COMP 3400 Mainframe Administration[1]

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

[1]These slides are based in part on materials provided by IBM's Academic Initiative.

Today

- HTTP Server
- Websphere Application Server (WAS)
- Websphere MQ

Short History of WebSphere

- WebSphere started out as HTTP server and became one of the first z/OS products running under the UNIX interfaces
- In 1997, a JVM plugin was added to run servlets
- Version 4 saw the move to J2EE branding
- Version 5 (2003) of the application server was synchronized with other platforms
- Today, WebSphere is not only a J2EE application server but middleware implementing the concepts of a Service Oriented Architecture, Enterprise Service Bus and Business Process Management
- Portable

Web applications on z/OS

In the past:

- Existing (pre-Web) applications tied to z/OS (CICS, DB2)
- New developments were made on other platforms

Now: integration of both types of applications on z/OS.

z/OS HTTP Server

HTTP Server can run in three modes:

- *Stand-alone server* – for simple Web sites
- *Scalable server* – for interactive Web sites with servlets and JSPs and dynamic traffic volume
- *Multiple servers* – combination of multiple stand-alone and scalable servers

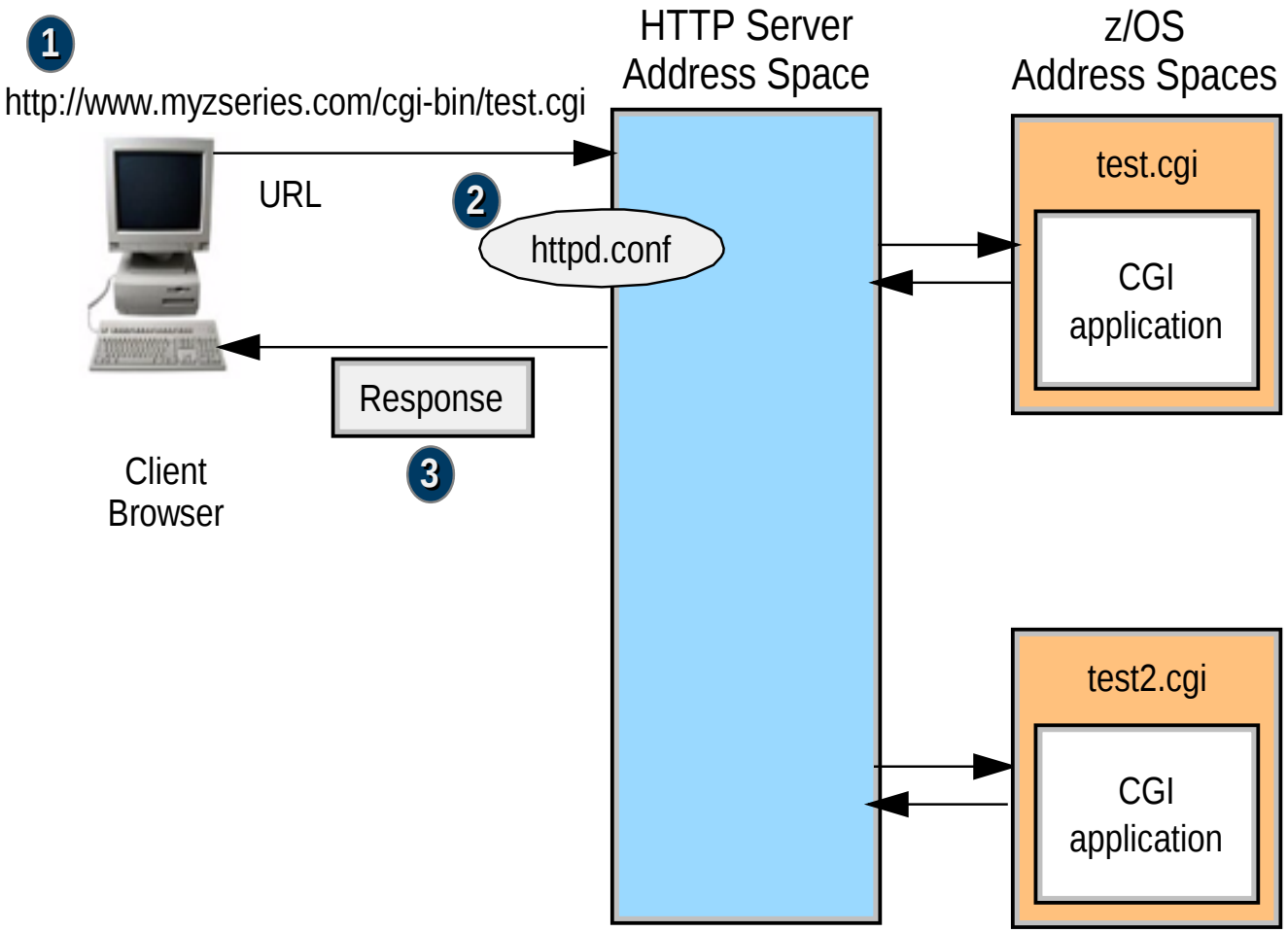
Serving Static Web Pages

- Works mostly like any other HTTP Daemon – HTTP Server retrieves the file and sends it to the user with HTTP header
- Main difference is that HTTP Server usually first has to convert files from EBCDIC to ASCII

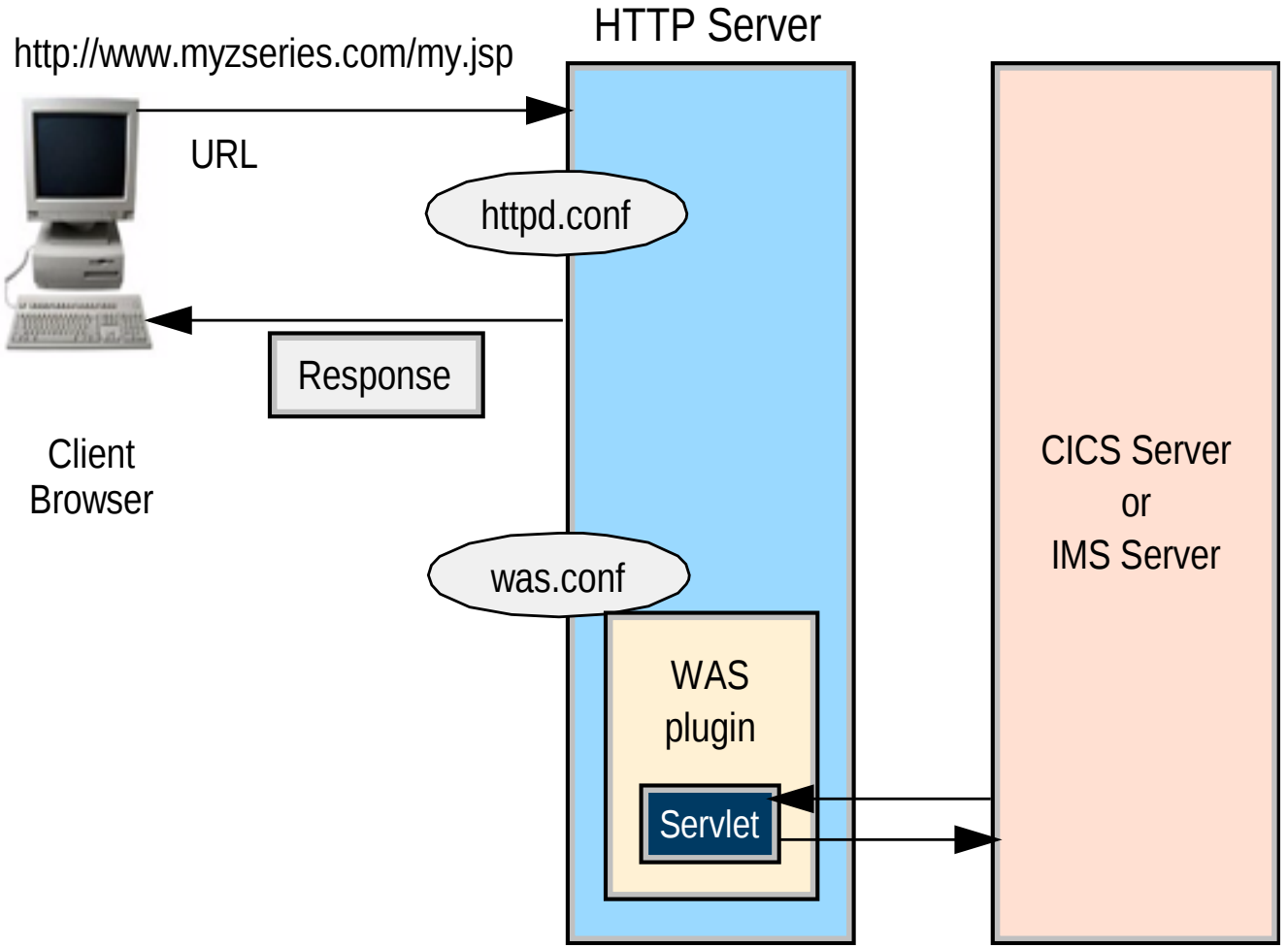
Serving Dyanmic Web Pages

- HTTP Server supports CGI (each request requires a separate address space) and FastCGI (multiple requests managed in the same AS)
- HTTP Server provides a plug-in interface to execute servlets either inside of the HTTP Server AS or communicate with a J2EE Server (using WebSphere HTTP Server plug-in)

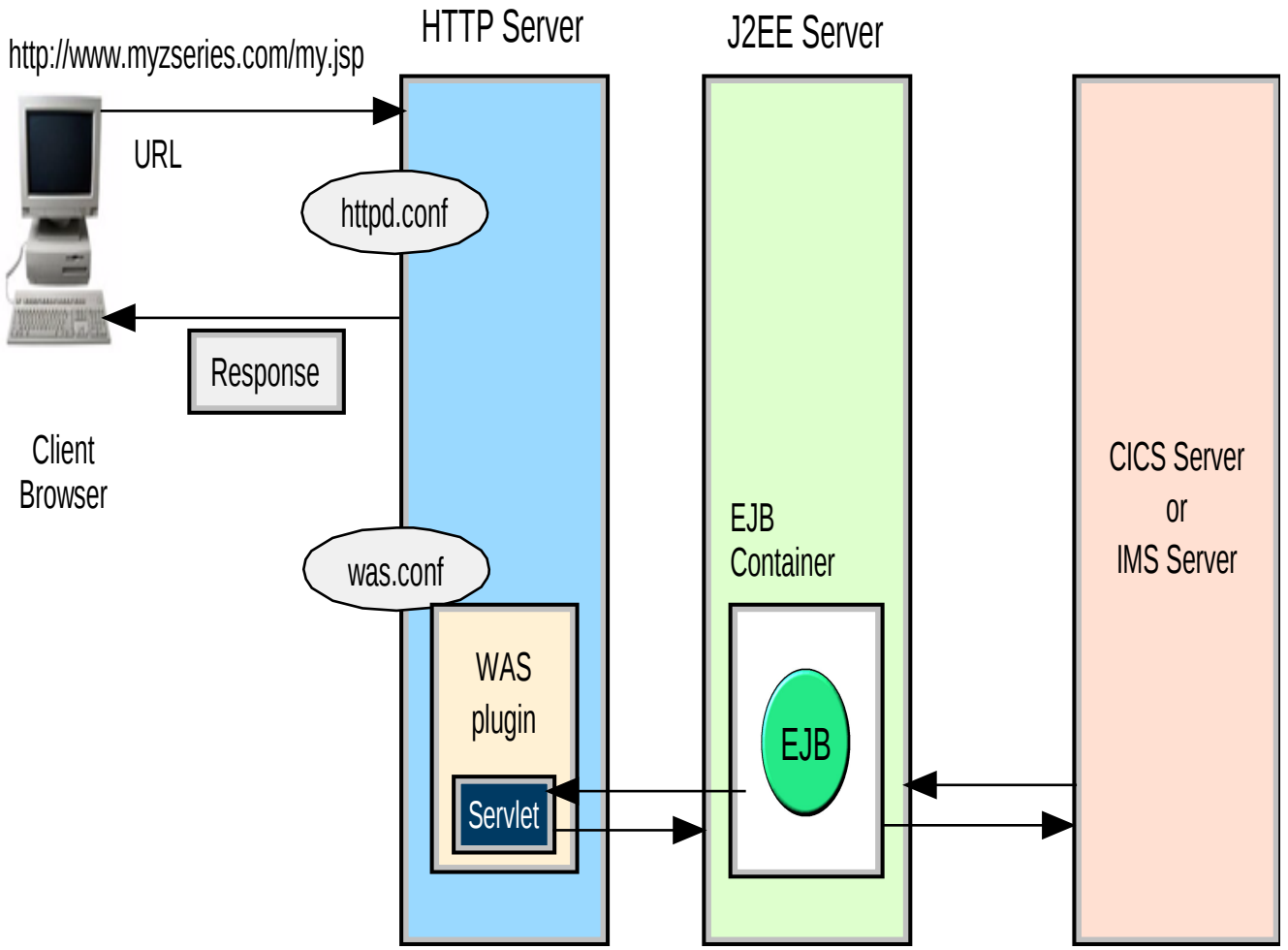
Example: CGI


1
http://www.myzseries.com/cgi-bin/test.cgi
URL
2
httpd.conf
Response
3
Client
Browser
HTTP Server
Address Space
z/OS
Address Spaces
test.cgi
CGI
application
test2.cgi
CGI
application

Example: WAS plugin



Example: WAS plugin with J2EE server



HTTP Server Features

- Performance and usage monitoring (using System Management Facilities (SMF))
- Tracing and Logging
- Server Side Includes (SSI), Cookies, HTTPS
- Simple Network Management Protocol (SNMP), Lightweight Data Access Protocol (LDAP)
- Persistent connections (HTTP 1.1), Virtual hosts, Proxy support
- Thread level security (independent security environment per client)
- Caching (HFS files, z/OS data sets, etc.)

