the Availability Digest

www.availabilitydigest.com

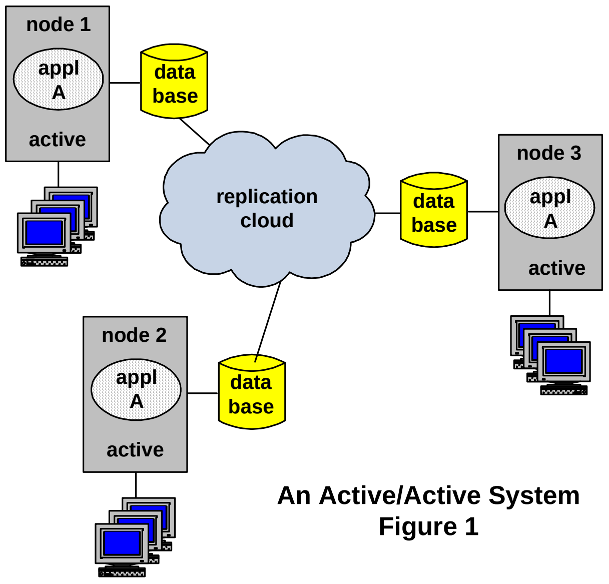
Achieving Fast Failover in Active/Active Systems - Part 2

September 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.

An active/active system (Figure 1) comprises two or more geographically-separated processing nodes, each actively participating in a common application using a local synchronized copy of the application database. The database copies in the replication network are kept in synchronism via data replication.

Should a processing node fail, all that needs to be done is to switch users from the failed node to one of the surviving nodes. Since it is known that the surviving nodes are operational because they are actively processing transactions, failover is risk-free and reliable.



An Active/Active System
Figure 1

However, how can users be redirected to other processing nodes so quickly that they are not aware of the fault and the recovery action, or at least are not inconvenienced by it? In Part 1 of this series, we talked about redirecting users via client intelligence or via network intelligence. In this Part 2, we explore server redirection in which faults are detected and users redirected via intelligence in the servers.

Server Redirection

Fault Detection

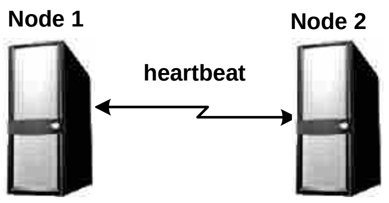
Heartbeats

Before users can be switched from a failed node, it must be known that the node has failed. With client redirection and network redirection described in Part 1, this is the responsibility of the

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

clients or the network. With server redirection, the nodes themselves must be able to detect a fault in another node.

The common technique to do this is to use heartbeats between the nodes, as shown in Figure 2. Each node generates a heartbeat – in effect, an "I'm alive" message – to all of the other nodes in the application network. Should a node detect the loss of a heartbeat, then it assumes that the other node is down and it can initiate recovery action.



Heartbeat Monitoring
Figure 2

A heartbeat can be a simple ping, or it can be generated at a higher level in the application stack. The problem with a simple ping is that the system may be able to respond to the ping even if it cannot respond to transaction requests, either due to congestion or to some higher level fault. The farther up the application stack that the heartbeat is generated, the more reliable it will be. The ultimate is for each application to generate its own heartbeat so that the failure of individual applications in a node can be detected.

A heartbeat may be a simple "I'm alive" message, or it may carry additional information describing the node's health. For instance, it might include the failure of redundant components that, though not fatal, result in single points of failure. It might include congestion information (queue lengths) or performance metrics. A node can be designed to take over the users of another node only if it is healthier than the other node.

Heartbeat messages may be bidirectional in that such a message sent to another node expects a response from that node, perhaps with equivalent information. Alternatively, the data replication channel may serve as a heartbeat. So long as replicated data is being received from a node, it can be assumed that the node is up. If a node has no replicated data to send, it sends a heartbeat message instead.

Since the heartbeat message is the primary fault-detection method in an active/active system, it is imperative that it be reliable. Should the heartbeat network fail, both nodes on either end of the connection will assume that the other node has failed, a potentially disastrous condition leading to a "tug-of-war". The tug-of-war syndrome is discussed in the next section.

Therefore, heartbeats should be sent over a non-switched redundant network (i.e., over direct connections). If the replication channel is used, this is often a redundant network anyway. As a further backup, the user network that connects clients to the processing nodes might be used to broadcast heartbeat messages should the primary heartbeat network fail.

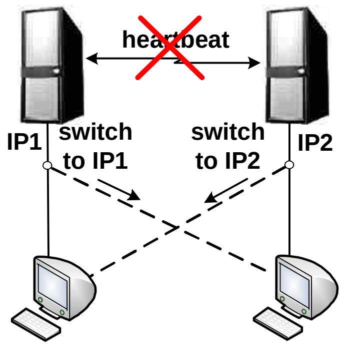
Typically, nodes are forgiving when monitoring heartbeats. It is expected that a heartbeat will occasionally be lost or be late. Therefore, it is common to not declare a fault until a certain number of heartbeats, typically three, are lost.

