Network Level Resiliency for High Availability (HA) in Ethernet Networks

ZNYX NETWORKS

Providing HA and Service Continuity
for Embedded Ethernet Solutions

TABLE OF CONTENTS

1 Introduction 1

1.1 Defining Availability 2
1.2 Causes of Downtime 3
1.3 Improving Availability 3
1.3.1 Fault Tolerance 5
1.3.2 Redundancy Schemes 5
1.3.3 Enhancing Network Application Level Availability with OA/Node 7
1.3.4 Enhancing Network Application Level availability with OpenArchitect/HA 8
1.3.5 Combining OpenArchitect/HA and OA/Node 10

2 HA for End Nodes 11

2.1 Types of Failures 11
2.2 Link Failure Recovery Requirements 11
2.3 Link Failure Detection 13
2.4 Fault Recovery 14

3 HA for Switches 15

3.1 Virtual Router Redundancy Protocol (VRRP) 15
3.1.1 OpenArchitect HA and Surviving Partner 16
3.2 Configuration Options 16
3.2.1 Active-Standby 16
3.2.2 Active-Active 17
3.2.2.1 Traditional VRRP Load Sharing 18
3.2.2.2 Load Sharing in a Switched Environment 19
3.2.2.2.1 Switched Interconnect 19
3.2.2.2.2 Active-Active Node Links 20
3.3 Switch Failure Modes 21
3.3.1 External Link Failure Detection 22
3.3.2 Internal Node Link Failure Detection 22
3.3.3 Inter-switch Link Failure Detection 23
3.3.4 Alarms 23
3.4 Internal Node Link Failure Recovery 23
3.4.1 Switch Fail-Over 24
3.4.2 Port Based VLAN Fail-Over 24
3.4.3 Port Fail-Over 24

3.4.4 Custom Failure Recovery Policies 24
3.4.4.1 Switch Fail-Over Due to "Goodness" 24
3.5 Inter-Switch Link Failure Recovery 25
3.5.1 Master Switch Failure 25
3.6 External Link Failure Recovery 25
3.7 Maintenance Process 25
3.7.1 Field Replacement 26
3.7.2 Automatic fail-back 26
3.7.3 Manual fail-over 26
3.8 Provisioning 27
3.8.1 Configuration 27
3.8.2 Software Distribution 27

4 HA Issues with PICMG 2.16 28
4.1 Basic PICMG 2.16 Topology 28
4.2 Optional Fabric Interconnect 29
4.3 Power On Synchronization 29

5 PICMIG 2.16 Chassis Interconnection Options 30
5.1.1 Single Chassis Interconnect 30
5.1.2 Multiple Chassis Interconnect (Star Configuration) 31
5.2 Multiple Chassis Interconnect (Helix Configuration) 32
5.2.1 Mitigating bandwidth decrease in the Helix topology OA/Node 33
5.2.1.1 Additional Bandwidth for "End" Chassis 33
5.2.1.2 Right handed Fail-Over 33

Introduction

The traditional isolation of data and telecommunication services on separate networks is vanishing. The new operational model is using one unified network powered by the ubiquitous Internet Protocol (IP). Service providers making this transition appreciate the operational and financial benefits of managing just one network. But they also are demanding a new level of IP service quality akin to conventional telecom services such as voice, which are rated by availability to the end user. For example, users always expect dial tone when they pick up a handset. They also expect the call will not be disconnected during a conversation. The same presumption of continuous availability governs use of video services. Service providers adopting the all-IP network model must incorporate traditional rigorous levels of continuous uptime as they merge multiple services on one network. Downtime causes user dissatisfaction and costs money. Downtime at the wrong time, such as during a 911 call, could mean the difference between life and death.

Like the proverbial weakest link in a chain, system availability is determined by its least available "service providing" component. ZNYX Networks solutions ensure that the IP network is not the weakest link in an embedded system. This white paper discusses network level availability and how to create High Availability (HA) embedded systems with ZNYX Networks OpenArchitect/HA Suite.

1.1 Defining Availability

Availability is the *proportion of time that a system can be used for productive work*. However, network reliability is required to achieve availability. Reliability is defined as the *provisional probability that an element will operate without fault in a given period*.

Two different measures are used to quantify these values:

- Mean Time Before Failure (MTBF), which is the a*verage time from an initial fault free state to a failure*. This is a measure of reliability.
- Mean Time To Repair (MTTR), *the average time taken to diagnose a failure and repair it*.

Availability is then defined using those two parameters as:

Availability = MTBF / (MTBF + MTTR)

This equation shows that availability can be improved by either increasing MTBF or by reducing MTTR. It is impossible to create systems that never fail, so minimizing MTTR values will increase availability. Carrier class systems have traditionally provided MTTR targets of less than 50 milliseconds , which is the specification for SONET ring recovery time.

Availability is quantified using the percentage of uptime (or expected down time) provided by a system. For example, a system with 99.9% availability would expect to have about nine hours of downtime a year. A system with 99.999% availability, often referred to as 5 nines availability, limits downtime to about 5 minutes a year.

Carrier class equipment availability targets are in the 5 nines range.

1.2 Causes of Downtime

A system is composed of several hardware and software components. Faults in the network can be caused by any of these components. Some examples include:

1. Hardware failure
2. Network failure
3. Operating system failure
4. Application failure
5. Human error
6. Environmental calamities (earthquakes, floods, storms, etc.)
7. System overload
8. Security breach
9. Calamities caused by terrorism

Downtime could also result from a planned system upgrade or maintenance, especially, if these activities cannot be done while this system is running or require the system to be restarted or made inactive for some period of time.

1.3 Improving Availability

Regardless of what factor may cause downtime, correcting system outages involves the following workflow:

1. Detecting the fault
2. Fault diagnosis
3. Fault isolation
4. Recovery
5. Repair

Each of these steps contribute to the length of time it takes to recover from a failure. The average total time taken performing these steps for various failover scenarios is quantified in the MTTR for a system. Therefore, minimizing the time spent performing any or all of these steps will improve system availability.

Network availability plays a crucial role in the design of systems today, especially given the popularity of distributed applications facilitated by the Internet. The all-IP network essentially acts as the "system bus" to these applications. The following example is used to illustrate the challenges facing system designers.

In a simple network setup (Fig. 1), a switch or router is the network element connecting multiple nodes. For restoring availability in this setup, faults are first detected when things stop working. The administrator then performs some manual intervention to diagnose the fault. The administrator next isolates the problem, and if it is a simple problem such as removing faulty routes, the administrator can easily recover the system. However, if it is a more serious issue such as faulty hardware, repair may require replacing the faulty switch, reconfiguring the new one and restarting to restore the service of its predecessor. In this example, MTTR is very high, which means availability is very low.

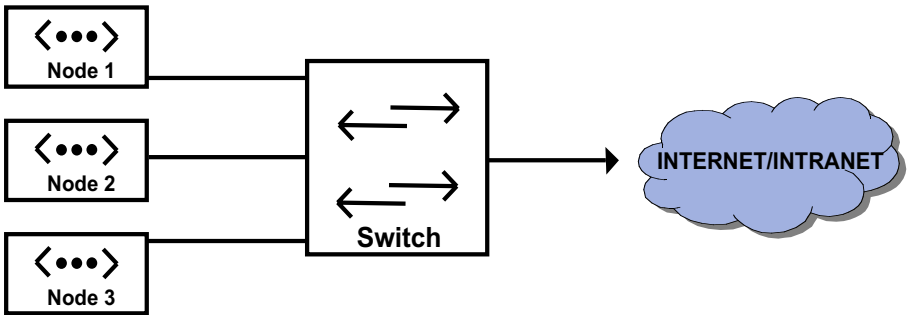


Figure 1. A Simple Network

The system described above is unacceptable for a mission critical application. Improving availability requires software that detects, diagnoses and isolates the fault before notifying an administrator to repair the system. A preferable design would include an automated system that circumvents faults.

