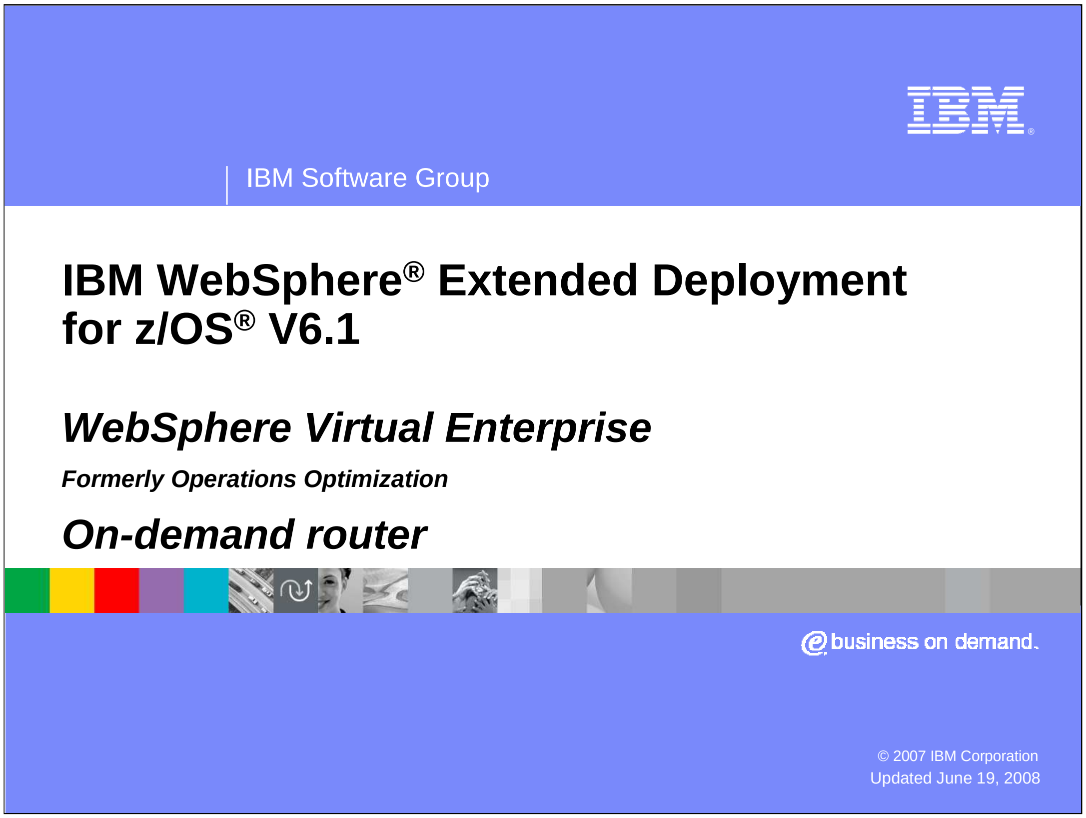


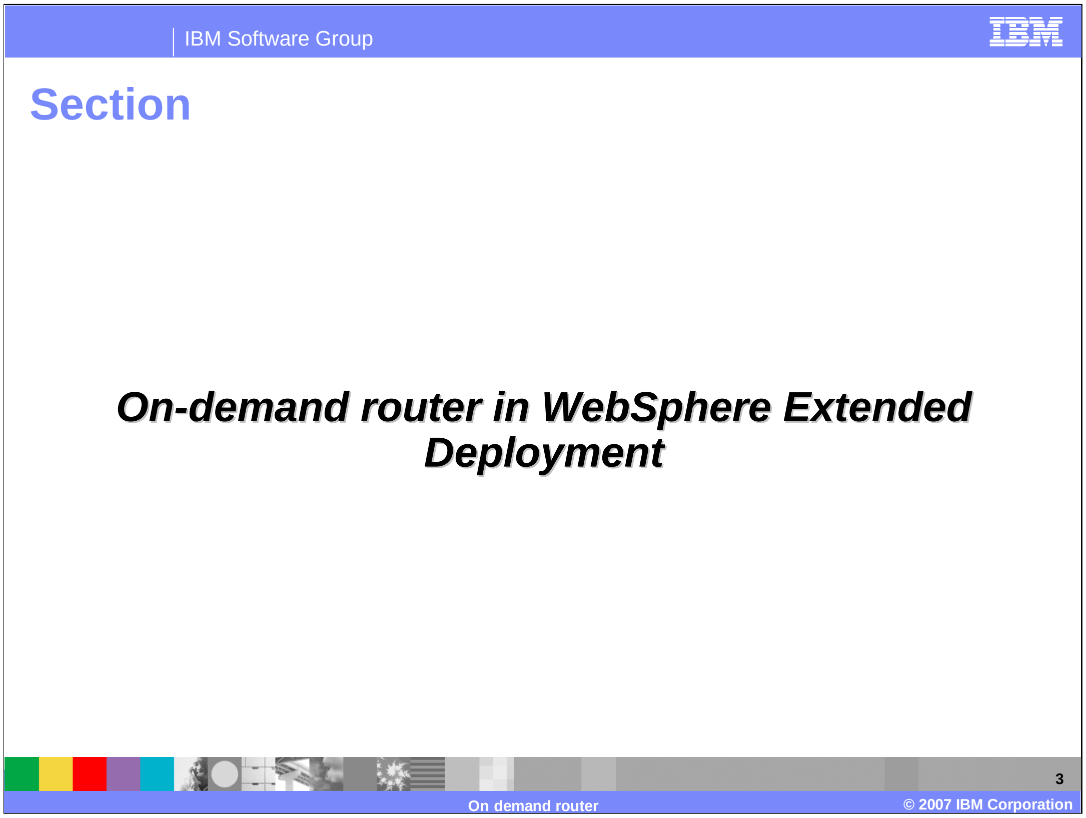
This presentation will cover the on-demand router component of WebSphere Extended Deployment V6.1 for z/OS.

This module was originally recorded for WebSphere Extended Deployment Operations Optimization, which is now called WebSphere Virtual Enterprise. Though the module uses the previous names, the technical material covered is still accurate.

Agenda

- On-demand router in WebSphere Extended Deployment
- On-demand router topologies
- On-demand router in z/OS
- On-demand router administration

This presentation will first discuss the on-demand router and how it interacts with other WebSphere Extended Deployment and z/OS features. Then some common topologies for using the on-demand router and how to create and administer an on-demand router will be covered.



This section will discuss how the on-demand router interoperates with other WebSphere Extended Deployment features.