Questions

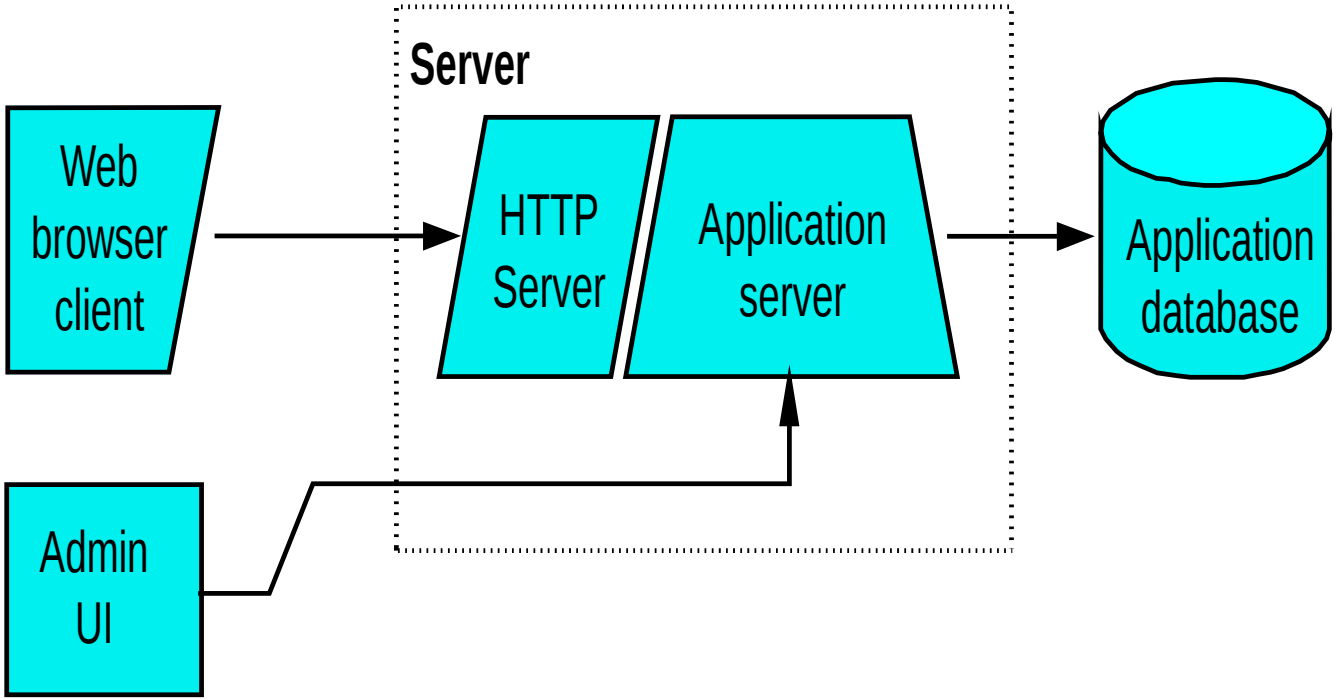
?

WebSphere Application Server (WAS)

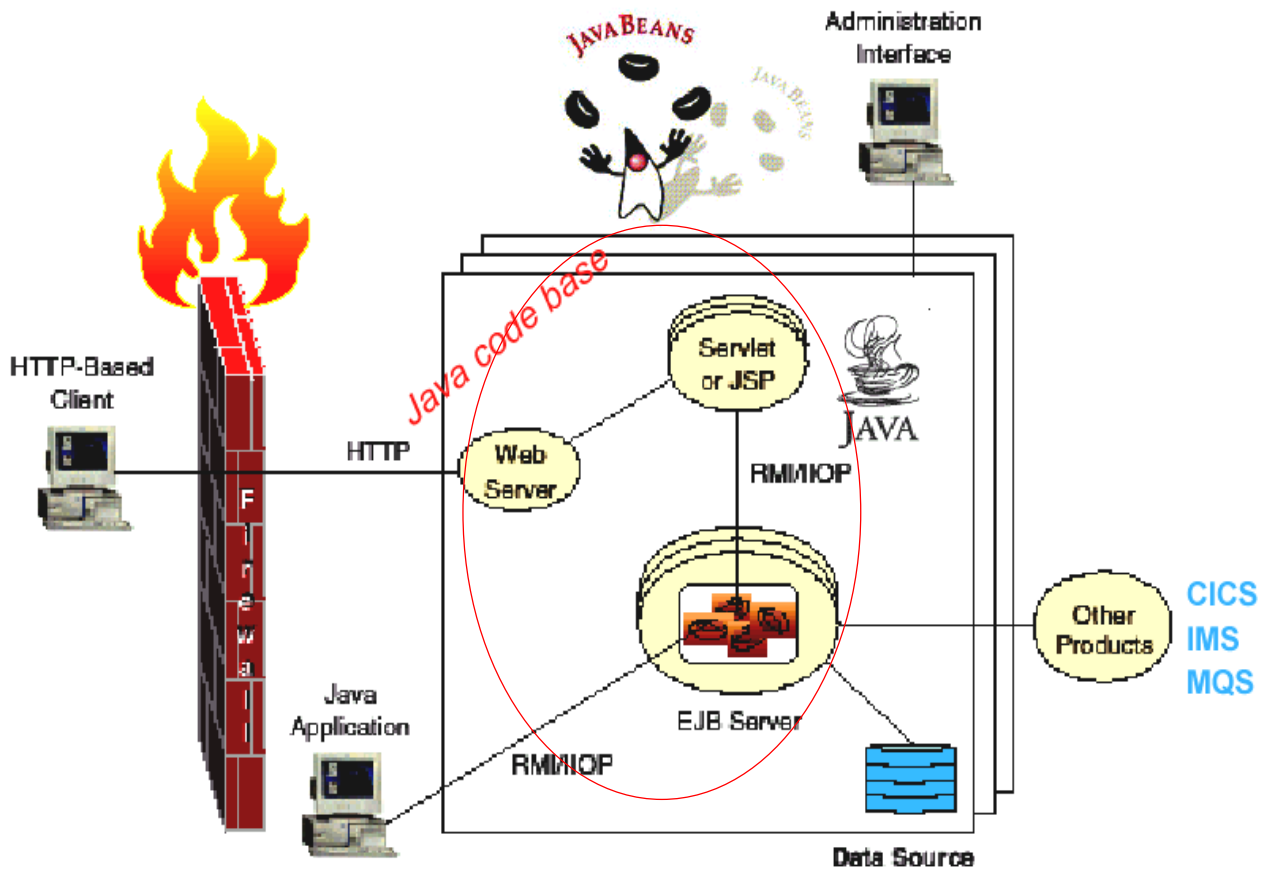
WAS has two main components:

- A plugin for the Web server (HTTP Server, Apache, ISS, ...)
- J2EE Application Server:
  - Java Virtual Machine (JVM)
  - Support for servlets, JSPs, EJBs, CORBA, ...
  - Support for DB2, CICS and IMS
  - Administrative tools: security, performance, scalability, recovery

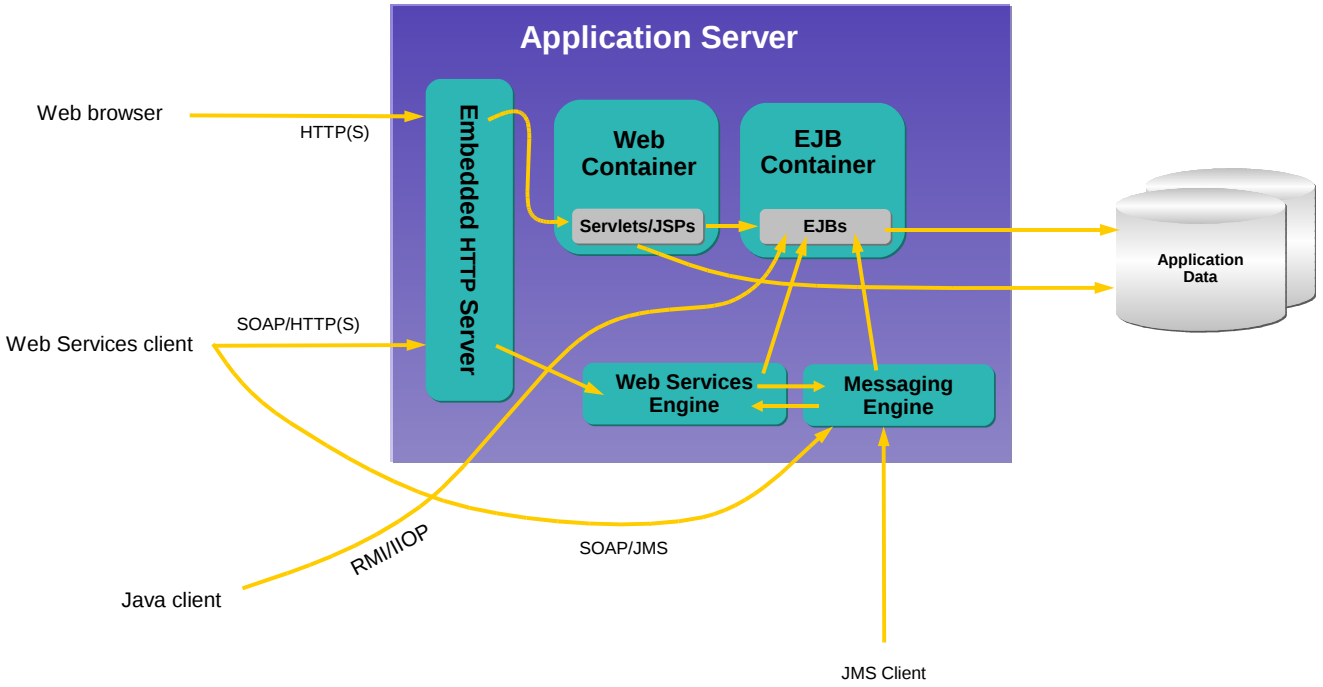
Basic Model for WAS



WAS in Context



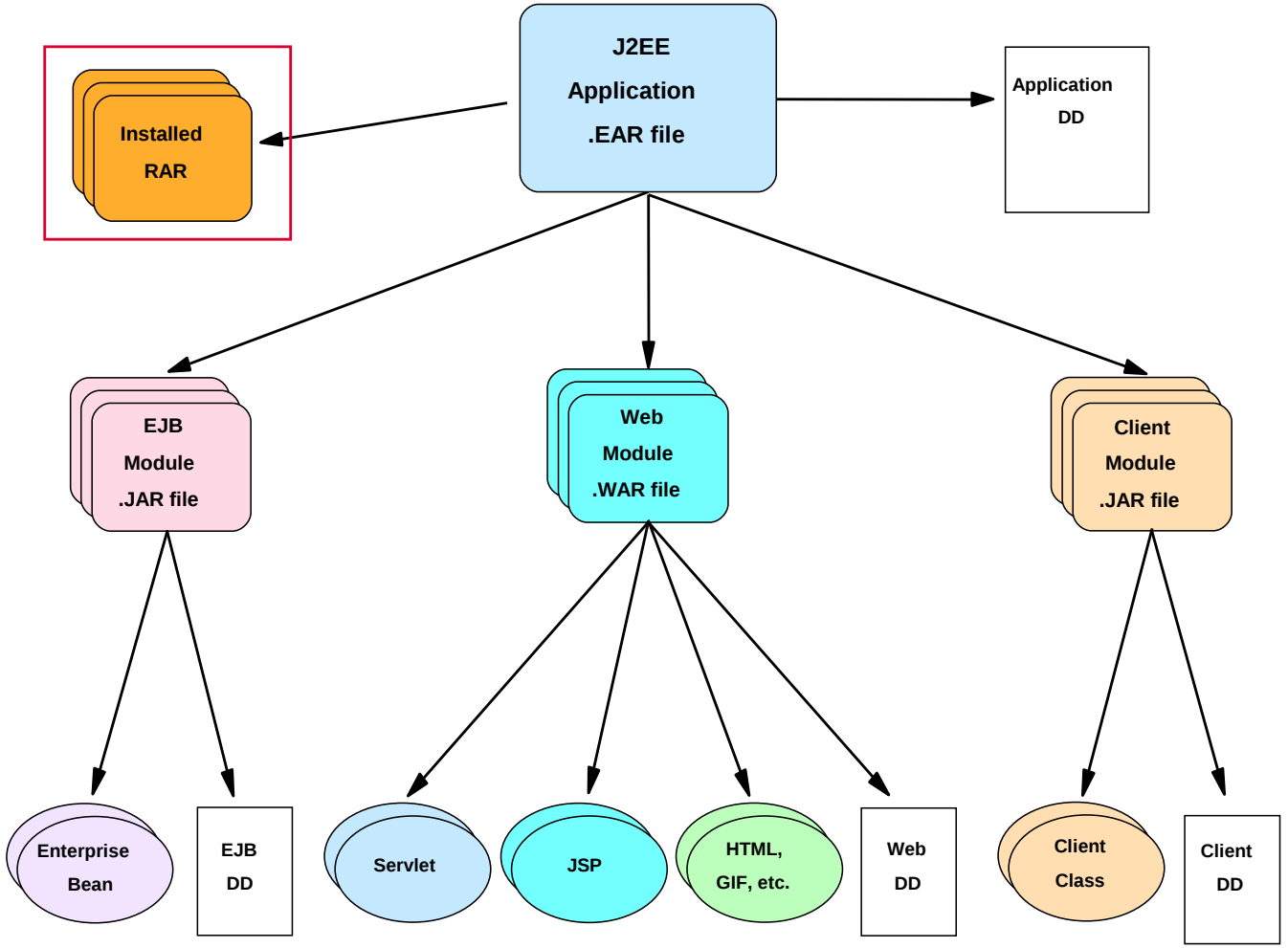
Accessing Server Resources



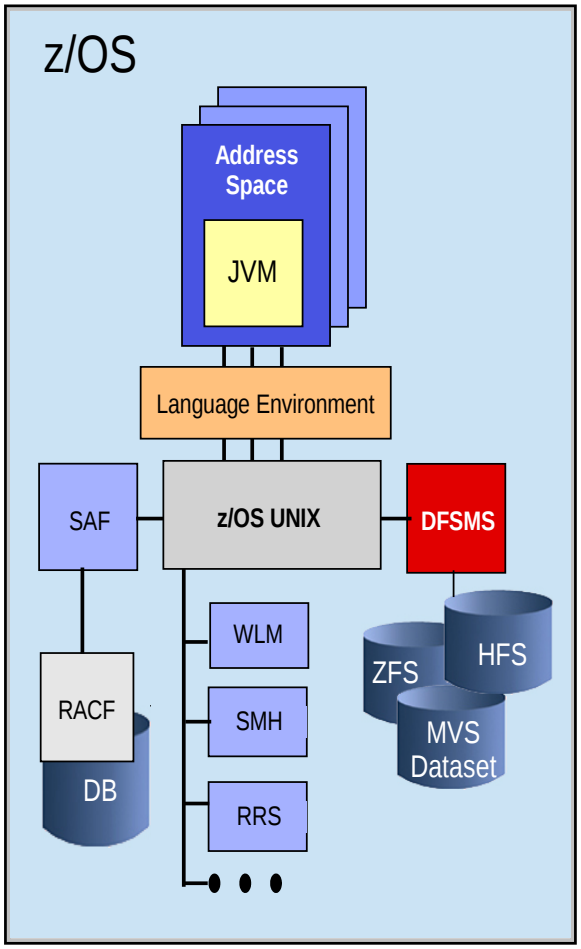
J2EE Application Model (*)

