WebSphere Application Server Version 8 High Availability Enhancements

David Follis
IBM

August 11, 2011
Session Number 9484

WebSphere Application Server Sessions

Day	Time	#	Title	Speaker	Room
Wednesday	3:00	9483	Using IBM's New Cross-Platform Installer on z/OS	Mierzejewski	Oceanic 5
Thursday	8:00	9482	WAS Version 8 – Overview	Follis	Europe 2
Thursday	9:30	9486	WAS Version 8 – Batch Update	Hutchinson	Europe 2
Thursday	11:00	9485	WAS Version 8 – New z/OS Exploitation/Differentiation Features	Follis	Europe 2
Thursday	1:30	9484	WAS Version 8 – High Availability Enhancements	Follis	Europe 2
Thursday	3:00	9488	WAS - Back to Basics Part 1	Loos	Europe 2
Thursday	4:30	9489	WAS - Back to Basics Part 2	Stephen	Europe 2
Friday	8:00	9490	WAS for z/OS - Level 2 Update	Stephen	Europe 2
Friday	9:30	9487	WAS for z/OS – PotPourri	Follis, Hutchinson, Loos, Mierzejewski, Stephen, etc.	Europe 2

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

CICS*
DB2*
GDPS*
Geographically Dispersed Parallel Sysplex
HiperSockets
IBM*
IBM eServer
IBM logo*
IMS
On Demand Business logo
Parallel Sysplex*
RACF*
System z9
WebSphere*
z/OS
zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon 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 throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

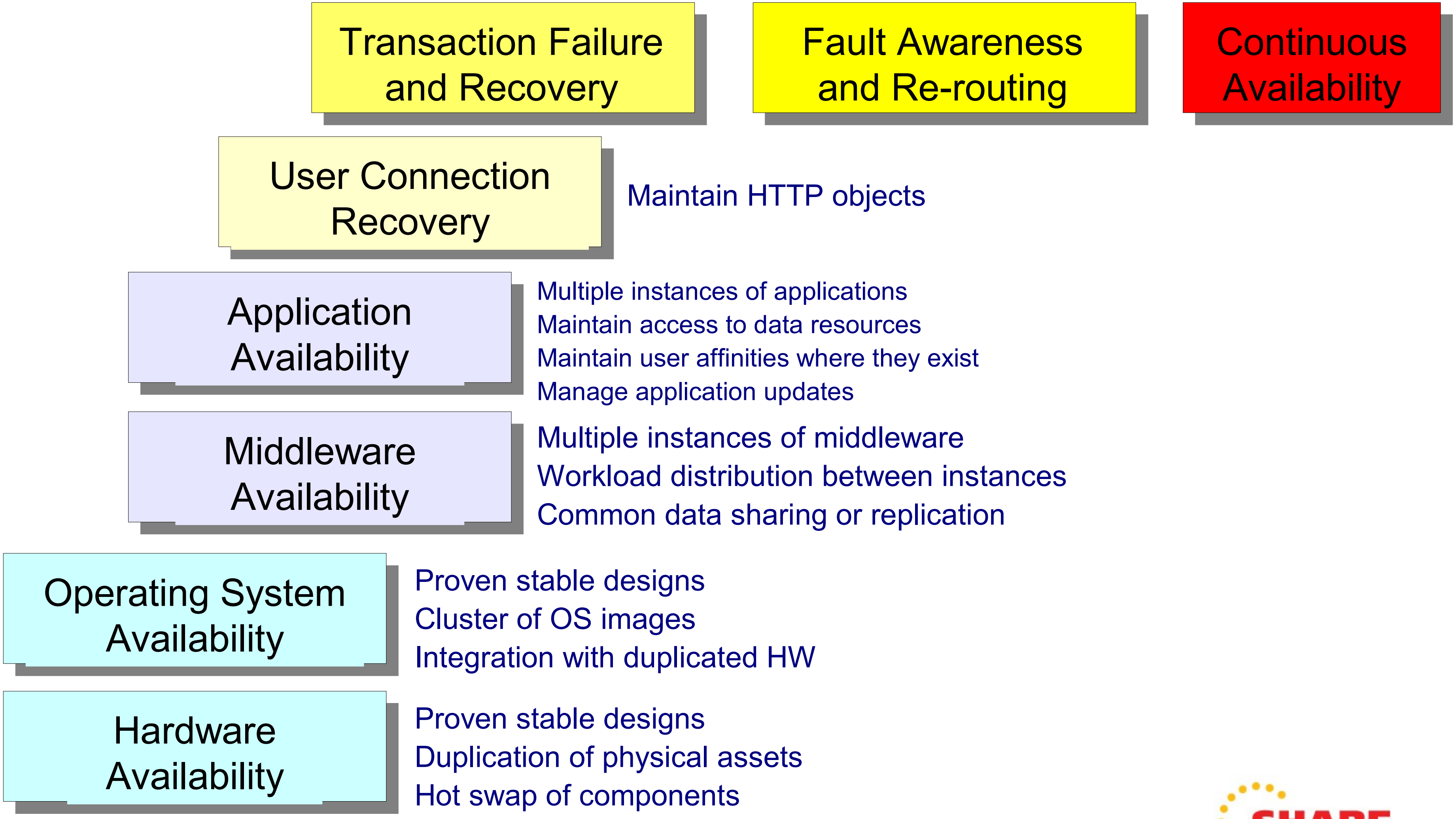
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Disclaimer

- The information contained in this documentation is provided for informational purposes only. While efforts were many to verify the completeness and accuracy of the information contained in this document, it is provided "as is" without warranty of any kind, express or implied.
- This information is based on IBM's current product plans and strategy, which are subject to change without notice. IBM will not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation.
- Nothing contained in this documentation 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 the IBM software.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon 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 throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

First Consideration -- What Constitutes an "Outage?"

This is key -- know what you're trying to plan to protect against. This will keep you from under-engineering as well as over-engineering.



Transaction Failure and Recovery

Fault Awareness and Re-routing

Continuous Availability

User Connection Recovery

Maintain HTTP objects

Application Availability

Multiple instances of applications
Maintain access to data resources
Maintain user affinities where they exist
Manage application updates

Middleware Availability

Multiple instances of middleware
Workload distribution between instances
Common data sharing or replication

Operating System Availability

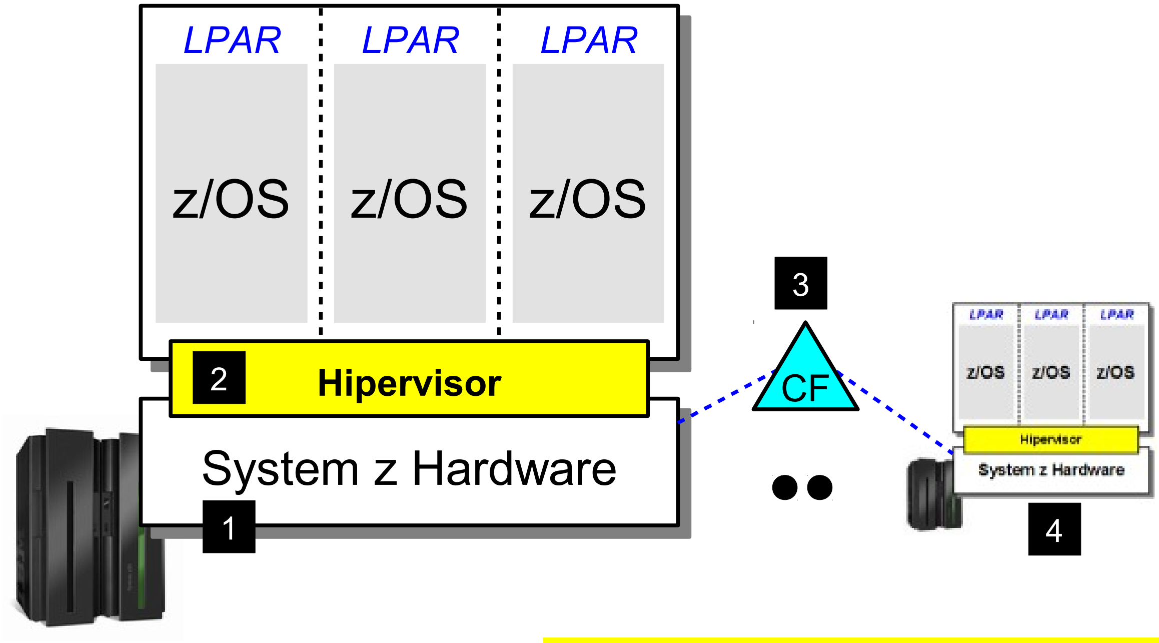
Proven stable designs
Cluster of OS images
Integration with duplicated HW

Hardware Availability

Proven stable designs
Duplication of physical assets
Hot swap of components

What Does System z and z/OS Bring to the Table

This is where WAS z/OS starts its HA journey ...



1. Hardware

A strong story about designed-in redundancy, hot-swappable components and mean time between failure measured in years.

2. Hipervisor

The virtualization layer that allows multiple logical partitions (LPARs) to be hosted on top of the physical hardware resources. Extremely stable with sophisticated dynamic qualities.