On-demand router

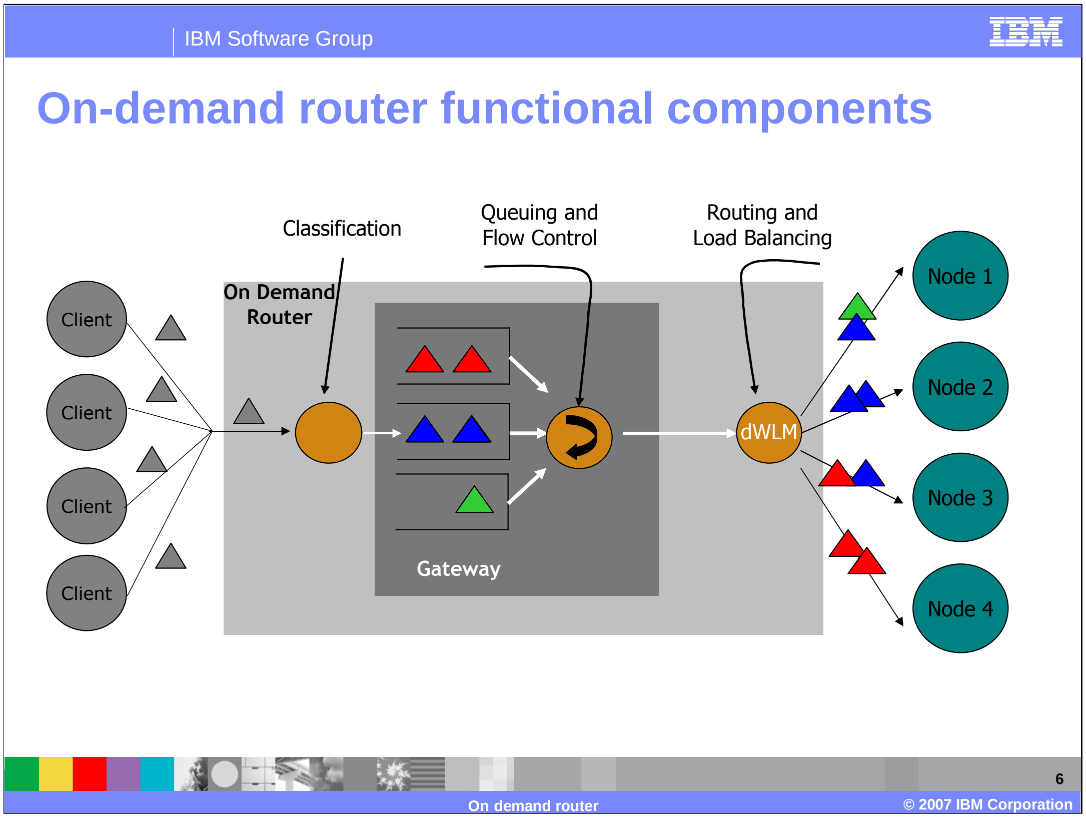
- The on-demand router is an intelligent HTTP proxy server
  - Enhanced version of the proxy server from WebSphere Network Deployment V6.1
  - Can replace or complement the HTTP server plug-in
  - Prioritizes requests and controls traffic flow according to operational policy
    - Ensures that quality of service policies are observed
    - Flow control features ensure that application servers are not overloaded
  - Integrates with application placement to route requests to dynamic cluster members

The on-demand router is an intelligent HTTP proxy server that is provided with WebSphere Extended Deployment. It is the point of entry into a WebSphere Extended Deployment environment, and is responsible for request prioritization, flow control, and distribution to application servers. It can momentarily queue requests for less important applications in order to allow requests for more important applications to be handled more quickly. It is aware of the current location of a dynamic cluster's server instances, so that requests can be routed to the correct endpoint. The on-demand router can also dynamically adjust the amount of traffic sent to each individual server instance based on processor utilization and response times. The on-demand router contributes to the health of a WebSphere cell; for example it can detect and react to a "storm drain condition". A storm drain condition can occur when a backend subsystem or resource terminates unexpectedly. Until the resource becomes available again, the server can return error codes quite quickly. A simple router can falsely assume the server is processing requests very efficiently and start sending more requests to the affected server. The server can become swamped with work when it is able to satisfy requests again. The on-demand router detects this condition and deals with the condition in cooperation with the health monitor. These and other advanced features distinguish the on-demand router from the HTTP server plug-in, and give the on-demand router the ability to ensure a more consistent quality of service for your enterprise applications. It can be used in place of, or in conjunction with the HTTP server plug-in, depending on your needs.

On-demand router (continued)

- World class edge of Sysplex distribution
  - Compliments Sysplex Distributor
  - Complements z/OS workload management
- On-demand router contributes to:
  - Java™ 2 Enterprise Edition (J2EE) batch
- On-demand router is required for:
  - Edition management
  - Health monitoring
  - Visualization
  - Scale out
  - Dynamic operations

The on-demand router is fully integrated into the z/OS architecture and brings a new facet to the z/OS workload management. As this chart points out the on-demand router is quite ubiquitous in WebSphere Extended Deployment. By itself the on-demand router enhances both Sysplex distributor and z/OS workload management. At the same time, it is required for many of the other WebSphere Extended Deployment Operations Optimization features to work.



This diagram illustrates the flow of an HTTP request through the on-demand router. When a request enters the on-demand router, it is classified according to rules that you have defined. These rules can be based on many criteria, including Uniform Resource Identifier, HTTP headers, cookies, or the client's IP address. When a rule is matched, the request is placed into a queue with all other requests that are in the same class of service. The on-demand router can drain these queues at different speeds to ensure that requests in each class are able to meet their goals. This can mean holding less important requests in the queues for longer than more important requests, so that more resources are available to serve the important requests. As requests are released from the queues, they are routed to application servers by the dynamic workload management component, which is constantly aware of the location of each dynamic cluster member. Each member has a dynamically assigned weight, which can increase or decrease the number of requests it receives based on how quickly it is responding to requests. This feature is labeled dynamic work load management or dWLM as opposed to the z/OS workload management.

Autonomic request flow manager (ARFM)

- Dispatches work out of the queues based on class weights
- Dynamically adjusts weights
  - Achieves response time goals for different classes of requests
  - Responds to varying load conditions and request surges
    - Limits the number of requests on a given server
    - Overload protection

The autonomic request flow manager is a component that runs inside of the on-demand router. It is responsible for queuing work by service policy as requests come into the on-demand router, and for controlling the flow of requests exiting the queues. It dynamically adjusts the rates at which requests flow through to application servers to ensure that response time goals are met and that no application servers are sent more requests than can be handled.

Autonomic request flow manager (continued)

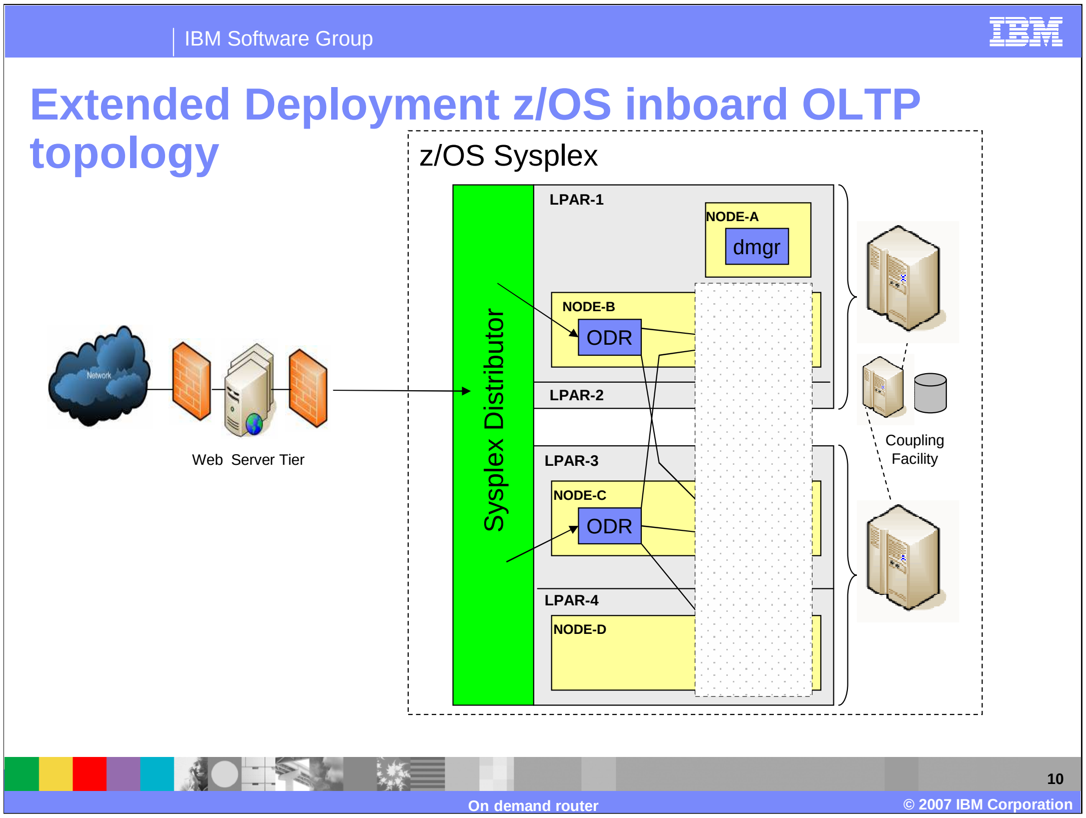
- ARFM Gateway queues and dispatches requests
  - Manages logical queues
  - Queues are dynamically created as service classes appear in the request flow
  - Acts on every request
- ARFM Controller sets gateway dispatch weights
  - Monitors request flow, run times and application placements
  - Sets service class dispatch weights
  - Operates at scheduled intervals

