8. Setting Up Redundant Networking

8.1 IP Address Takeover (IPAT)

Client machines and applications usually cannot choose on the fly between several IP addresses to access a server application. Instead, the takeover node has to provide for a functionality that makes sure it acquires the well-known IP address prior to re-starting the business critical application. This process is known as IP address takeover (IPAT). This basically works as follows.

A cluster node which has two IP interfaces per network boots with its configured boot address on its Service Interface and the standby address on the Standby Interface. If it is the higher priority node, the Service Interface will switch to the service address as soon as the node acquires the resource group. In turn, when a takeover node acquires the service IP address, its Standby Interface will take over the service address of this resource group. In the following example, the service address of the resource group is **192.168.20.11**.

Interface	Node 1 (primary)	Node 2 (secondary)
Service	192.168.20.10	192.168.20.20
Standby	192.168.21.10	192.168.21.20

This is the situation before Linux-HA is started. Both Service Interfaces are on their respective configured boot addresses. The Standby Interfaces are on their standby addresses. Now Linux-HA is started on both nodes of which Node 1 is the primary node. It will acquire the service address.

Interface	Node 1 (primary)	Node 2 (secondary)
Service	**192.168.20.11**	192.168.20.20
Standby	192.168.21.10	192.168.21.20

Now let us imagine that the Service Adapter of Node 1 fails. The Standby Adapter will take over the service address (event `swap_adapter`):

Interface	Node 1 (primary)	Node 2 (secondary)
Service		192.168.20.20
Standby	**192.168.20.11**	192.168.21.20

Both nodes (more exactly: all surviving nodes) will also run a `fail_standby` event against 192.168.21.10

Instead if Node 1 fails, Node 2 will acquire the service address contained in the resource group:

Interface	Node 1 (primary)	Node 2 (secondary)
Service		192.168.20.20
Standby		**192.168.20.11**

In this simple case we could as well move the service address to the Service Adapter of Node 2 but is simpler to move it to the Standby Adapter. First, the Standby Adapter is no longer needed anyway (Node 2 will run a `fail_standby` event against its Standby Adapter, and all other surviving nodes will run the same event for this adapter), and second, Node 2 might as well be running a resource group (Mutual Takeover). The logic to move the Service IP Address is simpler if the address is always moved to the same (i.e. the Standby) adapter of the surviving machine.

Four network adapters to make one IP address HA. Wow. In principle we could use only one adapter per node. But there are two good reasons why we don't want this. First, if the adapter in one node fails, it has no way to tell if only the adapter failed or the entire physical network. Second, if we added the logic to tell if it's an adapter or a network failure, we would immediately have to perform a node failover which takes much longer than the local adapter failover. We want to minimize downtime, remember? But there's a way to reduce the adapter overhead: set up Mutual Takeover whenever possible.

Commercial solutions sometimes offer an option to put both service and standby addresses on one adapter, using IP aliasing. While this is also possible with Linux, I do not recommend this, a) due to the difficulty described above and b) due to the low prices of ISA and PCI ethernet adapters. These prices render the potential hassle when using only one adapter completely useless.

There is a caveat to IPAT. If the takeover node has taken over the service address, it has two IP interfaces in the same subnet (see last example above). If the source IP address is critical for the client, a dedicated host or net route potentially needs to be set up via the service interface. When using Mutual Takeover, this can become a problem. Suggestions appreciated.

8.2 MAC Address Takeover

When an IPAT occurs, client machines are confronted with a different MAC address for the same IP address. According to RFC 826, all IP implementations should update their ARP caches if an ARP request occurs for which there is already an IP address in the cache. Thus, we could simply flush the takeover node's ARP cache after an IPAT and ping the clients. The ping leads to an ARP request which in turn updates the client's ARP cache.

The logic to perform the client ARP refresh looks like,

- flush local ARP cache
- for each client in client_list
  - set temporary host route via IP interface to be refreshed
  - ping client
  - delete temporary host route

Unfortunately, not all IP stacks support the functionality required by RFC 826, the old Windoze 3.11 WINSOCK.DLL being a good (bad!) example. Some IP stacks for DOS lack this too.

Unix client machines can also be refreshed by a simple shell script running on an arbitrary TCP port from

`inetd` which is triggered by the takeover cluster node (using `telnet <client_name> <port_number>`). The shell script simply flushes the client's local ARP cache; no input/output is required which makes this feature quite secure. Although this is fairly simple and straightforward, I do not recommend this because it requires additional administration effort for each client machine on the network.

If clients access the servers via a router, they need not be refreshed because the MAC address is only valid within a physical network segment. The router will have to be refreshed instead.