3. Coupling Facility / Parallel Sysplex

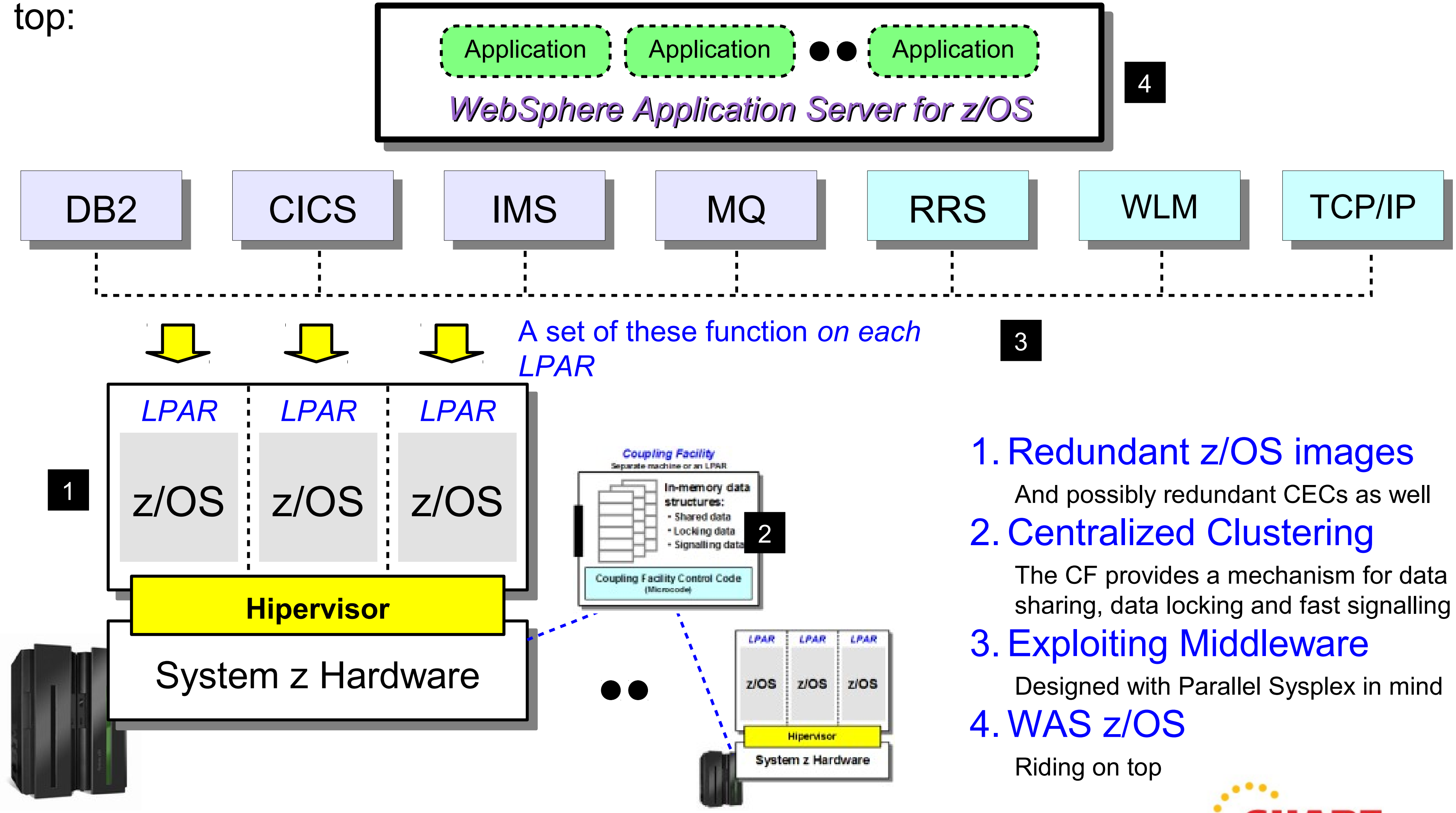
This is the heart of the HA story ... this is what provides the shared data, rapid signaling and clustering at the OS level.

4. Other CECs

Parallel Sysplex is not limited to a single CEC. Multiple CECs may be joined. The physical distance between the machines in the Sysplex may be expanded to span buildings or cities.

z/OS Middleware Components that Exploit the Parallel Sysplex

Let's do a quick survey of the key middleware components that are "Sysplex Aware" ... this then sets the stage for the discussion of WAS z/OS that rides on top:



1. Redundant z/OS images

   And possibly redundant CECs as well

2. Centralized Clustering

   The CF provides a mechanism for data sharing, data locking and fast signalling

3. Exploiting Middleware

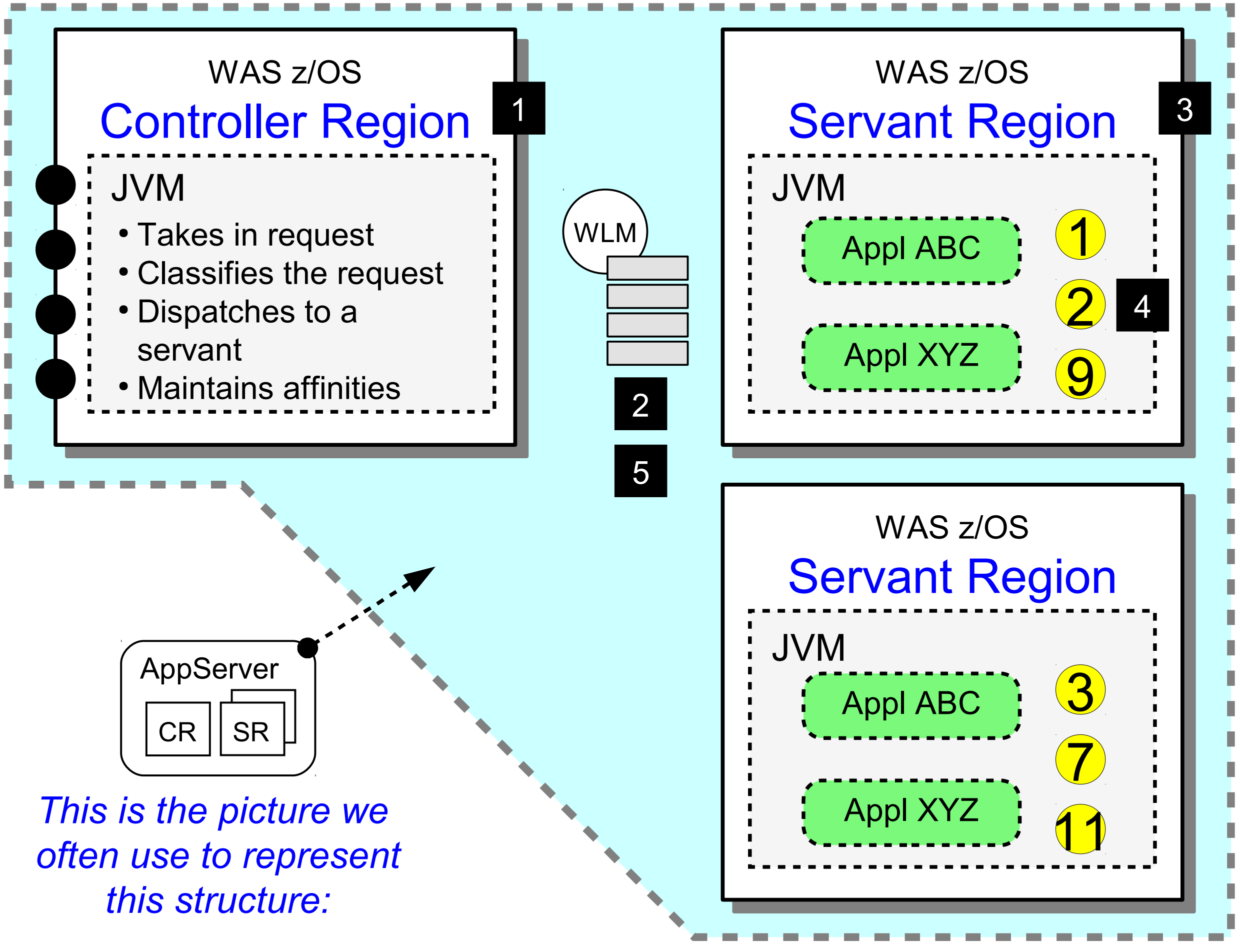
   Designed with Parallel Sysplex in mind

4. WAS z/OS

   Riding on top

The Split JVM Model -- Redundant JVMs Behind the Listener Ports

This is the first line of defense * -- redundant JVMs per application servers provides nearly seamless protection against JVM outages:



This is the picture we often use to represent this structure:

1. Controller

Consider this IBM plumbing code. It's primary role is summarized by the bullets to the left

2. WLM Work Queues

The controller makes use of WLM work queues between CR and SR. This provides a way to segment by classification as well as a queuing point to buffer against overruns.

3. Servant(s)

This is where the applications run. Multiple concurrent servants is possible and is what provides the redundancy.

4. User Session Objects

Not replicated to each servant, which means no unnecessary usage of heap. Sessions maintained in z/OS data spaces, so lost servant does not mean lost sessions.

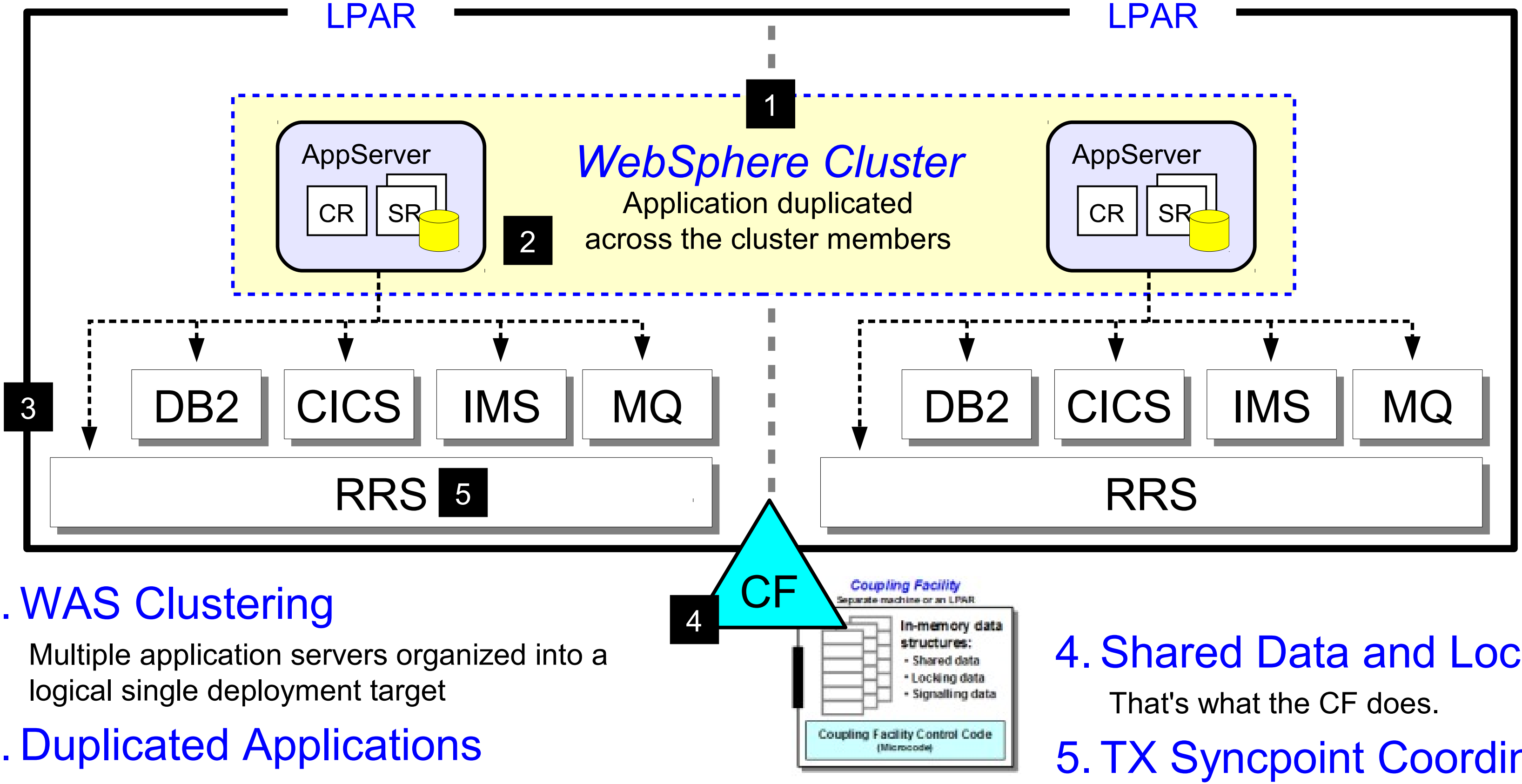
5. Auto Restart

WLM will automatically restart any failed servant regions

* One could say that the System z hardware and the z/OS operating system are the true first lines of defense against outage

