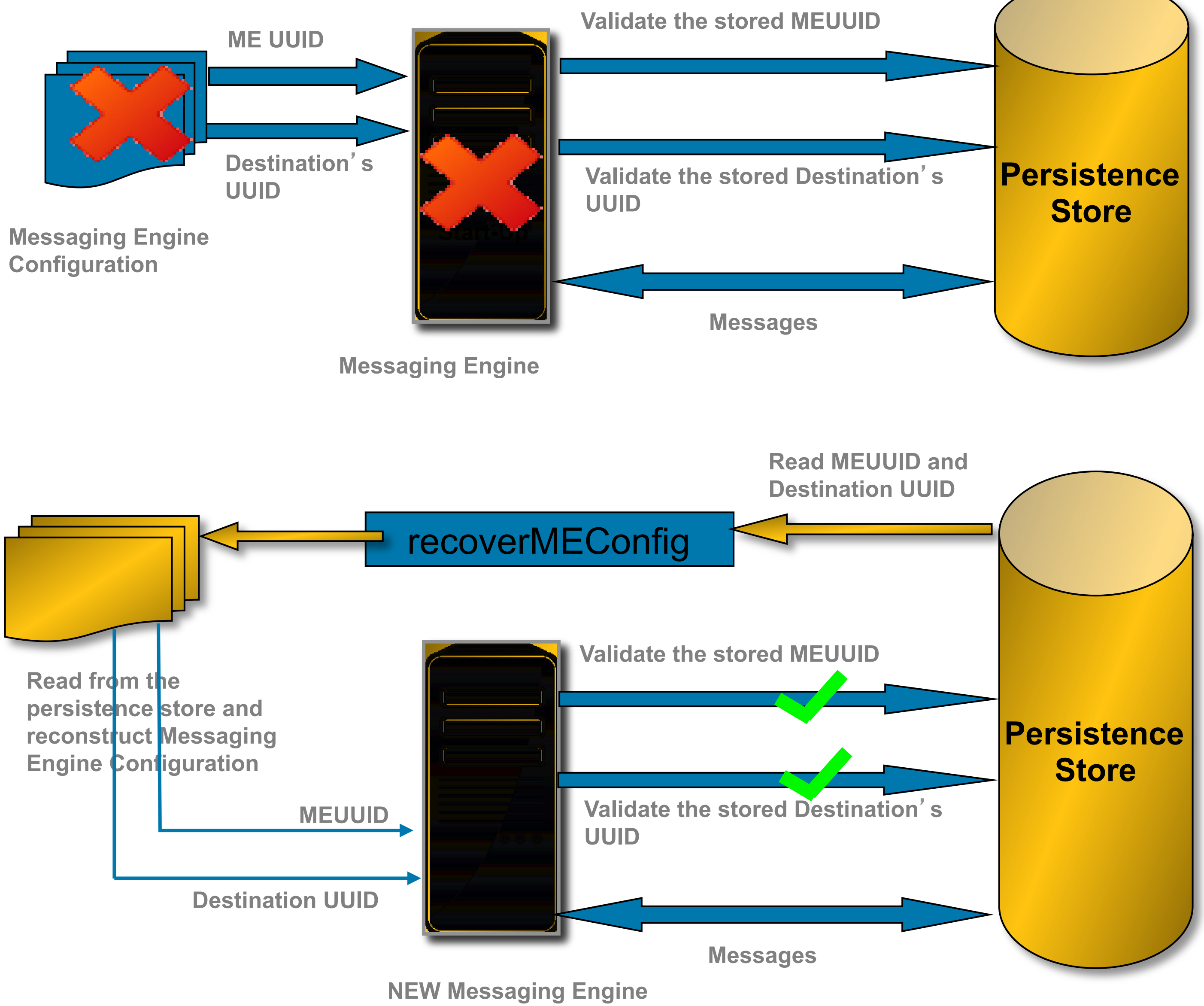
- When the active ME loses connectivity to DB, ME raises a local error and notifies the HAManager it cannot continue
- Instead of panicking the entire JVM, ME is made to gracefully stop
- HAManager triggers the standby ME to start and take ownership of the DB

- This ensures all the applications on Server1 run without any issues and the JVM is not killed ☺

ME to recover data from an orphaned persistence store



ME to recover data from an orphaned persistence store



ME to recover data from an orphaned persistence store

NEW

Advantages

- New admin command to recover both the MEUUID and destination UUID from the persistence store
- The configuration files are recreated based on the UUID retrieved from the persistence store
- Ability for configuring new ME to recover messages from an orphaned persistence store

We love your Feedback!

Don’t forget to submit your session and speaker feedback!

•Your feedback is very important to us – we use it to improve next year’s conference

Legal Disclaimer

- © IBM Corporation 2013. All Rights Reserved.
- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- If the text contains performance statistics or references to benchmarks, insert the following language; otherwise delete:
  Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- If the text includes any customer examples, please confirm we have prior written approval from such customer and insert the following language; otherwise delete:
  All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.
- Please review text for proper trademark attribution of IBM products. At first use, each product name must be the full name and include appropriate trademark symbols (e.g., IBM Lotus® Sametime® Unyte™). Subsequent references can drop "IBM" but should include the proper branding (e.g., Lotus Sametime Gateway, or WebSphere Application Server). Please refer to http://www.ibm.com/legal/copytrade.shtml for guidance on which trademarks require the ® or ™ symbol. Do not use abbreviations for IBM product names in your presentation. All product names must be used as adjectives rather than nouns. Please list all of the trademarks that you use in your presentation as follows; delete any not included in your presentation. IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Quickr, Sametime, WebSphere, UC2, PartnerWorld and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.
- If you reference Adobe® in the text, please mark the first use and include the following; otherwise delete:
  Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- If you reference Java™ in the text, please mark the first use and include the following; otherwise delete:
  Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- If you reference Microsoft® and/or Windows® in the text, please mark the first use and include the following, as applicable; otherwise delete:
  Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- If you reference Intel® and/or any of the following Intel products in the text, please mark the first use and include those that you use as follows; otherwise delete:
  Intel, Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- If you reference UNIX® in the text, please mark the first use and include the following; otherwise delete:
  UNIX is a registered trademark of The Open Group in the United States and other countries.
- If you reference Linux® in your presentation, please mark the first use and include the following; otherwise delete:
  Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.
- If the text/graphics include screenshots, no actual IBM employee names may be used (even your own), if your screenshots include fictitious company names (e.g., Renovations, Zeta Bank, Acme) please update and insert the following; otherwise delete: All references to [insert fictitious company name] refer to a fictitious company and are used for illustration purposes only.