the Availability Digest

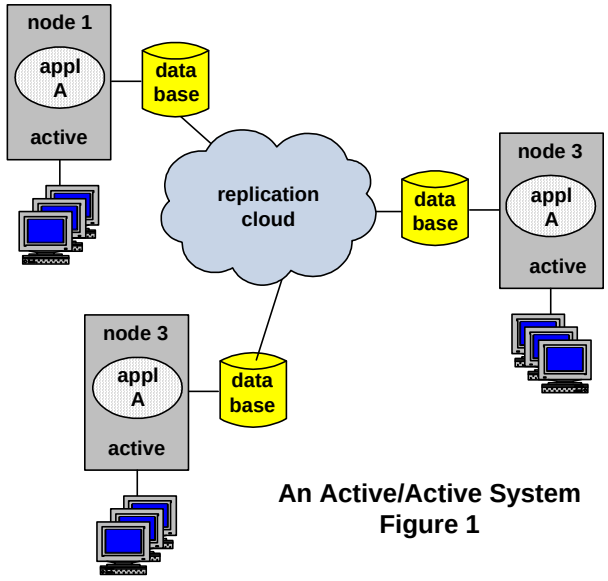
www.availabilitydigest.com

Achieving Fast Failover in Active/Active Systems - Part 1

August 2009

Active/active systems[1] provide continuous availability not because they avoid faults but because they can recover from faults so quickly that users don't notice that there has been an outage. This capability requires not only that failover to a backup component be rapid but that it be reliable.

As shown in Figure 1, an active/active system comprises two or more geographically separated processing nodes that cooperate in a common application. A transaction can be routed to any node in the application network and be processed just as it would be if it were routed to some other node.



An Active/Active System
Figure 1

This requires that each node has access to a local copy of the application database. The databases are kept synchronized via data replication. Whenever a processing node makes a change to its copy of the application database, that update is immediately replicated to the other database copies in the application network.

Should a node fail, all that is required to recover from the fault is to move the users or to reroute transactions to one or more surviving nodes. Failover can be accomplished in seconds or even in subseconds. Furthermore, failover is risk-free and reliable because it is known that the surviving nodes are operational. After all, they are actively processing transactions.

But how can users or transactions be moved so quickly between processing nodes? That is the subject of this article.

Transaction Redirection

We take the easiest case first. An active/active system can be structured so that users are not associated with any particular node. Rather, they submit their transactions to an intelligent router or to a load balancer that according to configured rules routes the transaction to some node in the application network (Figure 2). For instance, a transaction might be routed to the nearest

[1] What is Active/Active?, *Availability Digest*; October, 2006.

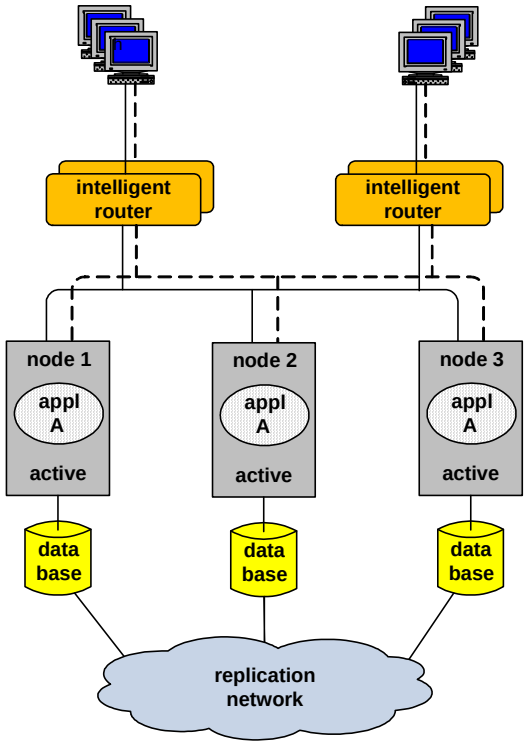
available node or to the least loaded node. Alternatively, transactions might simply be round-robined to a set of nodes.

Needless to say, the routing mechanism must be redundant and geographically distributed so that no single event can block a group of users from the application network.

Should a node fail, the routing facility will detect that no response is being received from the transactions being sent to that node and will mark the node out of service. Based on its routing rules, the routing facility will begin forwarding transactions that normally would have gone to the failed node to some surviving node

There must be some mechanism to return the failed node to service. For instance, the router might periodically ping the failed node or, better still, send the failed node an application-level query message. When the router receives a response, it can once again begin routing transactions to the recovered node.

Achieving continuous availability requires only that a fault be rapidly detected. Rapid fault detection can easily be attained by immediately rerouting a failed transaction to another processing node. If a series of transactions to a node fail, the router can declare the node down and monitor it for recovery. If succeeding transactions are successful, the node has suffered only a transient fault (or an application bug) and can be reinstated as an active node participating in the application.