1.3.1 Fault Tolerance

Fault tolerance is the ability of a system to withstand one or more faults without causing loss of service. Some systems are nearly immune to faults, while others are designed to degrade gracefully – even under multiple fault conditions. The key to designing any HA system is to anticipate potential failures and to implement appropriate recovery strategies.

Fault tolerance is achieved with a combination of software and redundant hardware components. The software detects, diagnoses and isolates faults. HA software then compensates for the fault and repairs it by switching over to the healthy standby hardware components.

1.3.2 Redundancy Schemes

Redundancy within a system can be achieved through two methods: N redundancy or N+M redundancy, where N is the resources and M is the number of spares. In the N redundancy scheme, multiple identical resources, (2N, 3N, 5N, etc.), are allocated for fault recovery. The simplest is the 2N setup, which duplicates resources in the system. The N+M redundancy scheme provides individual spares for a group of resources. N+1 is the simplest case, where there is 1 spare for the group of resources. Fig. 2 shows the difference between the two schemes.

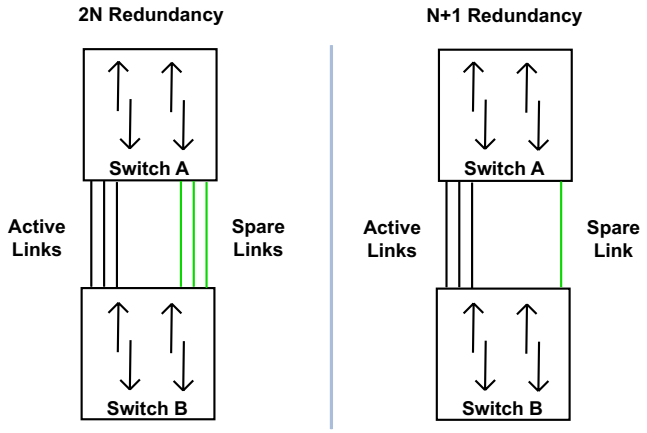


Figure 2. Difference Between 2N Redundancy and N+1 Redundancy

The advantage of 2N redundancy is that it is less complex and potentially faster to recover from faults, since the spare resources can be pre-configured. In addition, 2N redundancy may be able to tolerate multiple failures. The disadvantage is that it is more expensive. N+M, makes the most sense in systems where a large number of similar resources need protection. N+M is more complex to implement, especially if the spares are for many different configurations. As component costs continue to decline, the cost savings in N+M may not offset the added complexity.

In Fig. 2, the 2N redundancy system has three active links, each with a spare. The N+1 redundancy setup has three active links and one spare. If each link had different configuration parameters, the system on the left could have each spare configured appropriately, while the system on the right would have to have its spare dynamically configured during fault recovery. The left system could potentially lose all three active links and still provide full service, whereas the right system can lose only one link before a service-affecting outage occurs. The decision to implement one scheme or the other is dependent on the weighted balance of risk versus cost.

There is another advantage of 2N redundancy – a system can be configured into an *Active-Active* setup in which the spare resources could actually be put into service as opposed to just being idle. However, it is important to configure the system in such a way that the system does not depend on spare resources to provide service for this creates a single point of failure. Systems where spare resources are used only when the active ones fail is called an *Active-Standby* configuration.

1.3.3 Enhancing Network Application Level Availability with OA/Node

For the most part, network application designers have viewed the network as a necessary evil. They have been conditioned to believe that networks are inherently unreliable, and have had to rely on numerous end-to-end failure detection and recovery schemes to work around network problems. End-to-end failure detection often uses a heartbeat mechanism that is relatively slow – typically several seconds – to detect problems and which can only indirectly detect link level failures. As a result, although a problem can be detected, the exact cause and its ultimate resolution usually cannot be determined. In order to significantly improve link failure detection, ZNYX Networks' OpenArchitect/HA Suite, and associated OA/Node software provides an orthogonal approach to the end-to-end schemes.

OA/Node is driver-level software that provides the functionality to manage link-level fault detection and recovery, and is designed to augment end-to-end schemes. It provides **IP Transparent Fail-Over** capability, allowing applications to continue uninterrupted when a link failure occurs (see Fig. 3). The software hides the fact that there are multiple physical interfaces connected to the network from the application – the native Operating System only sees one virtual interface. This abstraction allows the existing network application to run unchanged on top of OA/Node. The physical interfaces may be on the same or separate network interface cards, allowing for fully redundant network hardware.

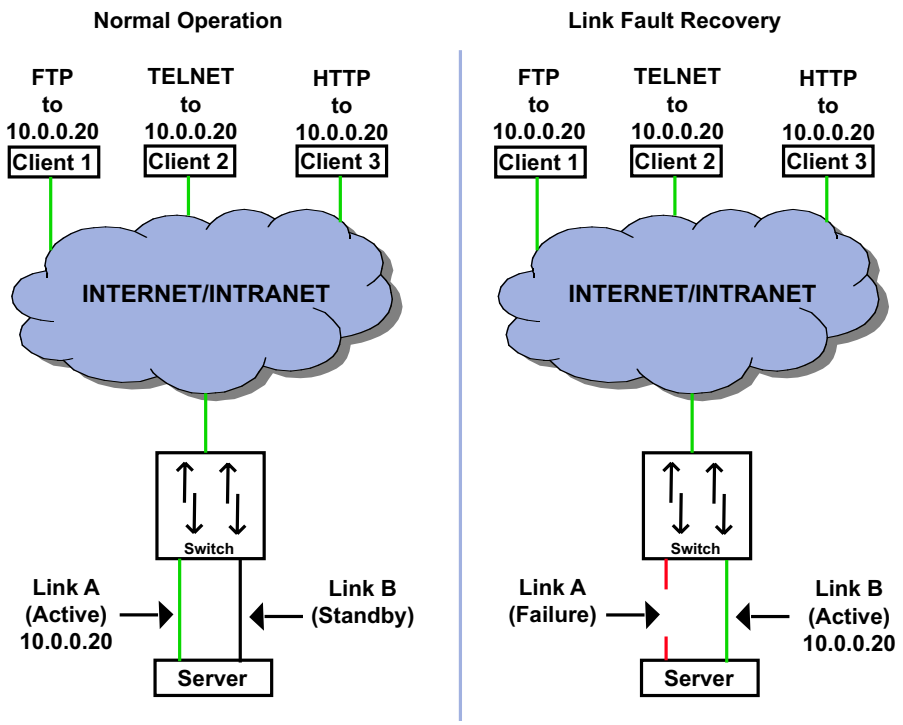


Figure 3. IP Transparent Failover

In the Fig. 3 example, client systems access the server via IP address 10.0.0.20. The left diagram shows that Link A is the active link that responds to client requests destined for IP address 10.0.0.20. When Link A goes down (i.e. from a cable cut), Link B activates and assumes the IP address as shown on the right. The applications on the clients and server run uninterrupted because the fail-over is transparent.

1.3.4 Enhancing Network Application Level availability with OpenArchitect/HA

The end-to-end and OA/Node failure detection can be further enhanced with help from an interconnected Ethernet switch. ZNYX Networks' OpenArchitect/HA software provides fault management and recovery for Ethernet switches, enabling full network redundancy.