The autonomic request flow manager has two logical components, the gateway and the controller. The gateway is responsible for putting each incoming request into a queue that represents its service class, and later dispatching requests from the queues. The controller operates periodically, monitoring request flow, response times, and application placements, and dynamically sets the dispatch weights for each logical queue, dictating how quickly requests are released from each of them.

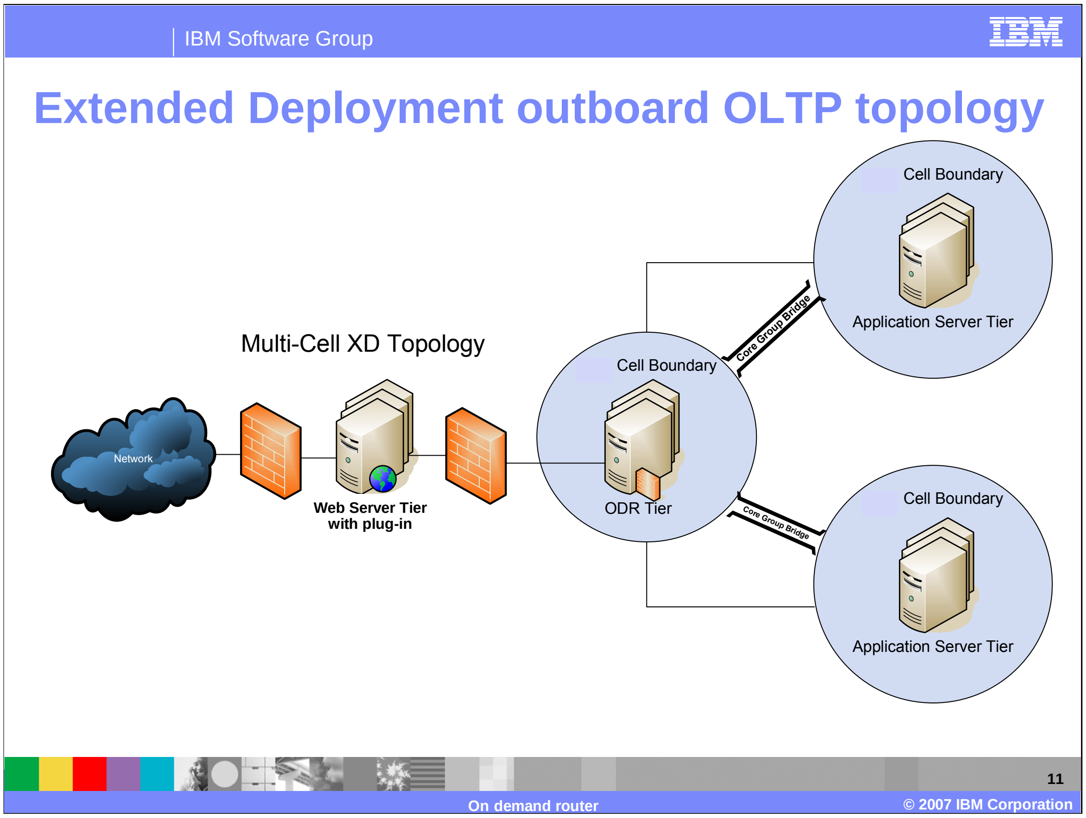
Section

On-demand router topologies

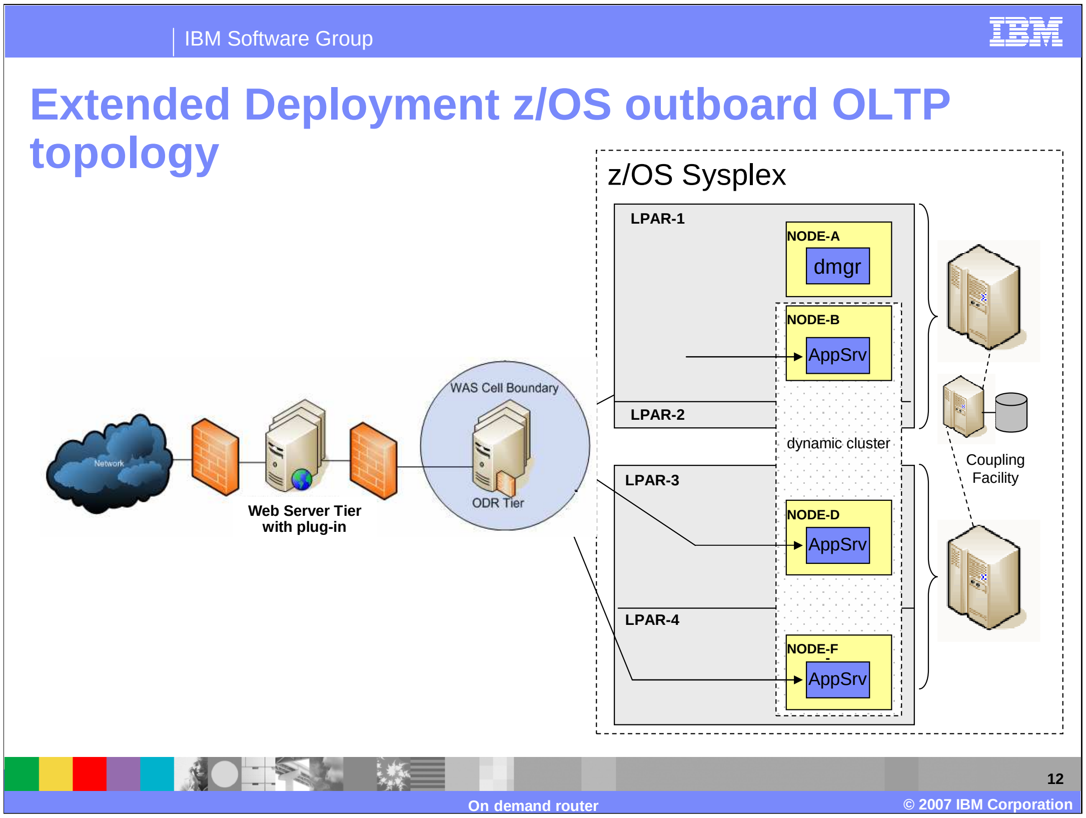
This section will present several possible topologies to use an on-demand router.