Transaction Routing
Figure 2

Thus, fault recovery is no different from that of normal application processing following a transaction failure. Either the transaction is resubmitted, or it is aborted. A node fault has no more impact on the users than an application fault.

User Redirection

In many active/active systems, users are assigned to a primary node. Should that node fail, the users must be switched over to some surviving node. This is *user redirection*.

Just as with transaction redirection, the failover of users from a failed node to a surviving node must be fast and reliable. There are four ways that we will describe for switching users between nodes:

- client redirection
- network redirection
- server redirection
- DNS redirection

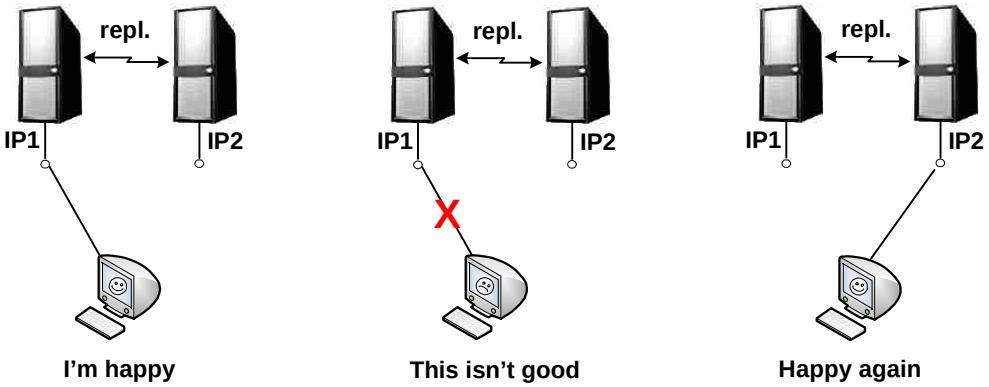
In Part 1 of this article, we will discuss client and network redirection. Server and DNS redirection will be covered in Part 2.

Client Redirection

Client redirection is perhaps the simplest method for switching users. However, it depends upon the ability to build some intelligence into the client, whether it be a browser, an ATM, or some other device.

Client redirection depends upon the client's ability to know at least two routes, each one to independent processing nodes. These routes may be known to the client as IP addresses, or they may be URLs. One route is to the client's primary processing node, and the other is to its backup node.

The client normally sends traffic to its primary node (Figure 3). However, if it should find that its primary node is nonresponsive, it shifts its traffic to its backup processing node. It should then periodically query its primary node with a ping or an application-level message. When it determines that its primary node has recovered, it can once again send its traffic to that node.



Client Redirection
Figure 3

This procedure is similar to the technique described above for transaction redirection. The client can be implemented to retry a failed transaction a configured number of times, or it can simply switch to its backup node on the first transaction failure. If the failure were transient in nature, the client will determine that upon the next query of its primary node and will return to using that node. The failover time imposes no more of an impact on the user than a transaction retry during normal processing.

If the active/active system is a multinode architecture with more than two processing nodes, a client can be given a list of backup nodes in priority order. In this way, multiple node failures can be accommodated provided that the surviving nodes can carry the system load.

An alternative strategy is to have a client round-robin its transactions to each node, one node after the other. Should it fail to receive a response from a node, the client removes the node from its round-robin list. It then begins to query the failed node periodically to detect the node's return to service, at which time the node is returned to the round-robin list.

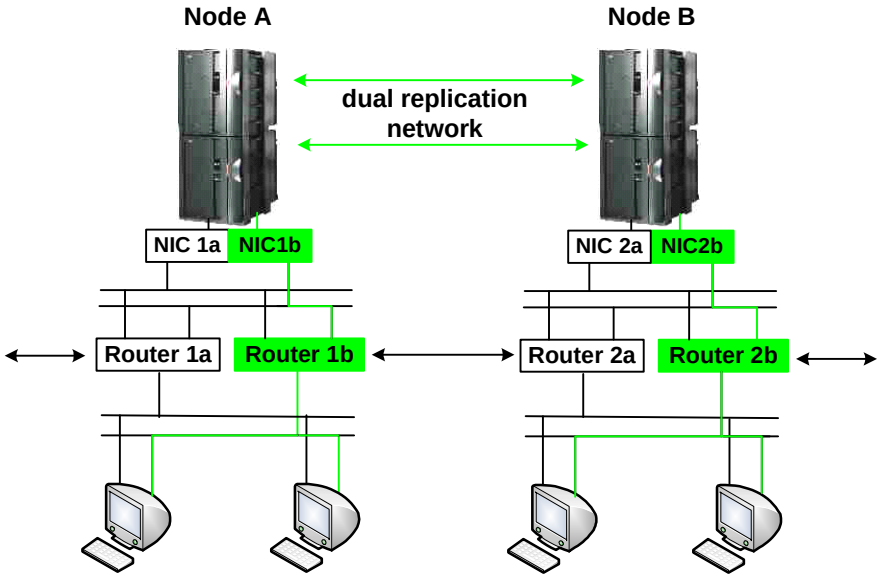
It is a good practice to also allow the client to switch processing nodes upon an external command. This provision can be useful to take nodes out of service for maintenance or to load-balance the application network.