One important heartbeat tradeoff is network load versus fault detection time. To truly achieve continuous availability, fault detection should be very rapid, typically measured in the subsecond range. This means that each node must send several heartbeats per second. However, rapid heartbeats may significantly increase network load. The more complex a heartbeat message is, the more heartbeats that must be missed before declaring a fault, and the shorter the desired fault-detection time all contribute to increased network load. However, if the number of nodes in the application network is small, heartbeat traffic may not be a significant factor in network load, and fast fault-detection times – and thus fast failover times – can be achieved.

The “Tug-of-War” Syndrome

If heartbeat messages are lost due to a network failure, both nodes on either end of the heartbeat connection will assume that the other node has failed and will try to switch the users currently connected to the other node to itself. Since each node is attempting to do this, a tug-of-war develops (Figure 3) in which users may be switched back-and-forth between nodes, a situation that might collapse the active/active network.

There are several solutions to this problem. One is to not use automatic failover, but rather to do it manually by operator action. In this case, the operator will be notified of a system fault. If upon investigation it turns out to be a loss of heartbeats, the operator can decide to continue on with two nodes or to switch users to one node until the heartbeat network can be repaired. Of course, if manual switchover is used, continuous availability has been sacrificed. It may take many minutes to analyze the fault, to decide on the appropriate corrective action, perhaps to get management approval, and then to effect the failover.

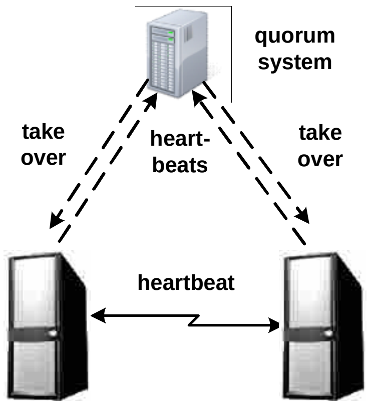


Tug-of-War
Figure 3

Another technique is to provide a cluster lock that must be acquired prior to switching users. This lock may be a global mutex on some other system accessible to the active/active nodes. If heartbeats are lost, both nodes will attempt to acquire the cluster lock, but only one will be successful. This is the node that will switch the users of the other node to itself.

A Quorum System

A third technique, which is used commonly in classic clusters,[2] is to include a quorum system in the active/active application network as shown in Figure 4. A quorum system is an independent system that is responsible for monitoring nodal heartbeats and for making switchover decisions. It is the quorum system that receives the heartbeat messages from all nodes.



Quorum Monitoring
Figure 4

If the quorum system detects a nodal failure, it will direct the surviving nodes to acquire the users connected to the failed node. If a fault in the heartbeat connection between a node and the quorum system should occur, the quorum system will assume that the node has failed and will switch that node’s users to surviving nodes. In any event, since failover decisions are being made by an independent third party (the quorum system), there is no tug of war.

Split-Brain Mode

The quorum system solves another problem that can arise in active/active systems, and that has to do with the loss of the data-replication network. Should the data replication network be lost, then changes made by one node cannot be replicated to the other node. If nothing else is done, the two nodes will continue to process transactions, queuing their changes to the other node until the time that the replication network is restored and the queued changes can be drained to the

[2] Active/Active Versus Clusters, *Availability Digest*; May, 2007.

opposite nodes. During the resynchronization process, data collisions are bound to occur due to both nodes updating the same data objects.

During the time that replication is down, the databases at the two nodes will diverge. This condition is known as “split brain.” There are two actions that can be taken when this type of fault occurs. One is to continue on as is. If data collisions cannot occur or if their resolution can be handled automatically, the only penalty is processing transactions against stale data.

However, if data collision resolution is complex, or if processing against stale data is unacceptable, then one of the nodes must be shut down. As with the tug-of-war syndrome, this can be a manual operator action.

However, if a quorum system is in use, it can resolve this problem automatically in the same way that it resolves tug-of-war situations. If two nodes report that they have lost data replication between them, the quorum system can direct that one of the nodes shut down and that its users be switched to the other node.

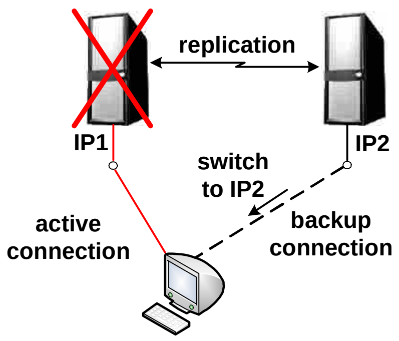
Commanding Switchover

Once a node has determined that it is to acquire users from another node, there are several ways in which it can proceed.