Clustering Across LPARs -- the Second Line of Defense

Clustering is a feature available on all platforms of WAS. The difference is the access to the Sysplex-enabled middleware components:


LPAR
LPAR
1
WebSphere Cluster
Application duplicated across the cluster members
AppServer
CR
SR
2
AppServer
CR
SR
3
DB2
CICS
IMS
MQ
RRS
5
DB2
CICS
IMS
MQ
RRS
CF
4
Coupling Facility
Separate machine or an LPAR
In-memory data structures:
Shared data
Locking data
Signalling data
Coupling Facility Control Code (Microcode)

1. WAS Clustering

   Multiple application servers organized into a logical single deployment target

2. Duplicated Applications

   WAS propegates application binaries to all members of the cluster.

3. Redundant Sysplex-aware Middleware

   Physically separate from other LPARs but sharing common data structures in the CF.

4. Shared Data and Locking

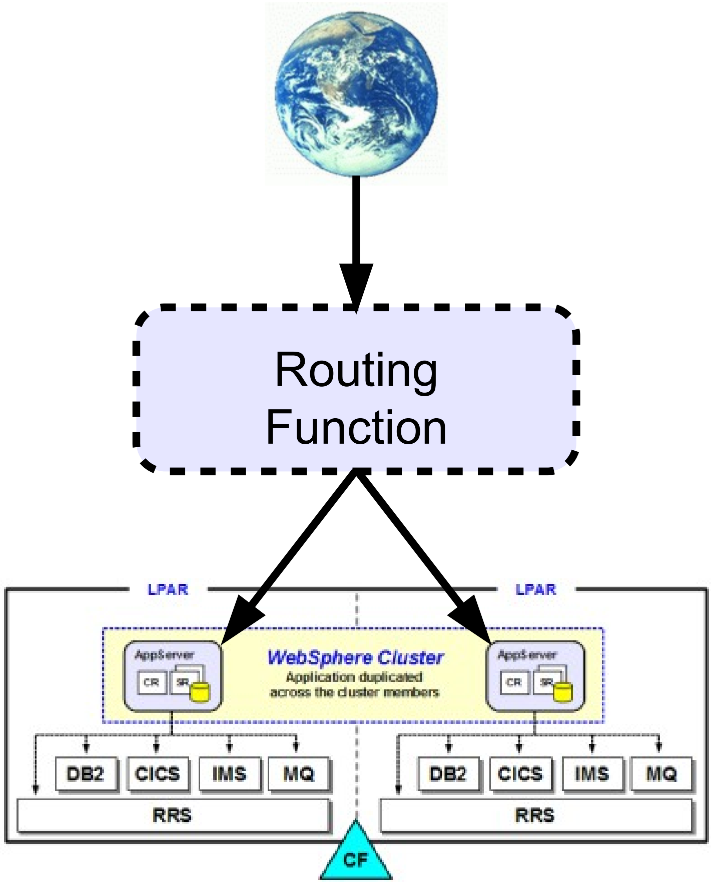
   That's what the CF does.

5. TX Syncpoint Coordinator

   Resource Recovery Services (RRS) is a transaction syncpoint coordinator. All the major z/OS subsystems make use of it to coordinate 2PC processing.

Routing from Outside In

WAS cluster members represent physically separate application servers hosting separate TCP ports. So *something* has to be "out front" routing:

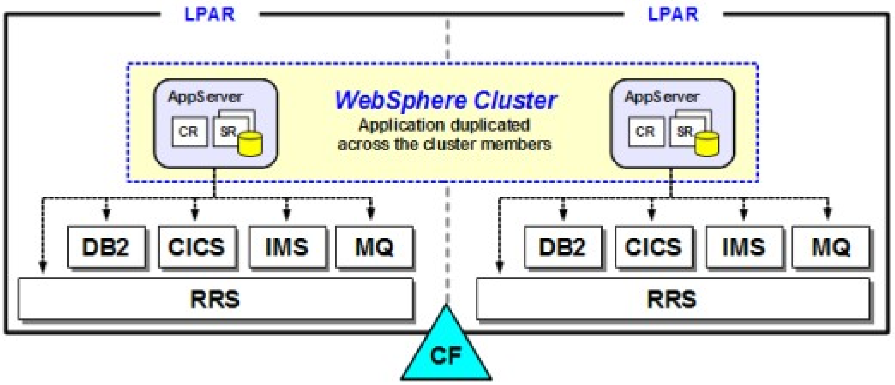


There is a great deal to consider in this space:

- Does it terminate the SSL connection?
  This allows for a more flexible routing to the backend.
- Do server affinities need to be honored?
  The most common is affinity based on session object location.
- Where does this routing function reside -- inside the DMZ or behind the secure firewall?
  Inside the DMZ suggests a minimum of protocols and ports punching through the back secure firewall.
- How much knowledge of the environment do you desire the function to possess?
  Server up or down? Or more -- J2EE application status?
- How intelligent do you want the routing to be?
  Three basic levels -- pure round robin; weighted round robin, intelligent placement based on advice.
- Plus other criteria not listed above

Four Topics to Consider

They are ...



Server Affinities

The most common are the result of creating HTTP session objects.

Affinity restricts request routing flexibility

Data Access Approaches

It's a trade-off ... the benefits of co-location vs. the flexibility of IP-based re-routing.

Other Application Dependencies

The application may be up and accessible but not "working" because of some other element in the design not available.

What do you do about this other than keep close tabs on the relationships and enforce careful change control?

Transactions and Transaction Recovery

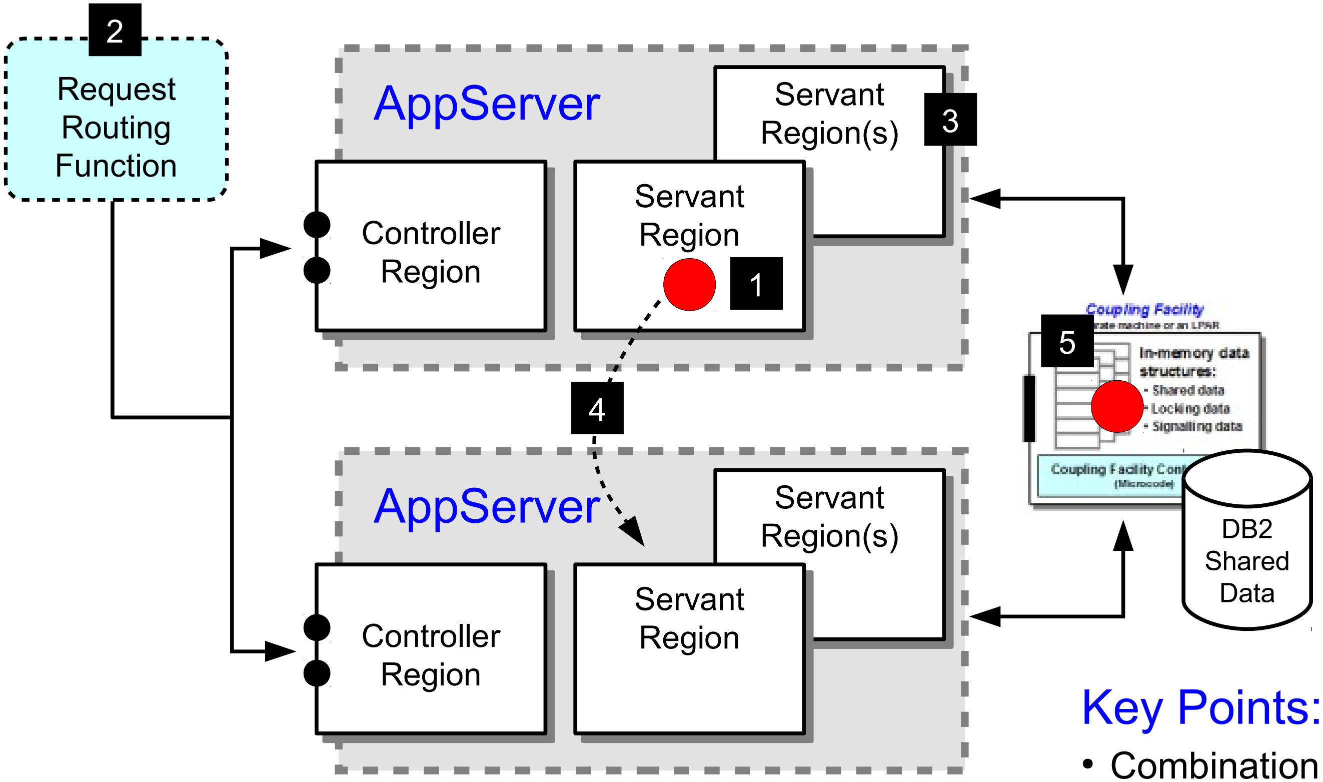
WAS has a mechanism to roll back in-doubt transactions. RRS is central to this.