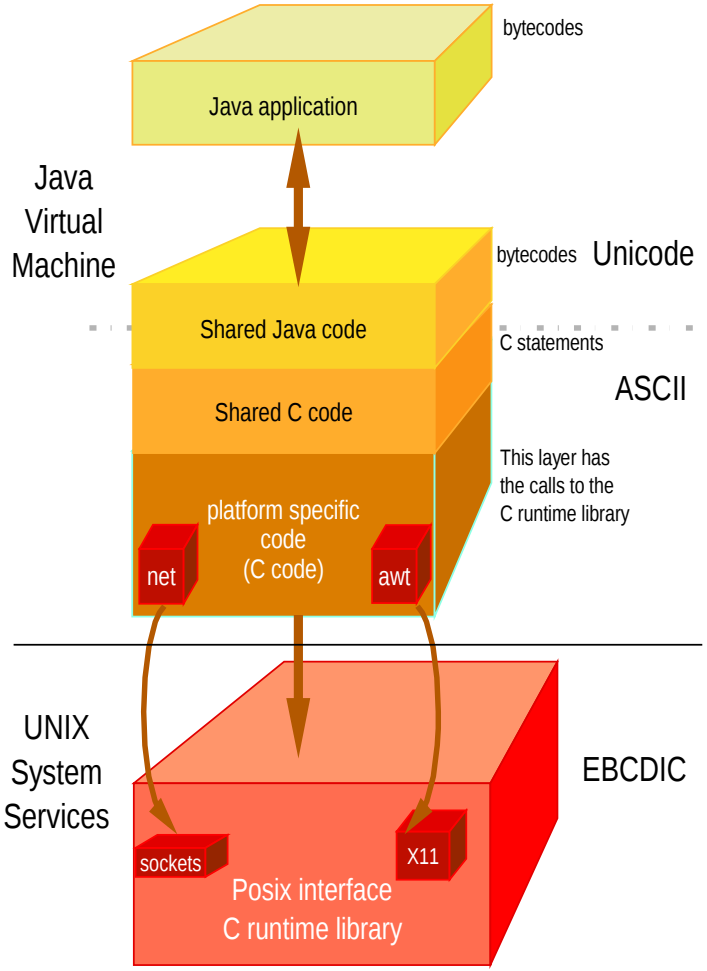
- Functional
- Reliable
- Usable
- Efficient
- Maintainable
- Portable

Enterprise Application Packaging

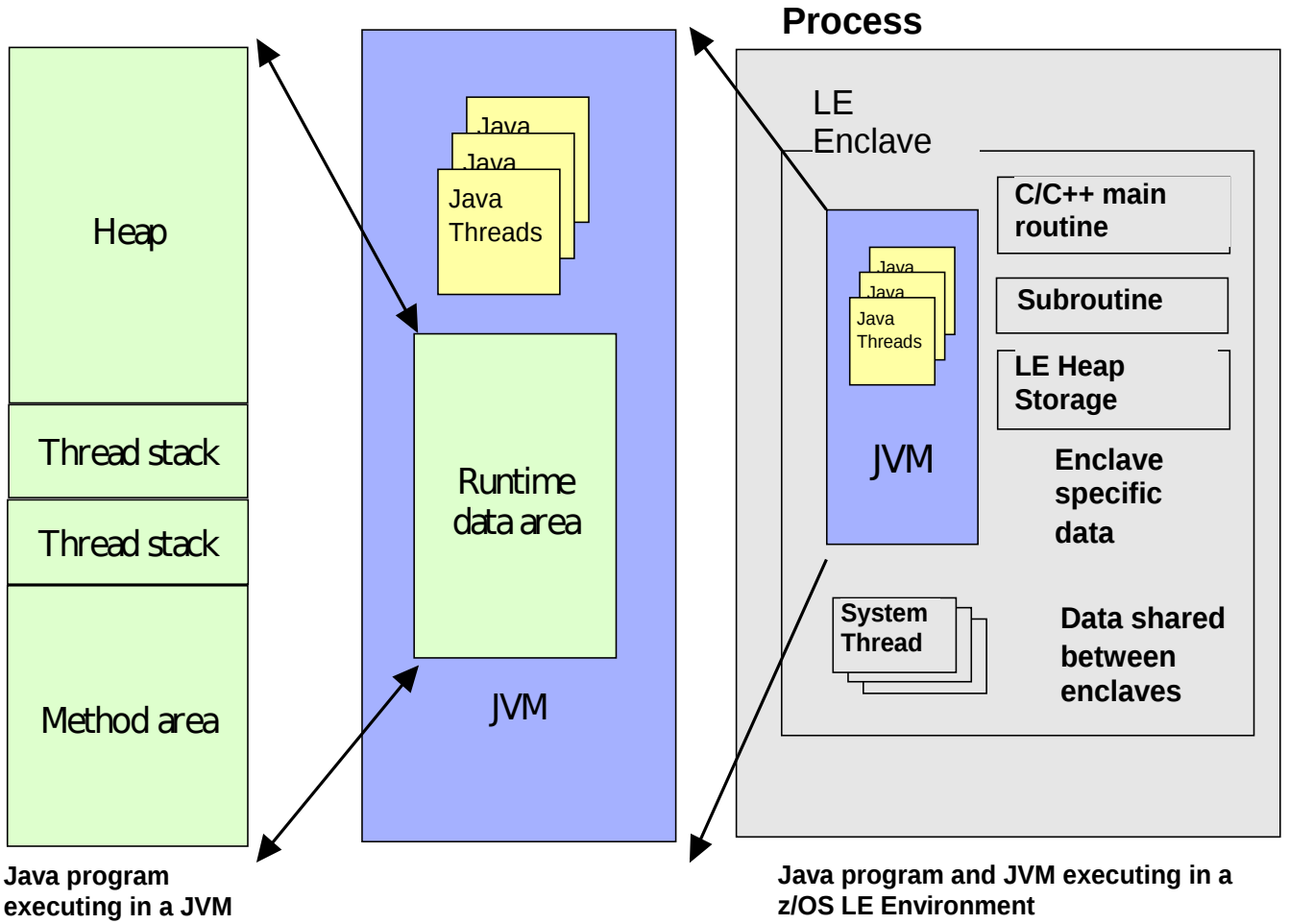


Java on z/OS

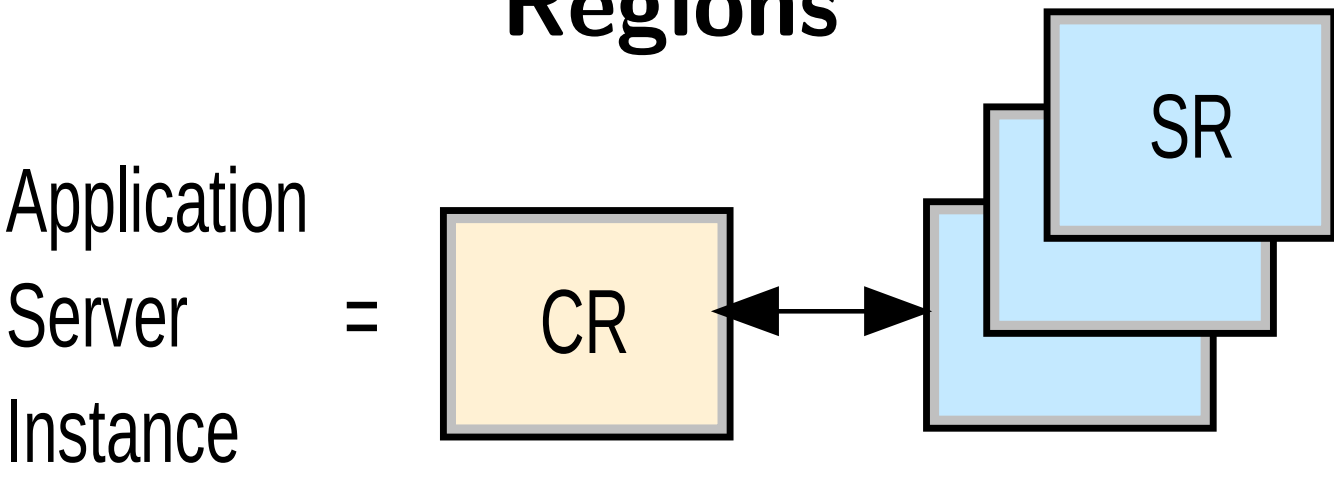

z/OS
Address Space
JVM
Language Environment
SAF
z/OS UNIX
DFSMS
WLM
SMH
RRS
RACF
DB
ZFS
HFS
MVS Dataset


bytecodes
Java application
Java Virtual Machine
bytecodes
Unicode
Shared Java code
C statements
ASCII
Shared C code
This layer has the calls to the C runtime library
platform specific code (C code)
net
awt
UNIX System Services
EBCDIC
sockets
X11
Posix interface
C runtime library

Java in the z/OS Language Environment



WAS Controller Region and Servant Regions



Controllers and Servants

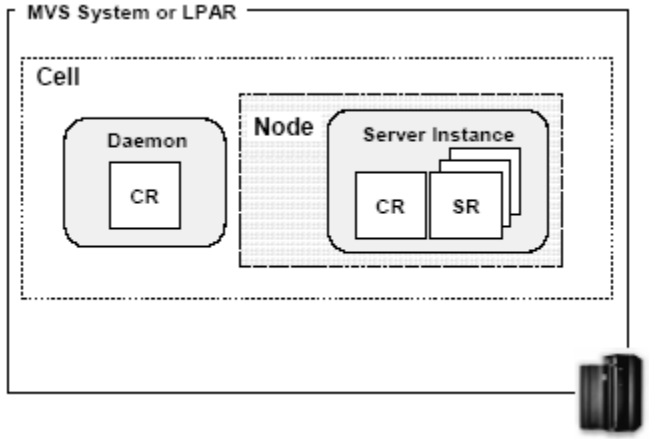
- The Address Space in which the JVM resides is called a "Servant"
- Multiple servants maybe started by WLM
- Work is queued by a "Controller" Region

WAS Terminology

- Servers – CR and multiple SRs
- Nodes – logical grouping of WebSphere-managed servers sharing a common configuration (can not span LPARs)
- Cells – a grouping of nodes into a single administrative domain (can span LPARs)
- Cluster – multiple copies of the same component (server)

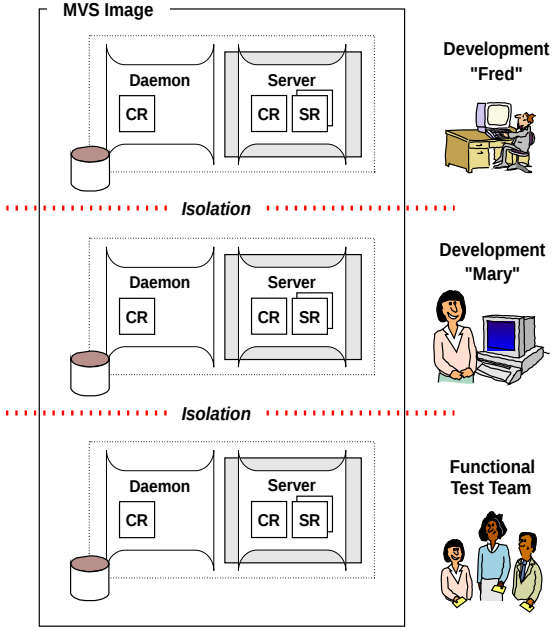
Base Application Server

A “standalone” server is a single server in a single node in a single cell:



Separate Environments (Cells) using Standalone Servers

You can create as many of these things as you like, and they can be 100% operationally and administratively isolated from one another:



Development "Fred"

Development "Mary"

Functional Test Team

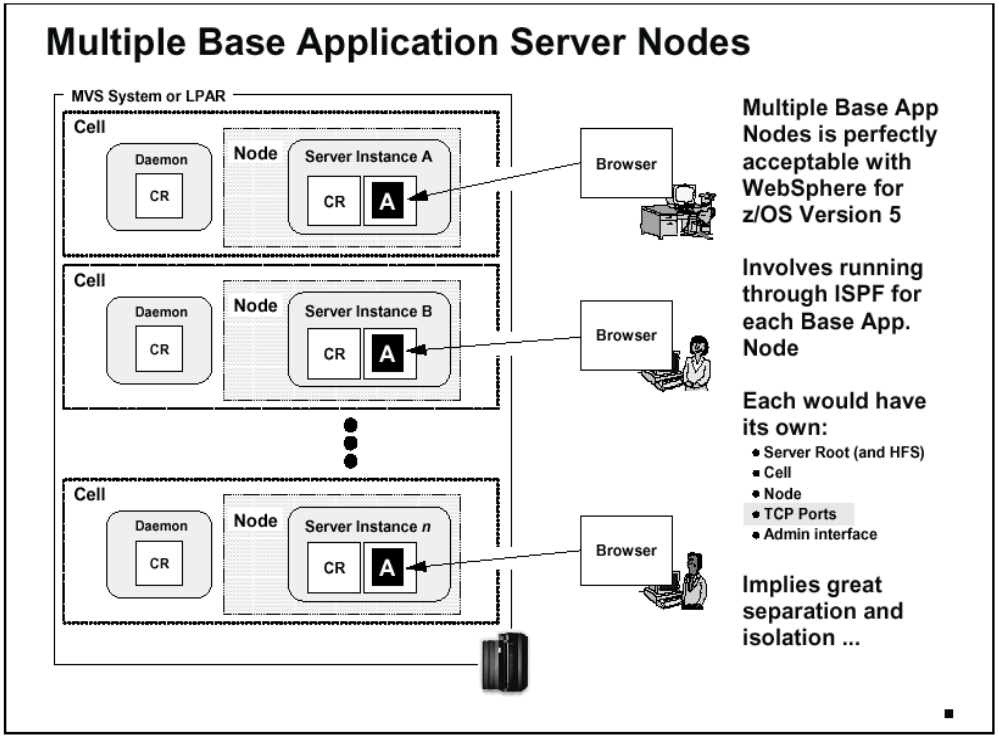
Even though they're on the same MVS image, their environments are separate from one another:

- **Separate Mount Point and HFS**
- **Separate JCL procedures**
- **Separate TCP ports**
- **Separate Admin Consoles**
- **Separate Userids / Passwords**
- **Start/Stop servers independently**

This is one of the primary benefits of creating Standalone Servers

There's another ...