Commanding Clients

The node can send supervisory messages over the user network to the clients that it wishes to acquire, as shown in Figure 5. These messages are commands to the clients to switch the IP address to which they are directing traffic to the IP address of the node doing the acquisition.

This technique requires that the clients have enough intelligence to be able to process these messages and to switch IP addresses. If there are many clients and there is no multicast or broadcast capability for the acquiring node to easily reach all of the impacted clients, then individual takeover messages will have to be sent by the acquiring node. This process could introduce an undesirable delay in the failover procedure.



Commanding Client
Figure 5

Commanding Routers

If the clients in the active/active network have no special network intelligence, they know only how to connect to a single IP address. This IP address can be a virtual address that does not correspond to any particular nodal address. Rather, it is the responsibility of the routers in the network to translate the virtual IP address into a real IP address for message delivery.

The advantage of using a virtual IP address is that network routing can be managed to determine to which real IP address a virtual IP address should be routed. This routing can be changed to reroute users from one node to another.

Thus, a node may send directives to routers in the network to redirect traffic destined to a virtual IP address from the failed real IP address to its real IP address. This rerouting depends upon the routers having the capability to accept such directives over the network. This technique solves the problems associated with the client-command technique. The clients do not have to have any specific network-redirection capability. They continue to send messages to a common virtual IP address. It is the routers that translate the virtual IP address to a specific node's IP address. Furthermore, network traffic created by rerouting commands is reduced since these commands need be sent only to the routers, not to all of the clients.

Gratuitous ARP

Another way in which a virtual IP address may be migrated to another node is to use a gratuitous ARP. To understand a gratuitous ARP, it is first important to understand the ARP protocol.

ARP is the Address Resolution Protocol used by routers and hosts to maintain in their cache a mapping of IP addresses to device interface addresses on the subnets to which they are connected. A device interface address is typically a media access control, or MAC, address provided by a network interface card (NIC). This mapping allows the router or host to determine to which physical device it should send a datagram that it has received based on the destination IP address in the datagram. Routers will periodically send their routing tables to neighboring routers. In this way, routing tables are self-discovering. They always keep up-to-date on the current network topology.

Since every router or host on a subnet can listen to all datagrams, the source and destination IP/MAC address pairs in each datagram are used by the routers and hosts on the subnetwork to update their caches with the IP/MAC address mapping.

If a host does not have a destination on its subnet to which to forward an incoming message with a specified IP address, it broadcasts an *ARP request* over its subnet and asks for the MAC interface on the subnet that is servicing that IP address. The appropriate device will respond to the sender with its MAC address. As with any datagram, the ARP request and reply will cause all routing tables to be updated with the source and destination address pairs contained in those datagrams.

A gratuitous ARP[3] is a form of ARP request in which the sender asks for the MAC address corresponding to its own IP address. As described above, the ARP request will carry the sender's IP address and its MAC address. Even though a reply to the ARP request will probably not be received, each router on the subnet will update its routing table by associating that IP address with the MAC address of the sender. Thereafter, any traffic destined for that IP address will be sent to the sender's MAC address. Sending a gratuitous ARP request is the mechanism often used by a host to advertise its presence on a subnet.

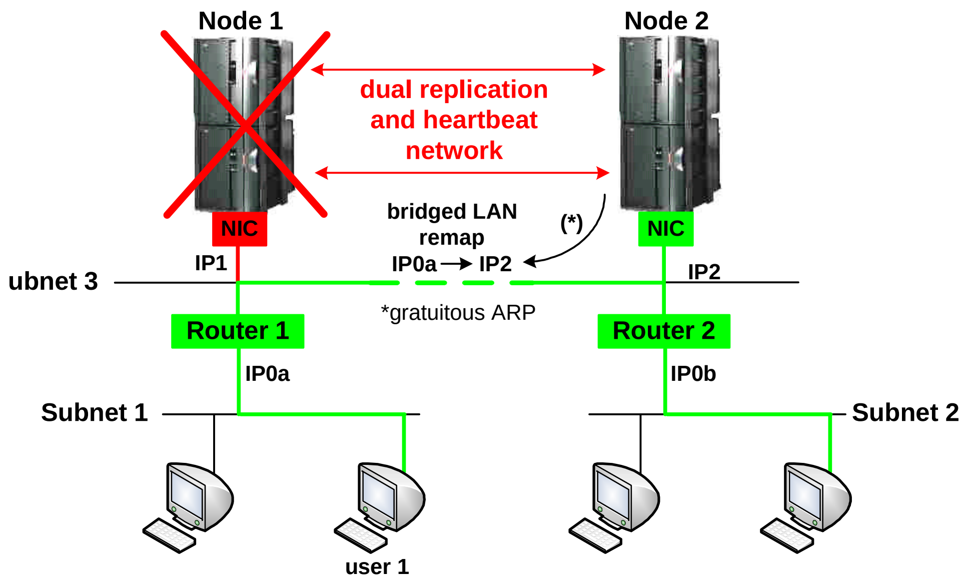
However, the gratuitous ARP can also be used to seize an IP address from another host. If a node in an active/active system determines that another node is down, it can reroute the virtual IP address of the users serviced by the downed node by sending out a gratuitous ARP advertising that it is now the destination for that IP address. Thereafter, all traffic from that virtual IP address will be forwarded to the new node. That node has redirected all users connected to the failed node to itself.