This is also why it's really important to lock down the definition of what constitutes an outage

It may well be that user access to a *part* of the functionality is acceptable

Server Affinities

The most common is HTTP session affinities, which are used to hold transient data. But without planning can create a need to route users back to the initial server



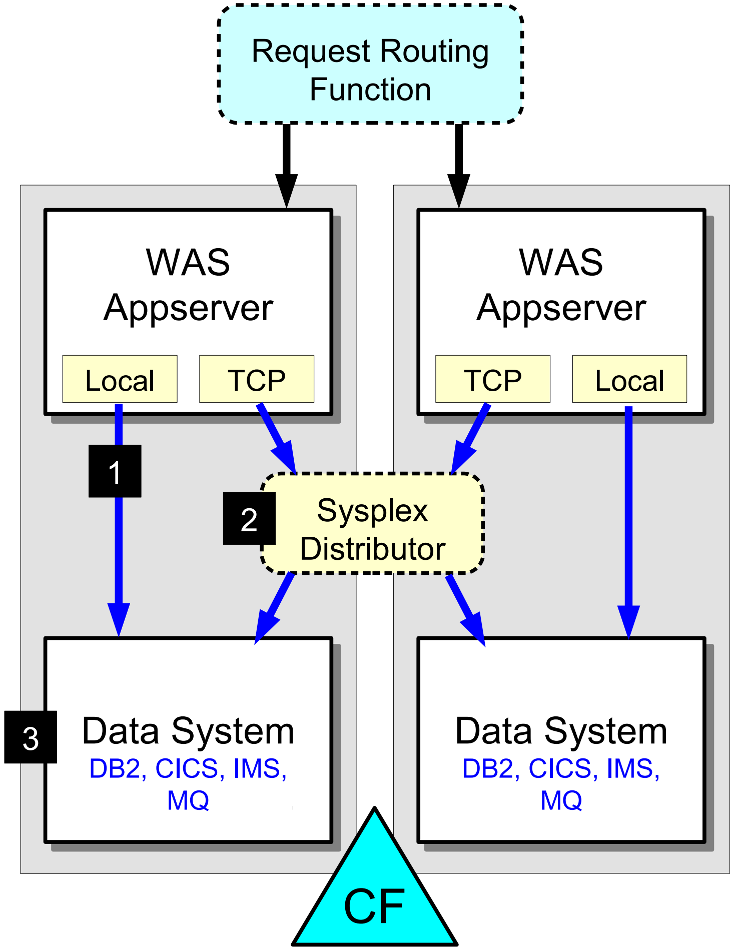
1. Session Object
   Created by application if designed to do so. It is a data object in JVM memory
2. Affinity Routing
   One option is to provide affinity routing. WAS Plugin does this, as does Proxy. Sysplex Distributor does not
3. Within Appserver
   If the servant in which a user object resides goes down, WAS z/OS automatically takes care of that and places user into a surviving servant
4. Replication Domain
   A feature of WAS across platforms. This serializes the object and copies it over the network to other servers in the defined domain.
5. Session Persistence
   Object persisted to DB2 and fetched back as needed

Key Points:

- Combination of affinity routing and replication or persistence is common
- On z/OS persistence performs as well or better than replication
- Create affinities only where necessary; be careful of object size

Data Access Approaches - The T2 vs. T4 Debate

You gain a degree of flexibility with an TCP-based connection but lose some of the advantages of a local cross-memory connection:



1. Local Connectors

Uses the cross-memory native interfaces. Available for DB2, CICS, IMS and MQ.

Advantages: Speed, avoid serialization, assert identity, single thread of execution, propegate enclave for DB2

Disadvantages: Loss of data system means application has no access to data unless alternative connections are made available. Routing function may not know backend data system is gone.

2. TCP-based Connectors

Uses the TCP network to flow requests to target listener. Available for DB2, CICS, IMS and MQ.

Advantages: Loss of TCP connection typically signals retry; SD will connect to surviving member. DB2 T4 takes this even further.

Disadvantages: Potential loss in performance, generally implies alias ID and PW.

Data locks may exist ... other work may proceed but work related to held data can not until failure subsystem restarted so locks can be freed.

3. What is Being Protected Against?

If you are concerned about loss of data system while WAS server stays up, then the TCP based with intermediate routing is a consideration.

If your primary concern is loss of the LPAR "tower" then the use of TCP connectors becomes less important.

New in Version 8

Terminology

Connection Management

The component of WAS that keeps track of connections

Connection Pool

A group of connections to a particular resource manager

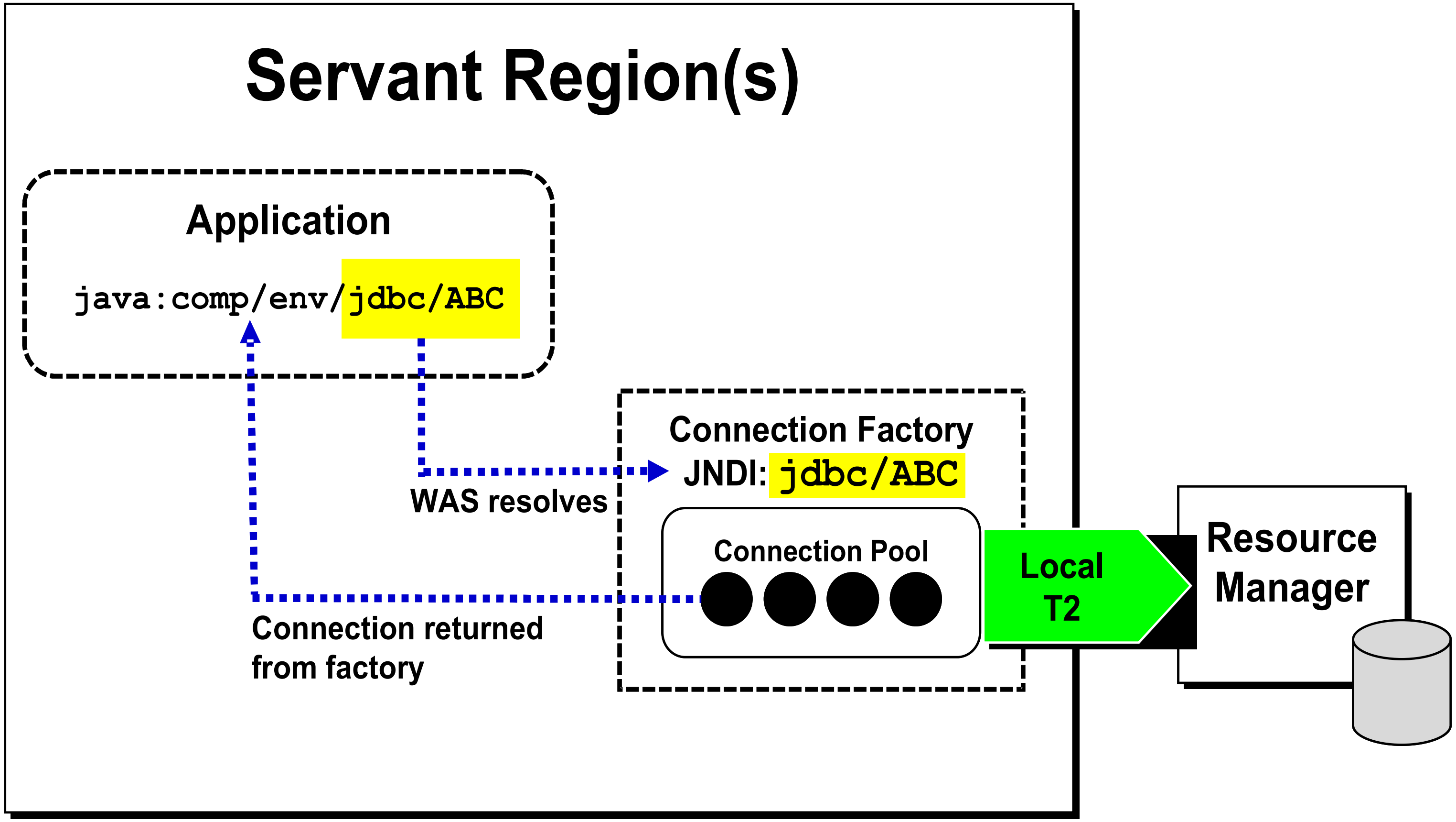
Resource Adapter

The code supplied by the resource manager that is used to access it. Supplied in a .rar file

Connection Factory

Defined for a particular resource adapter. Applications look up a connection factory to get connections to a particular resource manager. Connection information can either be configured with the factory or provided by the application.

Resolving Resource References



Servant Region(s)