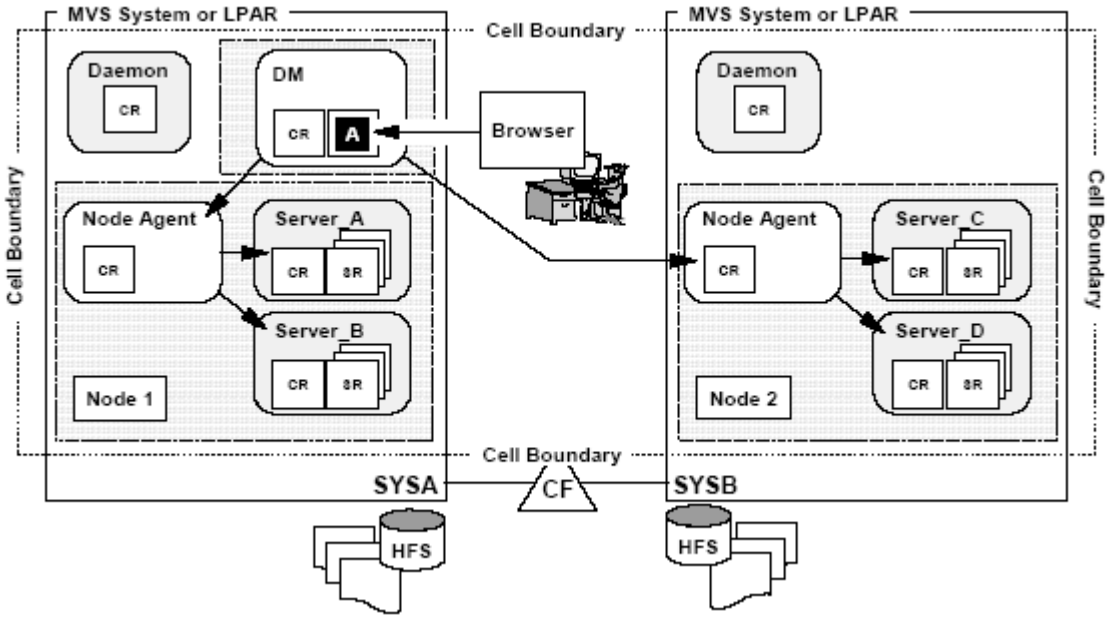
Multiple Base Application Server Nodes / Cells



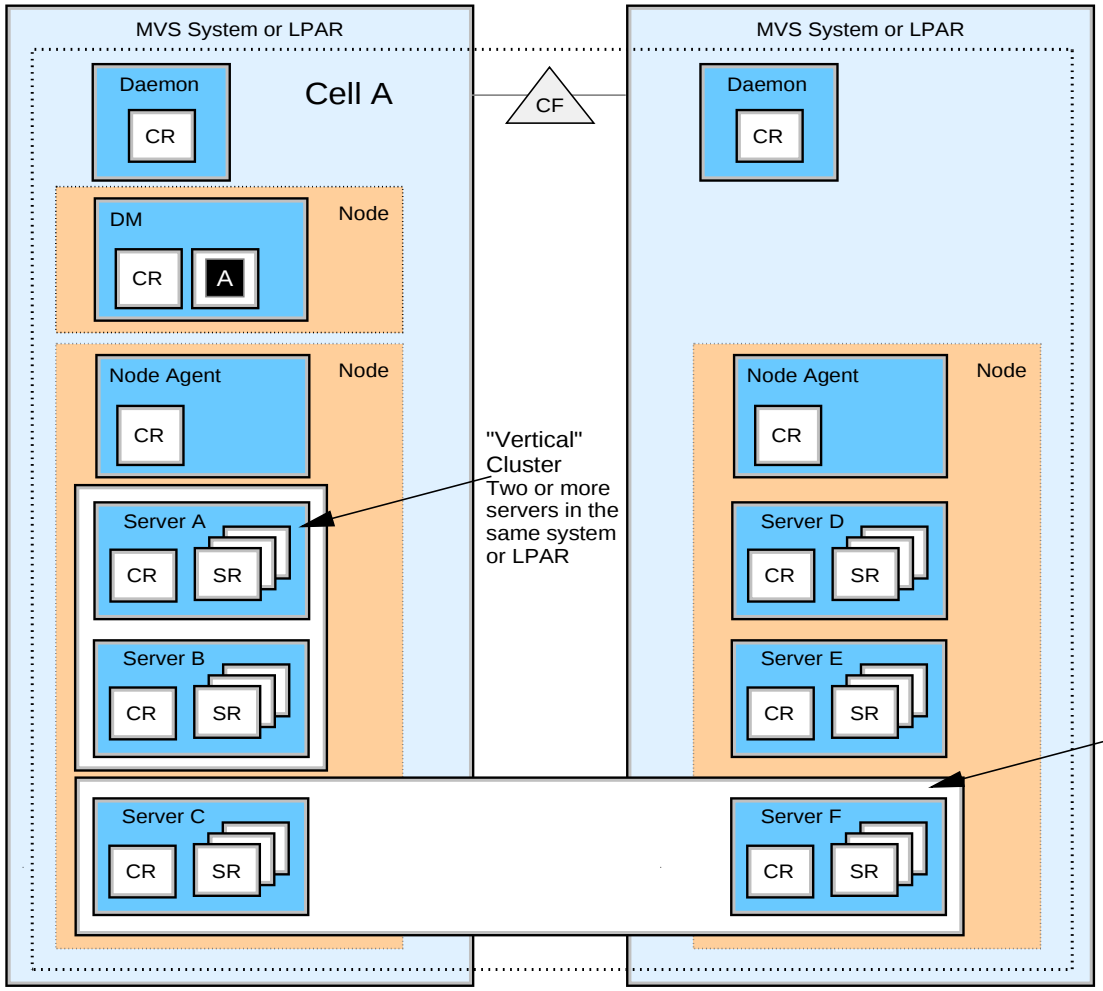
The Deployment Manager (DM)

- DM is a special kind of application server instance
- The administrative application runs in the DM
- DM manages application servers (CR/SR) grouped into *Nodes*
- The DM itself has its own node
- All other nodes have a *Node Agent* (with a CR)

WAS Network Deployment Overview



Clusters



Servers are clustered through the administrative interface.

Any given server may be a member of only one cluster at a time

- You can not have Server_C be a member of two different clusters, for example.

Hybrid of vertical and horizontal is permitted

Rules

- "Standalone" Servers need no DM
- Nodes never span LPARs
- DM needed to *administer* network deployment; nodes can continue to operate even if DM crashes!
- One DM per cell that uses network deployment

WAS and WLM

WebSphere uses three distinct functions of WLM:

- Routing – which server is best able to complete the work?
- Prioritizing – manage resources based on service level objectives
- Queuing – delay work until it can be processed

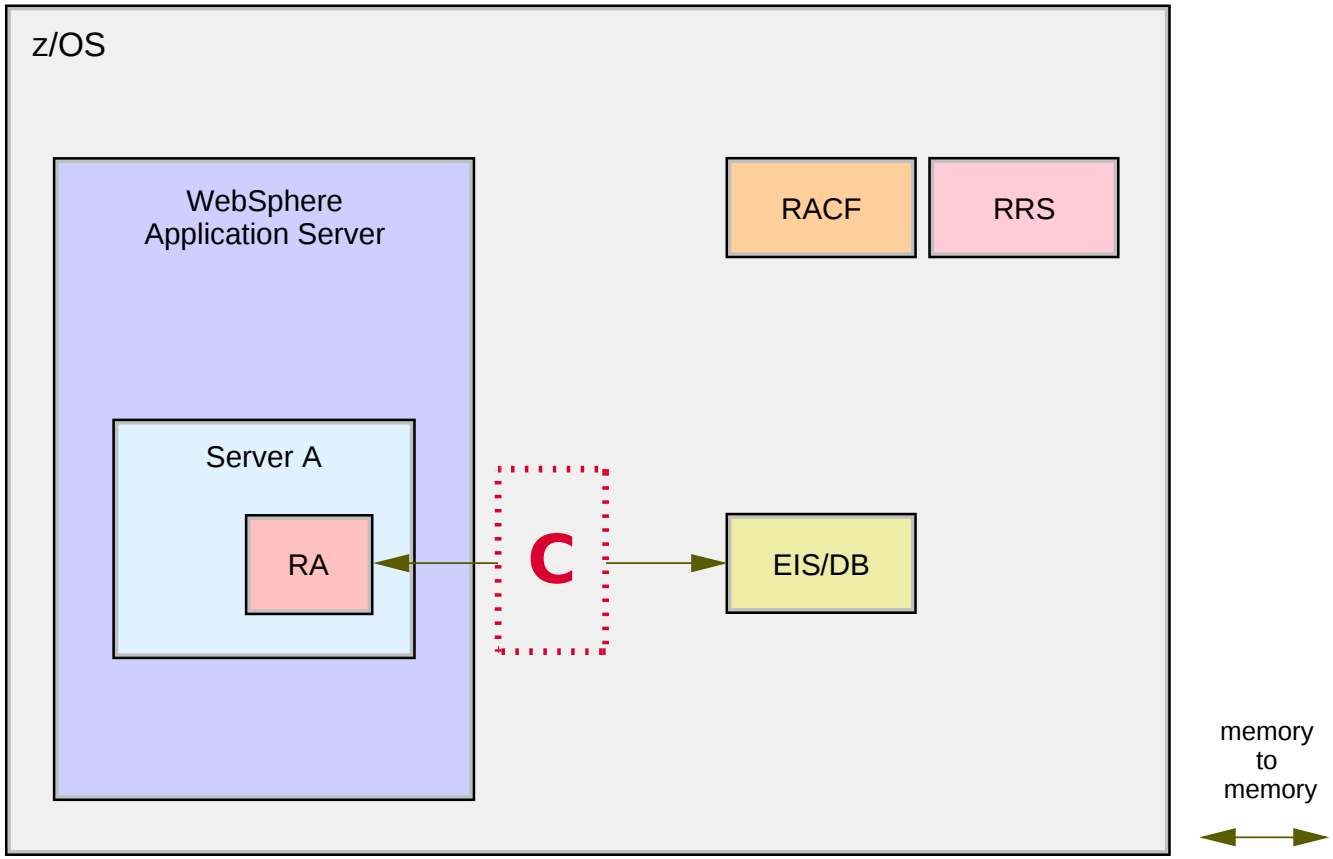
What are Connectors?

Objectives:

- Make the communication protocol / mechanism between client application and subsystem transparent to the application developer
- Provide portability of the client application
- Provide structured API and development tools for developers using connectors (J2EE Connector Architecture, J2CA)

Connectors may support remote operation (i.e. via TCP/IP) or use some form of optimized local communication.

Connectors and Resource Adapters



C=connector
RA=resource adapter

Local vs. Remote Connectors

	Local Connection	Remote Connection
Performance	No network overhead	network overhead
Availability	One LPAR	Multiple LPARs and network involved
Scalability	Duplicate all components on another LPAR	Components can be distributed across LPARs, allowing better workload balancing
Security	Thread identity can be used; no risky network connections	No thread identities; network connections require more security measures
Transactionality	2-phase commit supported	2-phase commit not always possible

Mainframe Connectors (*)

WAS supports the following connectors:

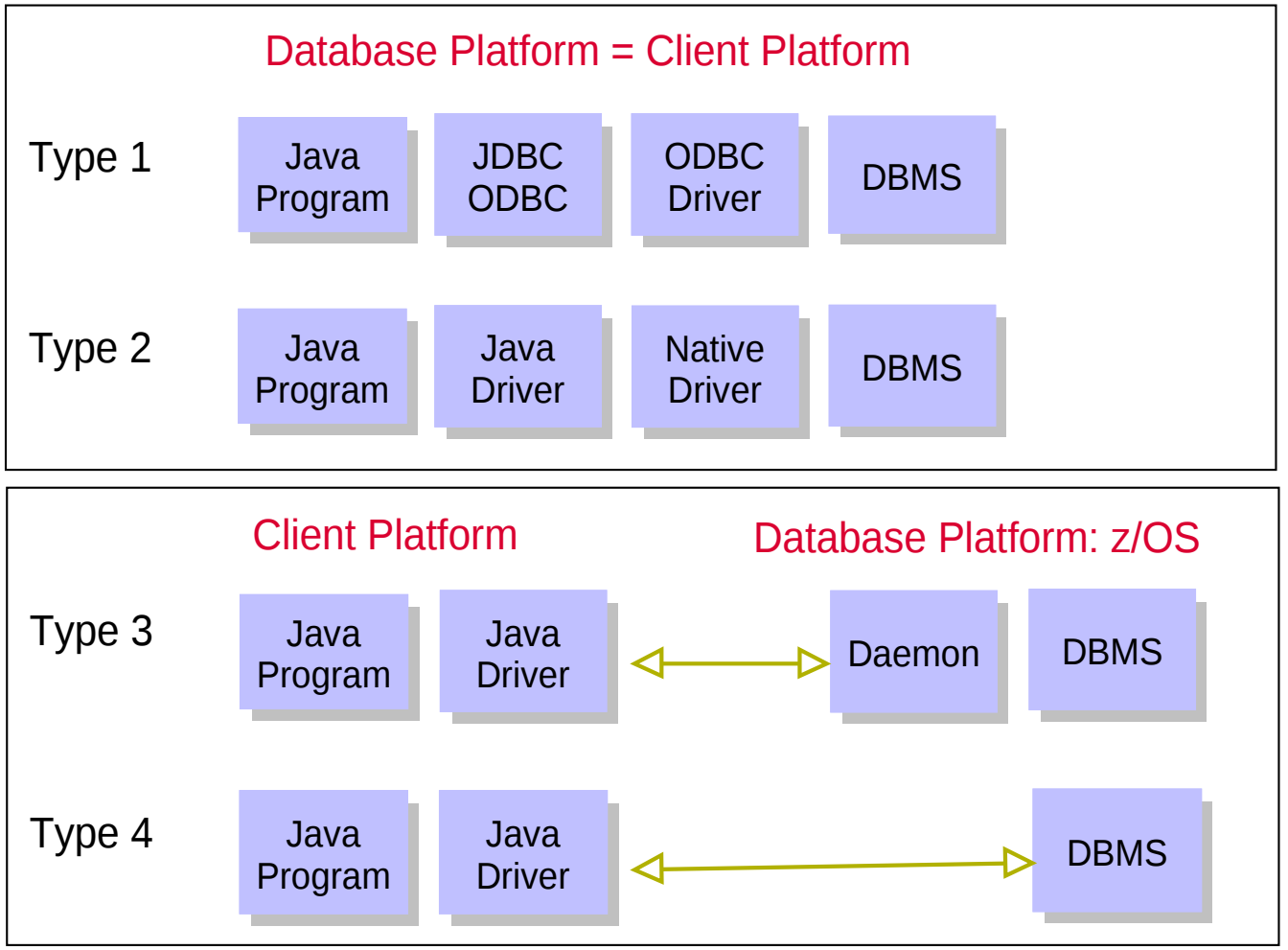
- CICS Transaction Gateway
- IMS Connect
- DB2 JDBC

