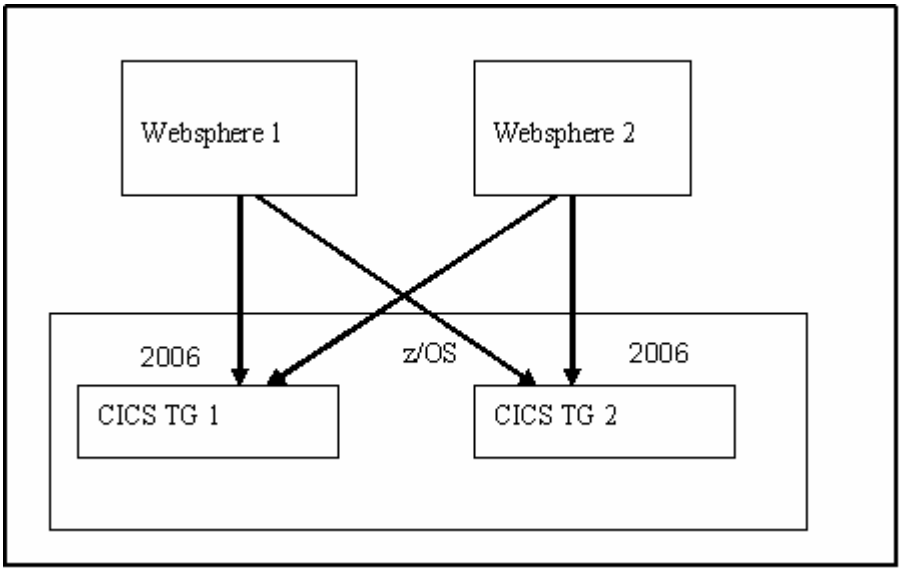
CICS Transaction Gateway on z/OS: Achieving high availability

z/OS provides a number of system facilities which assist you in setting up a CICS environment configured for high availability. This document addresses the addition of CICS Transaction Gateway to your CICS environment and assuring that the CICS TG also provide for the highest availability.

1 - Availability of the CICS TG instances

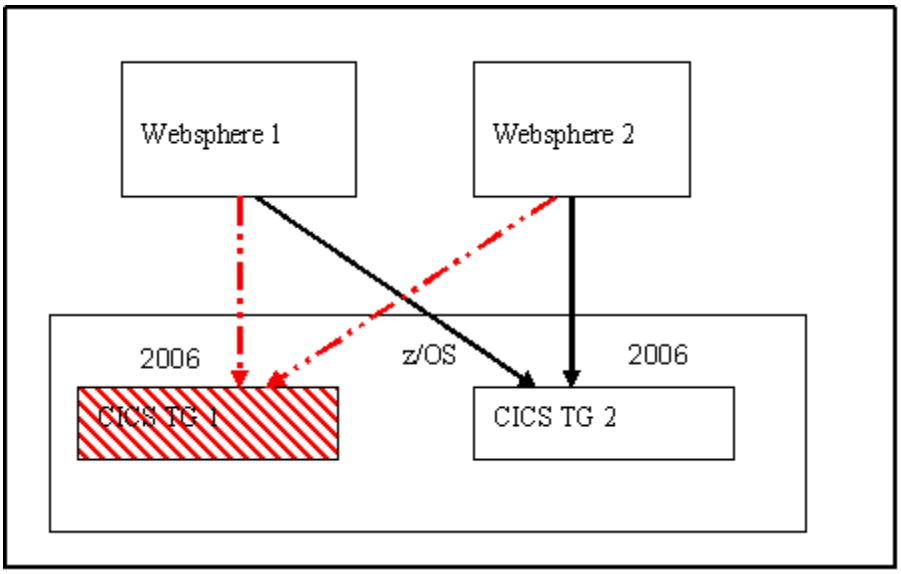
The first consideration is the availability of the CICS TG instances themselves. On z/OS, you may create multiple instances of the CICS TG in the same operation system image. Port sharing is a feature of the TCP/IP software on z/OS. It allows multiple servers (in our scenario, the TCP/IP stack views the CICS TG as a *server*) to listen on the same port. That means that the clients of that port have no indication that there is more than one server and they don't know or care which instance they are connected to. The algorithm is simple; whichever server (i.e. CICS TG daemon) has the fewest number of connections will get the next new connection. This approach is "connection balancing" and not load balancing per se. From a workload perspective, the presumption is that over a decent interval, each connection carries about the same amount of work.