Application

java:comp/env/jdbc/ABC

Connection Factory

JNDI: jdbc/ABC

WAS resolves

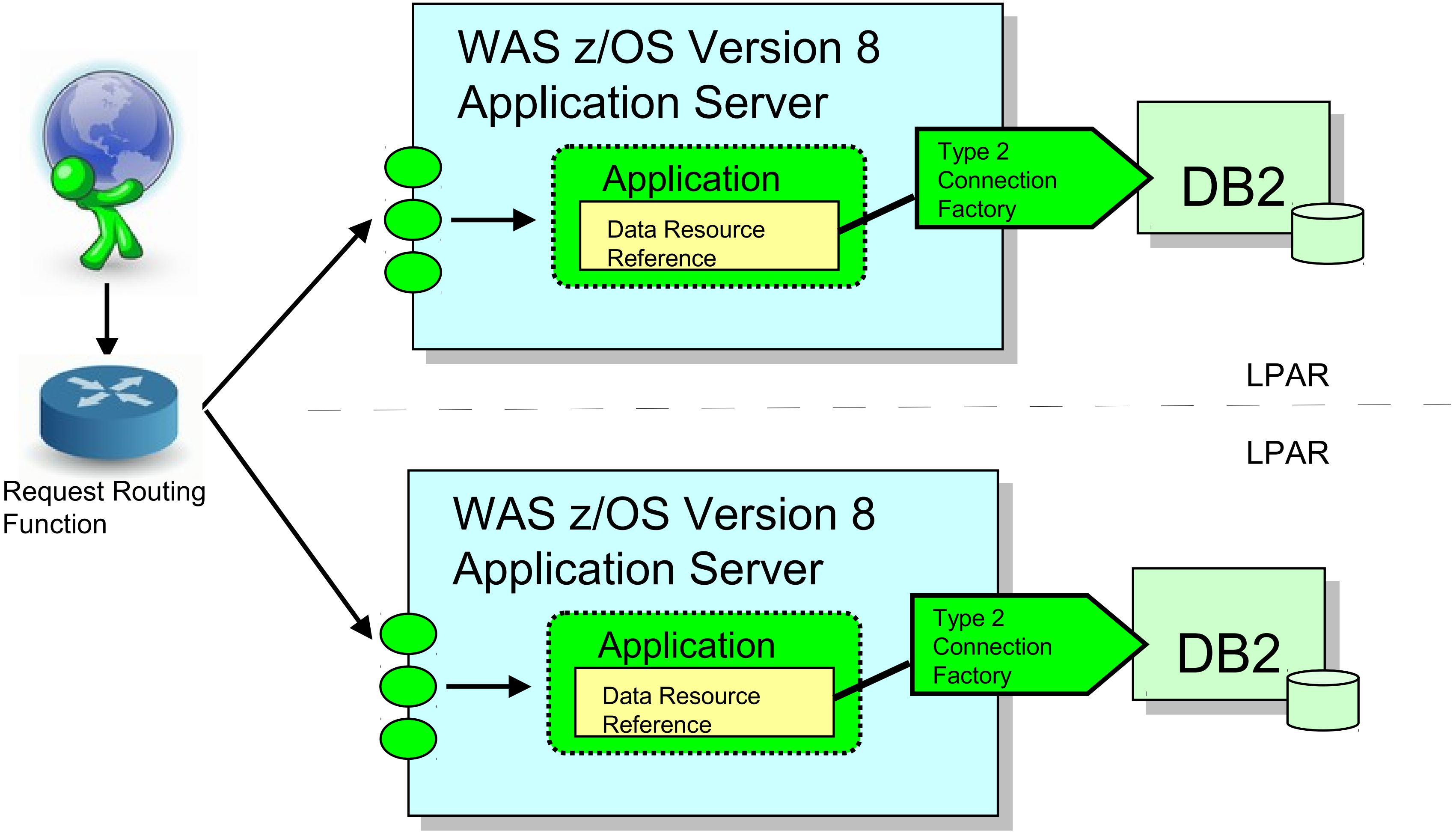
Connection Pool

Local T2

Resource Manager

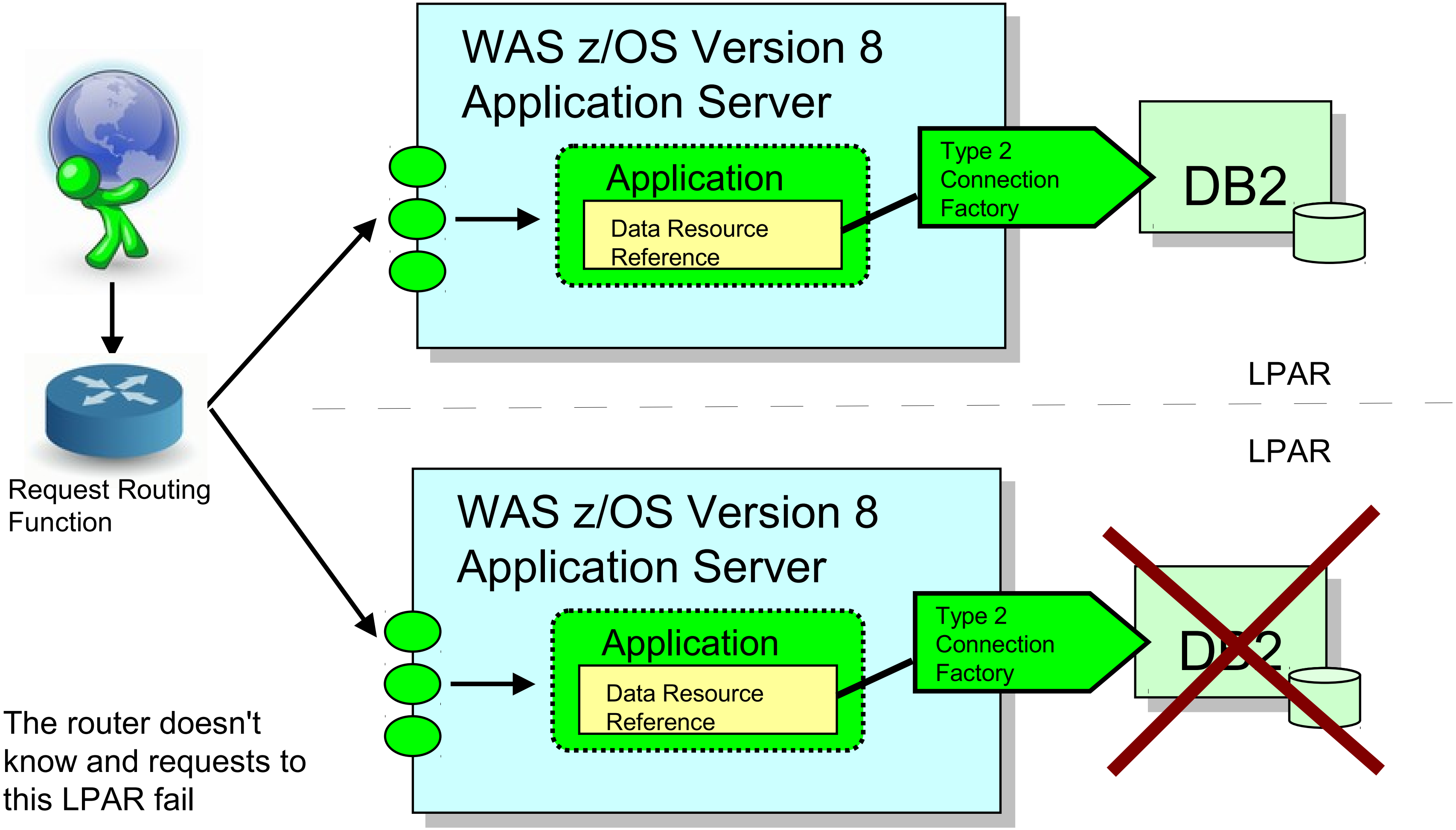
Connection returned from factory

A Typical Clustered Environment with Type-2 Connectors



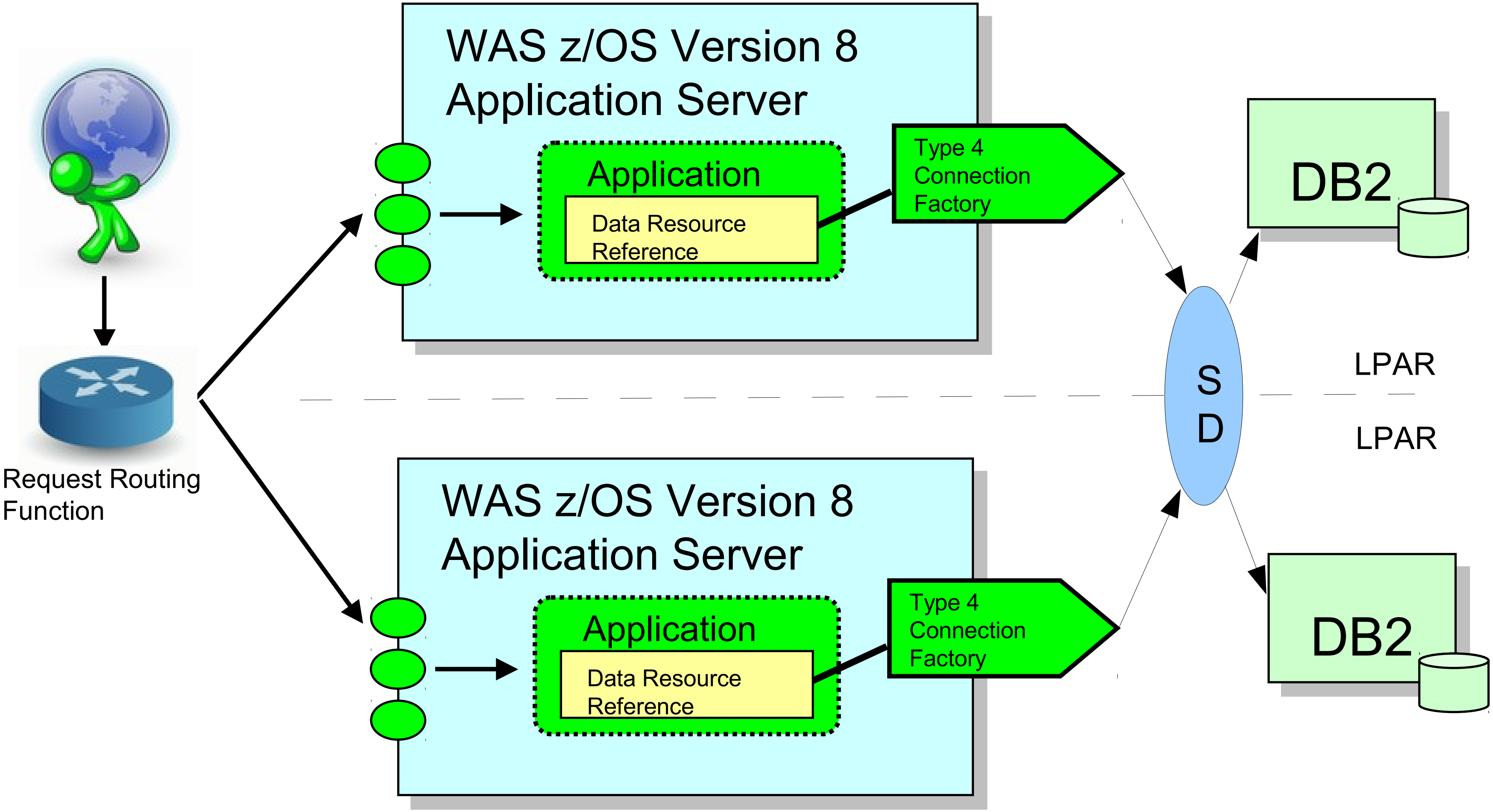
Just using DB2 as an example..

When something bad happens....



The router doesn't know and requests to this LPAR fail

A Common Solution...



Use Type-4 Connectors and Sysplex Distributor to eliminate close coupling between WAS and DB2... But this surrenders the value of co-location!

For Version 8 we asked three questions:

1) Can WAS tell when a back end resource manager has failed?

2) What can we do once we know?

3) Can we tell when the resource manager is back and undo what we did?

For Version 8 we asked three questions:

1) Can WAS tell when a back end resource manager has failed?

   YES! Connection Management already has the ability to recognize a failed resource manager and eliminate 'stale' connections from the pool.