Using J2CA to access CICS

Requires CICS Transaction Gateway; Version 6.1 supports:

- Local connections (only CTG libraries need to be present)
- Remote connections (requires CTG daemon)
- 2-phase commit supported for both local and remote mode

JDBC Driver Types



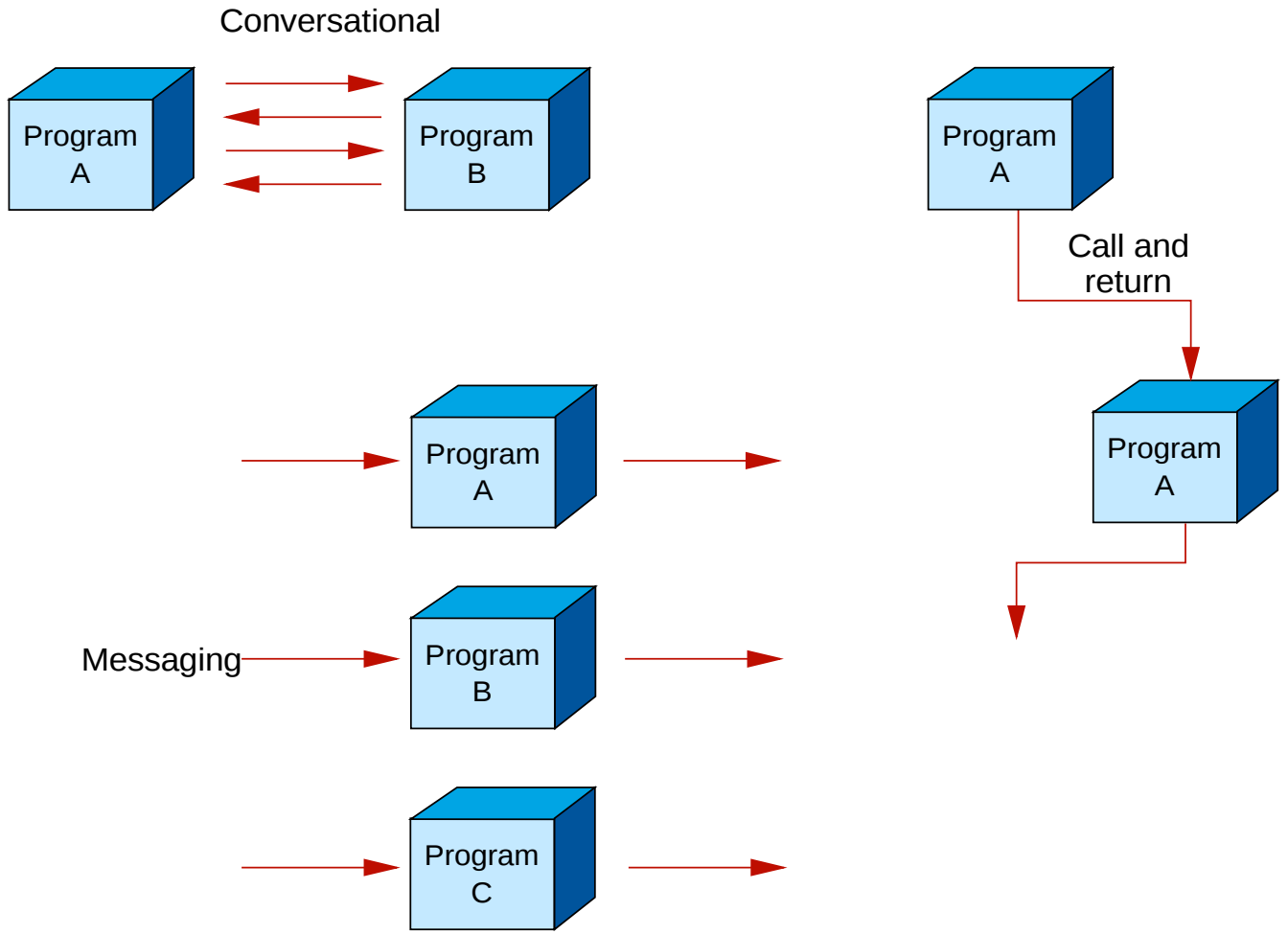
Why run WAS on z/OS? (*)

- Consolidation of workloads
- Continuous availability
- Performance
- z/OS Security
- Integration with legacy applications

Questions

?

Styles of Communication



Messaging

The principal objective of messaging is to exchange information:

- in an asynchronous way
- offering assured delivery
- offering failure independence of applications
- offering a triggering mechanism on the receipt of messages
- providing message persistence, transactional integrity and security

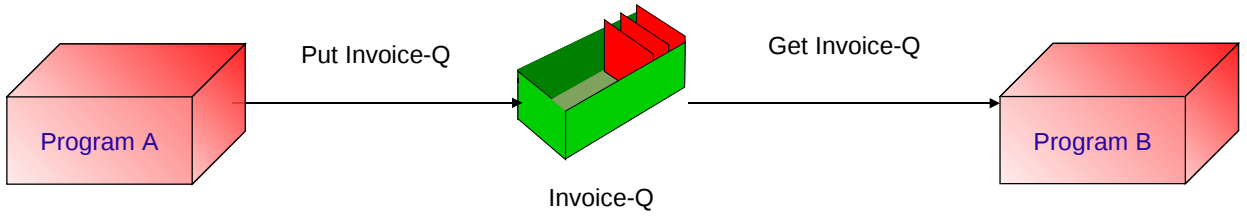
WebSphere MQ

- Implements messaging and queueing
- Provides a common set of APIs independent of platform or network complexity
- WebSphere MQ is available on over 40 platforms
- Enables a loosely-coupled form of application integration
- Enables flexible placement of business logic

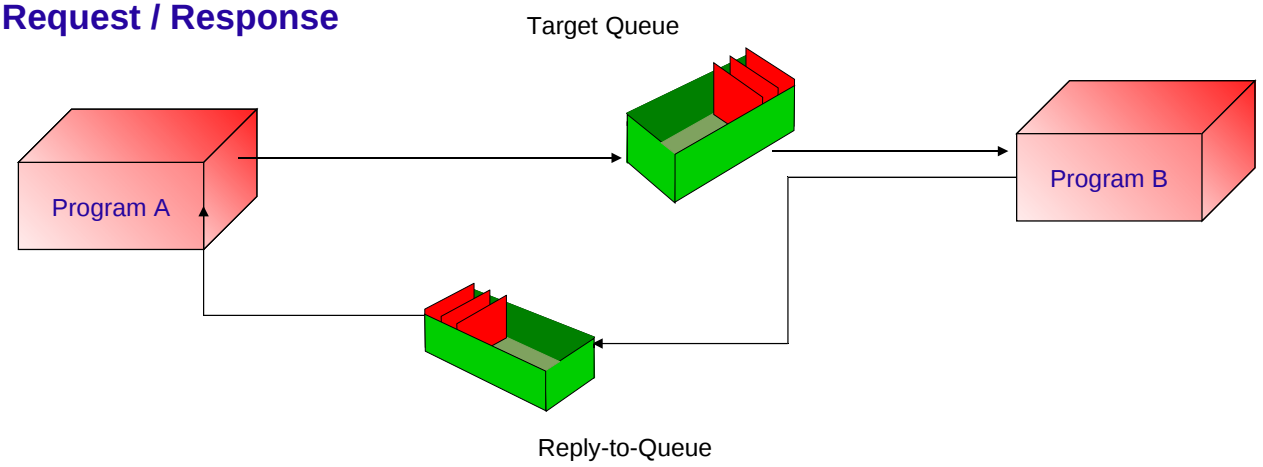
$\Rightarrow$ suited for a Service Oriented Architecture

Examples

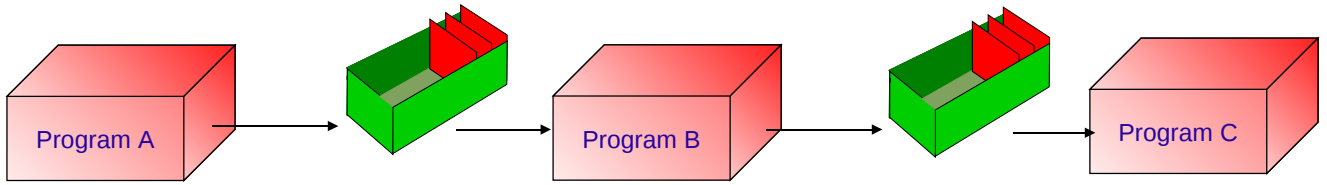
'Send and Forget'