In an availability configuration, when you put multiple CICS TGs onto the same LPAR, you should define them with identical configurations, including the same *listener* port so that you can take advantage of the port sharing facility. In the TCP/IP configuration, define this port with the ***SHAREPORT*** option.



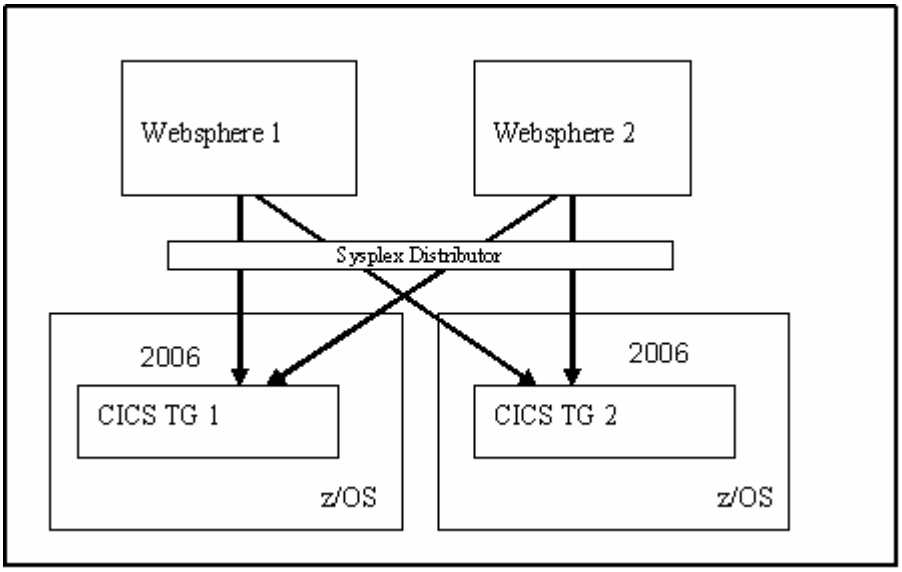
In this configuration, both CICS TG instances listen to the same port. During session set-up, the SHAREPORT facility causes the connections to be distributed somewhat evenly across both CICS TG instances. Note that this "somewhat evenly" doesn't necessarily mean that 50% of the connections from each server go to each CICS TG instance.

If one CICS TG fails, then the connections from WebSphere Application Server to that CICS TG instance will become disconnected.



The application server attempts to reuse a connection, discovers that it is not connected, and will then attempt to re-establish the connection. Since CICS TG #2 is still active and listening to the network, the connection will be bound to CICS TG #2 and the work flows to CICS unimpeded. If CICS TG #1 is recovered before WebSphere needs to reuse one of the broken connections, it will be the most likely candidate for the new connection.

If you define multiple instances of the CICS TG across several LPARs in a Sysplex, you should use Sysplex Distributor to provide for connection balancing across the LPARs.



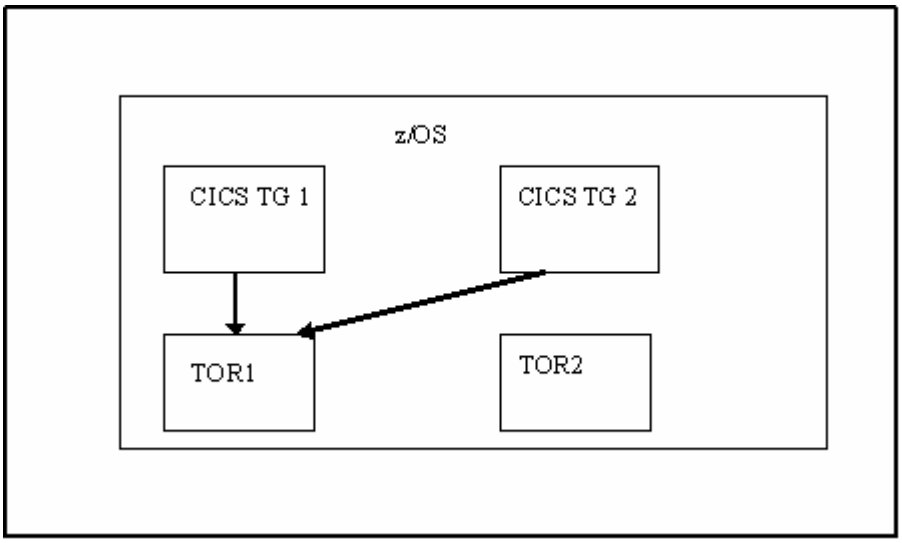
The session set-up between WebSphere Application Server and the CICS TG instances takes place just as in the single LPAR example. Before the connection request is passed through to the TCP/IP stack on a given z/OS LPAR, Sysplex Distributor will choose which of the participating hosts will be the target of

this request based on input from the z/OS Workload Manager (WLM) component. Failover processing takes place as previously described except for the participation of Sysplex Distributor.