2) What can we do once we know?

3) Can we tell when the resource manager is back and undo what we did?

For Version 8 we asked three questions:

1) Can WAS tell when a back end resource manager has failed?

   YES! Connection Management already has the ability to recognize a failed resource manager and eliminate 'stale' connections from the pool.

2) What can we do once we know?

   We already tell customers to use the PAUSELISTENERS function to stop the server from taking new work. The server could automatically pause. Maybe there are other things......

3) Can we tell when the resource manager is back and undo what we did?

For Version 8 we asked three questions:

1) Can WAS tell when a back end resource manager has failed?

   YES! Connection Management already has the ability to recognize a failed resource manager and eliminate 'stale' connections from the pool.

2) What can we do once we know?

   We already tell customers to use the PAUSELISTENERS function to stop the server from taking new work. The server could automatically pause. Maybe there are other things......

3) Can we tell when the resource manager is back and undo what we did?

   Maybe. With DB2 we can just try to get a new connection using the attributes in the Connection Factory configuration.
   We can use RESUMELISTENERS to undo the pause.

The Basic Flow

Detect

Determine there is a problem based on existing reporting of failures and a configured threshold of concurrent failures

(Connection Pool property 'failureThreshold')

Take Action

Take some configured action
(more on this coming up)

Monitor

Monitor the resource manager by testing at an interval until we recognize it is available again
(Connection Pool property resourceAvailabilityTestRetryInterval

Relational – try a new connection (DB2, other JDBC)
Non-Relational – testConnection method on the adapter
(Implemented by WOLA)

Undo

When the resource manager is back...
Undo whatever action we took

Taking Action: FAILOVER

- Define an alternate connection factory
    - Connection Pool attribute alternateResourceJNDIName

- When the resource manager can't be reached, hand out connections from the alternate..

- In flight transactions with the 'primary' resource manager will fail
    - naturally, they were in-flight when the RM they were working with died
    - In-doubt transactions will wait for the primary to come back to resolve

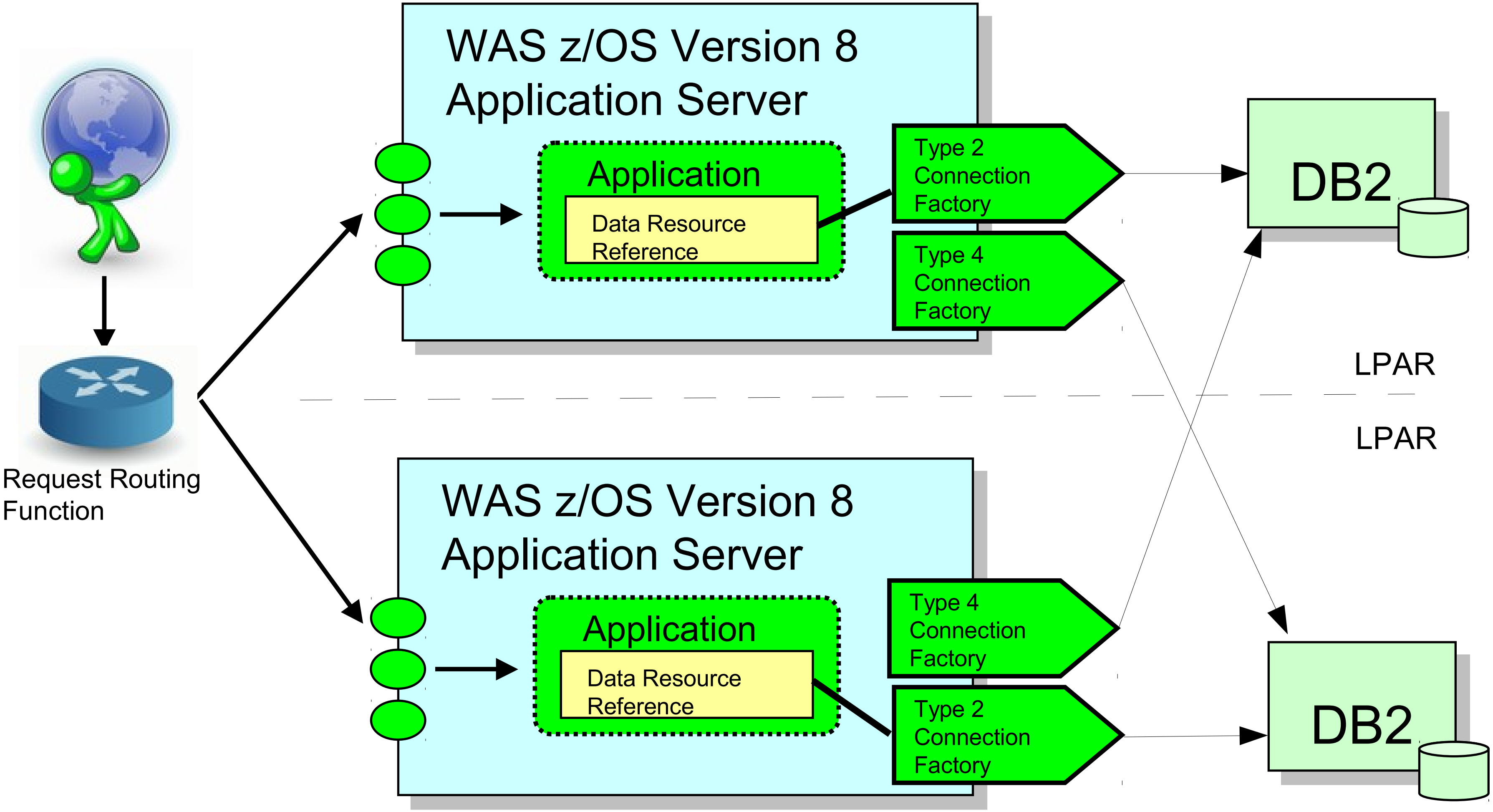
- When the 'primary' comes back, use it for new connections

- Connections from the alternate pool will be returned when the current user finishes

- The alternate factory must be reachable
    - e.g. defined in the same Node

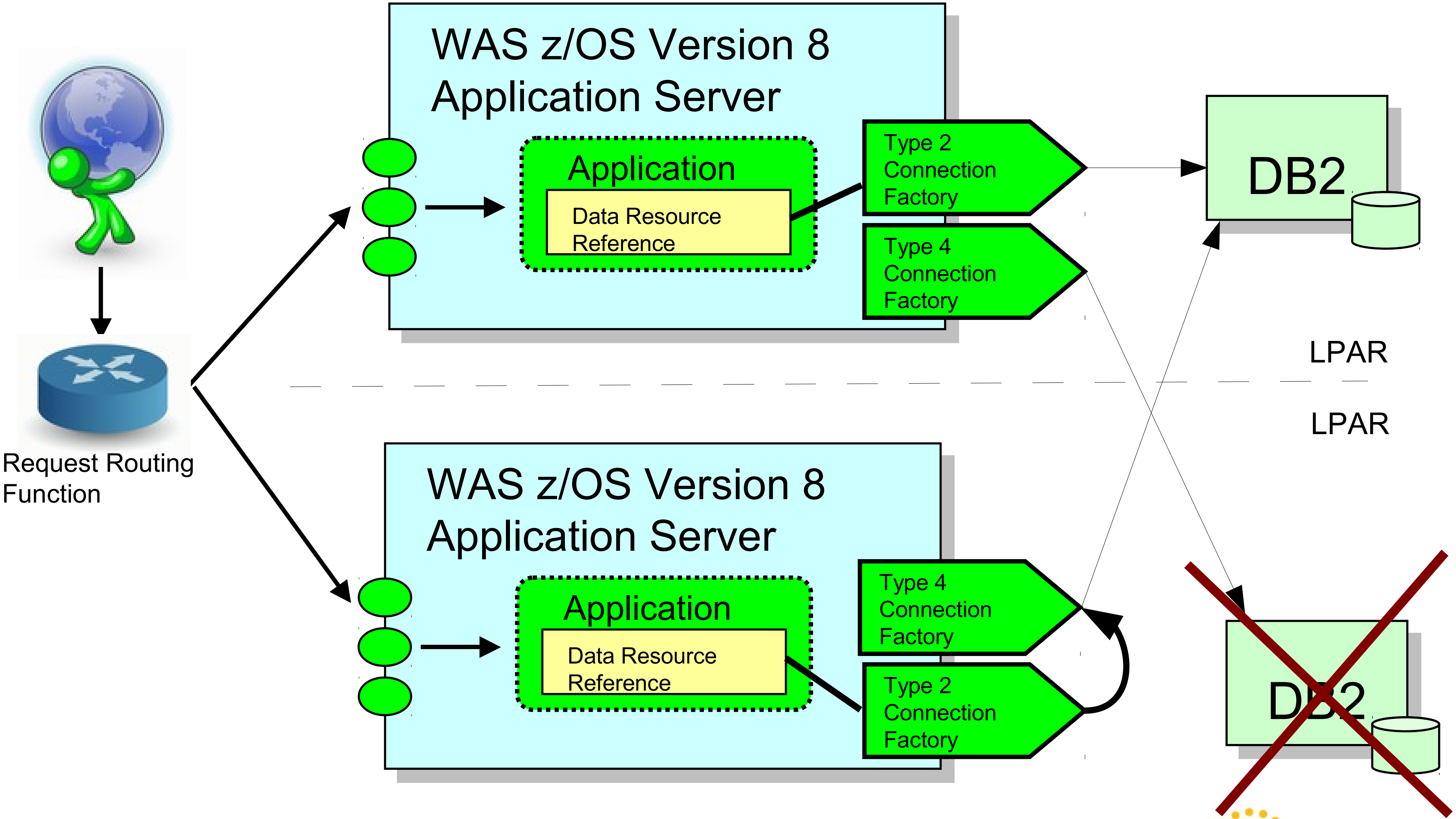
- The alternate factory can not be actively used
    - just for backup

Suppose we configure both connectors...