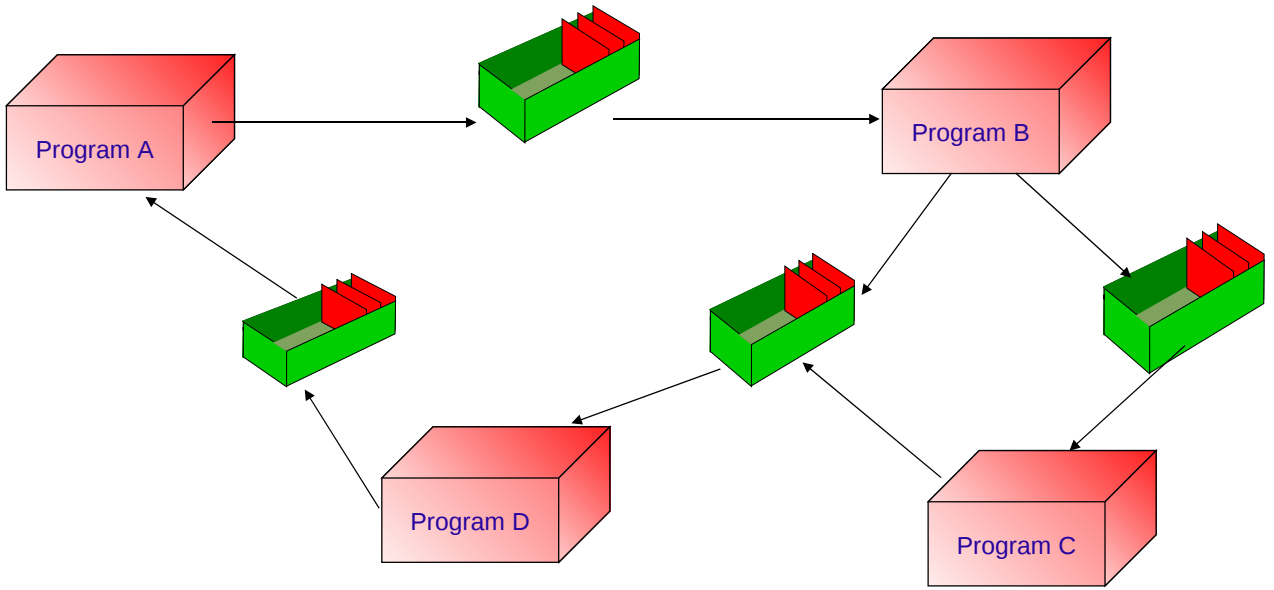


Request / Response

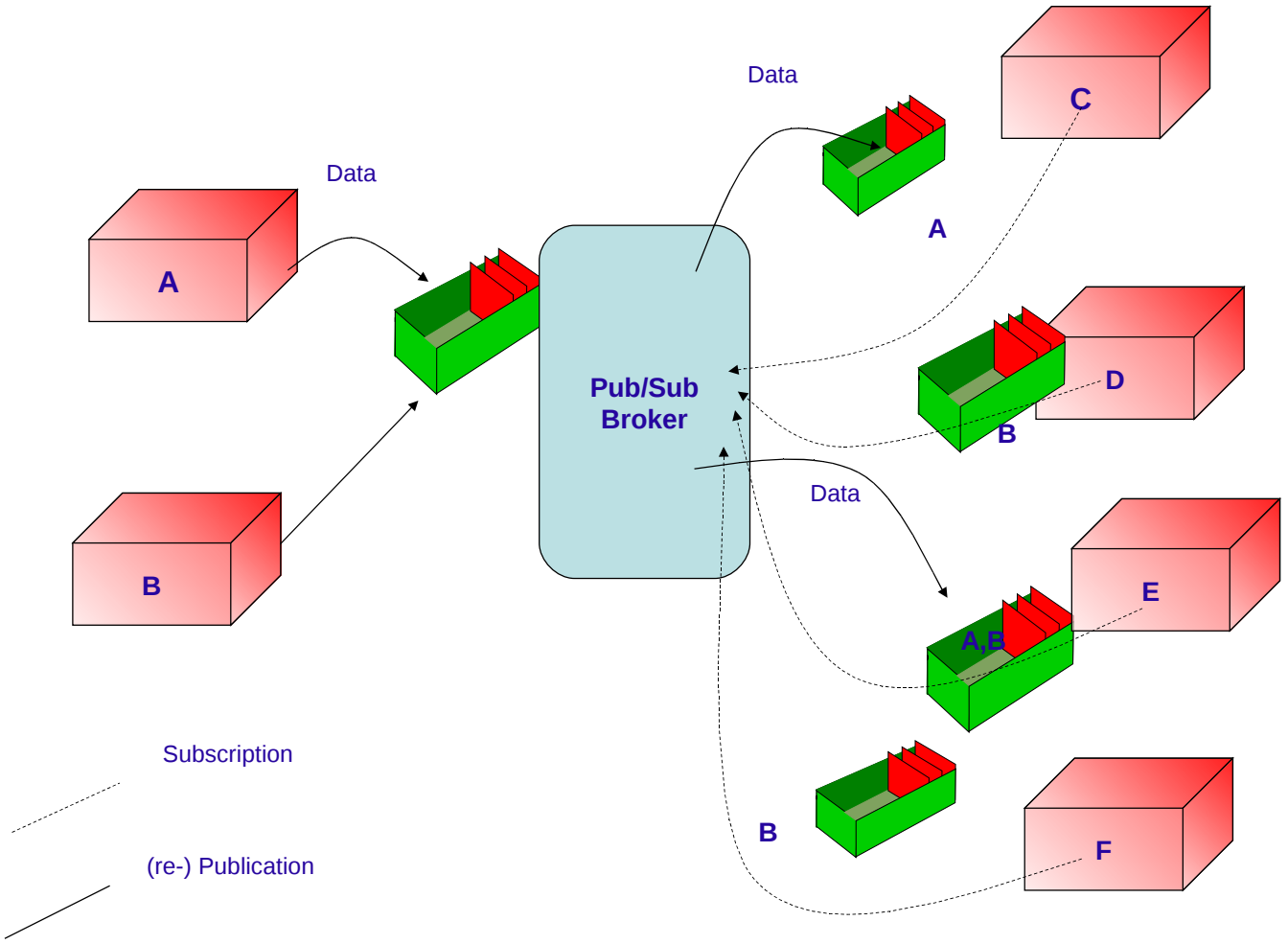


Examples

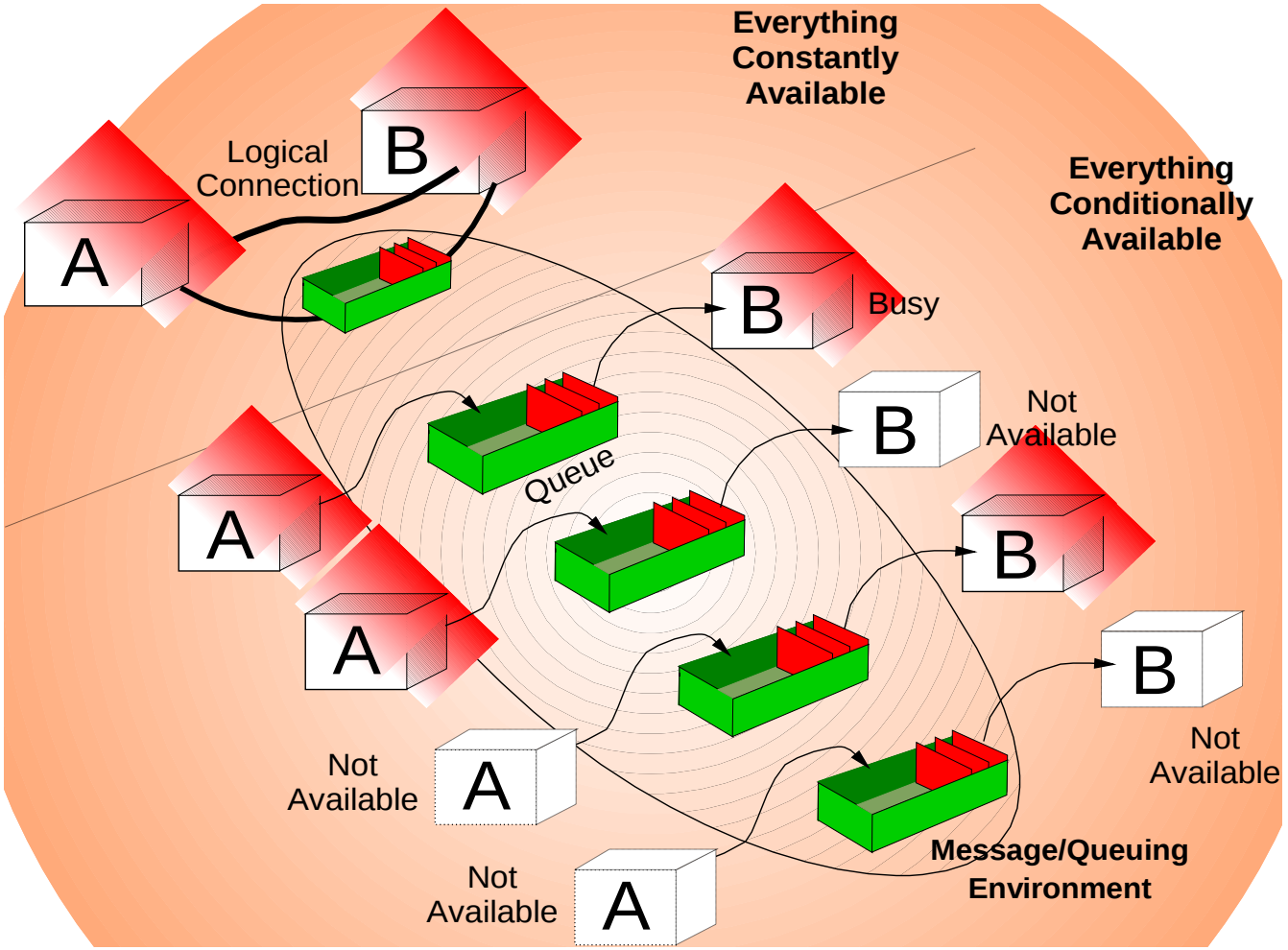

Program A
Program B
Program C


Program A
Program B
Program D
Program C

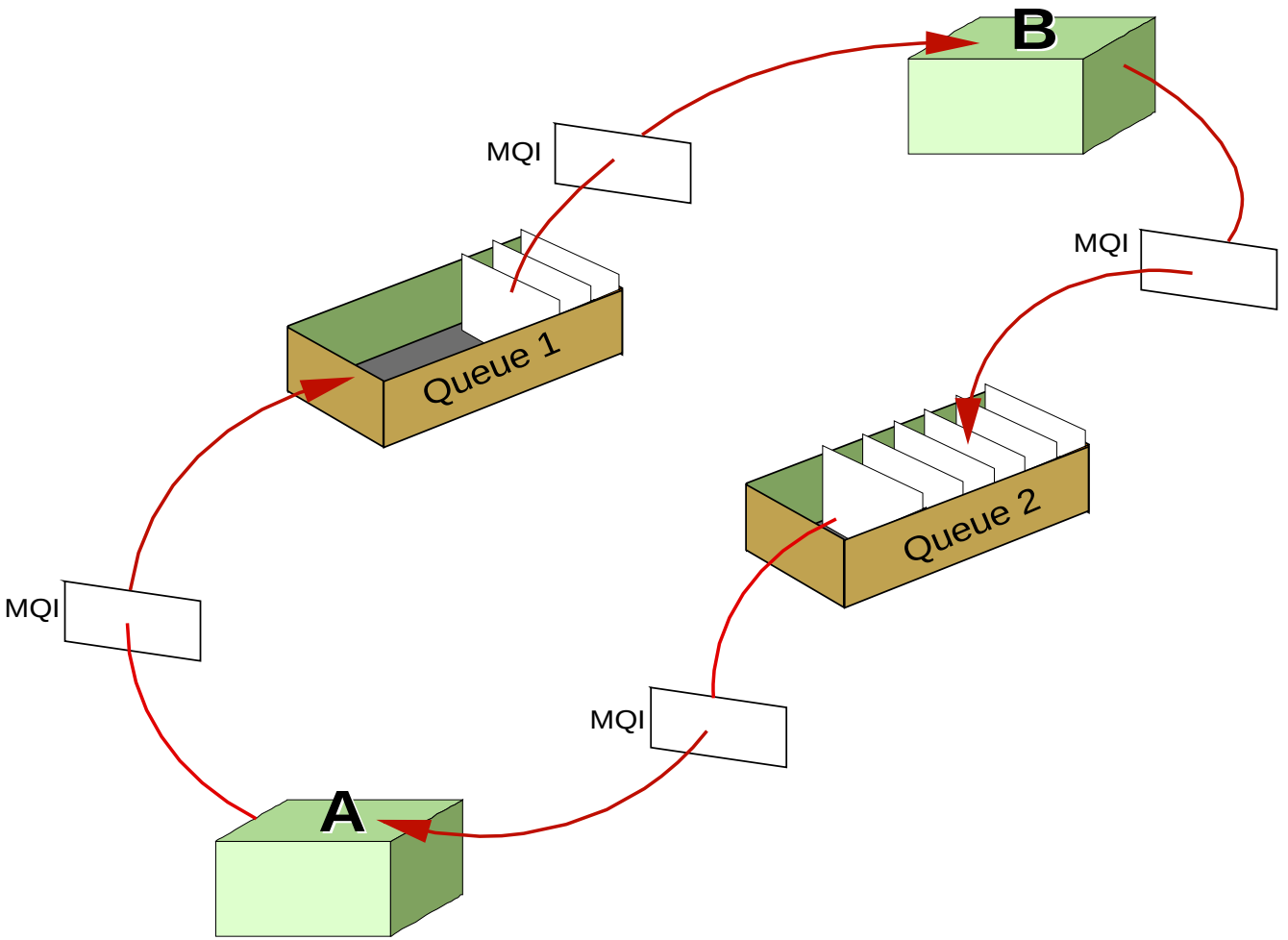
Publish/Subscribe



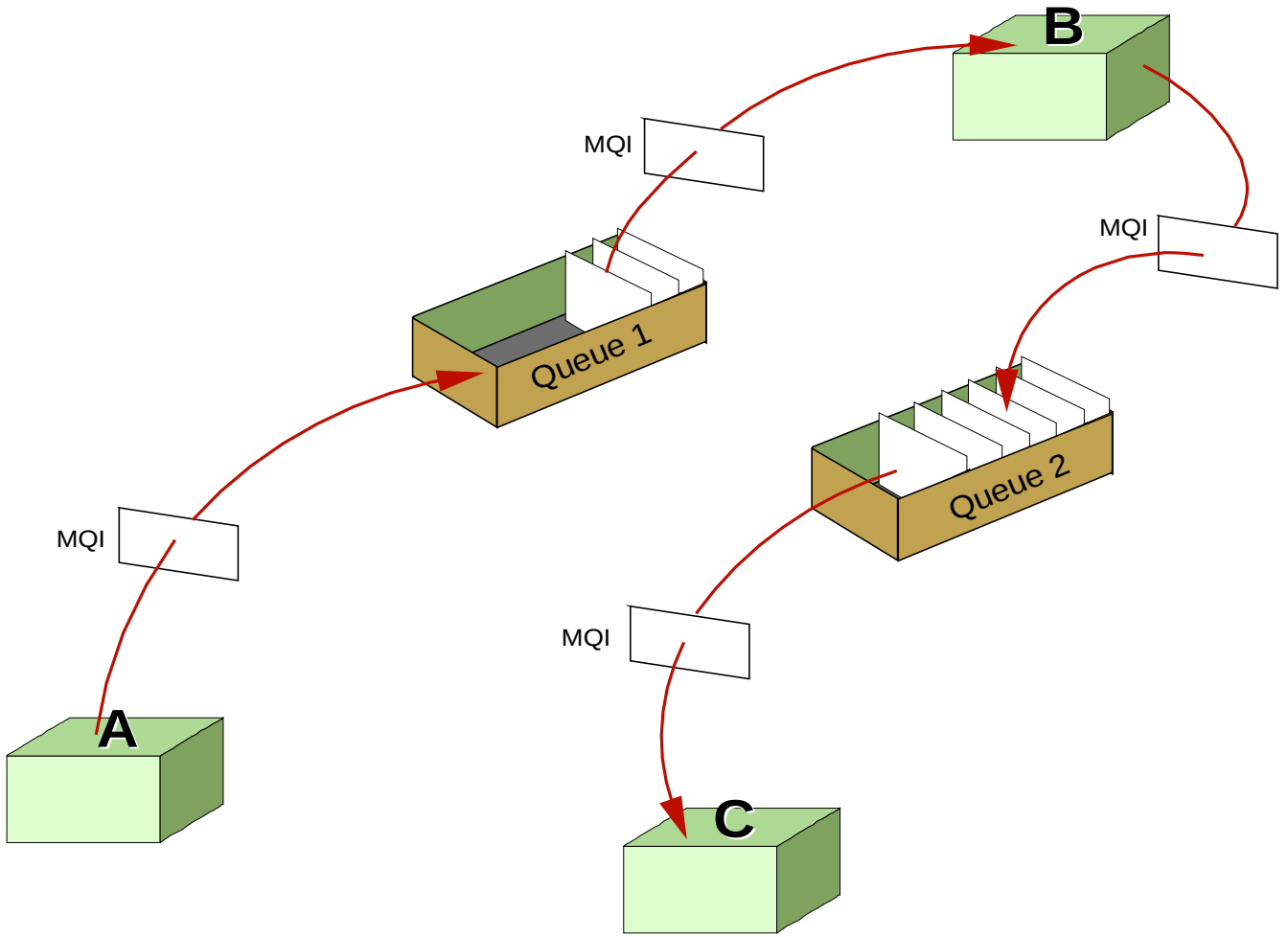
Time Independence



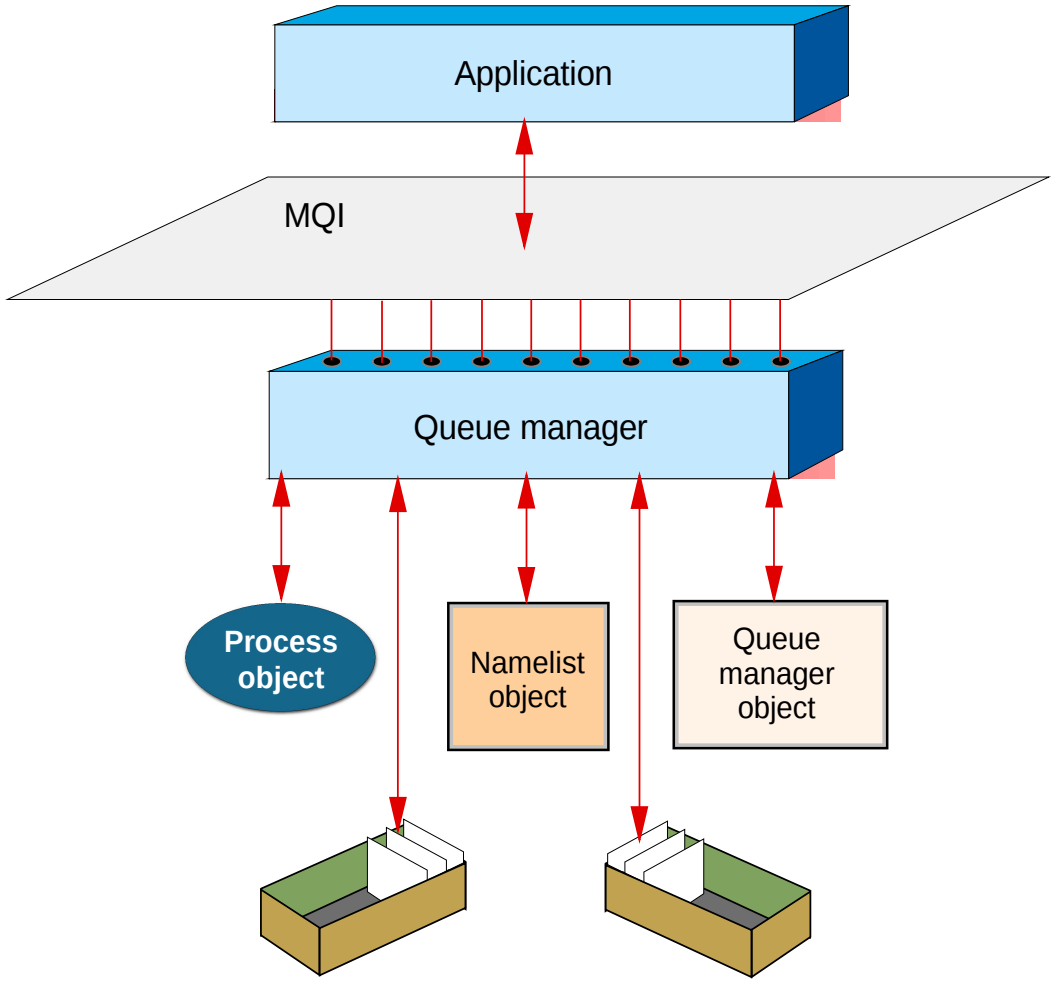
Synchronous Communication Model



Asynchronous Communication Model

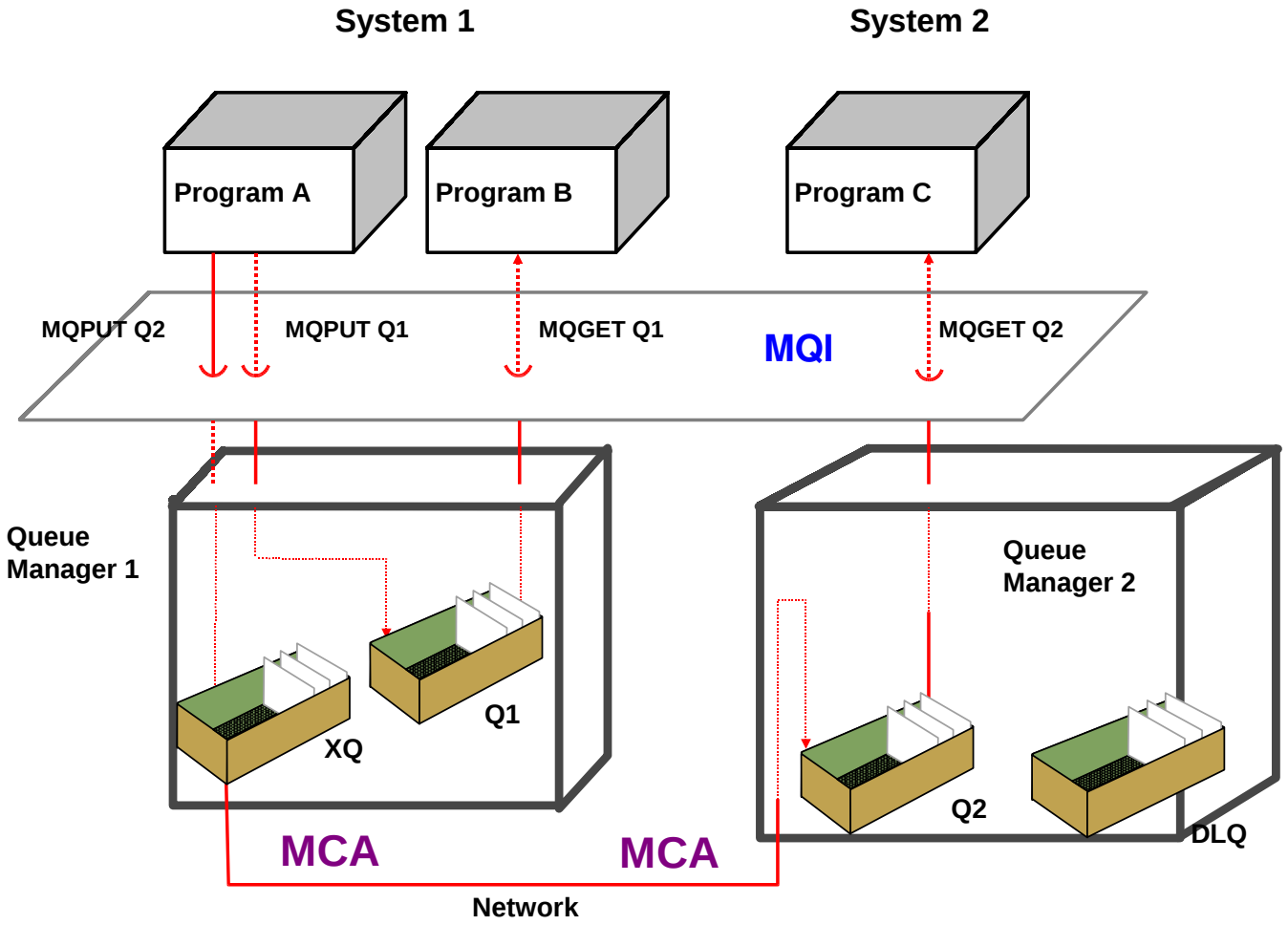


The Message Queue Interface (MQI)

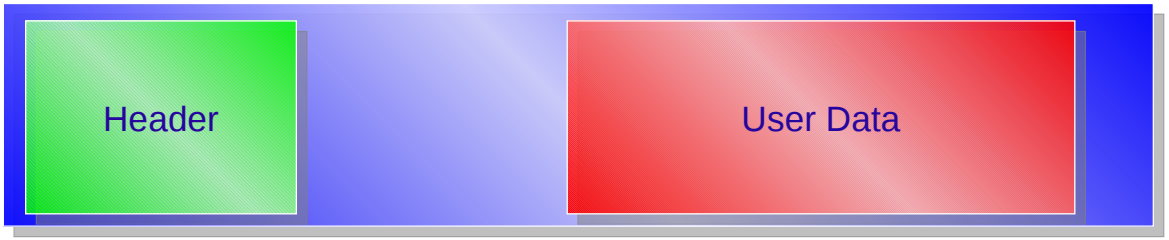


- Major calls
  - MQCONN
  - MQCONNX
  - MQOPEN
  - MQCLOSE
  - MQPUT
  - MQPUT1
  - MQGET
- Minor calls
  - MQBEGIN
  - MQCMIT
  - MQBACK
  - MQINQ
  - MQSET