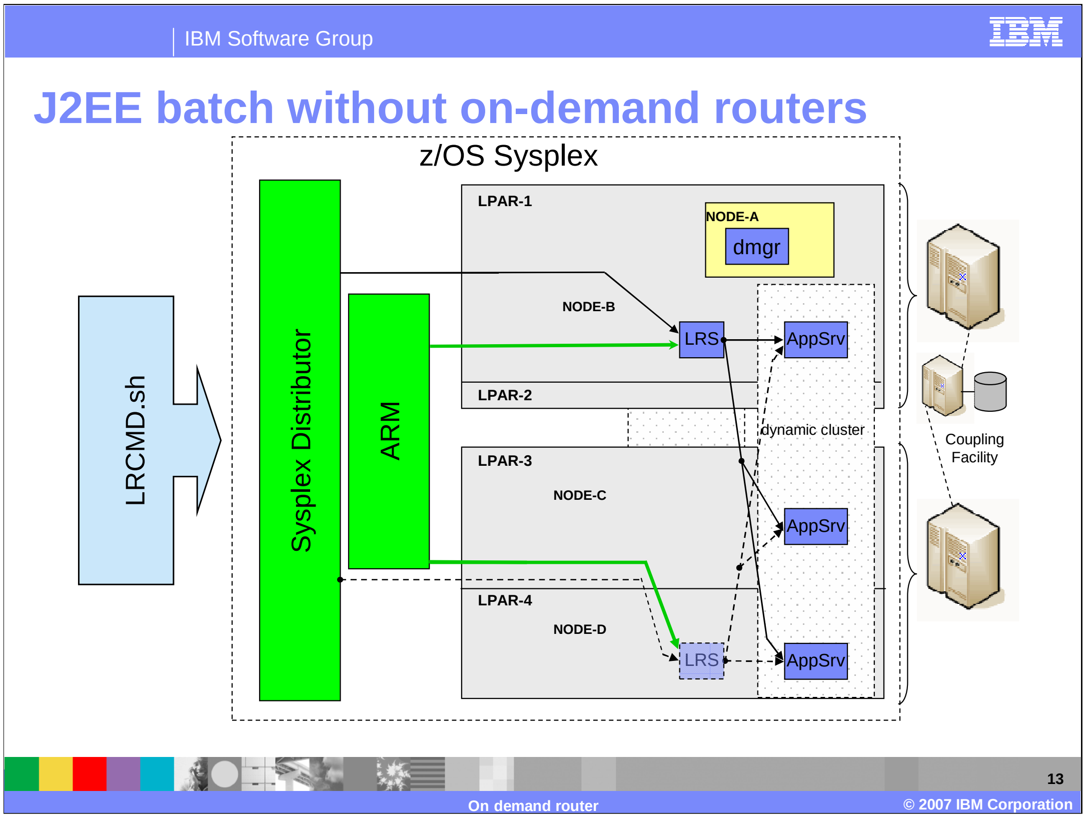
This slide presents the inboard on-demand router configuration. Node A is shown as deployment manager or 'dmgr'. The other three nodes contain servers, both on-demand routers, abbreviated here as ODR, and application servers shown here as 'AppSrv'. The inboard on-demand router configuration can take advantage of Sysplex distributor. Sysplex Distributor cluster addresses is a natural way to provide network transparency to the multiple on-demand routers. Note that while the on-demand routers are not clustered, they still provide high availability through redundancy; the multiple on-demand routers work cooperatively as peers off of shared state data. In addition an on-demand router can start additional servers in a dynamic cluster to accommodate increased load, for example if the load increases the on-demand router running on NODE-C can start a server on NODE-D. This topic is discussed more in the section on dynamic operations. This type of configuration with the on-demand router taking advantage of Sysplex distributor is the recommended kind of topology. However user constraints might require the on-demand router to be outboard.



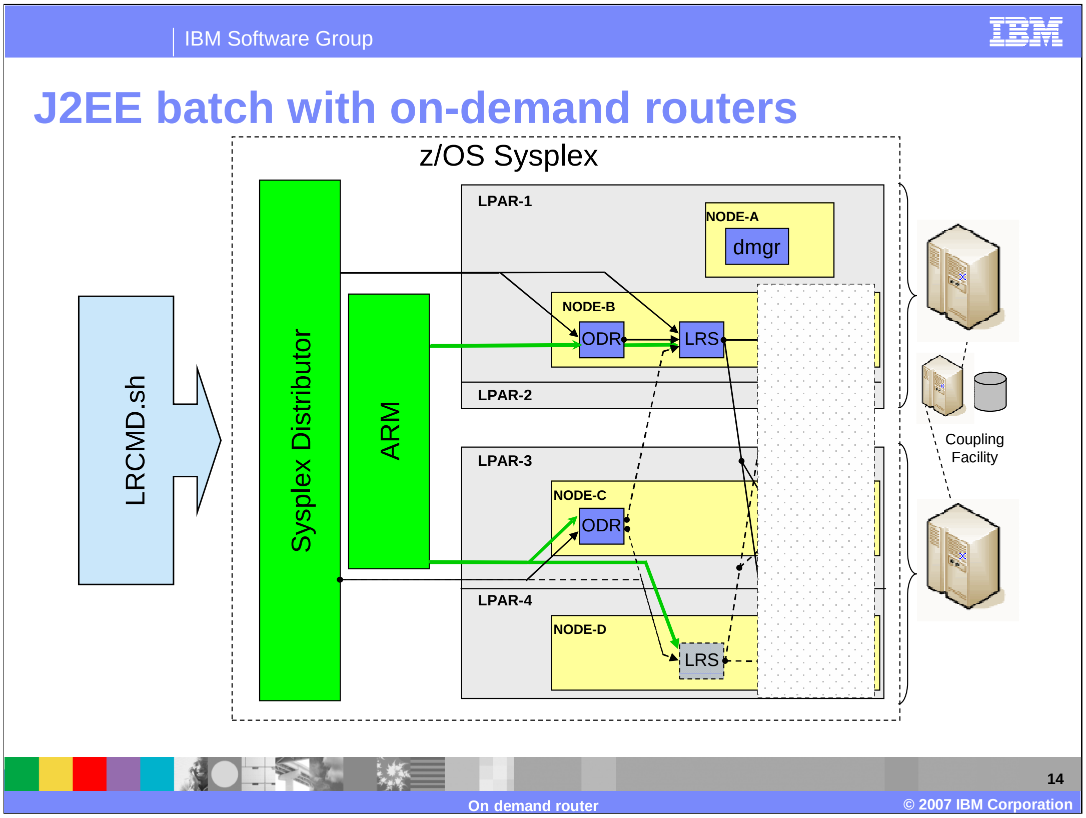
This slide shows the on-demand router isolated from other WebSphere Extended Deployment cells, communicating with them through core group bridges. Here the on-demand router could be on a z/OS platform and still take advantage of Sysplex distributor, or the on-demand router could be on a distributed platform. In either case the on-demand router interacts with z/OS workload management and carries on the two way communication that will be discussed shortly, even when the on-demand router is not on a z/OS platform.