2. Availability of the TORs.

The recommended configuration with CICS TG on z/OS is to have a given CICS TG instance connect to one and only one CICS region. We tend to call these *routing regions* instead of TORs since there aren't any *terminals* involved in the CICS TG connections, but I've kept the TOR designation in my charts since that is a more familiar terminology. Connections from the CICS TG address space to CICS are EXCI pipes. These pipes (actually MRO sessions) are attached to the Worker Threads in the CICS TG daemon.

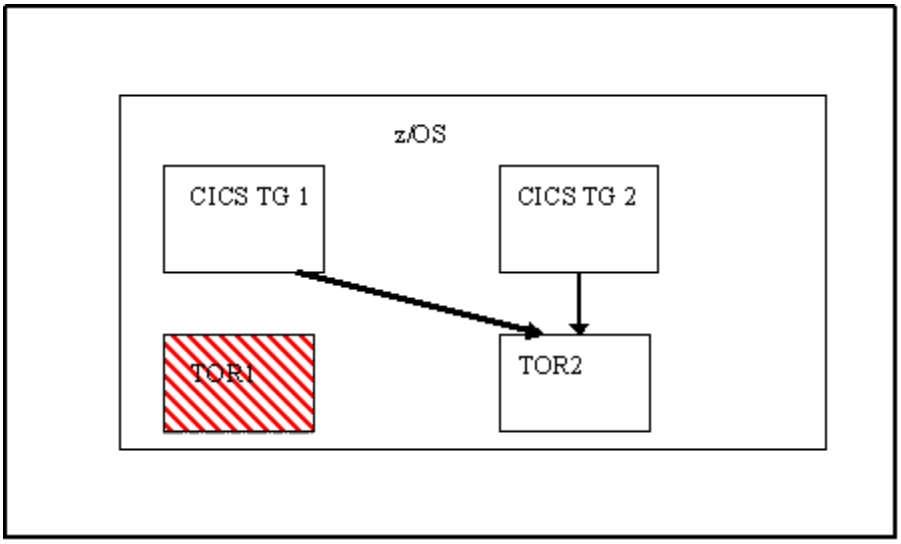
In a single z/OS LPAR, there's no compelling reason to have each CICS TG connected to a different TOR. From a capacity stand-point, a single TOR should suffice, though you still need that second TOR for availability even though you may not need it for capacity.



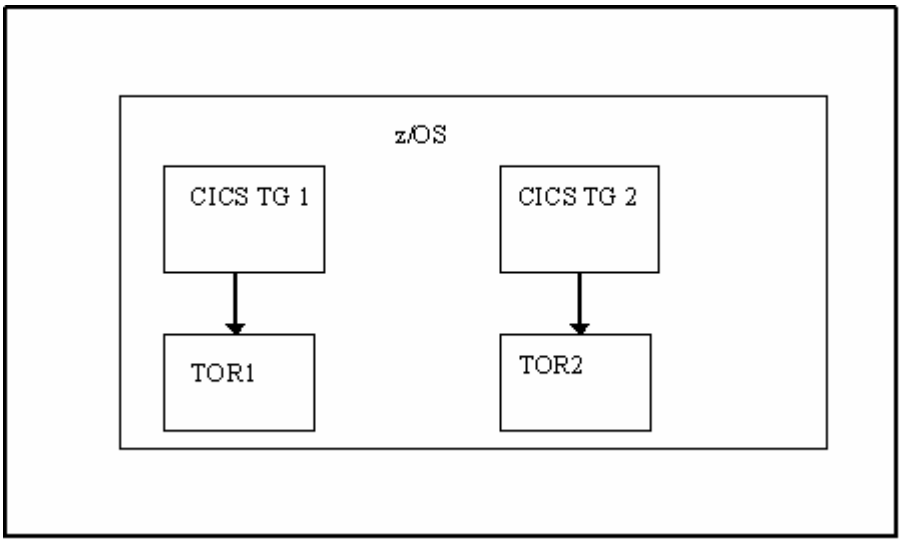
If the TOR abends, the EXCI pipes from the CICS TG are disconnected. When the CICS TG dispatches work to a worker thread and it attempts to send the ECIRequest to CICS, the CICS TG will discover that the pipe is gone. It will then attempt to re-allocate the pipe.