Transparency of Distributed Queueing



What is a Message?



A Series of Message Attributes
Understood and augmented by the Queue Manager
•Message Id
•Correlation Id
•Unique Routing information
•Reply routing information
•Message priority
•Message codepage/encoding
•Message format
....etc.

•Any sequence of bytes
•Private to the sending and receiving programs
•Not meaningful to the Queue Manager

- •Message Types
  - Persistent ... recoverable
  - Non Persistent
- •Up to 100MB message length

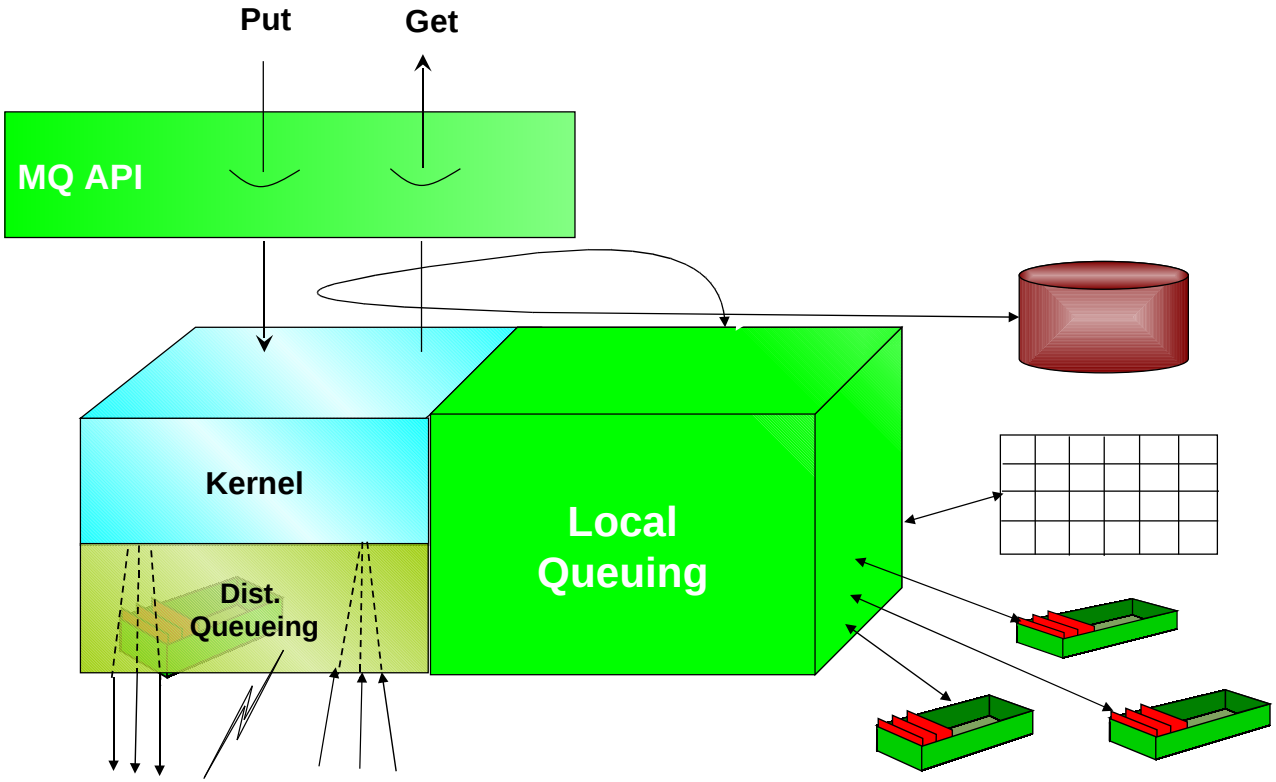
Types of Messages

- Datagram (send and forget)
- Request (send, wait for reply)
- Reply (response to request)
- Report (status, failure, etc.)

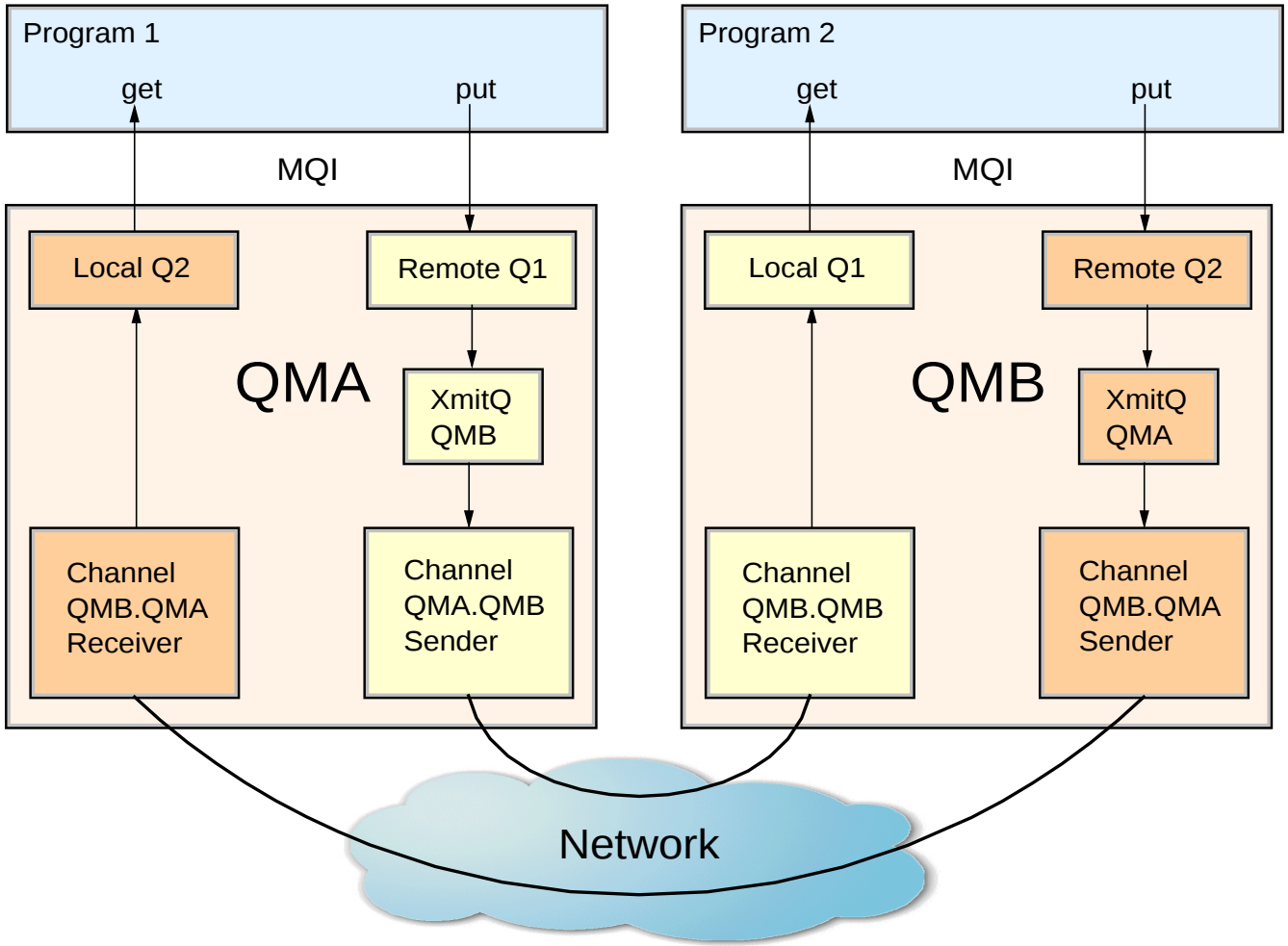
What is a Queue?