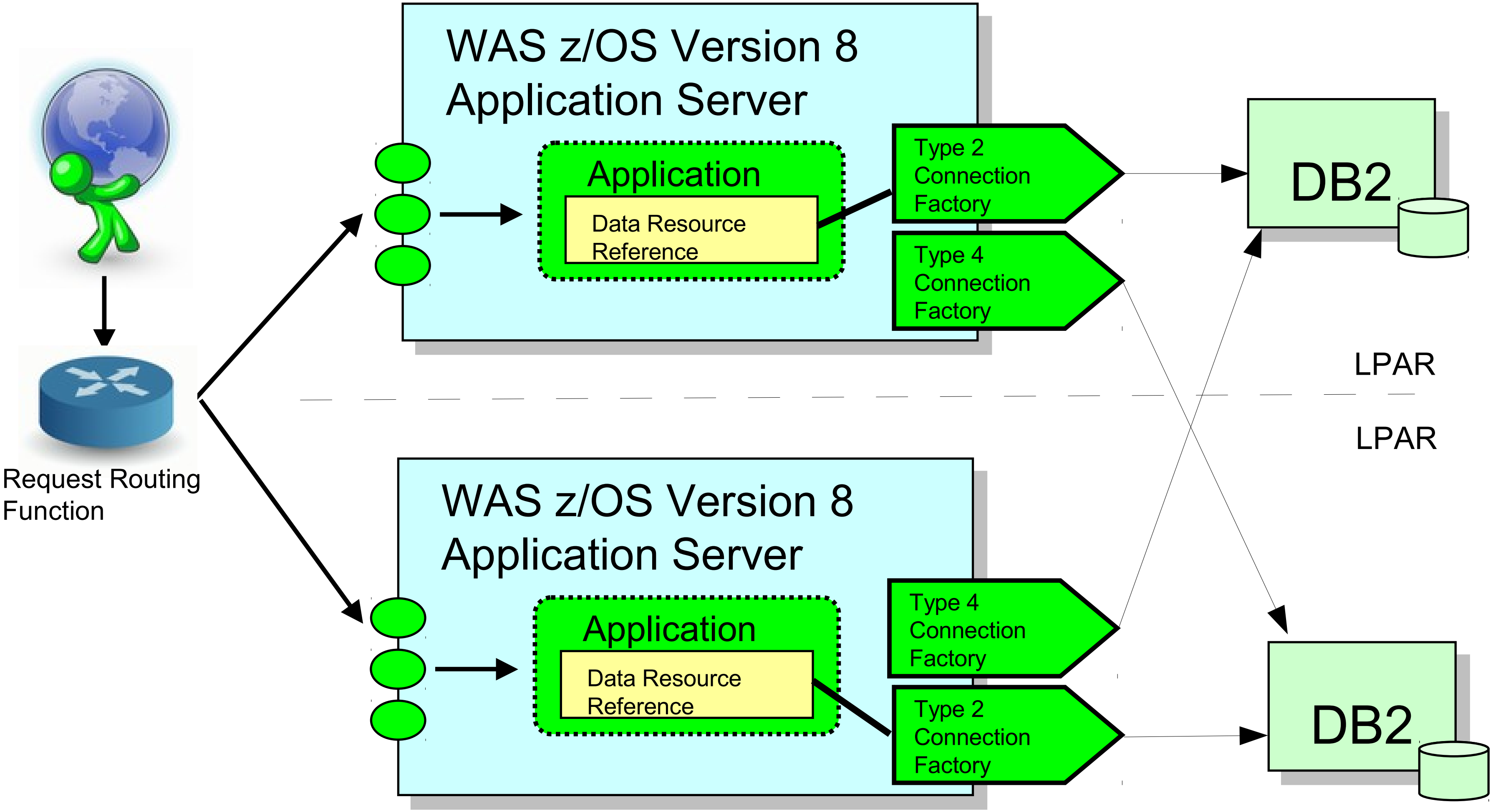
And the Type-4 connector is the 'alternate' for the Type-2

Then something bad happens....



New application connection requests use the Type-4 – AUTOMATICALLY!
WAS will also 'watch' for DB2 to come back....

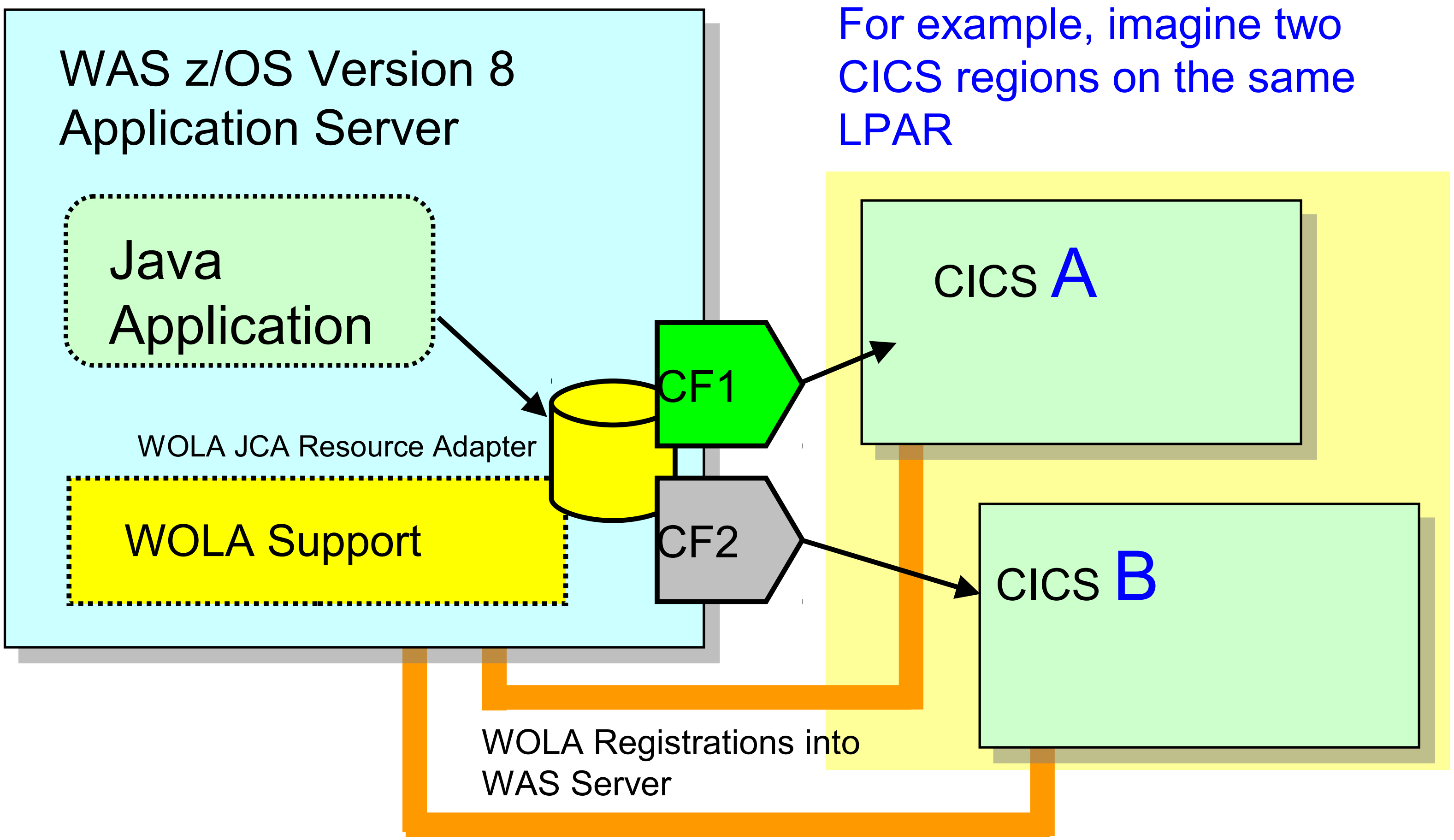
When the bottom DB2 is back...



Use of the Type-4 quiesces and we're back to normal

WOLA Variation on This New Function

WOLA participates in this as well in that a backup registered external address space now be used in the event the primary is lost:


WAS z/OS Version 8
Application Server
Java
Application
WOLA JCA Resource Adapter
WOLA Support
CF1
CF2
For example, imagine two CICS regions on the same LPAR
CICS A
CICS B
WOLA Registrations into WAS Server

WOLA is by definition “same LPAR,” and this gives you a degree of availability by allowing routing to secondary registered external address space

You can also trigger the 'failover' action manually

- Use the MODIFY command:
  - MODIFY server,FAILOVER,'connection factory JNDI name'

- And manually 'fail back':
  - MODIFY server,FAILBACK,'connection factory JNDI name'

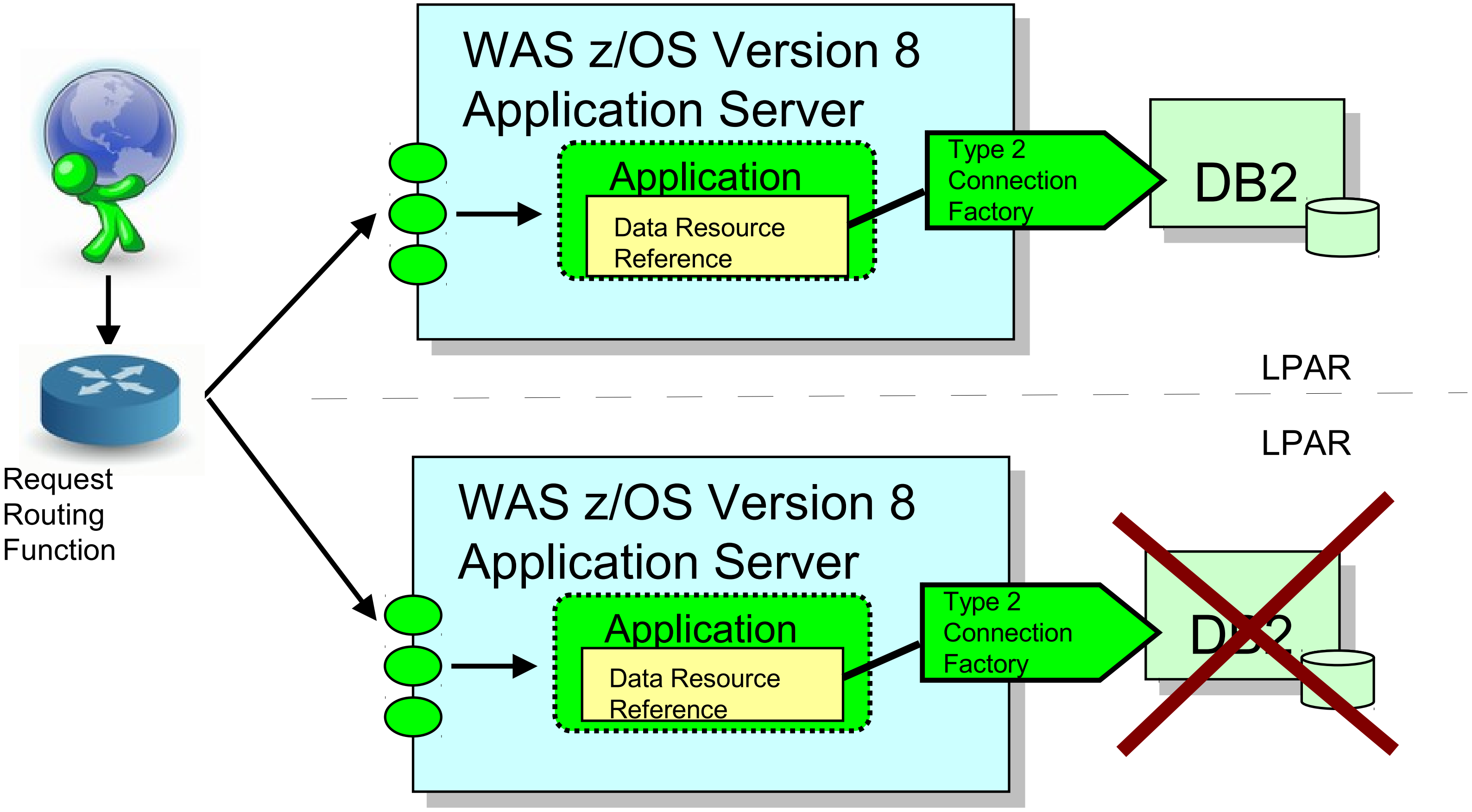
- Or use the MBean interface!