The goal of the OpenArchitect/HA software is to provide application transparent network fault recovery, which allow applications to continue uninterrupted when a switch or network link failure occurs. Network fault recovery can be quite complex due to interaction of the various network protocols that may be in use. OpenArchitect/HA speeds up the failure recovery of common network protocols and as a result, detects and recovers from network faults much faster than an end-to-end application can on its own.

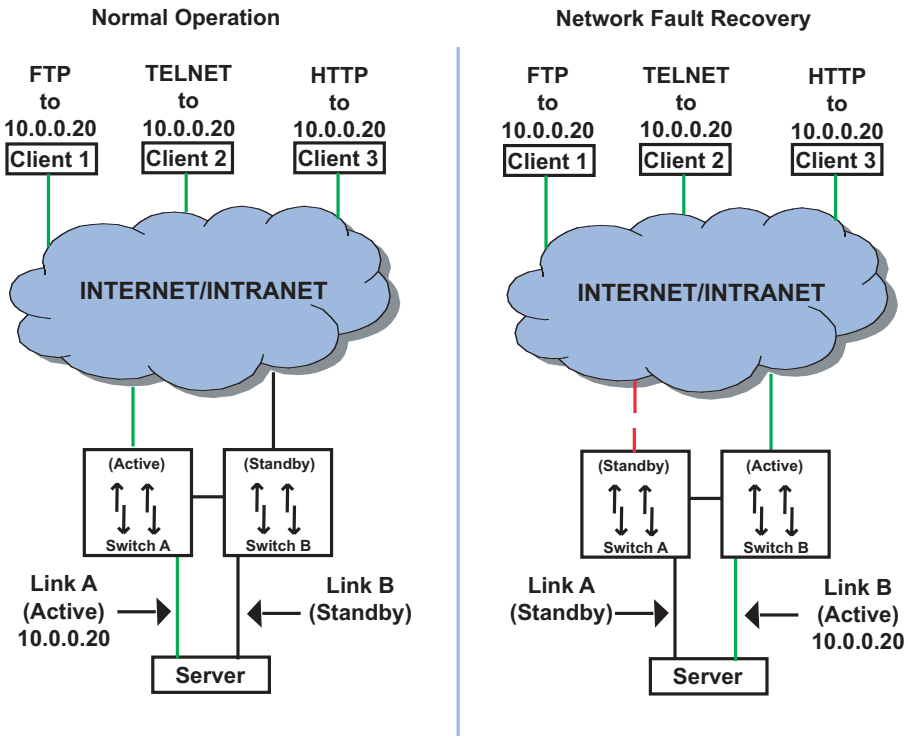


Figure 4. Network Fault Recovery Coupled with IP Transparent Failover

Fig. 4 shows an example of network fault recovery with OpenArchitect/HA running on a pair of switches in conjunction with OA/Node running on a server. The server has two network links – one to each switch – to provide 2N redundancy for the network. The client systems access the server via address 10.0.0.20. The data flow to the server in this case is through Switch A – the active switch as shown in the left diagram. The right diagram shows a fault

occurring between the OpenArchitect/HA enabled switch and the rest of the network. This kind of fault could be due to a cable cut, a problem with the service provider's equipment, or even when the active switch goes down due to a mechanical failure. Many things must happen in a coordinated fashion to provide a transparent fault recovery. The fault recovery results in Switch B becoming the active switch and Link B becoming the active link to the Server.

Technology in OpenArchitect/HA called Surviving Partner (SP) controls fault recovery by policy, which gives maximum flexibility to system designers for determining what should happen for different granularities of failure. Policy configuration is provided via a friendly user interface. System designers can incorporate many aspects of a complex setup to define the recovery strategy, including integration of dynamic routing behavior.

1.3.5 Combining OpenArchitect/HA and OA/Node

ZNYX Networks has invested several man-years of development to provide telco and service provider equipment manufacturers, and application developers with HA capabilities on its network adapters through its RAINlink-enabled drivers. This tradition continues with OpenArchitect/HA and OA/Node software, which provide a complete link management suite.

While OA/Node helps the end-to-end application improve its network availability, OpenArchitect/HA completes the system by extending HA beyond the node into the network. Furthermore, by partnering OpenArchitect/HA with OA/Node, all of the fault detection and recovery activity is transparent to both client and server applications.

By coupling OpenArchitect/HA on the switches with OA/Node on the nodes, system designers can utilize the software components to provide the fastest fault detection and recovery time possible. In a localized setup, the combination of OpenArchitect/HA and OA/Node has been measured to detect and recover from link failures within 50 milliseconds. In a configured network involving third party layer 3 uplink devices, recovery times of less than 1 second have been achieved.

2 HA for End Nodes

In the design of any HA system, it is important to understand where faults may occur and how to recover from them. This section examines potential faults from the end nodes perspective.

2.1 Types of Failures

There are two types of failures that can affect network connectivity:

1. Link failures
2. Node failures

A link failure is a failure of the network link in which no traffic is being sent or received by the end node, but where the end node is otherwise functioning normally. A node failure is a software or hardware problem that causes the node to stop sending or receiving traffic, although the link itself remains functional.

2.2 Link Failure Recovery Requirements

Each node should have at least two links to the network in order to protect against a single point of failure. The links should be on different switches in a switched network. Fig. 5 illustrates the evolution of node-to-network connectivity from a single link to full 2N network redundancy.

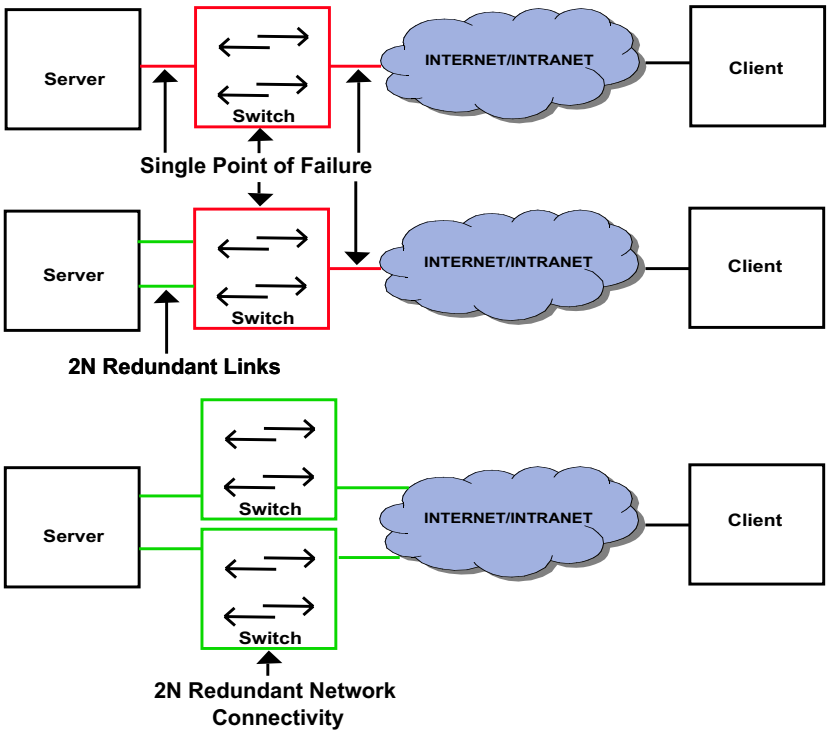


Figure 5. Nodal Network Connectivity Evolution

Multiple links require multiple network interfaces. The multiple links can be configured as either Active-Standby (where traffic is only allowed on one link while the others remain idle) or Active-Active (where traffic is active on all links).