- Place to hold messages
- Different Queue Types: Local, Alias, Remote, Dead-letter
- Can be predefined or dynamically defined
- Access can be FIFO, Priority, Direct, Destructive or non-Destructive, ...
- Parallel access is managed by the *queue manager*

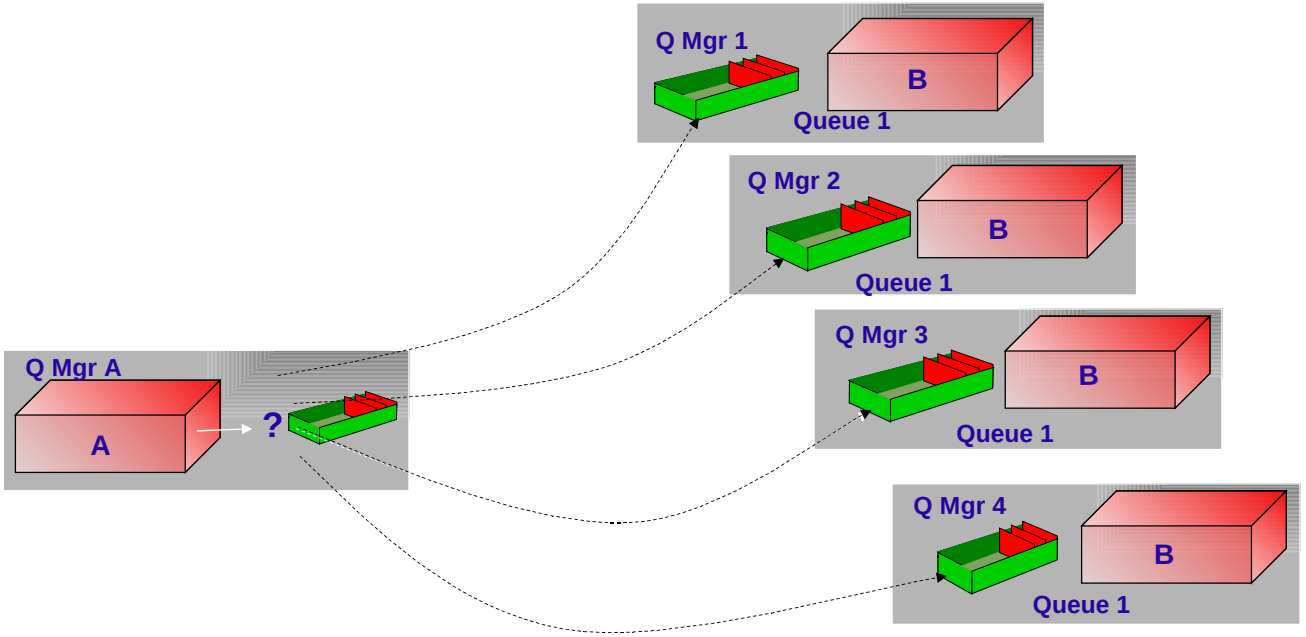
The Queue Manager



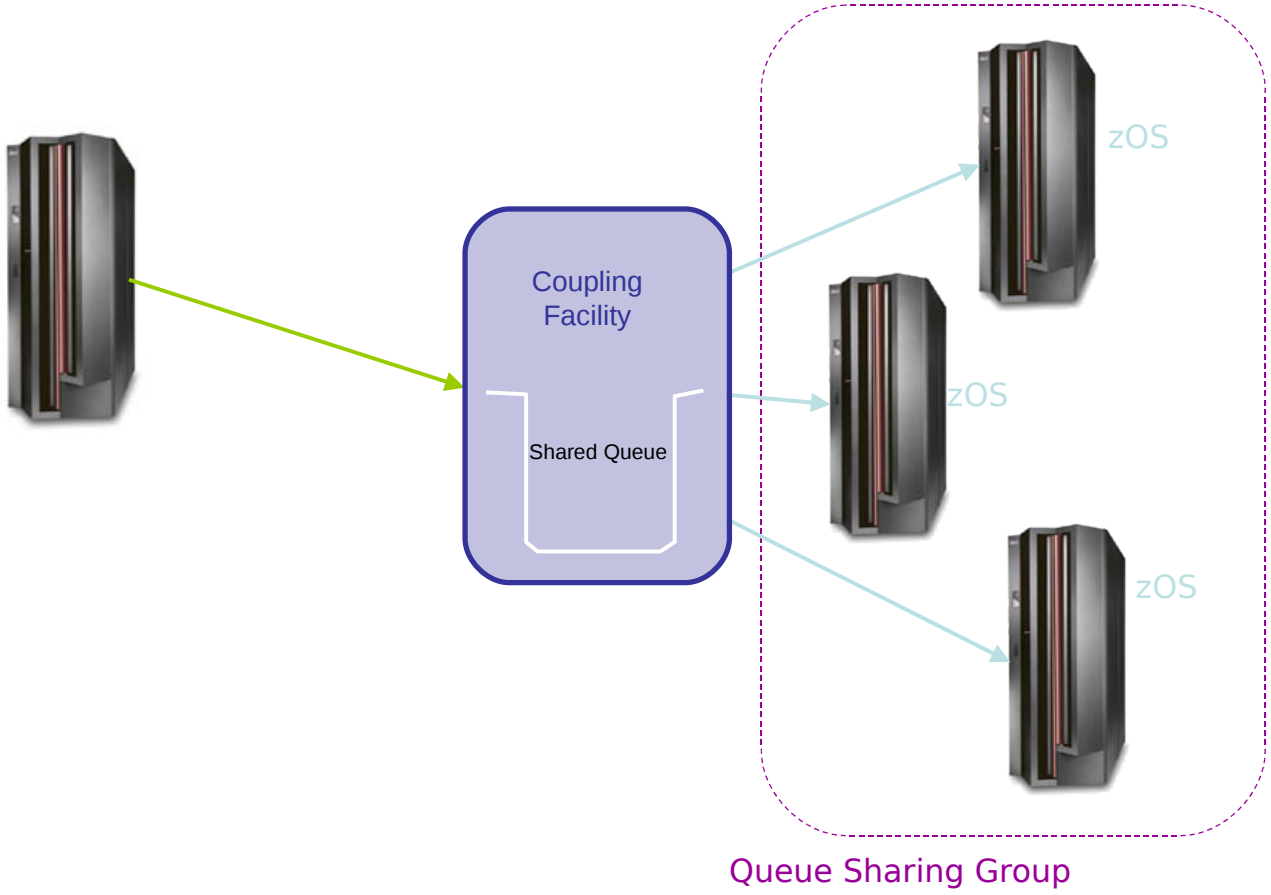
MQ Channels


Program 1
get
put
MQI
Local Q2
Remote Q1
QMA
XmitQ
QMB
Channel
QMB.QMA
Receiver
Channel
QMA.QMB
Sender
Program 2
get
put
MQI
Local Q1
Remote Q2
QMB
XmitQ
QMA
Channel
QMB.QMB
Receiver
Channel
QMB.QMA
Sender
Network

MQ Clustering for Improved Availability



MQ Clustering using Coupling Facility



Shared Queues and Server Failure

- Failure isolated to failed entity
- Automatic peer recovery for failing queue managers:
  - In-flight MQPUTs and MQGETs are rolled back
  - No marooned messages!

Security

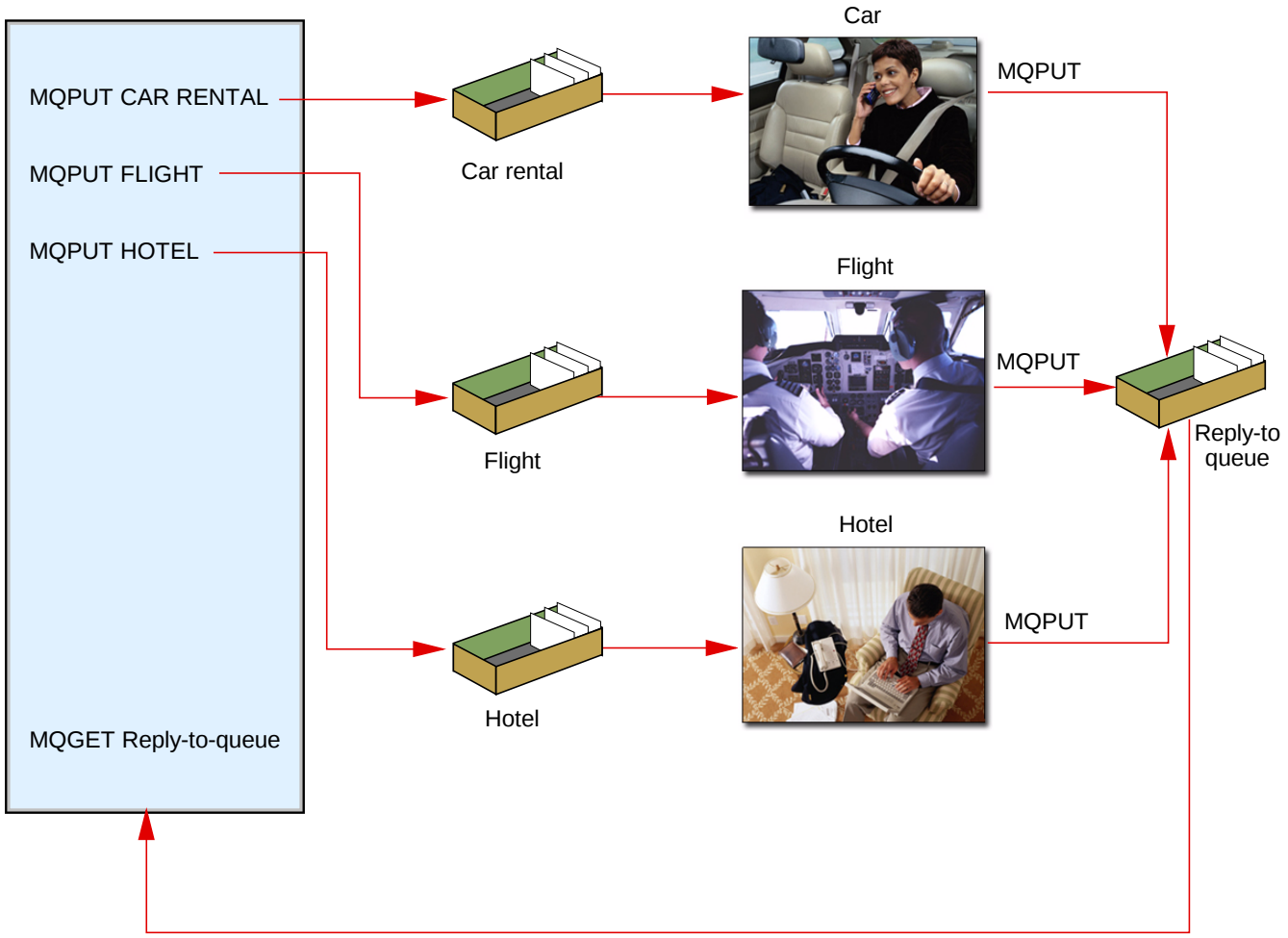
- MQ provides access control to its objects (queues, commands, messages)
- MQ provides channel security using SSL
- On z/OS, MQ uses SAF to map MQ security to native platform security (such as RACF)

$\Rightarrow$ RACF can map an SSL certificate to a userid

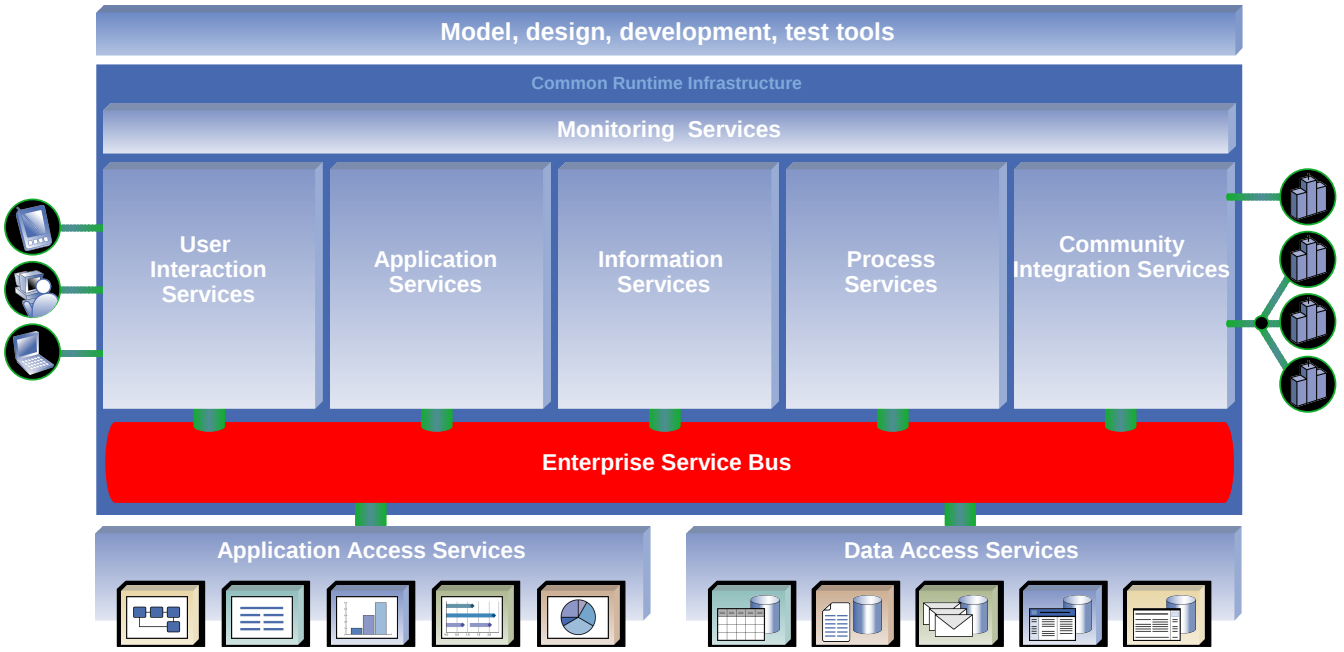
Transactional Support

- MQBEGIN, MQCMIT and MQBACK control transactions
- Messages and other transactional resources can be:
  - Managed by a Transaction Manager (WAS, CICS, IMS, ...)
  - Managed by WebSphere MQ (is also a TM)

Example


MQPUT CAR RENTAL
MQPUT FLIGHT
MQPUT HOTEL
MQGET Reply-to-queue
Car rental
Flight
Hotel
Car
Flight
Hotel
MQPUT
MQPUT
MQPUT
Reply-to queue

MQ and the "Big Picture"



Questions

?