The use of gratuitous ARP to redirect users to a surviving node is shown in Figure 6. (In this figure, redundancy of network components is not shown for simplicity; but in a highly-redundant architecture, subnets 1, 2 and 3, routers 1 and 2, and the NICs would be replicated.)

There are two nodes in the active/active network – Node 1 and Node 2. The community of users in Figure 6 is separated into two sets. One set of users connects via Subnet 1 to Router 1 using the virtual IP address IP0a. The other set of users connects via Subnet 2 to Router 2 using virtual IP address IP0b. The routers and nodes can talk to each other over Subnet 3, which is a logical

[3] W. Richard Stevens, TC*P/IP Illustrated, Volume 1*, Addison Wesley Longman; 1994.

LAN bridged over a WAN to connect the nodes. The nodes use separate redundant data-replication links for database synchronization.



Gratuitous ARP
Figure 6

During normal operation, Router 1 connects all IP0a traffic generated by the users on Subnet 1 to Node 1. Likewise, Router 2 routes all IP0b traffic generated by the users on Subnet 2 to Node 2.

However, should Node 1 fail, Node 2 issues a gratuitous ARP advertising that it is now the destination for all IP0a traffic. Router 1 will update its routing table (as will Router 2, but this is inconsequential) so that all traffic received from Subnet 1 with a destination address of IP0a will now be routed to the MAC address of Node 2. Node 2 has now seized all of the users originally connected to Node 1.

When Node 1 is restored to service, it will send the equivalent gratuitous ARP to return its users to it.

DNS Redirection

There is one additional redirection method to discuss, and that is using DNS (Domain Naming Services) redirection. Often, when a client wants to access a remote server, it knows that server via a URL (Universal Resource Locator). A URL is an easy-to-remember name, but is meaningless to an IP network. The URL must be converted to the destination node's IP address. This is the function of the DNS server. Every network has a DNS server (typically in a redundant configuration) or uses the DNS services of the network's ISP.

Therefore, one way to reroute users to another node is to have them address the server via a URL. Each user request will then be mapped to an IP address by consulting the DNS server. If it is desired to route the user requests to another node, all that needs to be done is to modify the URL mapping in the DNS server so that the URL points to the new IP address.

While this technique is simple to implement, it has one big problem. DNS entries are cached for fast access. It may take several minutes for a DNS update to be reflected in the DNS cache. During this time, URL requests will still be routed to the old node. Thus, continuous availability cannot typically be achieved with DNS redirection.

Summary

Continuous availability requires that there be rapid and reliable recovery from a fault. Recovery must be fast enough so that users are unaware, or at least are not inconvenienced, by the fault.

Fault recovery generally requires several steps:

- The fault must be detected.
- The cause of the fault must be determined.
- It must be decided whether it is faster to fail over to the backup system or to try to recover the failed system.
- Management approval may be required to do a failover.
- The failover procedure must be invoked. This can require:
    - rebuilding the database.
    - starting applications.
    - reconfiguring the network.
    - testing the backup system before putting it into service.

It is for these reasons that failover can often take hours.

With active/active systems, once a fault is detected, there is no need to concern oneself with the other steps. Just fail over. It is known that the system to which you are failing over is operational because it is already processing transactions.

Failover itself, whether using user redirection, network redirection, or server redirection, can be very fast. It can be achieved in subseconds. The costly step from a time viewpoint is fault detection.

Therefore, to achieve continuous availability, the rapid detection of faults is imperative. We have described in this series various ways that faults can be detected – by clients, by routers, by nodes. The secret to fast failover is to localize the failover logic to the device doing the fault detection. If the client is detecting faults, the client should initiate the failover. The same is to be said of routers and nodes.

Some fault detection approaches are proactive and some are reactive. A proactive approach can determine a fault and can failover before many users are affected by the fault. Heartbeats and router detection of downed paths are examples of proactive fault detection. A reactive approach detects a fault when it causes a user request to fail, The sensing of a fault by a client due to a rejected transaction is an example of reactive fault detection. Clearly, proactive approaches will provide better availability than reactive approaches.