Most operating systems require different addresses to be assigned to each interface. This causes problems for fault recovery. First, the applications running on the node may not behave well when the network address changes. Informing the network of new destination addresses also is difficult and time consuming. The solution is to have just one MAC and IP address for all the interfaces.

ZNYX Networks' OA/Node is link level software that provides this functionality. OA/Node provides **IP Transparent Fail-Over** capability, allowing applications to run uninterrupted when a link fails.

OA/Node provides a driver abstraction layer to the Operating System on the node. This abstraction layer hides the fact that there may be multiple physical connections to the network. It also hides the complexities of link failure detection and recovery from the OS or application. However, should the designer wish more control over the fault recovery process, OA/Node also provides an application level interface (RMAPI). Fig. 6 shows the relationship between OA/Node, the operating system, and applications.

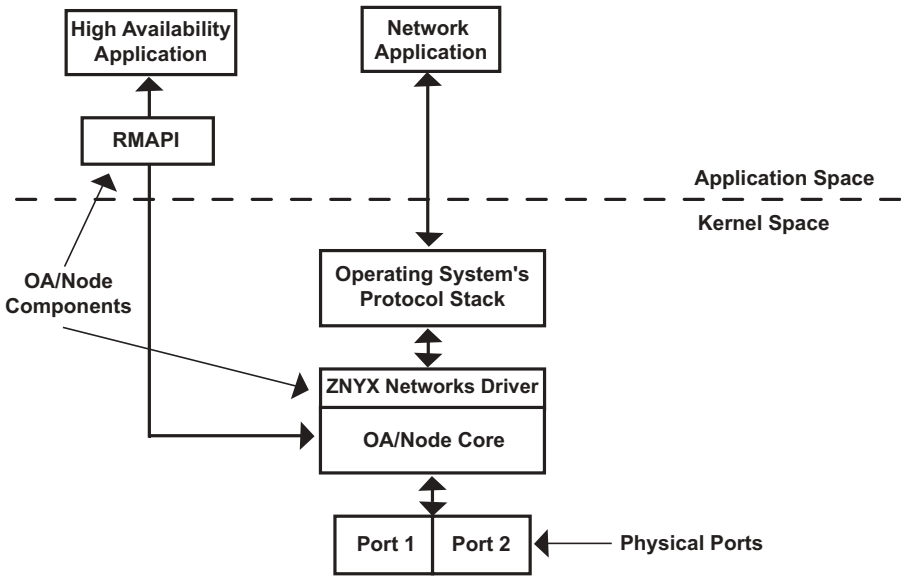


Figure 6. OA/Node Components in the Application Environment

2.3 Link Failure Detection

The key to fast failure detection and recovery is good hardware support. OA/Node software takes advantage of any hardware support provided by the network interface. Typically, the network interface can detect the state of the physical link (i.e. it can tell if the link is physically capable of sending or receiving data). Another detection mechanism is required if other problems in the network down stream from the physical link are blocking traffic. OA/Node provides this feature, which looks for a lack of activity on the link to indicate a problem downstream.

2.4 Fault Recovery

In setups where the node links are configured as Active-Standby, OA/Node software will "fail-over" to the standby link when a link failure is detected on the active link (see Fig. 7).

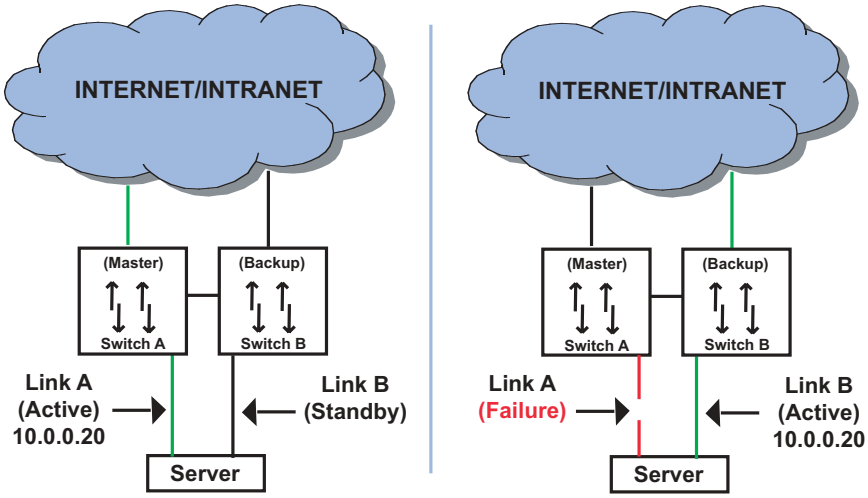


Figure 7. Active-Standby Fail-Over using OA/Node

In Active-Active setups, OA/Node software will redirect traffic from whichever link has failed to another working link. OA/Node must rely on the switches to which it is connected to ensure that the traffic is re-directed to the remaining links (see Fig. 8).

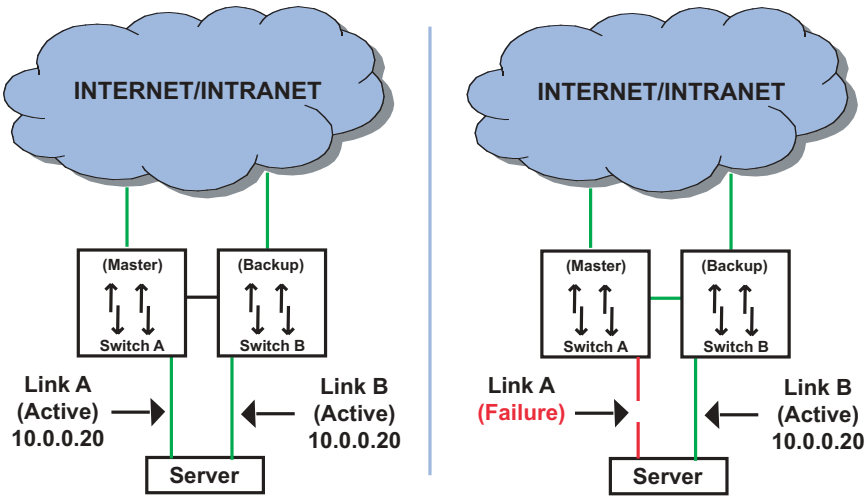


Figure 8. Active-Active Fail-Over using OA/Node

3 HA for Switches

This section explores HA from the switch perspective. The fundamental consideration for the design of the network topology is to ensure that each end node has a redundant data path–both to other nodes and to other parts of the network, if applicable. An HA configuration must have at least two switches in order to provide an alternate path in case of a switch failure.

3.1 Virtual Router Redundancy Protocol (VRRP)

A standard way of providing data path redundancy was specified by the IETF in RFC 2338, which defines VRRP (Virtual Router Redundancy Protocol). VRRP works in a switched environment even though the RFC specified routers and a shared medium. The basic idea behind VRRP is that two or more routers present a virtual IP address to end systems so that they are not "tied" to a specific piece of hardware to use as their gateway (see Fig. 9).