Client redirection can also be useful in conjunction with transaction redirection, described in the previous section, to allow a client to connect to a backup transaction router if necessary.

Network Redirection

With network redirection, the switching of users from one node to another is the responsibility of the network. A typical configuration for an active/active network is shown in Figure 4. It comprises two nodes, Node A and Node B. The nodes are geographically separated, and all components are replicated with a backup component so that there is no single point of failure.

The databases of the two nodes are kept in synchronization via data replication over a dual-WAN network. Clients are connected to dual local routers via dual LANs, and the routers are connected to each's local server via dual LANs feeding dual Network Interface Cards (NICs). The routers are also connected to routers at the remote node over a bridged WAN. Thus, there is a backup path for any single failure in the application network.



A Two-Node Active/Active System
Figure 4

During normal operation, clients at the Node A site connect to their server via Router 1b using one rail of the client LAN. Router 1b passes traffic to NIC 1b of Node A via one rail of the server LAN. Clients at the Node B site connect to their server in a similar fashion, using Router 2b to connect to NIC 2b.

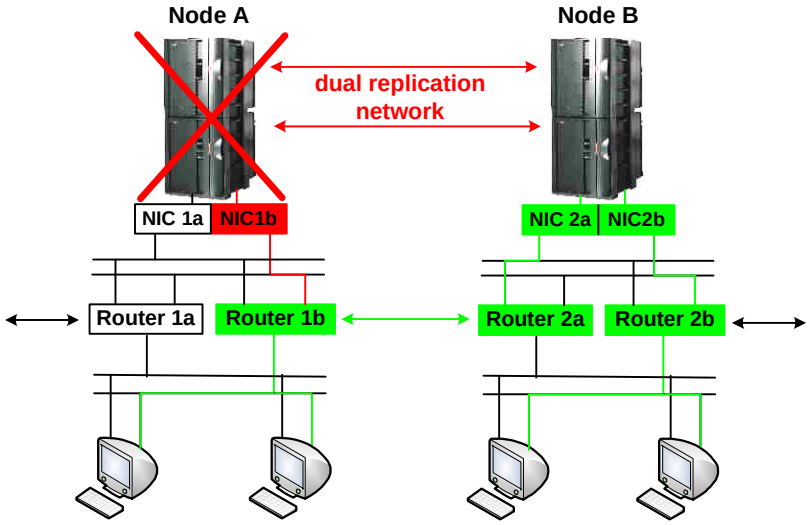
Let us now see what happens if Node A should fail, as shown in Figure 5. Router 1b finds that it can no longer talk to NIC 1b on Node A. Looking in its routing table, it finds that its alternate route in this case is to Router 2a in Node B.[2] Therefore, it passes all further traffic from its clients over the bridged WAN to Router 2a, which passes the traffic to NIC 2a. All users have now been switched from Node A to Node B.

Any other component fault is recovered in essentially the same way via alternate routing. Should a NIC or a server-side LAN rail fail, the router can get to the other NIC on the server via the alternate server-side LAN rail. Should a client-side LAN rail fail or should a router fail, clients can

[2] Alternatively, Router 1b could first try to communicate with NIC 1a over the other server-side LAN rail before deciding that the server is down and that the need exists to route traffic to Node B via Router 2a.

reach the other router via the alternate client-side rail. Thus, no single subnet fault will take down a node.

Of course, in order to support a client-side, dual-rail LAN, some intelligence is required in the client to allow it to choose a LAN rail on which to communicate. In some cases, it may be decided not to use dual-rail LANS if the client does not have this capability or if the reliability of a single LAN is deemed to be compatible with the availability requirements of the application. The same consideration can be applied to the dual-rail server-side LAN.



Router Redirection
Figure 5

The client can use a virtual IP address that is translated by the router so that the client need have no knowledge of where its traffic is ultimately being directed.

Connection and Session Loss

One problem with user redirection when switching to another node is that in the typical configuration, a node knows nothing about the connection and session context of a client that has just been switched to it. If no other provision is made, the client will get notification of the connection and session loss and will have to reestablish them.

If the client is a browser or a PC application of some sort, and if connection/session reestablishment must be done manually by the user, then the user is certainly aware of the outage, short though it may be. However, if the client has the intelligence to automatically detect this condition and to reestablish the connection and the session, the user will be unaware of the outage; and true continuous availability has been achieved.

What's Next?

In Part 1 of this article, we have described how rapid and reliable failover can be achieved by rerouting transactions or by switching users to surviving nodes via client redirection or network redirection.

In Part 2, we will discuss switching users via server redirection and DNS redirection. We will find that server redirection can be complex and that DNS redirection can exhibit long failover times.