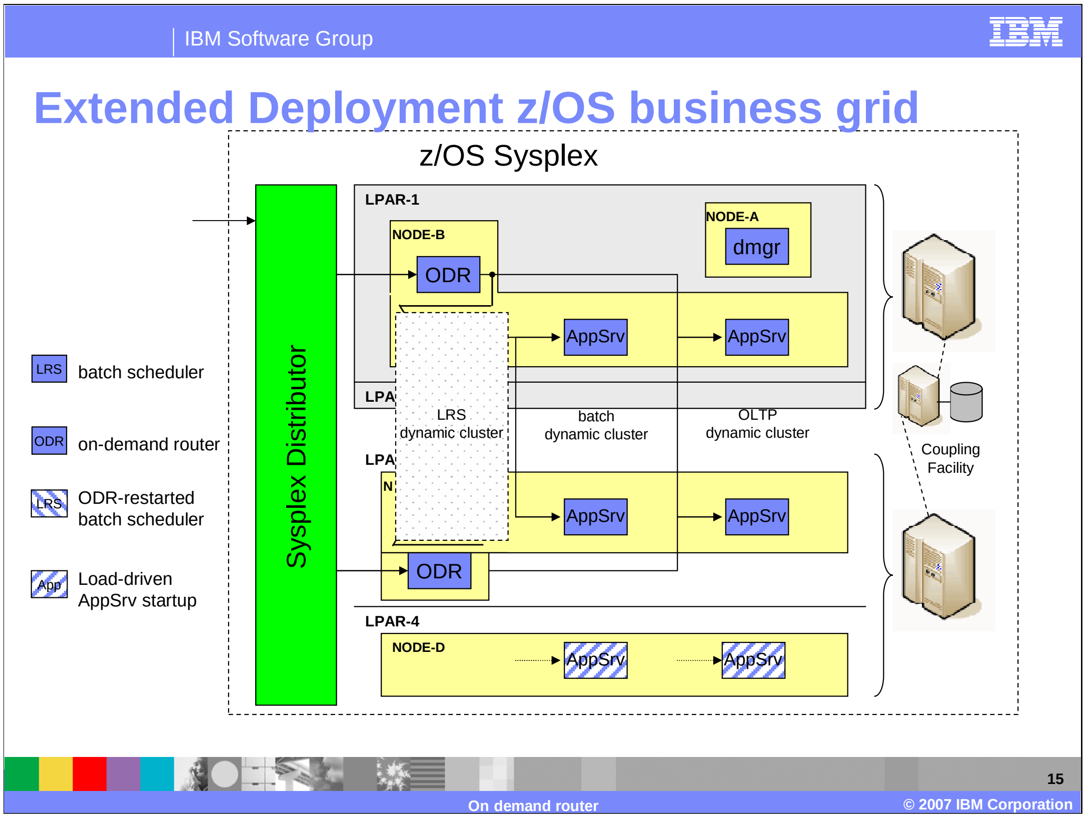
When the on-demand router is not configured on a z/OS platform it cannot take advantage of Sysplex distributor. But the on-demand routers still provide high availability through redundancy. On-demand routers can start additional servers in a dynamic cluster to accommodate increased load. And most importantly they can carry out the same bi-directional communication with the z/OS work load manager. Care should be taken when considering this configuration to account for the increased network traffic between the z/OS workload manager and the on-demand router. If the on-demand router is located in the same SYSPLEX as a cell, this communication does not require traversal of the TCPIP stack. If the on-demand router is on a distributed platform, the communication will traverse the TCPIP stack.



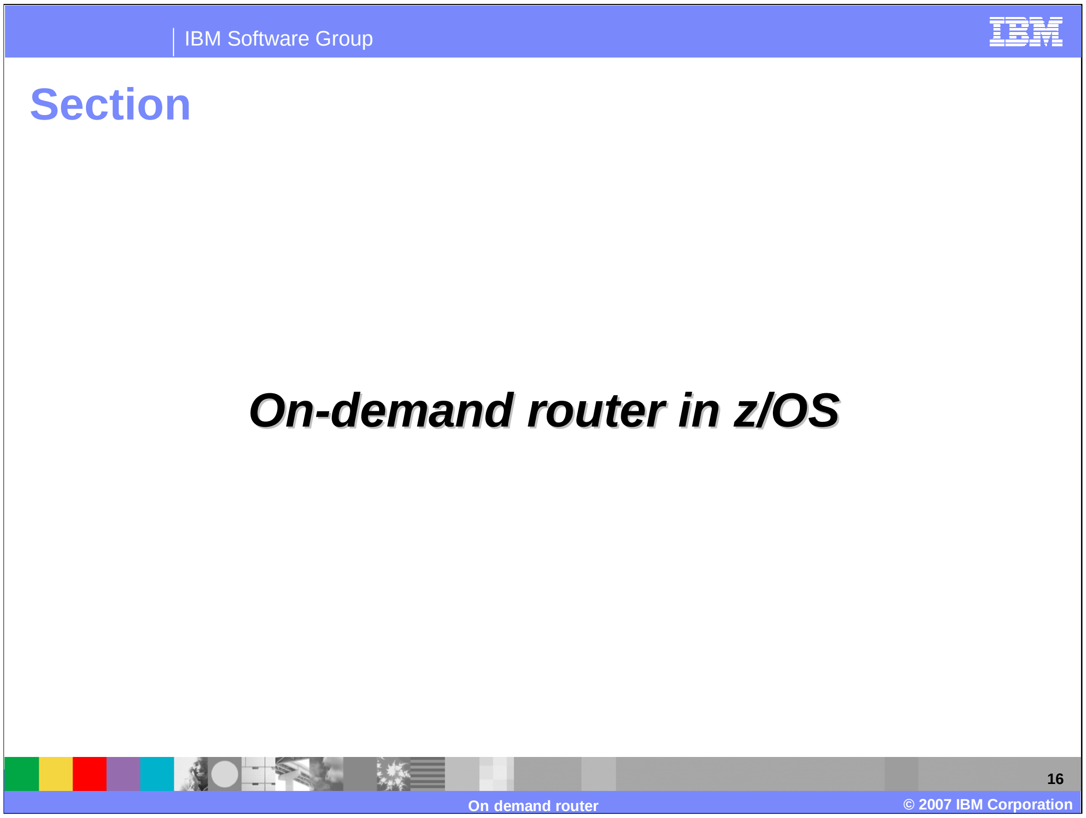
Because the grid scheduler, or LRS, is an application running on an application server, an on-demand router can improve the availability of J2EE batch. In particular, without an on-demand router present, you will not have continuous availability of the batch application. At the start of this picture the grid scheduler is running in NODE-B. The grid scheduler is an application and the rule is there can only be one running grid scheduler cell. Here a user enters batch commands though LRCMD.sh. These flow through Sysplex distributor to the grid scheduler running in NODE-B. The control from grid scheduler then follows the solid arrows to the application servers running the J2EE batch steps and all is fine. If for some reason the server running the grid scheduler terminates unexpectedly, the solid arrow paths are lost. Here, ARM restarts a grid scheduler on NODE-D in the picture as represented by the lower green arrow. Once the new grid scheduler “comes up”, the flow is similar to before except following the dashed arrows. Between the time the first grid scheduler crashed and the second grid scheduler is ready, all communication is lost. Note that instead of using ARM to restart a grid scheduler, the grid scheduler can be placed in a cluster and configured to have exactly one instance running. The result is the same when a second grid scheduler is required.



With on-demand routers present, you will have continuous availability of the batch application. This is the same picture as before except for the addition of two on-demand routers. The main difference is to reroute the communication path from the Sysplex distributor from the running grid scheduler to the on-demand routers. ARM restarts an on-demand router if it goes down. Since multiple on-demand routers can be running at the same time, availability is assured. When a grid scheduler goes down in this picture, the on-demand router restarts a copy, and while the grid scheduler is restarting, the on-demand routers queue up requests. In this scenario, the on-demand router has made an already great system even better.



This picture is collage of the onboard on-demand router topological options as discussed.



This section will discuss briefly how the on-demand router interoperates with z/OS.

IBM Software Group

HTTP distribution scorecard

	Sysplex distributor	HTTP with plug-in	WebSphere proxy server	On-demand router
Load balancing capabilities	⚑⚑⚑	⚑⚑	⚑⚑⚑	⚑⚑⚑⚑
DMZ Friendliness	⚑	⚑⚑⚑	⚑	⚑
Caching / edge serving	⊘	⚑⚑	⚑⚑⚑	⚑⚑⚑
Affinity routing	⊘	⚑⚑⚑	⚑⚑⚑⚑	⚑⚑⚑⚑
Awareness of environment	⚑⚑	⚑	⚑⚑⚑	⚑⚑⚑⚑
High availability	⚑⚑⚑⚑	⚑⚑⚑	⚑⚑⚑	⚑⚑⚑
Platform applicability	⚑	⚑⚑⚑⚑	⚑⚑⚑	⚑⚑⚑

17

This slide borrows heavily from the "***WebSphere z/OS - Comparing front end HTTP options"*** article by Don Bagwell cited in the reference section at the end of this presentation. As noted in the article, this chart is highly subjective and is offered to you as an initial guide to understand where the on-demand router fits into the z/OS structure. It is important to note that this presentation does not suggest that the on-demand router can replace the Sysplex Distributor, but rather that the two compliment each other.