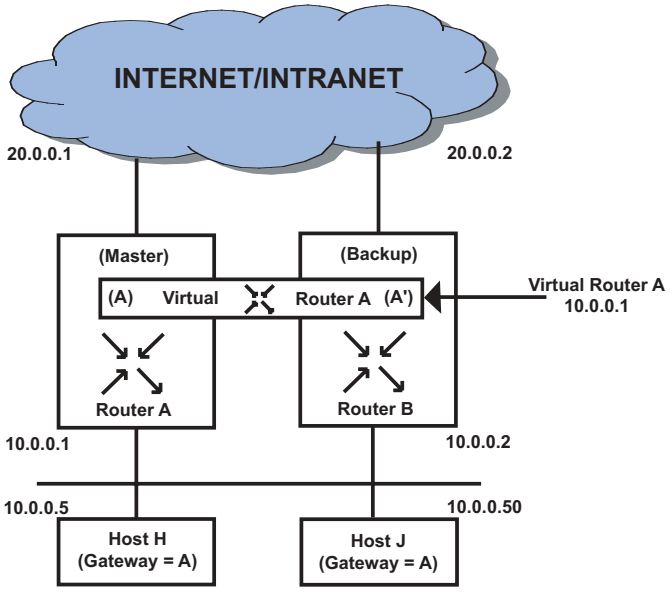


Figure 9. A Simple VRRP System Topology

VRRP ensures that if the router on the left (the "Master" in VRRP terminology) fails, the router on the right (the "Backup") will assume its IP address. End systems therefore do not know or care which piece of hardware is acting as the gateway and router failures become transparent.

3.1.1 OpenArchitect HA and Surviving Partner

As stated previously, VRRP was defined for a shared media environment. To provide similar functionality in a switched environment ZNYX Networks introduced its OpenArchitect/HA software, with restoration provided by the Surviving Partner feature. In order to be able to provide HA for switches, three topology rules must be followed:

1. A minimum of two switches must be used (a master and a backup). As with VRRP, OpenArchitect/HA supports multiple backups.
2. Each end node must have a connection to each switch for data path redundancy.
3. Each master switch must have one connection to each backup switch for OpenArchitect/HA management traffic.

3.2 Configuration Options

OpenArchitect/HA can be configured in one of the two modes, either as Active-Standby or Active-Active. In the Active-Standby mode, one of the switches forwards traffic while the other is in an "idle" state waiting for a failure in the active switch. In the Active-Active mode, both switches are forwarding traffic.

3.2.1 Active-Standby

In the Active-Standby mode, a primary data path is defined through the Master switch. A user-configurable policy determines the behavior of the switches when a fault occurs.

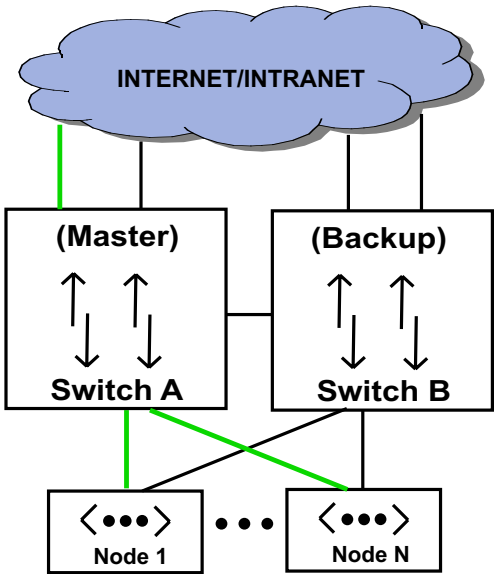


Figure 10. Active-Standby Configuration

In Fig. 10, switch A is designated as the Master (active switch) and switch B is designated the Backup (standby switch). The end nodes (from 1 to N, where N is limited by the number of ports on the switch) have a primary data path through the Master, as shown in the thick (green) lines. The primary data path is where the data traffic is forwarded under normal circumstances.

3.2.2 Active-Active

The Active-Active configuration is a synonym for load sharing. Load sharing allows network administrators to maximize the utilization of network resources without compromising on reliability, provided that an appropriate level of traffic engineering is performed. **The traffic engineering ensures that the total shared capacity does not exceed 100% of the capacity of a single switch or link. Otherwise, traffic may be lost when a failure occurs.**

In order to avoid problems such as packet re-ordering, load sharing is performed on a per-flow basis. This means that once a flow is established, all packets for that flow will use the same physical interface. The utilization of the interfaces and switch resources will vary accordingly since each flow has its own resource requirements.

3.2.2.1 Traditional VRRP Load Sharing

Fig. 11 shows how the traditional VRRP environment supports load sharing from a subnet perspective.

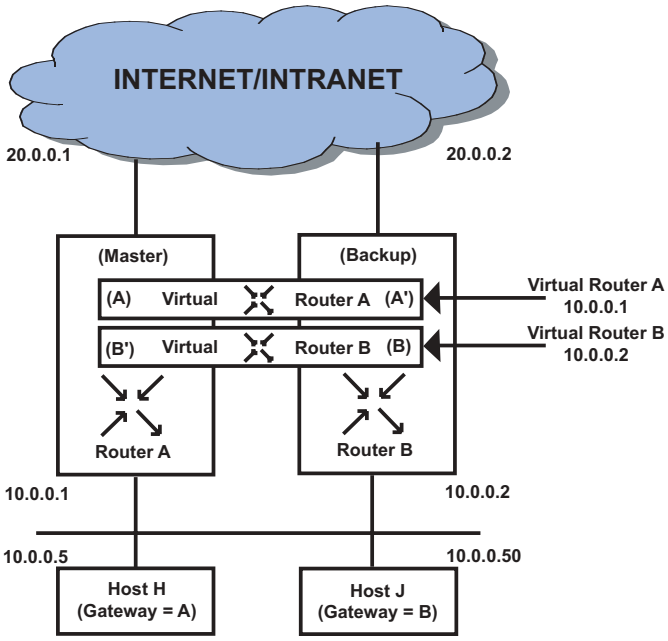


Figure 11. Traditional VRRP Load Sharing

This setup allows the LAN administrator to divide nodes into groups, with each group using a different virtual router as its gateway, thereby distributing the routing load over multiple routers. In this example, there are two physical routers each supporting two virtual routers. A virtual router is comprised of a Master and a Backup. In the diagram, the Master is designated as either A or B and the Backup is designated A' or B'. Thus, virtual router A is comprised of A and A', and virtual router B is comprised of B and B'.

Load sharing is achieved by specifying that the default gateway for half the nodes is virtual router A, and for the other half, it is virtual router B.

3.2.2.2 Load Sharing in a Switched Environment

A key consideration for load sharing in a switched environment is how to provide the connectivity for traffic between the internal nodes on the two different switches. The method chosen will depend on the amount of traffic expected between the two switches. Two possible configurations are discussed below.

3.2.2.2.1 Switch Interconnect

One option, is to utilize the switch interconnect link to pass data traffic (see Fig. 12). This may be augmented by additional connections between the two switches. Care must be taken to ensure that the links between the two switches can handle the internal traffic and that data traffic not interfere with OpenArchitect/HA management traffic. If there are ports available, it may be wiser to provide dedicated data links between the two switches.