Now -- let's back up and talk about pipe allocation. The CICS TG will allocate a pipe when it is needed. At pipe allocation time, an exit program named DFHXCURM is driven. (The default exit program provided by IBM does nothing at all.) The APPLID of the CICS region is taken from the Server field in the current ECIRequest and the pipe is connected to that CICS region if it is active. This exit is re-driven if the pipe cannot connect to the target CICS region. This allows the exit to substitute a different APPLID for the one which was specified in the ECIRequest.

So -- to achieve high availability for TORs, we code the DFHXCURM exit to handle failover in the event that a TOR fails. Note that if the exit substitutes a different APPLID, this value is **NOT** communicated to the CICS TG itself. The CICS TG does not know the actual name of the CICS region to which it is connected, only the name that it processed from the ECIRequest.



Of course, you could start out with each CICS TG instance connected to a different TOR as in this configuration.

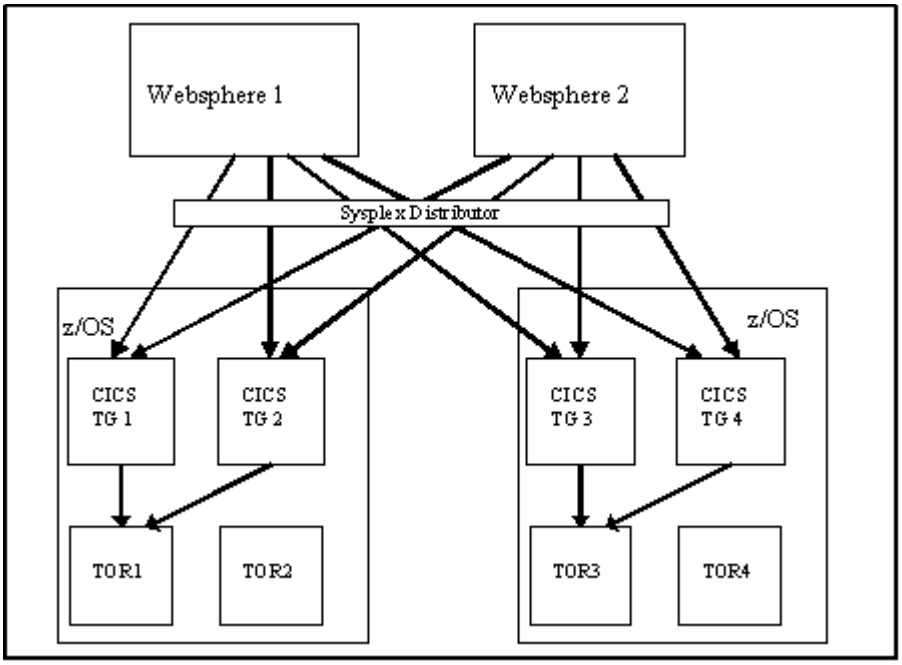


If a TOR fails, the DFHXCURM exit would be expected to substitute the name of the TOR which is still running. The result is the same, but your configuration is more complex:

- you need separate configurations for each CICS TG instead of running them both from a common configuration
- your exit needs to have code to handle the fact that the primary TOR is not the same for each CICS TG

One way to overcome this complexity is to utilize CICSPlex System Manager's knowledge of the active CICS topology. The CPSM API can be used to request a list of available CICS regions. By restricting the query to only those regions of type "TOR" which are on the same z/OS system as the CICS TG instance, the DFHXCURM exit can easily be implemented in a multiple-LPAR Sysplex environment without requiring explicit coding to deal with the multiple z/OS images.

Applying all of the availability mechanisms described above within a Parallel Sysplex, we wind up with a configuration with no single point of failure and provisions for failover processing should any node become unavailable.



Leigh Compton
IBM Advanced Techincal Support
lcompton@us.ibm.com

16 June 2005