In this chart, the on-demand router was given the high rating for "load balancing capabilities" because it has the ability to classify and prioritize work. It also has balancing capabilities not discussed in this presentation. It truly is a very powerful piece of functionality.

Although the proxy server and on-demand router have been deemed "DMZ un-friendly" here, this characterization is a point-in-time thing. This is understood to be an issue and will be addressed over time.

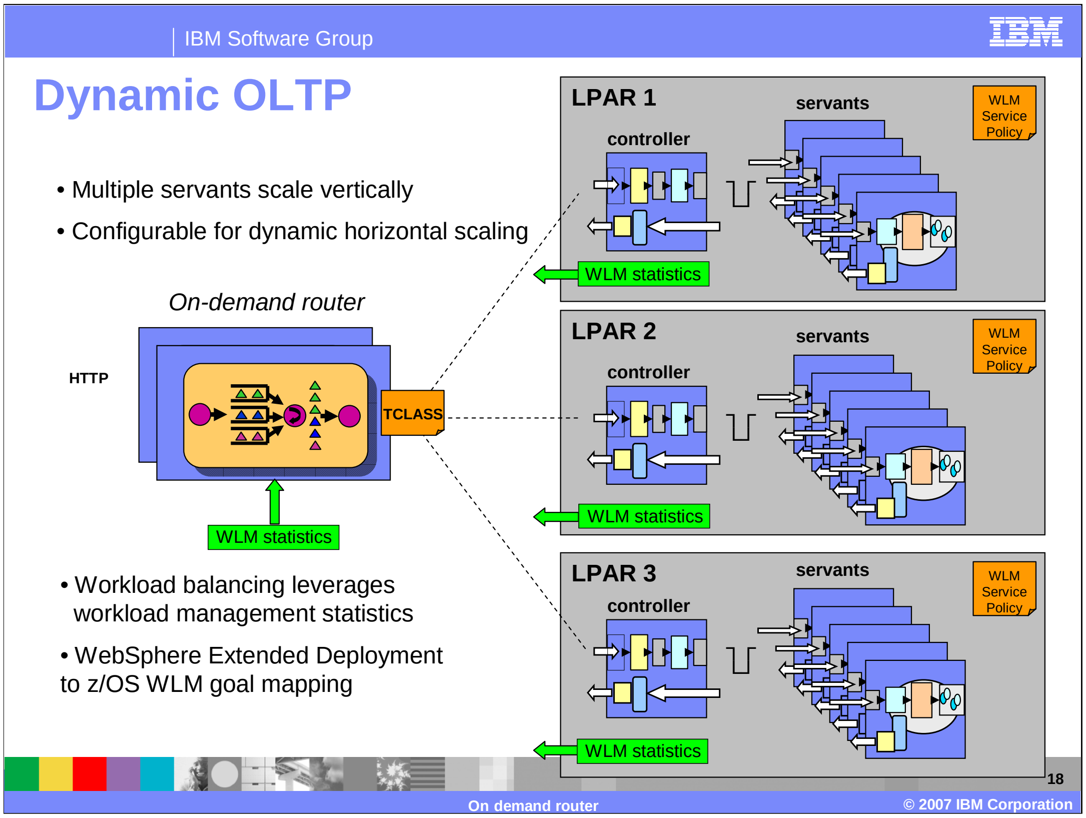
Some might wonder why Sysplex Distributor is not equal to or better than the proxy server or on-demand router for "awareness of environment". The proxy server and on-demand router is shown as relatively better than Sysplex Distributor because the Sysplex Distributor has no awareness of the state of the application within the application server, only the status of the application server address space itself. The on-demand router is given the highest rating because it is not only being fed WLM metrics (like Sysplex Distributor and proxy server), but it is also capable of detecting things like backend data store outages. (It does not directly detect a DB2 outage, for example, but algorithms within the on-demand router deduce such things and avoid sending work to servers where backend outages have occurred.)

Sysplex Distributor has the highest rating for high availability because it is a function that is so deeply a part of the Parallel Sysplex environment. It is proven, and it is a part of the known advantages of Parallel Sysplex. The other options can be made highly available by way of configuring multiples of the server, but in general the way to really make them highly available is to use Sysplex Distributor along with the multiple instances.

For platform applicability, the HTTP server with plug-in scores the highest because it runs on most platforms and in non-IBM HTTP servers. The Sysplex Distributor is only applicable to z/OS (and only to a Parallel Sysplex), so it only received one star.

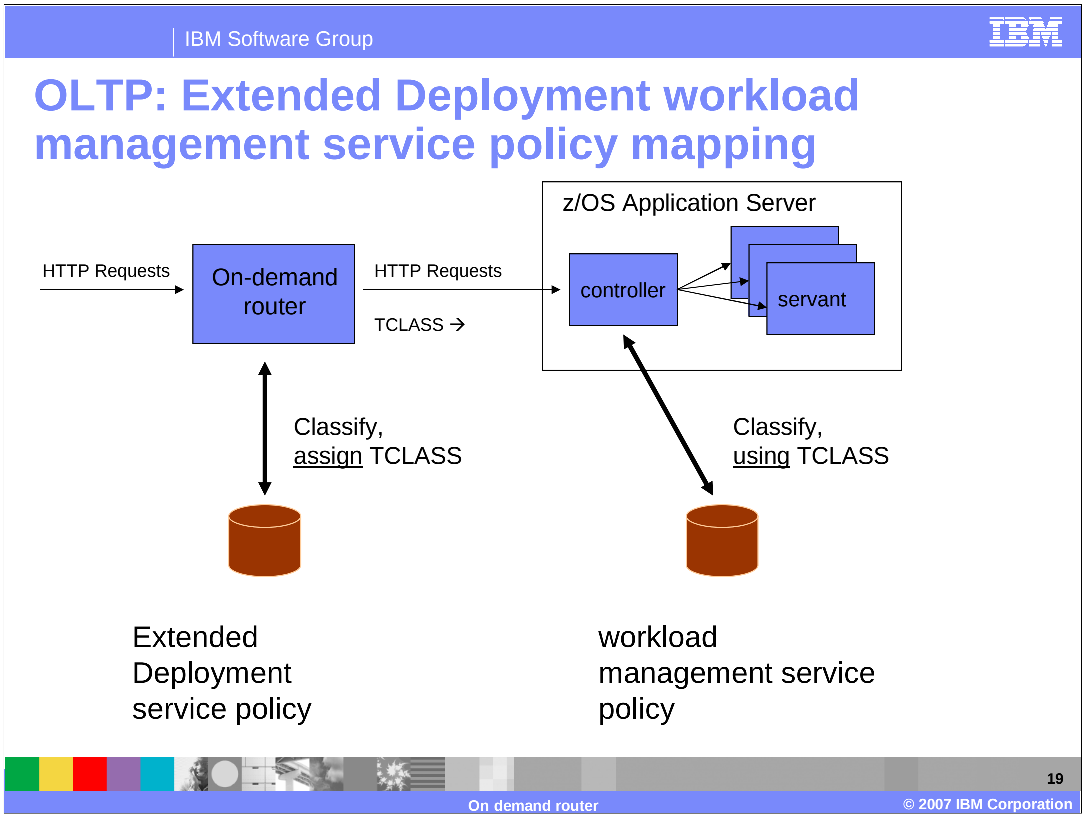
"Simplicity / low overhead" is a terribly subjective evaluation and will generate a lot of discussion as a metric. Sysplex Distributor is rated the highest primarily because it comes with z/OS Communication Server and is highly optimized. The proxy server and on-demand router are the lowest rated because in terms of raw throughput, they *probably* do not pump the requests as fast as the Sysplex Distributor would. But then again, when the on-demand router is configured to do all the classification and prioritization, it's doing a whole lot more than Sysplex Distributor is doing.