This situation can be remedied by taking over the MAC address as well. This is already possible for many ethernet cards. All one need do is, e.g.:

```
ifconfig eth1 hw ether 00:00:C0:0C:29:D0
```

It's done via a simple ioctl; c.f. the ifconfig man page for more details. Works fine here. Be careful to pick an unused hardware address for the service interface. Good examples are those starting with 0xdeadbeef, or replace the last two bytes with the host part of the IP address, e.g.

```
192.168.20.156 -> 00:C0:87:A3:01:56
```

8.3 TCP vs. UDP

Since TCP based server programs usually keep an internal state of the connections in memory, TCP connections are lost when a takeover occurs (this also applies for TCP based proxy daemons like socks). Users will have to log in again. UDP in turn is a stateless protocol which makes takeovers happen transparently. A common example for this is NFS V2. HTTP 1.0 is also very easy to use because connections are very short; if a connection fails, users are familiar with pressing the browser's "reload" button... HTTP 1.1 might be different.

8.4 Cluster Node's Lifesign: Heartbeat

Linux-HA will check all network adapters by sending heartbeat packets across all of them (e.g. every 0.5 seconds). Heartbeat packets will be sent between service and service adapters, and between standby and standby adapters. Due to routing considerations, service/boot and standby adapters in a node must reside in different IP (sub-) nets. This ensures packets sent to another node's service/boot adapter are sent out via the local service adapter. The same applies for the standby adapters.

Heartbeat packets will be sent via IP Multicast to all active cluster nodes. If heartbeats are no longer received from a node, a timeout counter starts, and after a number of heartbeat packets missing, a failure of this node is assumed. This will be cross-checked with all surviving nodes to make sure all nodes have the same perception of the failure. Actions will only be taken if all surviving nodes agree an the nature of the failure.

Heartbeat packets will be UDP (connection-less, no SYN flood problem) and carry digitally signed (and possibly encrypted) information about the events in the cluster. Digital signatures will be used to ensure nobody from the outside fools the cluster. Heartbeat packets with wrong signatures will be discarded, and the other nodes will be informed about a potential break-in attempt (as well as the system administrator). For additional security, the IP packet filter in the kernel might be set up to accept heartbeat packets only from the other active cluster nodes.

Other heartbeat / communication means may also be considered, e.g. MPI.

Message types being exchanged will be

- Cluster Membership: Joining And Leaving
- Event Detection

- Sychronisation

8.5 Keeping Track Of The Network Status

There are some basic guidelines concerning networks:

- If a single network adapter stops sending and receiving heartbeats, that adapter is marked as failed.
- If all the network adapters on a network stop sending and receiving heartbeats, that network is marked as failed.
- If all the network adapters on a node simultaneously stop sending and receiving heartbeats, that node is marked as failed.

8.6 Non-IP Heartbeat

The IP subsystem is a SPOF for a single machine. In the event of a IP subsystem outage, the affected node can no longer send nor receive heartbeat packets. This can lead to a situation where two nodes think the other node is down (partitioned cluster) and try to acquire resources which are owned by the other node.

This can be prevented in two ways.

- Run a non-IP heartbeat via
  - a point-to-point RS232 null modem cable running plain TTY (no SLIP or PPP). Virtually all PC's have two RS232 interfaces none of which is normally used on a Linux-HA node (no X11, no mouse). If there are not enough RS232 ports (mouse, radio clock receiver), a cheap ISA two- or four-port serial card may be used.
  - SCSI Target Mode, i.e. run a RS232-like TTY connection across a SCSI connection; the adapter must support this. SCSI target mode may run into heartbeat timeouts because the SCSI bus is arbitrated and may be used by an initiator for a very long time. Simply attaching a null modem cable between two neighboring nodes is the better solution. Since some folks want to use point-to-point SCSI connections as high speed networks, SCSI Target Mode is currently in the works (see also IP Encapsulation in SCSI Driver. According to the author, the "driver could conceivably support moving any type of data over a SCSI bus.").
  - Target Mode SSA which is far better suited than Target Mode SCSI for running heartbeats due to no arbitration being involved and data being transferred in 128 byte packets, which means the SSA device driver should provide for SSA Target Mode.

  Since non-IP heartbeat networks are point-to-point, each node must forward messages which are addressed to a node other than itself to this node.
- Set up a "host ping file" on each node, containing IP addresses or resolvable names of other machines in the network. If a IP subsystem failure is suspected, the machines mentioned in this file will be pinged. If answers are received, the local IP subsystem is fine, if not, it is down. Especially when the nodes have only one network adapter (this makes only sense for rotating resource groups) this is the only way to tell a network failure from an adapter or node failure.

If a node finds its local IP subsystem is down, it will release the resources and stop Linux-HA gracefully.