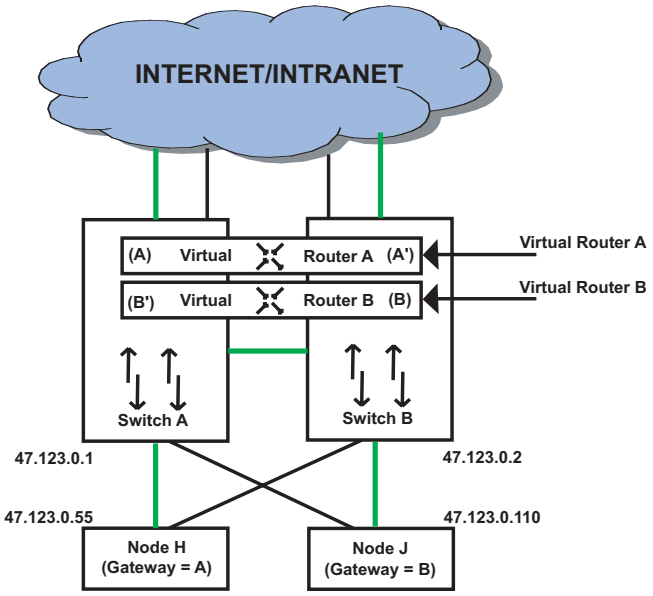


Figure 12. Load Sharing in a Switched Environment Using Switch Interconnect Links

In Fig. 12, the thick (green) lines show the primary data path for the nodes. Traffic between nodes whose primary data path is on different switches flows over the link connecting the two switches.

3.2.2.2.2 Active-Active Node Links

An alternative way of providing an internal traffic data path is to make both internal node links "active" instead of the alternate Active-Standby mode. The configuration in Fig. 13 requires more intelligence in the internal nodes but potentially reduces the load on the switch interconnect links. The basic idea is that the primary data path is defined via the default gateway. What would have been the standby link is now the secondary data path for internal traffic originating from nodes on the other switch.

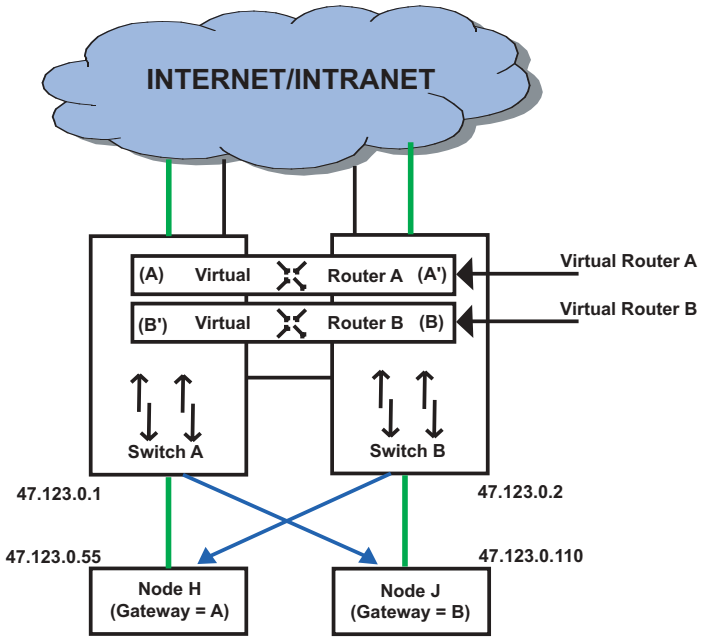


Figure 13. Load Sharing in a Switched Environment with Active-Active Node Links

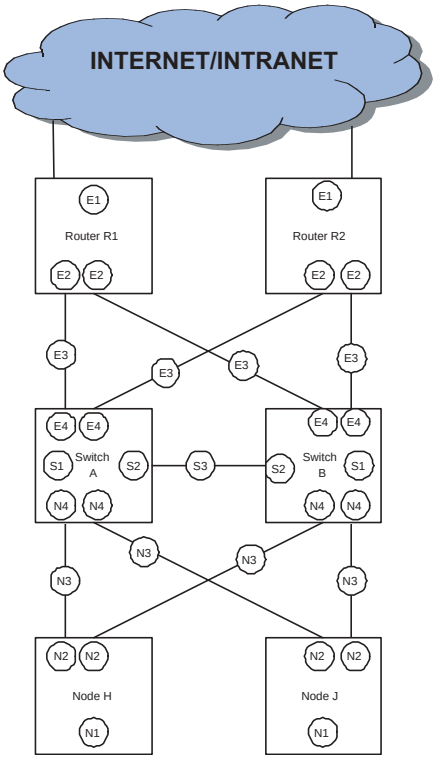
The thick (green) lines show the primary data path as defined by the default gateway. The arrows (blue) show the internal data path from nodes on the other switch. The interconnect link between the switches can be used for data traffic in case of a node link failure.

3.3 Switch Failure Modes

A failure can be defined as a change of state – from something that is ready for use (is working)to something that is not. When the system is first powered on or booted, it goes through an initialization phase. Care must be taken to ensure that any failure detection mechanisms do not interfere with this initialization. The system can only be deemed to be operational when initialization has completed. This is when the failure detection mechanisms are activated.

There are three types of failures that must be detected by the switches in the HA configuration:

1. External Link Failure
2. Internal Node Link Failure
3. Inter-switch Link Failure



Switch Failure Modes

External Link Failure
E-1 – Network element failure
E-2 – Network element port failure
E-3 – Cable cut/removed
E-4 – Fabric port failure

Internal Node Link Failure
N-1 – Node Failure
N-2 – Node port failure
N-3 – Connection failure
N-4 – Fabric port failure

Inter-switch Link Failure
S-1 – Switch Failure
S-2 – Fabric Port Failure
S-3 – Connection Failure

Figure 14. Location of Link Level Failures

Switch link failures are slightly more complex than end node link failures and the methodologies to determine link failure vary. However, the critical step is to detect failure as quickly and reliably as possible to avoid a service outage.

3.3.1 External Link Failure Detection

The switch must detect a link failure to the external network element for each of the following:

1. The port on the external network element connecting to the switch fails
2. The cable between the external network and the switch fails
3. The port on the switch connecting to the external network fails

Another problem affecting the external link is when the external network has a packet forwarding failure. This may be caused by either a software or hardware problem within the external network. The system architect must decide on which failure detection and recovery schemes are most appropriate for the system.

3.3.2 Internal Node Link Failure Detection

The switch must detect a failure of the link to the internal node for each of the following scenarios:

1. The port on the internal node connecting to the switch fails
2. The cable between the internal node and the switch fails
3. The port on the switch connecting to the internal node fails

Typically, power loss, a node reset, or a reboot may cause the link to go down so that these kinds of failures can also be detected.

3.3.3 Inter-switch Link Failure Detection

The HA switch configuration must detect the following failures:

1. Cable failure between the switches
2. Port failure on the link between the switches
3. A software failure on a switch

The link hardware will be able to detect both port and cable failures. A software failure on a switch is detected through a heartbeat mechanism. Redundant paths between switches are critical to ensure that a link failure does not cause a loss of heartbeat.

3.3.4 Alarms

Any time a link failure is detected, an alarm or notification should be sent to the appropriate alarm management software on the switch. OpenArchitect/HA provides the alarms through SNMP.

3.4 Internal Node Link Failure Recovery

An administrator-configurable policy determines the action of the switches when a fault occurs. There are three predefined policies to choose from:

1. Switch fail-over
2. VLAN fail-over
3. Port fail-over

3.4.1 Switch Fail-Over

This policy specifies that should any of the active links go down, the Master switch will fail over to the Backup switch. All attached systems should receive a notification that the Master has been changed so that any subsequent traffic will be sent to the Backup – which now becomes the Master switch.

3.4.2 VLAN Fail-Over

Using port-based VLANs may be desirable in some configurations. The VLAN Fail-Over policy has been defined for such situations. It specifies that when a link fails, all other links within the VLAN fail over to the Backup switch. Traffic that is destined outside of the failed-over VLAN will be forwarded back to the Master switch through the interconnect link(s).