Refer to the original, excellent article for details.

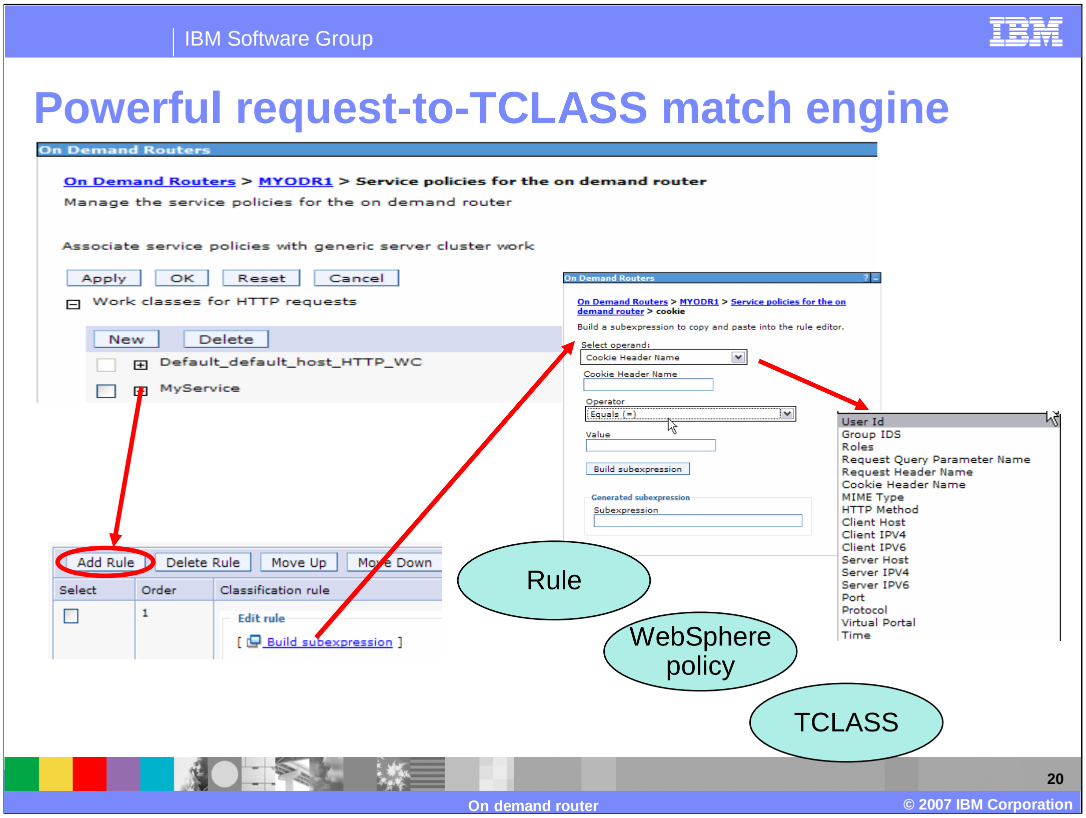


The on-demand router is required for dynamic OLTP and it can be either inboard with Sysplex Distributor, or outboard. The application servers provide vertical scaling by deploying multiple servant regions through z/OS work load manager. The on-demand router adds horizontal scaling to WebSphere. In particular, the on-demand router can start or stop additional servers as needed for horizontal scaling. The on-demand router accepts z/OS WLM input similar to the Sysplex distributor and feeds output back to z/OS WLM through the TCLASS parameters and WebSphere Extended Deployment to z/OS WLM goal mapping. It is important to note that the mechanisms used by the on-demand router and z/OS WLM are vastly different. z/OS WLM will decide to start a new servant region based on how the OLTP z/OS WLM service policies are being satisfied as compared to how the service policies are being met for all tasks in the purview of z/OS WLM. z/OS WLM can react much faster than the on-demand router. The on-demand router, however, considers the WebSphere offered load, the resources available on all the LPARs in its domain, and how all the WebSphere tasks are meeting their WebSphere service policies.

As has been shown in the previous slides, the on-demand router provides autonomic computing, in that the on-demand router can restart failed servers. This occurs either on the same node or a different node if an entire node is lost for some reason.



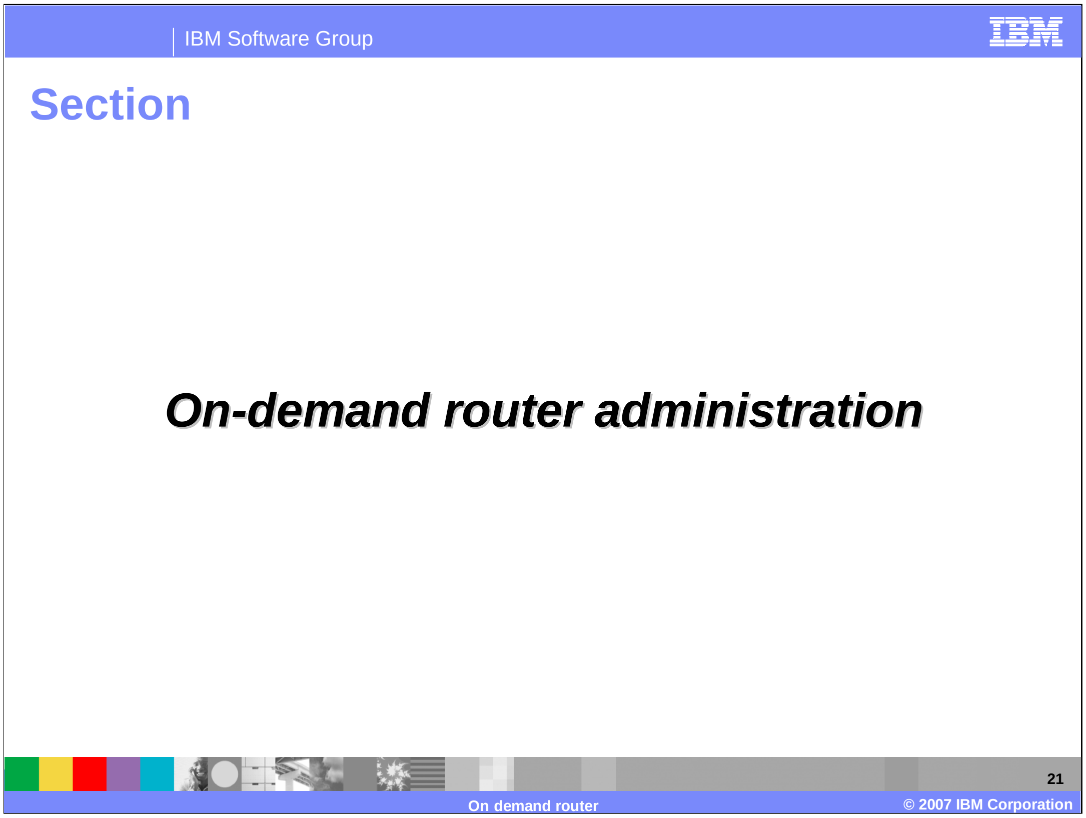
Here is a more detailed look at the feed back from the on-demand router back to z/OS WLM. As already seen, the on-demand router performs a classification and a prioritization of incoming OLTP requests. The result of the classification is to assign a TCLASS value to the incoming request. This TCLASS value is then passed along with the request to a server to process. The request's first stop in a server is to the controller region for the server. The next step is configurable, but by default, the controller region uses the TCLASS value to assign a z/OS WLM service policy to the request. The request is then queued to z/OS WLM and the flow follows the normal WebSphere for z/OS processing. If the default value is overridden, the controller region performs its own classification and selects a service policy. Both the on-demand router and a controller region have stats from z/OS WLM to make decisions with. However, the on-demand router also has the view of the complete WebSphere offered load and current server response times.