- Manual mode is useful for a planned outage

Are other actions available?

Yes! Configure failureNotificationActionCode on the connection pool

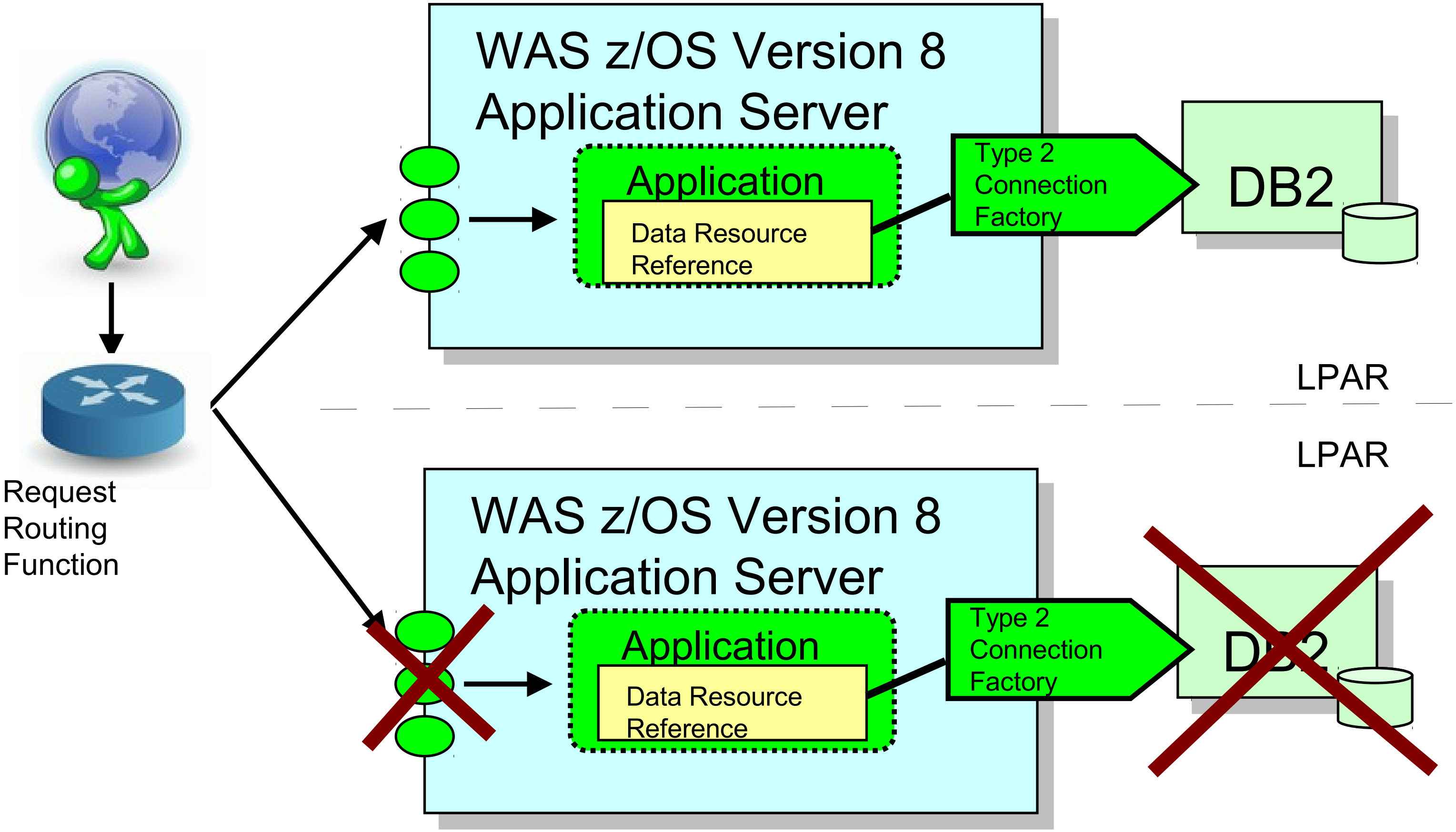
Action 1: Issue WTO BBOJ0130I



The WTO contains the server name and the connection factory JNDI name

Use automation to influence work routing away from that WAS server

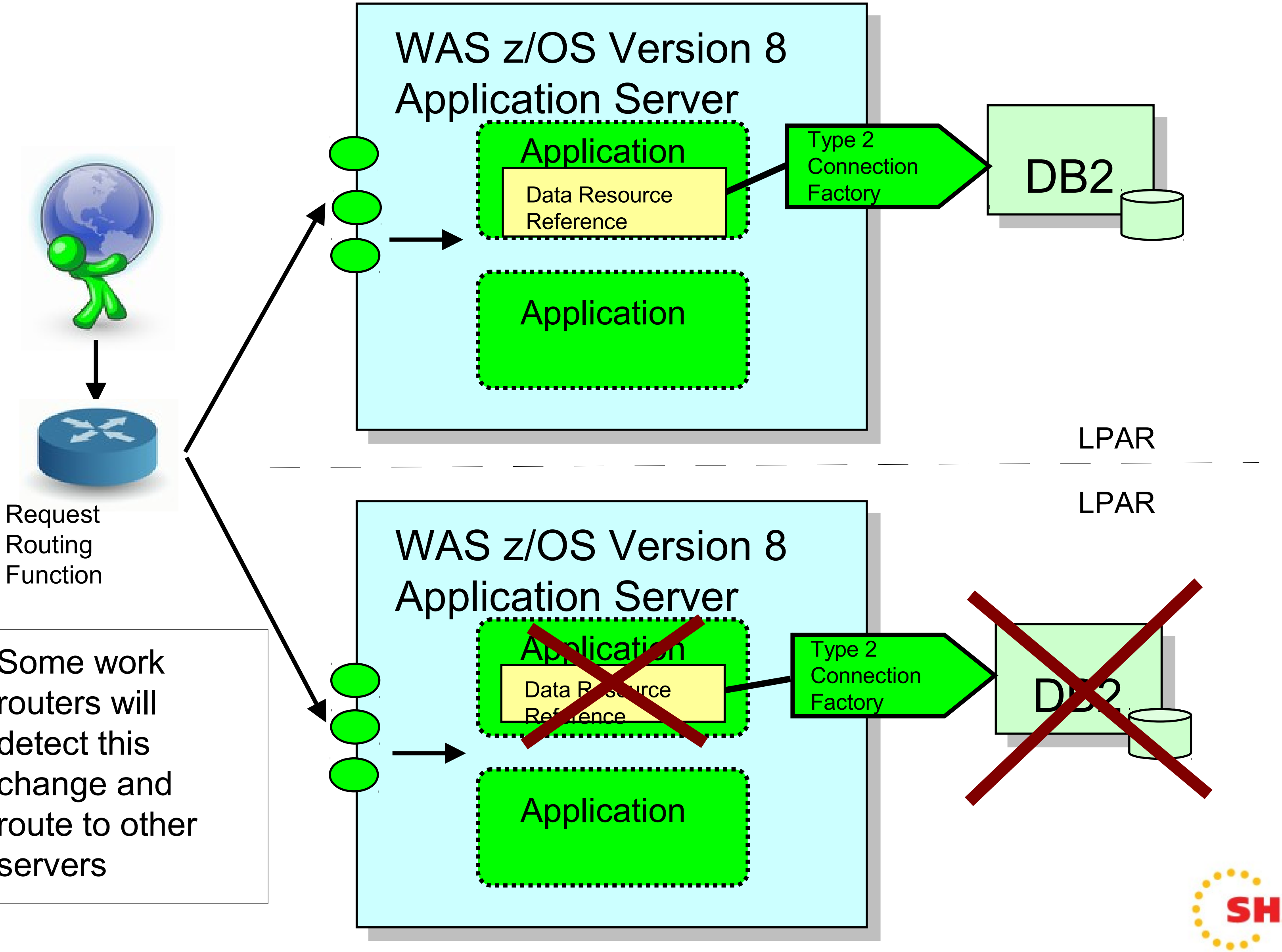
Action 2: Pause Listeners



Work routers see the ports close and route to the other server(s)

Resume Listeners is issued automatically when DB2 returns

Action 3: Stop Affected Applications



Some work routers will detect this change and route to other servers

A few other things...

- If your resource manager does not support the testConnection method (CICS)
    - Set enablePartialResourceAdapterFailoverSupport
    - Allows automatic failover, but requires manual 'failback'

- You can also disable/enable failover or failback by setting
    - disableResourceFailover
    - disableResourceFailback

- And there are Mbean and Modify command interfaces to disable/enable

- Improve performance at failover by setting
    - PopulateAlternateResource
    - Comes with some background overhead during normal operations

This is tricky stuff... Remember

TEST IT BEFORE YOU NEED IT