3.4.3 Port Fail-Over

Port fail-over can be viewed as a special case of VLAN fail-over, where each port is a unique VLAN. Thus, when the link fails, only that link is transferred over to the Backup switch. All traffic is forwarded back to the Master through the interconnect link(s).

3.4.4 Custom Failure Recovery Policies

OpenArchitect/HA also provides the flexibility for experienced users to define their own policies. One example of a custom policy would be to have the Master switch fail-over due to "goodness" (described below).

3.4.4.1 Switch Fail-Over Due to "Goodness"

If either VLAN or port fail-over is used, it may be useful to define a threshold triggering when to fail the entire switch. This feature is necessary to ensure that the interconnect links do not get overloaded when multiple port or VLAN failures occur. A "goodness" parameter is defined on the Master switch to specify the ratio of active-to-connected links to determine when to fail over to the Backup. This ratio should be established based on the projected amount of traffic that may need to utilize the interconnect link.

3.5 Inter-Switch Link Failure Recovery

The inter-switch link(s) are a key component of the HA solution. One link is used to provide signaling between the Master and Backup switches so that the Backup knows when the Master fails. The link(s) are also for data traffic in certain fail-over modes mentioned above.

Care must be taken in engineering the interconnect links to ensure non-interference with the heartbeat mechanism.

3.5.1 Master Switch Failure

OpenArchitect/HA uses a heartbeat mechanism between the Master and the Backup to indicate health of the Master switch. The system manager is able to set various criteria for the heartbeat mechanism to identify a failure. When the criteria have been met the Backup will assume that the Master has failed and will take over the role of the Master.

3.6 External Link Failure Recovery

External link failure recovery is policy-controlled. The system manager or architect can decide how to handle a failure under various circumstances. Some options are:

1. To fail-over to an alternate external link if one is available
2. To fail-over to an interconnect link to the partner switch
3. To fail over to the partner switch

3.7 Maintenance Process

The faulty switch must be replaced before the standby also fails. In order to reduce the chance of human error during the maintenance process, automatic configuration of the replacement switch is needed.

3.7.1 Field Replacement

There are different scenarios for replacing a defective switch. Field replacement should be controlled by a policy that is set by the network administrator. The supported scenarios are the following:

1. The replacement switch has been pre-loaded with the appropriate software and configura tion files
2. The replacement switch has been pre-loaded with the appropriate software but not the appropriate configuration files
3. The replacement switch has neither the appropriate software nor the appropriate configuration files

OpenArchitect/HA provides a mechanism called SurvivingPartner to allow the replacement switch to automatically download its software and configuration files from a central authority.

3.7.2 Automatic fail-back

A policy-based mechanism is provided to instruct the replacement switch whether it should automatically become the Master switch or wait for manual intervention. A mechanism for automatic fail-back is also provided. If automatic fail-back is set, the normal fail-over rules still apply.

3.7.3 Manual fail-over

A mechanism to trigger a fail-over on the Master switch is provided. This is useful for a controlled fail-over to do maintenance or to re-establish the original Master. This mechanism should be used with caution.

3.8 Provisioning

The OpenArchitect/HA switch management software provides easy to use configuration and distribution tools to reduce chances of human error.

3.8.1 Configuration

Configuring a switch is a process that is inherently prone to error. In addition to editing configuration files directly, a form based configuration application is available. A mechanism for configuring multiple switches from a single location is also available.

3.8.2 Software Distribution

Provisioning a switch includes ensuring that the correct versions of configuration files and software are installed and started. OpenArchitect/HA provides a mechanism for provisioning many switches (10's to 100's) that supports both initial network setup and in-service upgrades.

4 HA Issues with PICMG 2.16

The new PICMG 2.16 standard defines an HA topology for using Ethernet in the backplane of a CompactPCI chassis. This section explores the specific issues that must be addressed when building HA systems using this standard.

4.1 Basic PICMG 2.16 Topology

The basic PICMG 2.16 system provides for two switching fabrics with an optional interconnect link. Each node slot has two connections, one to each of the switching fabrics as shown in Fig. 15.

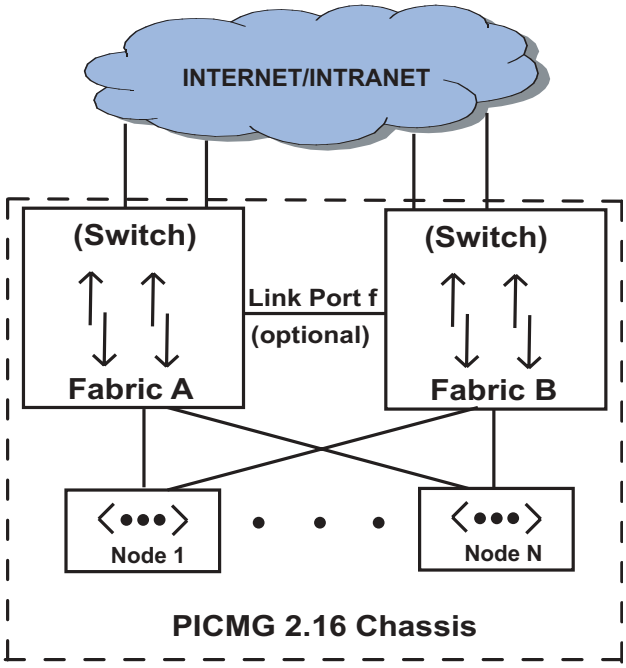


Figure 15. A Typical PICMG 2.16 System Topology

4.2 Optional Fabric Interconnect

The PICMG 2.16 standard supports one optional link (link port f) between Fabrics as shown in Fig. 15. Since the interconnect link is optional, chassis may be built without this internal interconnect. In these cases, an external link (or better two external links) will be required for HA.

System architects must design a way to distinguish a loss of heartbeat due to link failure versus a loss of heartbeat due to a software failure. The easiest way to resolve this issue is to provide an alternative data path for the traffic between the two switches via an external cable.

4.3 Power On Synchronization

In a system made up of discrete elements, it is easy to ensure that things boot up in the proper order. Typically, switches are booted up before servers, and servers are booted up before end stations to ensure that no race conditions occur and that nothing hangs while waiting for a dependency. System architects should use a chassis management scheme to define and enforce the appropriate blade boot order, as PICMG 2.16 places the network into one chassis with one power switch.

5 PICMIG 2.16 Chassis Interconnection Options

Chassis may be interconnected through whatever external links are available. Typically, this will be a higher speed link than what is on the PICMG 2.16 back plane. For example, on the ZNYX Networks ZX4500P embedded switch, the back plane is 10/100 Mbps while the external interconnect links are 1 Gigabit. The following interconnection options are supported in order to avoid single points of failure:

5.1.1 Single Chassis Interconnect

When interconnecting a chassis to another chassis, each fabric should have two connections, one to each of the fabrics in the other chassis (see Fig. 16).

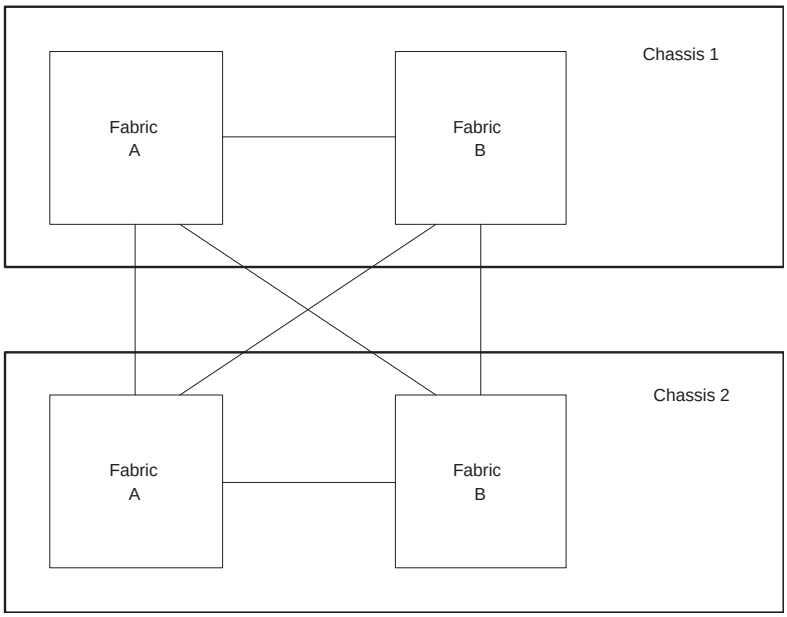


Figure 16. Interconnection of Two PICMG 2.16 Chassis

5.1.2 Multiple Chassis Interconnect (Star Configuration)

System architects will not be able to use a mesh configuration to connect multiple chassis if there is a shortage of external links available on the switching fabrics. One solution is to interconnect via an additional pair of switches such as the ZNYX Networks ZX4600 or ZX4800. Each fabric would need a connection to each switch for full fault tolerance as shown in Fig. 17.

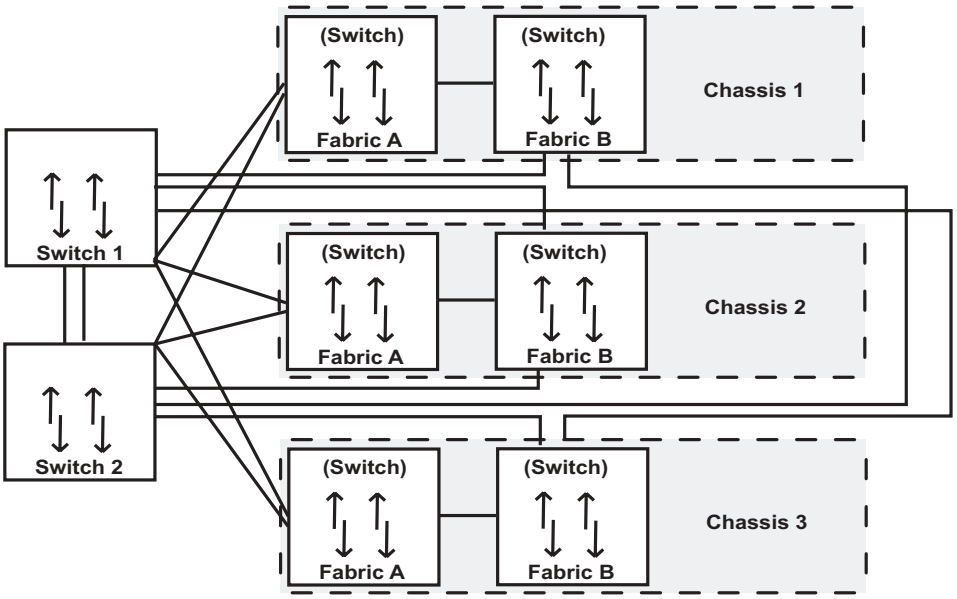


Figure 17. Star Topology for the Interconnection of Multiple Chassis

5.2 Multiple Chassis Interconnect (Helix Configuration)

An alternative to using additional switches to interconnect multiple chassis is to connect them in a Helix configuration (Fig. 18). This setup is simpler and less expensive than the star configuration. A disadvantage is that depending on the type of fault, traffic may have to traverse more switches. Another potential problem with this configuration is that a loss of a middle chassis may result in a decrease of bandwidth between the remaining chassis.

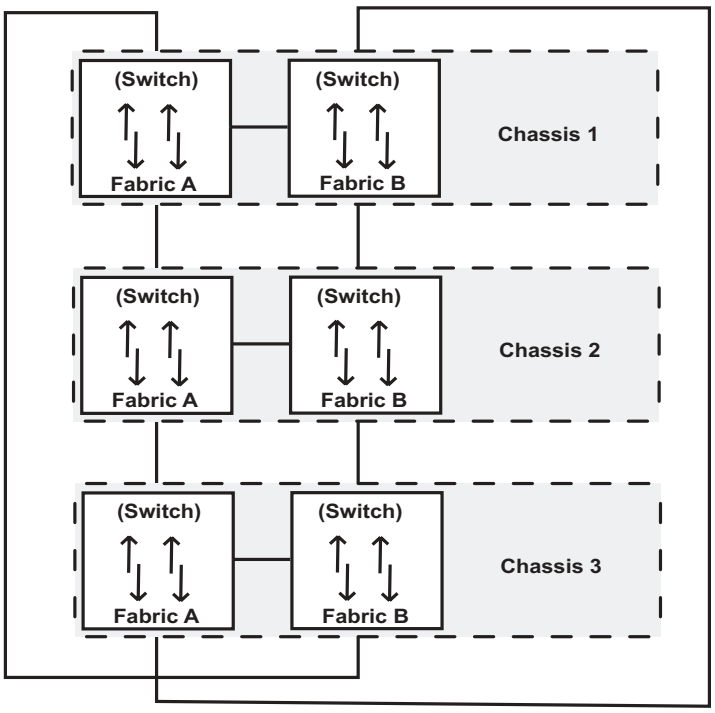


Figure 18. Helix Topology for the Interconnection of Multiple Chassis

5.2.1 Mitigating bandwidth decrease in the Helix topology

The worse case scenario for the Helix topology is failure of a middle chassis. This results in a cut in the primary link to chassis on either side of the defective unit. As shown in Figure 18, the "A" fabrics in chassis 1 and 3 do not have direct connections. Instead, they each connect to the "B" fabrics. In order for traffic to flow between nodes on the two "A" fabrics, it would have to use the interconnect link between the "A" and "B" fabrics. There are two ways to avoid this situation.

5.2.1.1 Additional Bandwidth for "End" Chassis

A way of avoiding the interconnect link to carry data traffic is to add additional interconnect links between the A and B fabrics of the "end" chassis. The "end" chassis are the two chassis that have their A fabrics connected to B fabrics in another chassis. If not all the fabric ports are used on the back plane of the chassis then a rear transition board (RTB) can be used to provide additional external links to connect the A and B fabrics between the end chassis.

5.2.1.2 Right Handed Fail-Over

An alternative approach is to have all chassis to the "right" (it may actually be below if they are rack mounted) of the failed chassis "fail-over" to use their B fabrics. The advantage of this approach is its simplicity (in terms of the number of connections) and that the bandwidth defined by the primary path is maintained.

The technologies, architectures and mechanisms described above provide system architects with unprecedented flexibility and power to implement High Availability in embedded Ethernet systems. Contact your ZNYX Networks sales representative for more information or for a demonstration of next-generation HA solutions implemented with OpenArchitect/HA Suite.

For more product information, contact a ZNYX Networks representative.

ZNYX Networks, Inc.
48421 Milmont Drive
Fremont, California 94538

Phone (510) 249-0800
Fax (510) 656-2460

Email: sales@znyx.com
Web: www.znyx.com

© 2001 ZNYX Networks, Inc. All rights reserved. ZNYX, OpenArchitect, HotSwap, and CarrierClass are trademarks of ZNYX Networks, Inc. All other trademarks are properties of their respective owners. Document Number: 280-0063-001 Date: 6/18/02