Managing the mapping of requests to TCLASS values is much easier with the on-demand routers graphical interface. WebSphere Application Server uses a flat file to assign TCLASS values to tasks. This file has to be hand edited and the server restarted to implement new TCLASS mappings. WebSphere Extended Deployment simplifies this process by offering a rule builder in the administration console. You can define generic service policies for an on-demand router as shown here or you can define specific policies for an application. The on-demand router is aware of both. As you can see, there is a rich set of operands for the rule builder, this example uses “Cookie Header Name”. Many of the operands can be modified, here you need to supply the name of the cookie you are looking for.

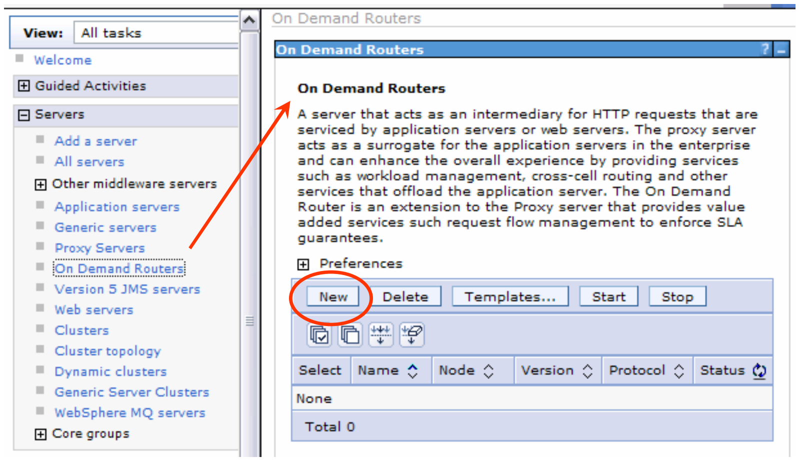
The rules for WebSphere policy are kept in two data bases. WebSphere maintains its policies though the administrative console. For all this to work correctly it is important to make sure that the WebSphere server policies and the z/OS WLM service policies are matched and matched in a way that makes sense. If there is a mismatch, a default policy will be used by z/OS WLM. It is your responsibility to ensure the matching occurs.

Note that part of the actual panels have been removed to allow the picture to fit on the slide.



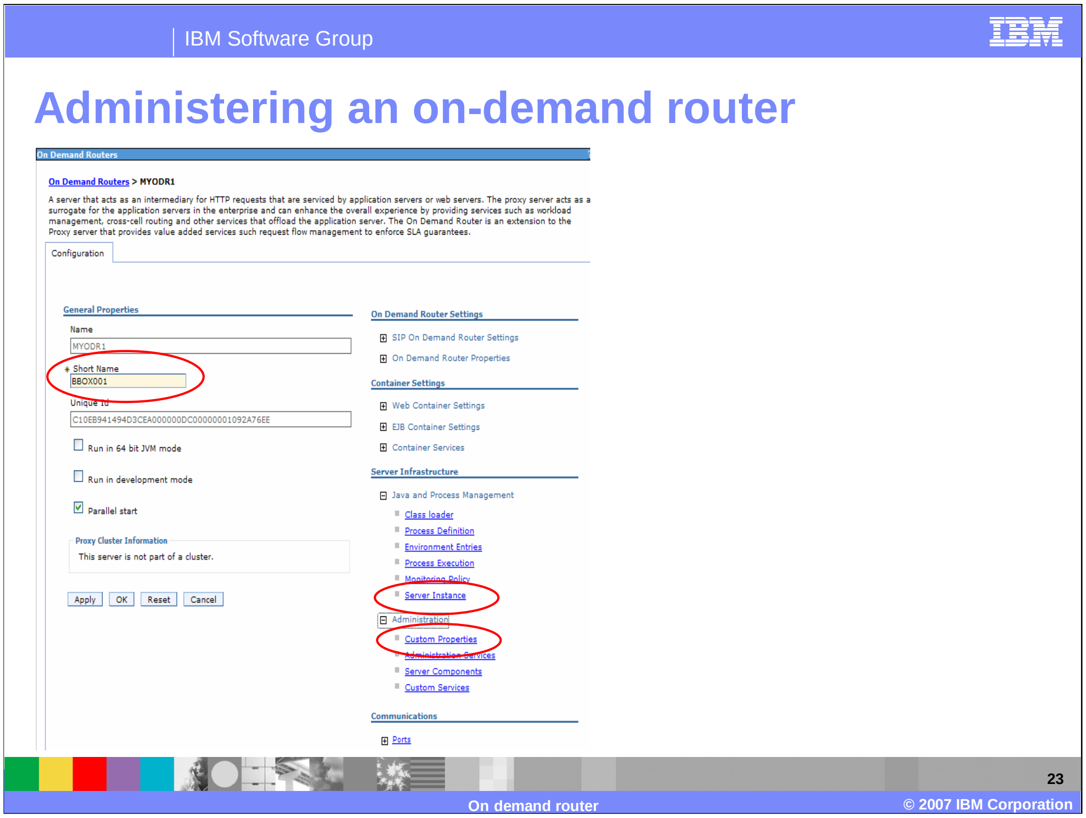
This section will introduce the creation and administration of an on-demand router. You can also view the demonstration "Administering an on-demand router".

Creating a new on-demand router



- Administrative console or wsadmin “createodr.jacl”

You can create a new on-demand router from the wsadmin.sh script “createodr.jacl” or you can use the administrative console. The administration page for on-demand routers can be found under Servers in the left pane of the administrative console. To create a new on-demand router, click on new and follow the wizard instructions. Note, the administration console will ask if your on-demand router will also support the session initiation protocol or SIP, but this feature is not supported on WebSphere Extended Deployment V6.1 and should not be selected.



Selecting an on-demand router from the on-demand routers list produces the main administration page for an on-demand router. Here you can set the short name. From the Server Instance link you can select how many copies of this server to run. From the Custom Properties link you can set cluster transition name and from the ports selection you can set the on-demand routers ports. For more detail on this, refer to the demonstration named, “Administering an on-demand router”.

Setting ports with Python

```
AdminTask.modifyServerPort(serverName, "-endPointName BOOTSTRAP_ADDRESS -port ",
    portNumber)
```


This Python script statement is an example of how to set port values without using the administrative console. The statement shown can be expanded to set all the port values you might require.

Summary

- On-demand router is critical to many WebSphere Extended Deployment features
- Workload management integration
  - Workload management metric exploitation by on-demand router
  - Extended Deployment workload management service policy mapping for OLTP/HTTP

The on-demand router offers world class edge of Sysplex routing. The on-demand router is required by many WebSphere Extended Deployment features and enhances all OLTP traffic. The on-demand router is best deployed with and enhances Sysplex Distributor. And the on-demand router interoperates effectively with z/OS WLM.

References

- ***WebSphere z/OS - Comparing front end HTTP options***, document ID PRS2663 by Don Bagwell of Washington System Center
  - http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS2663

Here is a link to the article mentioned on slide 17.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject= Feedback about XD61z_On_Demand_Router.ppt

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers