ZebOS-XP®
Network Platform

Version 1.4
Extended Performance

Architecture Guide

December 2015

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900
http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:
support@ipinfusion.com

Contents

Preface

This guide introduces the architecture of the ZebOS-XP Network Platform to network equipment manufacturers.

Audience

This guide is for engineers and other networking professionals who need an overview of the functions and features of ZebOS-XP.

Conventions

Table P-1 shows the conventions used in this guide.

Table P-1: Conventions

Convention	Description
Italics	Emphasized terms; titles of books
Note:	Special instructions, suggestions, or warnings
`monospaced type`	Code elements such as commands, functions, parameters, files, and directories

Contents

This document contains these chapters and appendices:

- Chapter 1, *ZebOS-XP Architecture*
- Chapter 2, *ZebOS-XP Key Features*
- Chapter 3, *ZebOS-XP Abstraction Layers*
- Chapter 4, *ZebIC*
- Chapter 5, *ZebHA*
- Chapter 6, *Management Modules*
- Chapter 7, *ZebOS-XP Versus ZebOS 7.10.x*

Related Documents

For details about each module, refer to the command references, configuration guides, and developer guides for the module.

This document cites major standards supported by ZebOS-XP. For details about all supported standards, see the *Feature Matrix*.

Use the *Installation Guide* for hardware and software requirements, configuring the ZebOS-XP modules and protocol daemons, and building the ZebOS-XP software.

Note: All ZebOS-XP technical manuals are available to licensed customers at http://www.ipinfusion.com/support/document_list.

Support

For support-related questions, contact support@ipinfusion.com.

Comments

If you have comments, or need to report a problem with the content, contact techpubs@ipinfusion.com.

CHAPTER 1 ZebOS-XP Architecture

This chapter introduces ZebOS-XP and describes its high-level architecture.

ZebOS-XP Network Platform Overview

The ZebOS-XP network platform is Layer 2 (data link) and Layer 3 (network layer) control plane software that allows network equipment manufacturers to rapidly add networking capabilities to communications products. ZebOS-XP takes advantage of separate data plane processors (network processing units and switch ASICs). ZebOS-XP is targeted at Tier 1 and Tier 2 network equipment manufacturers who provide solutions in:

- Carrier transport, access, and Carrier Ethernet
- Mobile backhaul and access
- Data center and cloud networking, including solutions for enterprise private clouds, hybrid clouds, and public clouds

The ZebOS-XP networking protocol modules conform to leading IEEE, IETF, MEF, and industry-specific standards. ZebOS-XP supports more than 200 protocols for:

- Layer 2 switching: Virtual Local Area Networks (VLANs), Spanning Tree, Data Center Bridging, TRILL, PTP
- Layer 3 routing: OSPFv2/v3, RIP/RIPng, BGP4+, ISIS
- Multi-Protocol Label Switching (MPLS) which operates at a layer between traditional definitions of Layer 2 and Layer 3
- Carrier Ethernet
- Data Center Ethernet

ZebOS-XP is built on an SDN architecture that features:

- High degree of scalability
- Modularity and programmability
- Embedded abstraction layer to communicate with the underlying operating system
- Hardware abstraction layer to run on a variety of silicon switching chipsets for forwarding table updates

The modular architecture of ZebOS-XP lets you select and integrate only the protocols you need. Each protocol module runs as a separate daemon with its own address space and stack. This isolates issues with one protocol module from all others. You can readily add a customized protocol module to address specific needs.

Note: For a list of all daemon file names, see *Installation Guide*.

ZebOS-XP provides configuration management at multiple layers:

- Command line interface
- SNMP
- External simple management interface C API
- Internal C API

High-Level Architecture

Figure 1-1 shows the high-level architecture of ZebOS-XP.

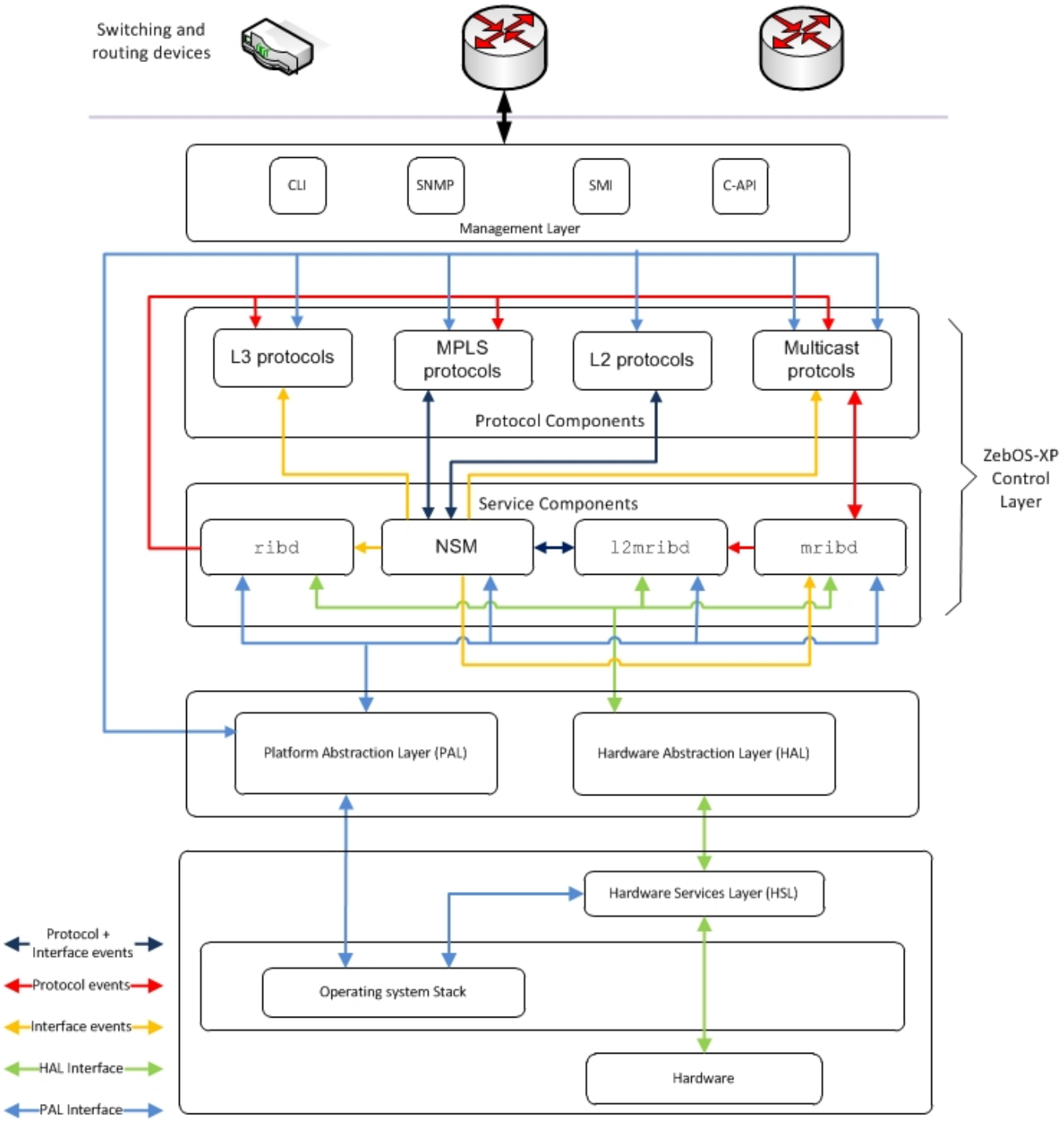

Figure 1-1: High-level architecture

At the top of Figure 1-1 is the Management Interface used to manage, configure, and operate the ZebOS-XP routing and switching protocols.

The major components in the control layer of ZebOS-XP are grouped into these categories:

- Protocol components, which include:
  - Layer 3 Protocols
  - Multi Protocol Label Switching Protocols
  - Layer 2 Protocols
  - Multicast Protocols
- Service components, which include:
  - Network Services Module (NSM)
  - Unicast Routing Information Base Daemon (`ribd`)
  - Layer 2 Multicast Routing Information Base Daemon (`l2mribd`)
  - Layer 3 Multicast Routing Information Base Daemon (`mribd`)

The bottom of Figure 1-1 shows the abstraction layers of ZebOS-XP:

- Platform Abstraction Layer
- Hardware Abstraction Layer

CHAPTER 2 ZebOS-XP Key Features

This chapter describes the key features of ZebOS-XP.

Management Interface

ZebOS-XP provides a comprehensive set of tools to manage, configure, and operate the routing and switching protocols.

The management interface includes:

- Command Line Interface
- Simple Network Management Protocol
- Simple Management Interface
- C Language Application Programming Interface

Command Line Interface

The command line interface (CLI) offers complete, unified management of ZebOS-XP. Each command is usually associated with a specific task. The CLI has two major components:

- The Integrated Management Interface process (`imi`) is a centralized, persistent daemon for managing the ZebOS-XP configuration that runs as a server of NSM and the protocols. The `imi` daemon connects to protocol daemons through inter-process communications.
- IMI shell (`imish`) is a interactive daemon program for the `imi` daemon that enables connection locally from the console of a device running ZebOS-XP or remotely from terminal emulator programs such as `ssh` or `telnet`.

Through `imish`, a system administrator can configure and monitor all of the ZebOS-XP daemons through one centralized connection. The `imish` daemon maintains a persistent connection to all protocol daemons, stores configuration data, and offers extensive monitoring and logging capabilities.

The CLI can use the secure authentication methods of the operating system to manage and validate user names and passwords.

Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) provides a standardized framework and a common language for monitoring and managing devices in a network. The SNMP framework has three parts:

- SNMP manager: A system used to control and monitor the activities of network devices. This is sometimes called a Network Management System (NMS).
- SNMP agent: The component within a managed device that maintains the data for the device and reports data to SNMP managers.
- Management Information Base (MIB): SNMP exposes management data in the form of variables on the managed device which network management agents can extract from the ZebOS-XP protocols for all standard defined MIBs.

As shown in Figure 2-1, an SNMP manager on the network sends query packets to gather status data. Each protocol daemon responds to these queries as defined by the corresponding MIB for the protocol.

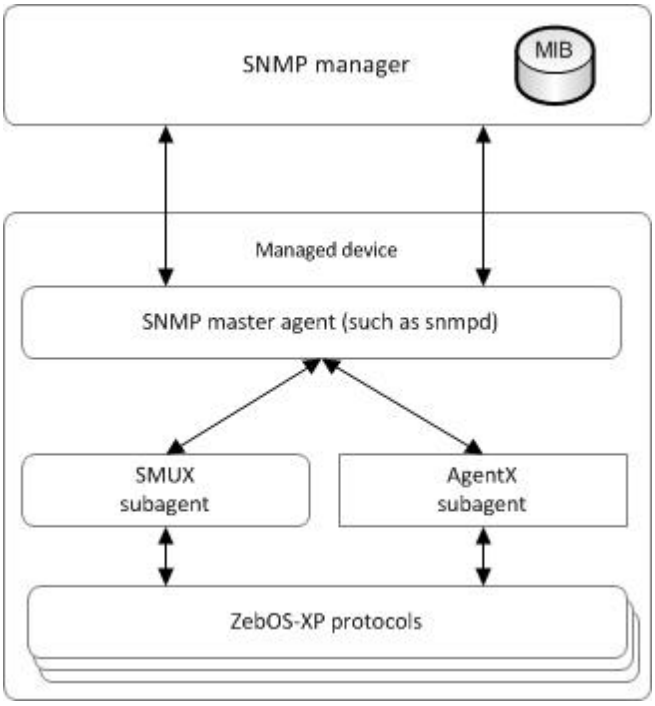

Figure 2-1: SNMP subagents

ZebOS-XP supports these subagents:

- The SMUX (SNMP Multiplexing) protocol defined by RFC 1227 is enabled by default. The SMUX agent responds to UDP-161 packets, and transmits them to ZebOS-XP over TELNET connections using port 199.
- The AgentX (Agent Extensibility) protocol defined by RFC 2741 uses TCP-705 for communication between the subagent and the master agent.

Note: You cannot configure SNMP with both SMUX and AgentX. One or the other must be used.

Each protocol daemon in ZebOS-XP has an SNMP API that give developers the flexibility to integrate with any management plane. In addition, ZebOS-XP can log both the systems' events and errors.

For more about installing and configuring SNMP agents and subagents, see the *Installation Guide*. For more about the MIBs a ZebOS-XP protocol daemon supports, see the developer guide for the protocol.

Simple Management Interface

The Simple Management Interface (SMI) is a C language API that you use to write applications that configure and manage the ZebOS-XP routing and switching protocols. For more see Chapter 6, *Management Modules*.

C Language Application Programming Interface

Every ZebOS-XP protocol module provides a C language Application Programming Interface (API) for configuring properties of the protocol. For example, the `ospf_api.h` include file defines an API for configuring the OSPF protocol. Use these APIs to integrate with any management plane to control the ZebOS-XP software.

Note: You must call the functions in a ZebOS-XP C language API in the same process space as ZebOS-XP (that is, from a ZebOS-XP daemon).

Network Services Module

Network Services Module (NSM) is the core of the ZebOS-XP architecture. NSM is the base module that simultaneously and independently communicates with every ZebOS-XP routing and switching process. The protocol components use APIs exposed by the NSM client, which act as conduits to transfer data between the protocol modules and NSM.

NSM also interacts with the Platform Abstraction Layer (PAL) to communicate with the underlying operating system. Figure 2-2 shows NSM interacting with PAL and the protocols.

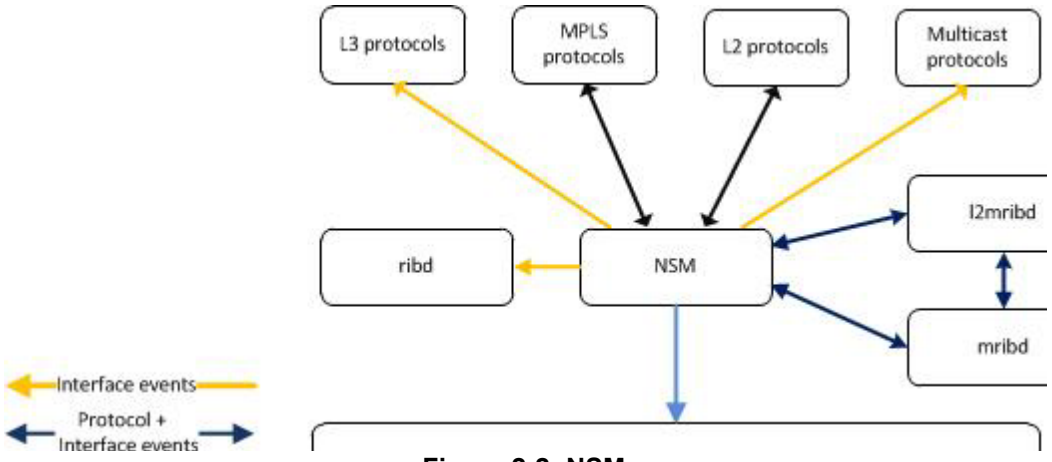

Figure 2-2: NSM

NSM provides the following services to the protocol components:

- Interface management
- Layer 2
- VLAN management for Layer 2 switching
- MPLS label management
- Traffic engineering
- QoS management
- Virtual Routing/Virtual Route Forwarding (VR/VRF) Management
- Virtual Private LAN Service (VPLS) and Virtual Private Wire Service (VPWS)

The protocol modules use the NSM client module to register for relevant services they want from NSM. The NSM server module manages the list of connected NSM clients and publishes information to them.

For more information about NSM, see the *Network Services Module Developer Guide.*

Interface Manager

The NSM Interface Manager provides a logical view of the interfaces in the system to the control plane. The Interface Manager configures and manages the physical and logical interface and provides the following functions:

- Manages and configures interfaces that are not managed by the network processor or ASIC
- Manages the learning, configuration, and management of the hardware interfaces present on the network processor or ASIC
- Provides an abstracted TCP/IP interface that can be populated by different TCP/IP stacks with the same operating system

- For Layer 3 interfaces, populates the interface in the TCP/IP stack for slow path and exception packet handling
- Maintains the database and hierarchies for all interface types and their relationships to each other and functions to maintain and change these hierarchies from the control plane
- Provides interfaces to link scanning tasks for interface up/down asynchronous events

Layer 2 Protocols

ZebOS-XP includes these Layer 2 features:

- Virtual Local Area Networks
- Link Aggregation
- Multi-Chassis Link Aggregation
- Spanning Tree
- Transparent Interconnection With Lot of Links
- Carrier Ethernet
- Data Center Bridging
- Port Authentication (authd)
- Precision Time Protocol
- Synchronous Ethernet
- Shortest Path Bridging (IEEE 802.1aq)
- Edge Virtual Bridging
- Ethernet Automatic Protection Switching
- Ethernet Local Management Interface
- Remote Monitoring

Figure 2-3 shows the ZebOS-XP Layer 2 high-level architecture.

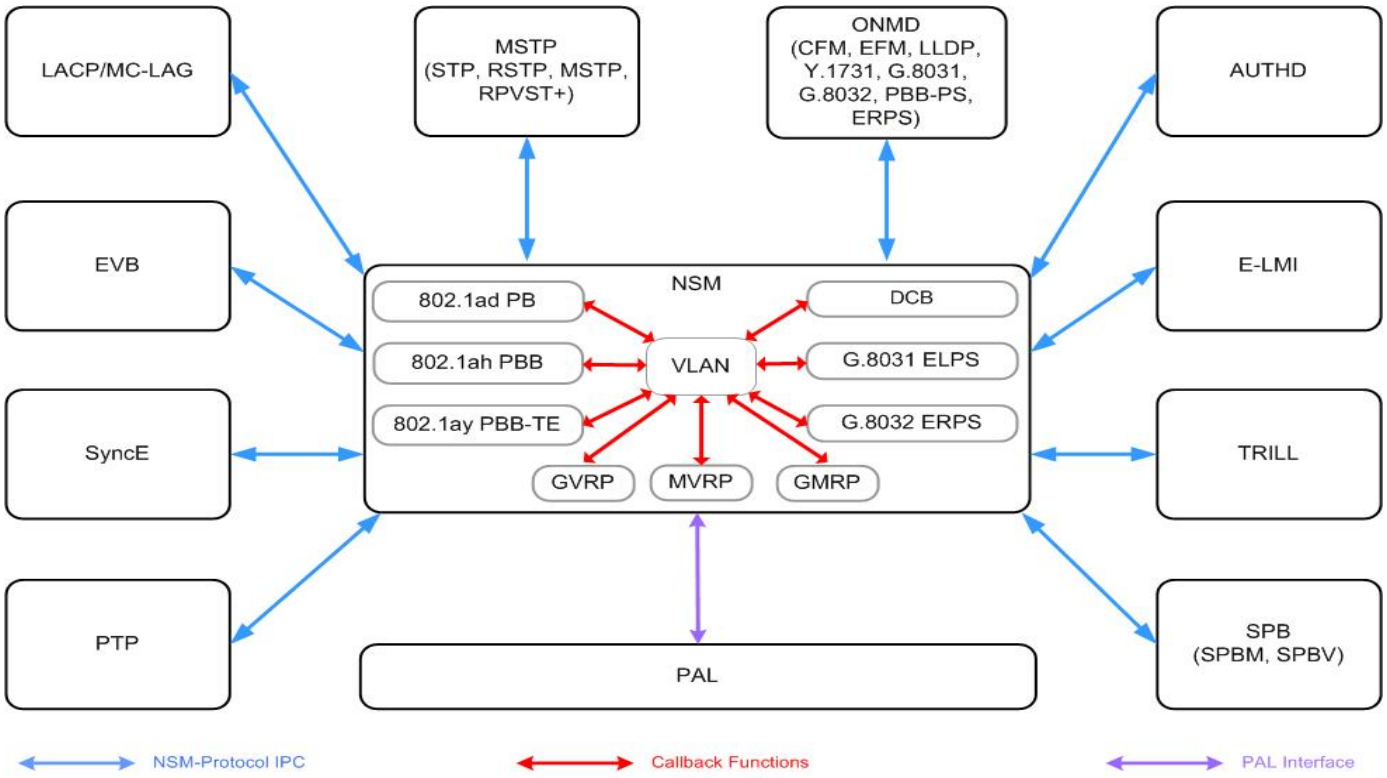

Figure 2-3: Layer 2 architecture

Virtual Local Area Networks

The VLAN modules offer a consistent network-wide management tools to manage virtual LANs (Local Area Networks) and bridged VLANs:

- VLAN bridging allows network devices to segment into VLANs, regardless of their physical location.
- VLANs, in accordance with IEEE 802.1Q, enable multiple bridged LANs to transparently share the same physical network link without leaking information between LANs. Traffic between VLANs is restricted to bridges that forward unicast, multicast, or broadcast traffic only on the LAN segments that serve the VLAN to which the traffic belongs.

ZebOS-XP VLAN modules make it easy to administer logical groups of stations that can communicate as if they were on the same LAN. They make it easier to manage a move, add, delete, or other updates to members of these groups.

The following highlights the features of the VLAN modules.

MAC Bridging (802.1d)

The ZebOS-XP VLAN modules support all IEEE 802.1D LAN MAC (Media Access Control) protocols, shared media, and point-to-point LANs. MAC bridging allows multiple LANs to be connected together. MAC bridging filters data sent between LAN segments, reduces network congestion, and allows networks to be partitioned for administrative purposes.

Identifying VLANS

VLAN identifiers (VIDs):

- Offer convenient, consistent, and network-wide methods for bridges to identify rules to classify user data for VLANs
- Extend source and destination MAC addresses by treating addressing for different VLANs independently

- Identify and select from multiple active topologies; and identify parameters that restrict access from one part of a network to another

Provider Bridging (802.1ad)

Provider Bridging (PB) enables a service provider to use the architecture and protocols of 802.Q to offer the equivalent of separate Local Area Networks (LANs), bridged LANs, or virtual bridged LANs to multiple customers. Provider bridging requires no active cooperation between customers and requires minimal cooperation between an individual customer and the service provider.

When VLANs were originally defined in the 802.1Q, the number of unique VLAN identifiers was limited to 4096. In large provider networks, each subscriber needs a separate address, thus this limit could prevent a provider from having more than 4096 subscribers.

To overcome the 4096 VLAN identifier limit, the frame format for 802.1ad inserts an additional VLAN header into a single 802.1Q Ethernet frame. There are two types of VLAN headers:

- The C-VLAN or inner header which is closest to the payload portion of the frame identifies the customer VLAN
- The S-VLAN or outer header which is closest to the Ethernet header identifier the provider VLAN

The frame format for 802.1ad is also called “Q-in-Q” or “double tagged”.

With the two VLAN identifiers in combination for each provider-customer pair, it is possible to define up to 16,777,216 labels.

Provider Backbone Bridging (802.1ah)

Provider Backbone Bridging (PBB) adds the capability to create Ethernet backbones for service access networks. PBB extends 802.1ad Provider Bridging in these ways:

- The 802.1ah header adds a service instance identifier (I-SID) which is a label that maps to a customer VLAN identifier. An I-SID virtualizes VLANs across a network. VLANs are mapped into I-SIDs by configuring only the edge of the network at the Backbone Edge Bridges (BEBs). This makes the maximum number of service instances 16 million.
- The 802.1ah header encapsulates backbone source and destination MAC addresses (termed B-SA and B-DA) along with the customer source and destination MAC addresses (C-SA and C-DA). The 802.1ah format is sometimes called “MAC-in-MAC” because of this. The encapsulation of customer MAC addresses in backbone MAC addresses means that the backbone does not need to learn customer MAC addresses. Customer MAC addresses are learned at BEB ports only.

Provider Backbone Bridge-Traffic Engineering (802.1Qay)

The Provider Backbone Bridging-Traffic Engineering (PBB-TE) standard supports the construction of “traffic engineered” backbone topologies, load balancing, protection switching, bandwidth management, and so on to serve the needs of large service providers. Provider networks rely on direct control of path routing so that traffic engineering can be used to allocate bandwidth, ensure diverse backup path routing, and select path performance as required by service level agreements. The architecture of bridges and their associated protocols as specified by 802.1ah allows the further specification of interoperable bridge capabilities that support traffic engineering at the required scale.

Registration Frameworks

The Layer 2 support in ZebOS-XP includes these registration frameworks:

- Generic Attribute Registration Protocol: GARP is a generic framework for bridges to register and de-register attributes, such as VLAN identifiers and multicast group membership.
- GARP VLAN Registration Protocol (802.1Q): GVRP is a GARP application that provides VLAN registration service. GVRP uses GARP Information Declaration (GID) and GARP Information Propagation (GIP), which provide the

common state machine descriptions and the common information propagation mechanisms defined for GARP-based applications.

GVRP provides support for VLAN pruning and dynamic VLAN creation. A switch can exchange VLAN configuration information with other GVRP switches, prune unnecessary broadcast and unknown unicast traffic, and dynamically create and manage VLANs.

- GARP Multicast Registration Protocol: GMRP allows participants to dynamically register and de-register information with the Media Access Control (MAC) bridges attached to the same LAN segment. A switch can exchange multicast group information with other GMRP switches, prune unnecessary broadcast traffic, and dynamically create and manage multicast groups.
- Multicast Registration Protocol (802.1ak): MRP has protocols, procedures, and managed objects to support multiple registrations, and allow the participants in an MRP application to register attributes with other participants in a bridged LAN.
- Multiple VLAN Registration Protocol (802.1ak): MVRP registers multiple VLANs and provides for the rapid healing of network failures without interrupting services to unaffected VLANs. In addition, MVRP improves the convergence time of the GVRP module.
- Multiple Multicast Registration Protocol (802.1ak) MMRP manages group Media Access Control (MAC) addresses. In addition, MMRP improves the convergence time of GMRP.

VLAN Prioritization (802.1p/Q)

ZebOS-XP includes priority signaling for traffic at the data-link layer. IEEE 802.1Q specifies a priority value of between 0 and 7 inclusive that can be used by QoS (Quality of Service) disciplines to differentiate traffic. Although this technique is often called “802.1p”, there is no standard by that name published by the IEEE. Instead, the technique is incorporated into 802.1Q standard which specifies the tag inserted into an Ethernet frame.

Port and Protocol Classification (802.1v)

Port and Protocol Classification is an amendment to 802.1Q to classify incoming packets by methods other than source port information, specifically, classification based on data-link-layer protocol identification.

Link Aggregation

The Link Aggregation module allows one or more links to be aggregated together to form a Link Aggregation Group (LAG), such that a MAC client can treat the Link Aggregation Group as if it were a single link. The Link Aggregation Control Protocol (LACP) allows bundling of several physical interfaces to form a single logical channel providing enhanced performance and redundancy. The aggregated interface is viewed as a single link to each switch. The spanning tree also views it as one interface. When there is a failure in one physical interface, the remaining interfaces stay up, so there is no disruption.

Multi-Chassis Link Aggregation

Multi-Chassis Link Aggregation (also called MC-LAG, MLAG, or Distributed Resilient Network Interconnect [DRNI]) extends the link aggregation concept to ensure that connectivity between two networks can be maintained despite the failure of a node. With MC-LAG, at either one or both ends of a link aggregation group, a single aggregation system is replaced by a *portal* that is a collection of one to three portal systems.

Spanning Tree

The ZebOS-XP Spanning Tree support are a combination of these modules:

- Spanning Tree Protocol (STP)
- Rapid Spanning Tree Protocol (RSTP)

- Multiple Spanning Tree Protocol (MSTP)

The following highlights the features of the Spanning Tree Protocol modules.

Note: All ZebOS-XP spanning tree modules support 802.3x flow control, broadcast storm recovery, and port mirroring.

Spanning Tree Protocol (802.1d)

The ZebOS-XP Spanning Tree Protocol (STP) module creates spanning trees within mesh networks of Layer 2 connected bridges, disabling any links that are not a part of the tree and leaving a single active connection between any two unique network nodes.

STP devices exchange BPDU (bridge protocol data unit) messages. The Spanning Tree Algorithm calculates the best path and prevents multiple paths between network segments. STP elects a root bridge, finds paths and determines the least cost path to the root bridge, then disables all other paths.

Network managers may design a topology that uses redundant links as automatic backup paths in the case of active link failure. Automatic backup takes place without the pitfalls of bridge loops, or the need to manually enable or disable backup links.

ZebOS-XP STP supports all STP switch port states, including:

- Listening, learning, blocking, forwarding, disabled

For more information about the Spanning Tree modules, see the *Layer 2 Developer Guide*.

Rapid Spanning Tree Protocol (802.1w)

The Rapid Spanning Tree Protocol (RSTP) accelerates the re-configuration and restoration of a spanning tree after a link failure.

Multiple Spanning Tree Protocol (802.1s)

The Multiple Spanning Tree Protocol (MSTP) is a supplement to the IEEE 802.1ad standard. MSTP allows VLAN bridges to use multiple spanning trees, by providing the ability for traffic belonging to different VLANs to flow over potentially different paths within the virtual bridged LAN.

Transparent Interconnection With Lot of Links

TRILL (TRansparent Interconnection of Lots of Links) eliminates the problems associated with using the Spanning Tree Protocol in a data center network. Spanning tree protocols restrict all traffic to a loop-free tree and in doing so creates blocking conditions that require the over provisioning of links. With TRILL, you can create a fully meshed network where all links are available on all paths, eliminating the need to over-provision links and improving the utilization of data center networking equipment.

TRILL provides Layer 2 bridging using IS-IS link state routing. TRILL encapsulates native frames in a transport header that contains a hop count, routes the encapsulated frames using IS-IS, and decapsulates the native frame before delivery.

An RBridge (routing bridge) is a device that implements TRILL. Each RBridge has a copy of the global link state database that it uses to calculate the optimum next hop towards the destination RBridge. TRILL runs directly at Layer 2, and is therefore not dependent upon IP addressing.

TRILL features include:

- Multi Path Forwarding-even distribution of traffic across available paths; maximum usage of available alternate pages
- Retention of state information-for faster convergence, stability and optimization of IP Multicast
- Minimize looping by decrementing the hop count

- Implementation of routing bridges

For more information about TRILL, see the *Transparent Interconnection of Lots of Links Developer Guide.*

Carrier Ethernet

ZebOS-XP offers a comprehensive set of Carrier Ethernet (CE) protocols, including MEF, IETF, and IEEE.

For more information about Career Ethernet, see the *Carrier Ethernet Developer Guide.*

Provider network operator can benefit from these VLAN features:

- Provider Bridging (802.1ad)
- Provider Backbone Bridging (802.1ah)
- Provider Backbone Bridge-Traffic Engineering (802.1Qay)

Link Level Discovery Protocol (802.1AB)

Link Layer Discovery Protocol (LLDP) is an agent running on an IEEE 802.1 bridge that provides a mechanism for all the bridges connected to the LAN to send and receive connectivity and management related information to each other.

Connectivity Fault Management (CFM)

Connectivity Fault Management (CFM) addresses the per-customer and per-service OAM granularity required to manage Layer 2 Ethernet services. CFM enables service providers to configure:

- Maintenance End Points (MEP) on a per-port, per-VLAN, or per-domain basis
- Maintenance Intermediate Points (MIP) on a per-port and per-level basis

CFM can operate over LAN segments, C-VLAN (customer VLAN), S-VLAN (service VLAN), B-VLAN (backbone VLAN), and backbone service instances identified by service instance IDs (I-SIDs). Support for enhanced CFM OAM fulfills the requirements of the of the Y.1731 standard with periodic, on-demand, and asynchronous messaging. CFM MEF-UNI is also supported.

Ethernet to the First Mile (802.3ah)

Ethernet to the First Mile (EFM) is a set of extensions to the IEEE 802.3 MAC and MAC (Media Access Control) sub layer. The EFM protocol introduces the concept of an Ethernet Passive Optical Network (EPON). EFM describes subscriber access technologies and the physical layer specifications for subscriber access. In subscriber access networks, this requires implementing only the far-end OAM features, which includes remote failure detection, remote loop back, and link monitoring.

Data Center Bridging

Data Center Bridging (DCB) is a collection of standards-based extensions for Ethernet protocols. DCB is a lossless transport layer to allow the convergence of LANs and SANs into a single unified fabric. DCB is a flexible framework that defines the capabilities required for switches and end-points to be part of a data center environment. DCB protocols can carry Fibre Channel, TCP/IP, and Inter-process communication traffic over a single, converged Ethernet network.

ZebOS-XP DCB provides these key benefits:

- Enables Fibre Channel over Ethernet (FCoE) to make possible the migration of SANS to more cost-effective Ethernet
- Enables high-speed connections over Carrier Ethernet between geographically dispersed data centers

Data center networks and backplane fabrics employ applications that depend on the delivery of data packets with a lower latency and lower probability of packet loss than is typical of IEEE 802 VLAN bridged networks. DCB supports the use of a single bridged local area network for these applications, as well as traditional LAN applications.

DCB features include:

- Lossless fabric for traffic on converged networks
- Virtual links by traffic class that includes pause-per-class technology, congestion management and event notification
- Improved performance, including stability, enhanced throughput, and robustness

The following subsection highlights the features of DCB.

For more information about DCB, see the *Data Center Bridge Developer Guide.*

Priority-based Flow Control (IEEE 802.1Qbb)

The Priority-based Flow Control (PFC) standard specifies protocols, procedures and managed objects that enable flow control per traffic class on full-duplex links. Priority-based flow control helps eliminate frame loss due to congestion by operating on individual priorities. Along with other Data Center Bridging technologies, PFC helps the flow of higher-layer protocols, which are highly loss sensitive, while not affecting traditional LAN protocols that utilize other priorities. Data Center Bridging networks (bridges and end-nodes) include limited bandwidth-delay product and limited hop count.

VLAN tag priority values identify traffic classes. Priority-based flow control helps eliminate frame loss due to congestion by a mechanism similar to the IEEE 802.3x PAUSE, but operates on individual priorities.

Enhanced Transmission Selection (IEEE 802.1Qau)

The Enhanced Transmission Selection (ETS) standard supports the allocation of bandwidth among traffic classes. When the available load in a traffic class does not use its allocated bandwidth, ETS allows other traffic classes to use available bandwidth.

A network can prioritize traffic to offer different service characteristics to different traffic classes. However, a user may want to share bandwidth among the priorities that are carrying bursty, high-offered loads, instead of servicing them with a strict priority. Therefore, traffic set at a specific priority level that does not use its set allocation allows traffic set at other priorities to utilize the unused bandwidth.

Data Center Bridging eXchange (DCBx)

The Data Center Bridging eXchange (DCBx) protocol is used by DCB-enabled devices to exchange information with directly connected peers. The DCBx may also be used to detect mis-configurations and to configuring the peers. DCBx uses the Link Layer Discovery Protocol (LLDP) as a signaling mechanism.

Quantized Congestion Notification (802.1Qau)

Quantized Congestion Notification (QCN) addresses the problem of sustained congestion by moving corrective action to the network edge.

Port Authentication (authd)

The ZebOS-XP Layer 2 802.1x module provides port-based network access control for LAN devices. The IEEE 802.1x standard offers centralized control of user authentication and access.

Precision Time Protocol

The Precision Time Protocol (PTP) is a protocol used to synchronize clocks throughout a computer network. On a local area network, PTP achieves clock accuracy in the sub-microsecond range, making it suitable for measurement and

control systems. PTP (IEEE 1588v2) introduced the concept of “boundary clocks” and “transparent clocks” that serve to further improve system and network scalability and the accuracy of clock synchronization.

PTP is an event-based protocol for time synchronization. PTP allows a better precision than other time synchronization protocol because of its “on-path support” and precision in sending the exact timestamp. On-path support in PTP is provided through boundary and transparent clocks. Boundary clocks synchronize time to the grandmaster and transmit it further on the network. There is no need for the grandmaster to send synchronization messages to all slave clocks. Boundary clocks in between can act as master.

Transparent clocks do not synchronize with the grandmaster or master. They just add correction to the timestamp by taking into account link delays and residence time.

For more information about PTP, see the *Precision Time Protocol Developer Guide.*

Synchronous Ethernet

Packet-switching networks such as Ethernet were designed to transport asynchronous data where strict synchronization is not needed because traffic is highly buffered. However, with Carrier Ethernet evolution, Ethernet is replacing the SONET/SDH/PDH infrastructure and carrying TDM traffic. This means that packet networks need frequency synchronization functions, especially at their edge. The adaptation of TDM signals into a packet network is called Circuit Emulation Services (CES). Synchronous Ethernet (SyncE) is an evolution of conventional Ethernet with SONET/SDH/PDH-based synchronization. Synchronous Ethernet is used to synchronize and send frequency information to nodes on the network. Synchronous Ethernet provides only frequency synchronization, not time or phase synchronization.

In Synchronous Ethernet, a reference-timing signal traceable to a Primary Reference Clock (PRC) is injected into the network element using an external clock port. The Synchronous Ethernet interface extracts the frequency of this reference-timing signal and locks it to a system clock. This reference frequency is distributed to downstream Synchronous Ethernet nodes. Using this method, timing originating from a PRC is distributed via the intermediate Ethernet equipment clocks (EEC), to the other nodes in the network.

For more information about SyncE, see the *Synchronous Ethernet Developer Guide.*

Shortest Path Bridging (IEEE 802.1aq)

SPB (Shortest Path Bridging) enables you to simplify how you create and configure networks—across the enterprise and for the cloud—by requiring service provisioning only at the edge of the network. Shortest Path Bridging (SPB) is a control plane protocol that combines an Ethernet data path with an Intermediate System To Intermediate System (IS-IS) link state protocol running between bridges. SPB does not depend on spanning tree protocols to provide a loop-free topology, but instead uses IS-IS Link State Packets (LSPs) to discover and advertise the network topology and compute the shortest path trees from all bridges in the SPB area. RFC 6329 describes the IS-IS extensions to support 802.aq SPB.

There are two types of SPB depending on the type of Ethernet data path:

- Shortest Path Bridging - VID (SPBV) uses an 802.1ad “Q-in-Q” data path
- Shortest Path Bridging - MAC (SPBM) uses an 802.1ah “MAC-in-MAC” data path

For more information about SPB, see the *Shortest Path Bridging Developer Guide.*

SPB, like TRILL, uses the IS-IS link-state routing protocol to advertise both topology and logical network membership. SPB packets are encapsulated at the edge in either MAC-in-MAC 802.1ah or tagged 802.1Q/802.1ad frames and transported only to other members of the logical network. Unicast and multicast are supported and all routing is on symmetric shortest paths.

The key benefits of the SPB solution are:

- Supports thousands of multi-terabyte switches and enables a non-blocking two-tier network fabric

- Uses existing IS-IS protocol and interoperates with existing service provider equipment
- Already available through OAM, enabling service providers to quickly roll out managed services

IEEE 802.1aq can be used anywhere that Spanning Tree is currently being used. Operators can take existing STP/MSTP-based networks and migrate to SPB.

With SPB, the data center network can be treated as one big Layer 2 switch by combining hundreds of smaller switches in a “non-blocking” topology. This big distributed switch can be used to interconnect hundreds of different customers cheaply with Layer 2 VPNs. The topology is fully compatible with all 802.1 data center bridging protocols. OAM (Operation, Administration, and Management) and address isolation is provided through MAC-in-MAC.

Edge Virtual Bridging

Software switches cannot achieve the level of network capabilities built into enterprise-class Layer 2 data center switches. To solve the management challenges with Virtual Ethernet Bridging (VEB), Edge Virtual Bridging (EVB) was developed with a primary goal to combine the best of VEBs with the best of external Layer 2 network switches.

EVB is based on VEPA (Virtual Ethernet Port Aggregator) technology. EVB enables the virtual switches to send all traffic and forwarding decisions to an adjacent physical switch. This moves the forwarding decisions and network operations from the host CPU to the switch. EVB also leverages the advanced management capabilities in the access or aggregation layer switches.

For more about EVB, see the *Edge Virtual Bridging Developer Guide.*

Ethernet Automatic Protection Switching

Ethernet Automatic Protection Switching (APS) detects signal failure or signal degrade on a working channel and switches traffic transmission to a protection channel.

Ethernet Linear Protection Switching

Ethernet Linear Protection Switching (ELPS) as defined by ITU-T G.8031 (2007) specifies these techniques:

- Linear 1+1 (One-plus-One) protection switching operates with either uni-directional or bi-directional switching; normal traffic is copied and fed to both working and protection transport entities
- Linear 1:1 (One-to-One) protection switching operates with bi-directional switching; normal traffic is transported either on the working transport entity or on the protection transport entity, using a selector bridge at the source

For more about ELPS, see the *Carrier Ethernet Developer Guide*.

Ethernet Ring Protection Switching

Ethernet Ring Protection Switching (ERPS) protects Ethernet traffic in a ring topology by ensuring that no loops are within the ring. Loops are prevented by blocking traffic on either a predetermined link or a failed link. ERPS integrates Ethernet Operations, Administration, and Maintenance (OAM) functions with a simple APS protocol. An Ethernet ring uses normal learning, forwarding, filtering, and flooding mechanisms and a forwarding data base (FDB).

ZebOS-XP supports multiple Ethernet ring instances, in accordance with ITU-T G.8032 (2012).

For more about ERPS, see the *Carrier Ethernet Developer Guide*.

Ethernet Local Management Interface

Ethernet Local Management Interface (E-LMI) is an OAM (Operation, Administration, and Management) protocol used for communications between two User Network Interfaces (that is, UNI-C and UNI-N). E-LMI provides both UNI and EVC (Ethernet Virtual Connection) status information to customer edge (CE) devices. This information enables

automatic configuration of CE operation based on the Metro Ethernet Network configuration. The ZebOS-XP E-LMI module follows the Metro Ethernet Forum 16 specification.

For more about E-LMI, see the *Ethernet Local Management Interface Developer Guide*.

Remote Monitoring

Remote monitoring (RMON; defined by RFC RFC 2819) provides remote monitoring and management of network devices and enables these devices to communicate with each other to exchange network information.

For more about RMON, see the *Layer 2 Developer Guide*.

Layer 3 Protocols

ZebOS-XP supports both IPv4 and IPv6 versions of:

- Border Gateway Protocol
- Open Shortest Path First
- Routing Information Protocol
- Intermediate System to Intermediate System
- Virtual Router Redundancy Protocol
- Bidirectional Forwarding Detection

Figure 2-4 shows how Layer 3 routing protocols interact with the Unicast Routing Information Base Daemon and Network Services Module.

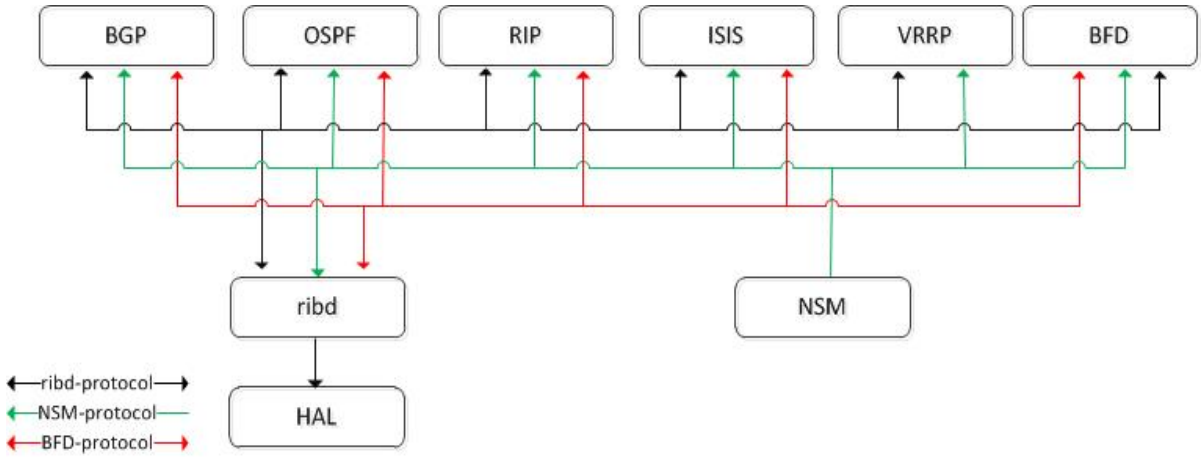

Figure 2-4: Layer 3 routing architecture

The protocol modules communicate with Network Services Module for interface updates and with the Unicast Routing Information Base Daemon for route updates and redistributed routes.

In addition to the standard Layer 3 routing protocols, ZebOS-XP offers:

- Virtual Routing (VR) support
- Traffic Engineering (TE) extensions
- Constrained Shortest Path First (CSPF) topology support for the Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) protocols

Unicast Routing Information Base Daemon

In ZebOS-XP, the unicast Routing Information Base daemon (`ribd`) is responsible for maintaining a central unicast Routing Information Base.

A Routing Information Base (RIB) is a data structure stored in a network device that lists the routes to particular network destinations and metrics (distances) associated with those routes. A RIB contains information about the topology of the network immediately around it. Maintaining a RIB by discovering network topology is the primary purpose of dynamic routing protocols such as BGP, RIP, and OSPF. Static fixed routes are added to a RIB by commands. (A RIB is also called a routing table.)

A Forwarding Information Base (FIB) is used to find the proper interface to which an input interface should forward a packet. In contrast to RIBs, FIBs are optimized for fast lookup of destination addresses. (A FIB is also called a forwarding table.)

Protocol modules create their own routes, and communicate this protocol-specific information to `ribd` via Inter-Process Communication (IPC) messaging functions. The `ribd` RIB contains all routing information received from routing peers, for example, destination prefix, nexthop information, and distance.

Figure 2-5 shows how the Layer 3 protocols, `ribd`, and the kernel communicate.

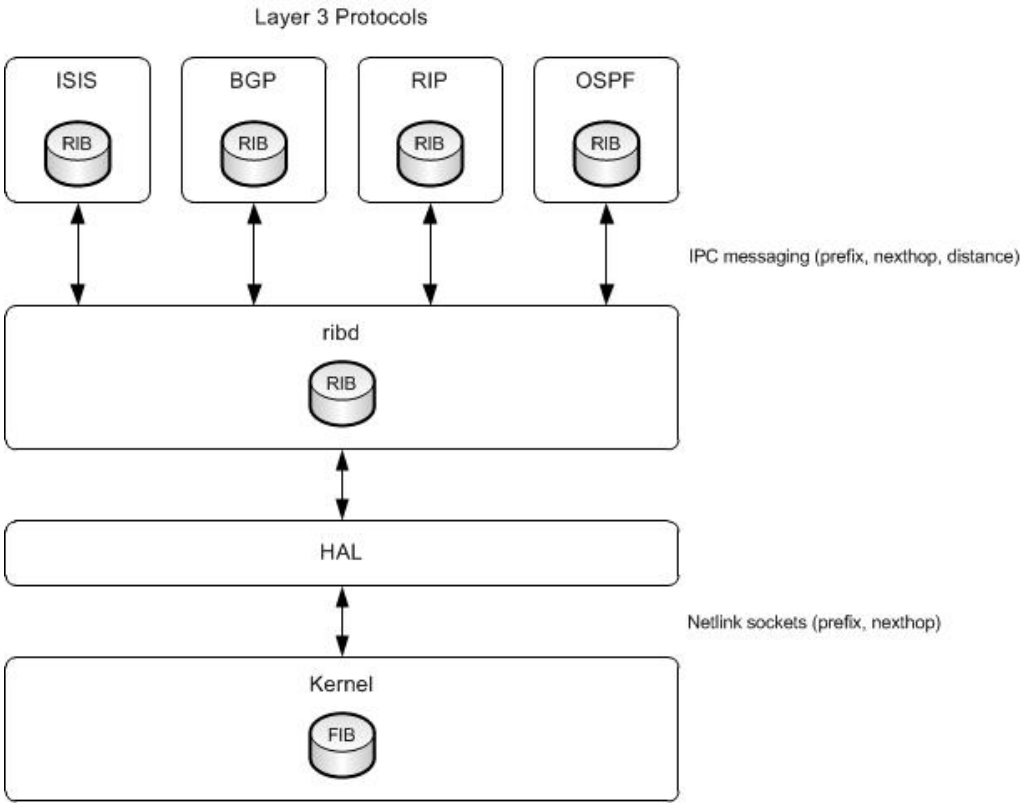

Figure 2-5: Protocol, ribd, and kernel interaction

The tasks of the `ribd` daemon are to:

- Communicate with ZebOS-XP routing and switching modules to get routing information updates
- Provide configuration for static routes
- Process the routing information and maintain all the received routes from clients as part of the RIB
- Maintain the FIB
- Process the routes and select the FIB route and program the kernel.

- Redistribute routes
- Handle interface up and down events

For every known prefix, `ribd` maintains a route node entry in its RIB. The `ribd` process populates this table upon receiving routes from:

- Protocols such as BGP, OSPF, and RIP
- Static routes configured using commands
- The kernel FIB
- Connected routes derived from interface information

Routing protocols use different metrics to calculate the best path for a destination. The best path is sent to the `ribd` process.

The kernel FIB contains the minimum amount of information required to make a forwarding decision on a particular packet, for example, destination prefix and nexthop information. This abbreviated form of the information in the RIB is transferred from `ribd` to the kernel FIB via Netlink sockets.

For more about the ribd daemon, see the *Unicast Routing Information Base Developer Guide*.

Border Gateway Protocol

Border Gateway Protocol (BGP) is a core exterior gateway protocol (EGP) used on the Internet. BGP maintains a table of IP networks, or prefixes, which designate network reachability among Autonomous Systems (AS). BGP is a path-vector protocol that makes routing decisions based on path, network policies, and/or rule sets.

ZebOS-XP supports Border Gateway Protocol (BGP4+) versions 4 and 4+ and offers optional VPN extensions for MPLS-VPN support. The VPN extensions work with NSM, MPLS-LDP, RSVP-TE, and MPLS forwarder modules.

ZebOS-XP BGP4+ features include:

- IPv4 and IPv6 support
- Route Reflection
- Route Refresh
- Community Attributes
- Community Attributes in Multi-Home Routing
- Extended Communities
- Protection of BGP Session via the TCP MD5 Signature Option
- Capabilities Advertisement
- Route Flap Damping
- E-BGP Multi hop
- Stateful implementation
- Multi-protocol BGP extensions
- BGP/MPLS VPNs
- BGP Graceful restart
- BGP4+ Inter-Domain Routing (IDR):
    - Virtual Routing and Virtual Routing and Forwarding Support
    - Full MIB support

For more information about BGP, see the *Border Gateway Protocol Developer Guide.*

Open Shortest Path First

Open Shortest Path First (OSPF) version 2 is a link-state routing protocol that runs internally on a single autonomous IPv4 system. Each router designated to run OSPF maintains an identical database describing the autonomous system's topology. From this database, a routing table is calculated by constructing a shortest-path tree. OSPF version 3 contains extensions to support IPv6 networks. The protocol is also extensible to other address families. ZebOS-XP supports the following OSPF standards:

ZebOS-XP OSPF features include:

- Opaque Link State Attributes (LSA)
- Multiple Instance Support
- Intra- and inter-area routing
- Type 1/2 external routing
- OSPF “not so stubby area” (NSSA) Option
- Inter domain Routing Protocol (IDRP) for IP-OSPF Interaction
- Opaque link state availability (LSA) Option
- Manual and automatic virtual links
- Equal Cost Multi Path (ECMP)
- Broadcast, point-to-point and point-to-multi-point models
- MD5 authentication
- Incremental SPF
- Traffic Engineering extensions
- Virtual Routing (VR) and Virtual Routing and Forwarding (VRF) support
- Graceful restart
- Full MIB support
- Optional Virtual Private Network (VPN) support
- Optional Constrained Shortest Path (CSPF) support

For more information about OSPF, see the *Open Shortest Path First Developer Guide.*

Routing Information Protocol

The Routing Information Protocol (RIP) version 2 is one of the most commonly used interior gateway protocols (IGP) for routing on internal IPv4 networks, and to a lesser extent, networks connected to the Internet. RIP employs a distributed variant of the Bellman-Ford algorithm to provide distance vector routing capabilities. RIP also supports subnet information, thus allowing Classless Inter-domain Routing (CIDR).

RIPng contains extensions to support IPv6 networks.

ZebOS-XP RIP/RIPng features include:

- Split horizons with poisoned reverse
- Triggered updates
- ECMP (Equal Cost Multi path Routing) support
- Virtual Routing and Virtual Routing and Forwarding Support
- Full MIB support

For more information about RIP, see the *Routing Information Protocol Developer Guide.*

Intermediate System to Intermediate System

Intermediate System-to-Intermediate-System (IS-IS) is a link-state routing protocol that runs internally on a single autonomous system. IS-IS routers maintain identical databases that describe the autonomous system's topology. A routing table is calculated from the database by constructing a shortest-path tree.

ZebOS-XP IS-IS features includes:

- Traffic Engineering extensions
- IS-IS MIB support
- Multi-protocol support for IPv4 and IPv6
- Equal cost Multi path
- Dynamic Host name Exchange Mechanism
- IS-IS Mesh Groups
- Traffic Engineering
- Restart Signaling
- Multi-topology routing
- Virtual Routing and Virtual Routing and Forwarding Support

For more information about IS-IS, see the *Intermediate System and Intermediate System Developer Guide.*

Virtual Router Redundancy Protocol

The Virtual Router Redundancy Protocol (VRRP) allows a virtual router composed of two or more VRRP routers on the same subnet to prevent failure by providing at least one standby virtual router if the master virtual router fails. VRRP eliminates the single point of failure most common in a static default routed environment.

In ZebOS-XP, VRRP functions are handled by the `vrrpd` daemon whose tasks are to:

- Bring up or down VRRPv4 or VRRPv6 sessions in the respective VR mode
- Create and delete VRRP sessions dynamically
- Transmit and receive VRRP packets to and from the virtual-router peers based on the time-out value or the current state of the VRRP router and performs functions related to the VRRP Finite State Machine (FSM).

VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN. The VRRP router controlling the IP addresses associated with a virtual router is called the master, and it forwards packets sent to these IP addresses. The election process manages dynamic fail-over in the forwarding responsibility should the master become unavailable. Any of the virtual router's IP addresses on a LAN can then be used as the default first-hop router by end-hosts. The advantage of using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end-host.

The NSM client code in the VRRP daemon handles updates from NSM and builds the local interface table. The VRRP master sends the VIP and VMAC to HSL which is installed in the kernel. This allows the master to process the forward the traffic sent to VRRP Virtual Router.

ZebOS-XP supports VRRP as specified in RFC 5798.

For more information about VRRP, see the *Virtual Router Redundancy Protocol Developer Guide.*

Bidirectional Forwarding Detection

Bidirectional Forwarding Detection (BFD) reduces the reliance upon the relatively slow Hello mechanism in routing protocols to detect failures where no hardware signaling is available to assist. BFD works with BGP, OSPFv2, and IS-IS to enable them to configure BFD sessions, and for the sessions to receive the bidirectional forwarding failure notifications.

BFD provides the following features:

- A single mechanism to detect liveliness over any media and in any protocol layer
- Rapid detection of communication failures between adjacent systems to quickly establish alternative paths
- Passive, Active, Synchronous, Asynchronous, and Demand modes of operation
- Improved system performance when faster detection is required, because data-plane reachability detection is detached from control-plane functionality
- A separate process that works in tight conjunction with routing protocols and NSM to detect forwarding plane reachability to protocol next hops
- ZebOS-XP protocol modules support BFD irrespective of where BFD packet-sending operations take place: in the interfaces, data links, or to some extent, in the forwarding engines themselves
- BFD is Graceful-Restart unaware: whenever BFD timers expire, a session-down event is triggered to the protocol module, and BFD maintains sessions for the protocol while it undergoes Graceful Restart
- A fast mechanism to detect liveliness of static next-hops

For more information about BFD, see the *Bidirectional Forwarding Detection Developer Guide.*

Equal Cost Multi-Path

Equal-Cost Multi-Path (ECMP) is a forwarding mechanism for routing packets along multiple paths of equal cost and ensures load balancing across multiple paths. The ZebOS-XP implementation leverages Forwarding Plane Load Balancing. Link-state protocols which use a cost-based metric such as OSPF and ISIS explicitly allow ECMP routing.

ZebOS-XP installs the maximum number of ECMP routes supported by a kernel. This allows for load balancing to be performed with more than one nexthop to reach a destination. If the router receives and installs multiple paths with the same administrative distance and cost to a destination, load-balancing is possible.

IPv6 Transition

With ZebOS-XP, you can integrate into your existing gateways and Customer Premises Equipment (CPE), protecting your network investment. ISPs and other access providers require a migration strategy to enable them to continue to provide full connectivity throughout the transition to IPv6. The is applicable for both new IPv6-based services and existing IPv4 services moving to IPv6 with:

- IPv6 Rapid Deployment provides IPv6 tunnels over existing IPv4 access network between your CPEs and your gateways to provide your customers secure access to IPv6 networks and services.
- IPv4 Residual Deployment provides stateless IPv4 over IPv6 tunnels for your existing IPv4 customers with efficient global IPv4 address sharing. The advantage of stateless tunneling is that it does not require managing per user state; abusive users can easily be identified. Furthermore, stateless tunneling simplifies high availability and scalability of the gateway since the need to scale is only based on traffic volume.

Multicast Protocols

ZebOS-XP provides these multicast protocol daemons:

- Layer 2 Multicast Routing Information Base Daemon
- Layer 3 Multicast Routing Information Base Daemon
- Protocol-Independent Multicast Daemon
- Multi Protocol Label Switching Protocols

Layer 2 Multicast Routing Information Base Daemon

The responsibilities of Layer 2 multicast Routing Information Base daemon (`l2mribd`) include:

- Updates hardware with snooping information.
- Interacts with NSM, for bridge, VLAN, and port related information and their association with each other.
- Interacts with MSTP for spanning tree related information.
- Interacts with MSTP to receive VLAN-to-instance mapping and TCN messages.
- Communicates with the `mribd` daemon to receive IGMP status on the VLANs. If IGMP is enabled on a VLAN, IGMP snooping functionality is disabled and vice versa.
- Processes IGMP and MLD packets and maintains group membership for all interfaces based on which multicast traffic forwarding is done. It also add/deletes the group membership information in hardware.

Figure 2-6 shows the Layer 2 multicast architecture.

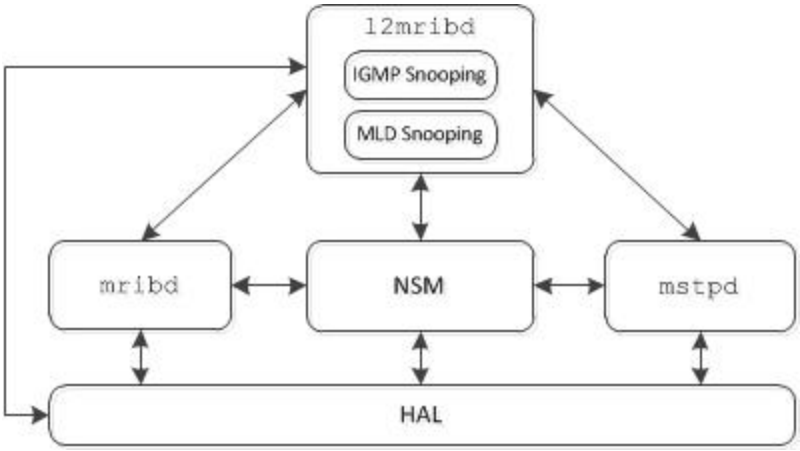

Figure 2-6: Layer 2 multicast architecture

The Layer 2 multicast modules support IPv4 or IPv6, or combination IPv4 and IPv6.

The features of the `l2mribd` daemon include:

- Internet Group Management Protocol (IGMP) Snooping: ability to passively snoop on IGMP packets to learn the IPv4 multicast group membership information. With IGMP snooping, multicast traffic in a group is only forwarded to ports that have members in that group. ZebOS-XP supports IGMP snooping functionality for IGMP versions 1, 2, and 3.
- Multicast Listener Discovery (MLD) Snooping: ability to passively snoop on MLD packets to learn the IPv6 multicast group membership. With MLD snooping, multicast traffic is only forwarded to ports that request receipt of packets from the source. ZebOS-XP supports MLD snooping functionality for MLD versions 1 and 2

For more information about Layer 2 multicast support, see the *Multicast Protocol Developer Guide.*

Layer 3 Multicast Routing Information Base Daemon

The multicast protocols communicate with the `mribd` daemon which communicates with the multicast forwarder. A common multicast routing information base allows multiple multicast protocols to function simultaneously.

Figure 2-7 shows the Layer 3 multicast architecture of ZebOS-XP. The `mribd` daemon holds the multicast RIB and consolidates the routes from multicast routing protocols such as Protocol-Independent Multicast Daemon and Multi Protocol Label Switching Protocols and installs them in the kernel and sends them to HSL to install in the hardware. NSM provides the VR/VRF as well as the interface updates to the `mribd` daemon, the `pimd` daemon, and the kernel multicast forwarder.

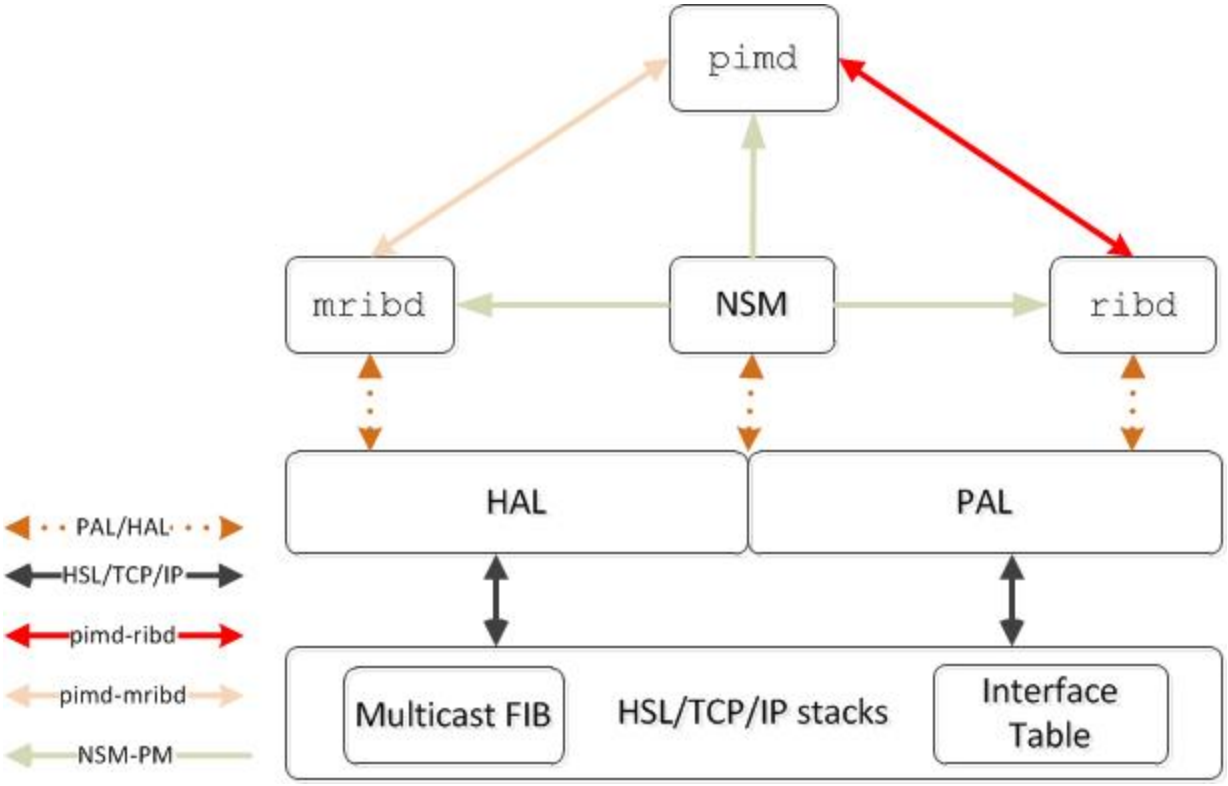

Figure 2-7: Layer 3 multicast architecture

The `mribd` daemon supports the following functions:

- Adds, deletes, and updates multicast routes in the route table and programs the kernel
- Multicast Virtual Interface (VIF) management for IPv4 networks
- Multicast Interface (MIF) management for IPv6 networks
- Multicast Route Entry management for IPv4 and IPv6
- Multicast Forwarder Event handling for IPv4 and IPv6
- Multicast Forwarding Entry statistics for IPv4 and IPv6
- Multicast Tunnel management for IPv4 only
- IGMP versions 1, 2, and 3 services (IPv4): Used by IP hosts and adjacent multicast routers to establish multicast group memberships
- IGMP Proxy

- Signals IGMP membership to multicast routing protocols like PIM
- MLD versions 1 and 2 services (IPv6): used by a by a device to discover listeners for a specific multicast group in the same manner as IGMP is used in IPv4
- MLD Proxy

For more information about Layer 3 Multicast, see the *Multicast Protocol Developer Guide.*

Protocol-Independent Multicast Daemon

Protocol-Independent Multicast (PIM) is a family of multicast routing protocols for IP networks that provides one-to-many and many-to-many distribution of data over a network. PIM is termed *protocol-independent* because it does not have its own topology discovery mechanism, but instead uses routing information supplied by other routing protocols.

ZebOS-XP support these variants of PIM:

- PIM Sparse Mode (PIM-SM: RFC 4601) efficiently establishes distribution trees across wide area networks (WANs) by routing packets to multicast groups. PIM-SM constructs a tree from each sender to the receivers in a multicast group and packets from the sender follow the tree to interested recipients. PIM-SM is for situations where multicast groups are thinly populated across a large region. Although it can operate in LAN environments, it is most efficient in WAN environments.
- PIM Dense Mode (PIM-DM: RFC 3973) is a data-driven routing protocol that builds source-based multicast distribution trees that operate on the flood-and-prune principle. PIM-DM uses dense multicast routing and implicitly builds shortest-path trees by flooding multicast traffic domain wide, and then pruning back branches of the tree where no receivers are present. PIM-DM effectively distributes data to target recipients in a concentrated area. PIM-DM is straightforward to implement but generally has poor scaling properties.
- PIM Source-Specific Multicast (PIM-SSM: RFC 3569) is a subset of PIM-SM that allows deployment of SSM in a network with hosts that do not support IGMP version 3. PIM-SSM builds trees that are rooted in just one source, offering a more secure and scalable model for a limited amount of applications (mostly broadcasting of content). In PIM-SSM, an IP datagram is transmitted by a source S to an PIM-SSM destination address G, and receivers can receive this datagram by subscribing to channel (S,G).
- Bidirectional PIM (BIDIR-PIM: RFC 5015), is a variant of PIM-SM which dispenses with both encapsulation and source state by allowing packets to be natively forwarded from a source to the RP using shared tree state. These benefits can be observed when using a Many-Many multicast deployment.

The `pimd` daemon holds the PIM states and takes care of PIM packet processing. The `pimd` daemon also holds the Tree Information Base (TIB) with all multicast routes. The `pimd` daemon operations include:

- Interacts with NSM for interface updates
- Receives local group membership information from IGMP (MRIB). Installs multicast interfaces (VIF) and multicast routes to MRIB.
- Processes PIM packets from PIM neighbor routers

The PIM features are:

- Any Cast RP
- Bootstrap router
- PIM MSDP
- PIM border

For more, see the *Protocol Independent Multicasting Developer Guide*.

Multi Protocol Label Switching Protocols

Multi-Protocol Label Switching (MPLS) operates at a layer operated between traditional definitions of Layer 2 (data link layer) and Layer 3 (network layer). The MPLS modules provide integration into enterprise, edge, and core communications equipment.

The ZebOS-XP MPLS module supports the following protocols and features:

- Label Distribution Protocol
- Traffic Engineering
- Resource ReSerVation Protocol-Traffic Engineering
- Differentiated Services
- Layer 2 Virtual Private Network
- Layer 3 Virtual Private Network
- MPLS-TP Operation, Administration and Management

For more information about MPLS, see the *Multi Protocol Label Switch Developer Guide*

Figure 2-8 shows the MPLS high-level architecture.

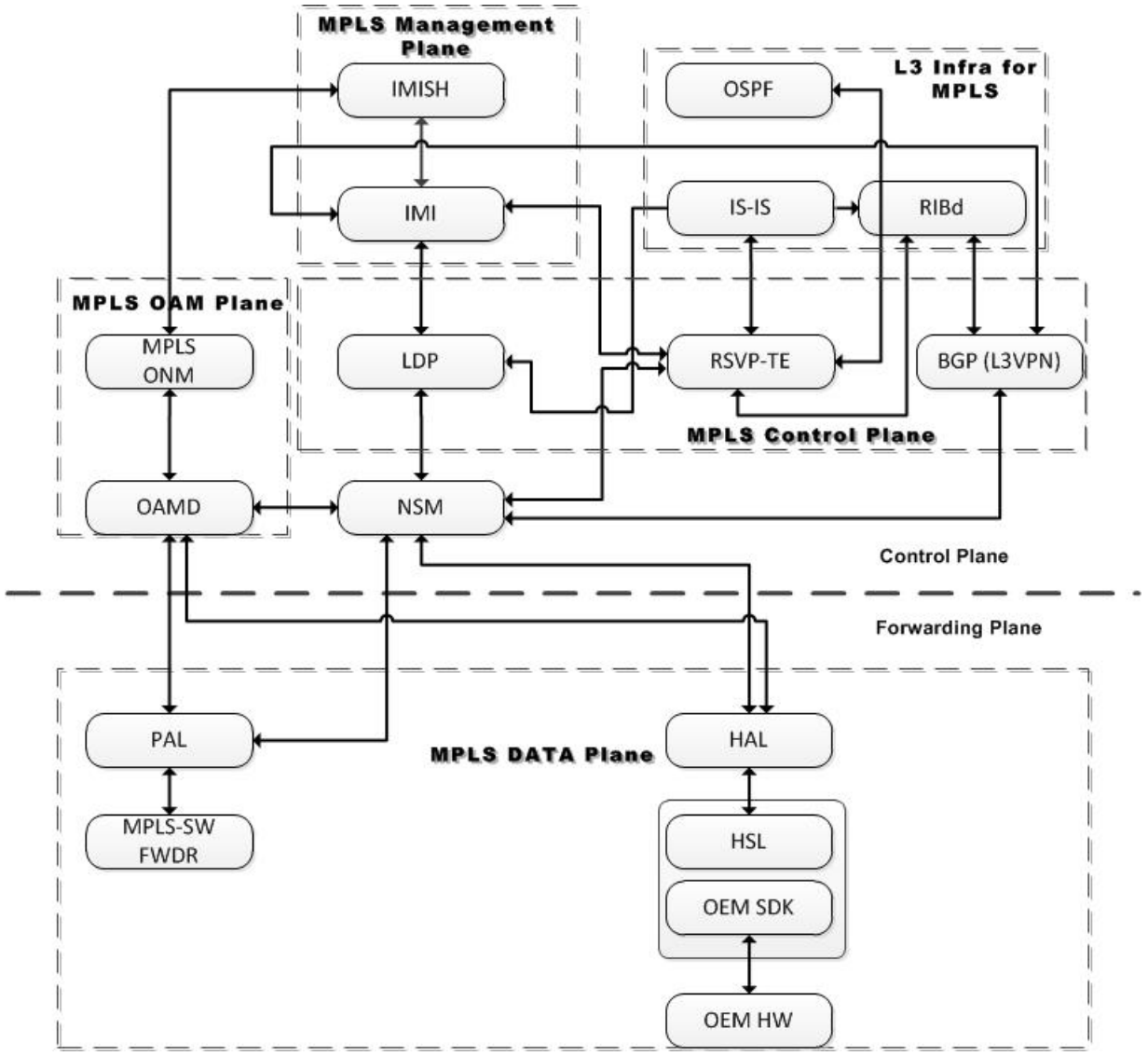

Figure 2-8: MPLS High Level Architecture

The following summarizes the architecture.

- IMISH: Provides the commands to configure the MPLS/MPLS-TP on-demand OAMs. This module interacts with the MPLS ONM to interface with the OAMD module to configure an on-demand OAM.
- MPLS ONM: Interface between IMISH and OAMD, which performs the OAM functionality. The OAMD uses the MPLS ONM to send the results of an active on-demand OAM session to IMISH to display the results to the end-user.
- OAMD: Performs on-demand and proactive OAM activities. It also provides the configuration and show commands for all proactive OAMs. This module interacts with NSM when the MPLS/MPLS-TP transport path information is required to transmit an OAM payload and to act on a received OAM payload. It uses PAL to send and receive the OAM payloads via the MPLS software forwarder.
- NSM: Houses all the MPLS/MPLS-TP transport path information. It handles the OAMD module, providing the required transport path information for OAM payload transmission and reception.

- LDP: Distributes labels between peers to form a Label Switched Path (LSP). NSM houses the MPLS Routing Information Base. LDP sends FTN/ILM obtained as result of LSP signaling to NSM. NSM installs these routes in MPLS software forwarder/hardware which will be used to forward data packets. This module also establishes LDP sessions between non-directly connected peers (T-LDP).
- RSVP-TE: Establishes traffic engineered label switched path. RSVP-TE queries CSPF to compute the explicit routes. Similar to LDP, RSVP-TE also sends FTN/ILM to NSM and NSM installs in MPLS software forwarder/ hardware.
- PAL: Interface between the OAMD and the MPLS software forwarder. It provides the necessary APIs to OAMD to send the OAM payloads to the MPLS software forwarder. It also aids the OAMD in registering for callbacks required to process incoming OAM payloads.
- MPLS software forwarder: Transmits OAM payloads received from OAMD over an MPLS/MPLS-TP transport path. This module also de-multiplexes any MPLS/MPLS-TP OAM payload received over a transport path and sends it to the OAMD for further processing.

Label Distribution Protocol

With the Label Distribution Protocol (LDP), two label-switched routers (LSR) exchange label mapping information. The two LSRs are called LDP peers and the exchange of information is bi-directional. LDP is used to build and maintain databases of LSRs that are used to forward traffic through MPLS networks.

LDP works with other routing protocols (RIP, OSPF, and BGP) to create the label-switched paths (LSP) used when forwarding packets. An LSP is the path taken by all packets that belong to the Forwarding Equivalence Class (FEC) corresponding to that LSP. In this way, ZebOS-XP LDP assigns labels to every destination address and destination prefix provided by the `ribd` daemon. The LDP interface to the MPLS forwarder adds labels to and deletes labels from the forwarding tables.

Traffic Engineering

Traffic Engineering (TE) is a signaling protocol that supports explicit routing capability to establish Label-Switched Paths (LSPs) in a MPLS network. RSVP-TE creates explicitly routed paths, which might not agree with the route suggested by the Interior Gateway Protocol (IGP) such as OSPF or IS-IS.

Resource ReSerVation Protocol-Traffic Engineering (RSVP-TE) provides signaling information for traffic engineering in an MPLS network. A router that supports MPLS as either an LER or an LSR uses RSVP-TE to request quality of service (QoS) tasks from the network for a flow that a network operator has provisioned.

TE extensions use the RSVP and LDP dynamic-signaling protocols to communicate to the ZebOS-XP MPLS forwarder or a third-party MPLS forwarder. The RSVP-TE extension enables MPLS to scale into large and complex IP-based communications equipment.

For computing explicit routes to reach the destination:

1. RSVP-TE queries Constrained Shortest Path First (CSPF) (OSPF or IS-IS) for a complete, end-to-end, explicit route based on constraints specified by the network administrator.
2. CSPF refers to the Traffic Engineering Database (TED) for computing the explicit route.
3. The resulting explicit route is used by a signaling protocol (RSVP-TE or LDP) to set up LSPs.

NSM maintains local TE parameters and it updates the TED of IS-IS and OSPF. If there is any change in TE-parameters, NSM updates the same in IS-IS/OSPF. IGPs exchange the TE parameters with their peers.

MPLS for Traffic Engineering has these advantages:

- LSPs are not constrained because of destination based forwarding paradigm can be created through manually or through automatically by the underlying protocols

- LSPs can be efficiently maintained
- MPLS allows traffic aggregation and disaggregation unlike classical destination which only allows IP forwarding and permits only aggregation
- Easy integration of “constraint-based routing” framework with MPLS

Figure 2-9 shows the MPLS Traffic Engineering high level architecture.

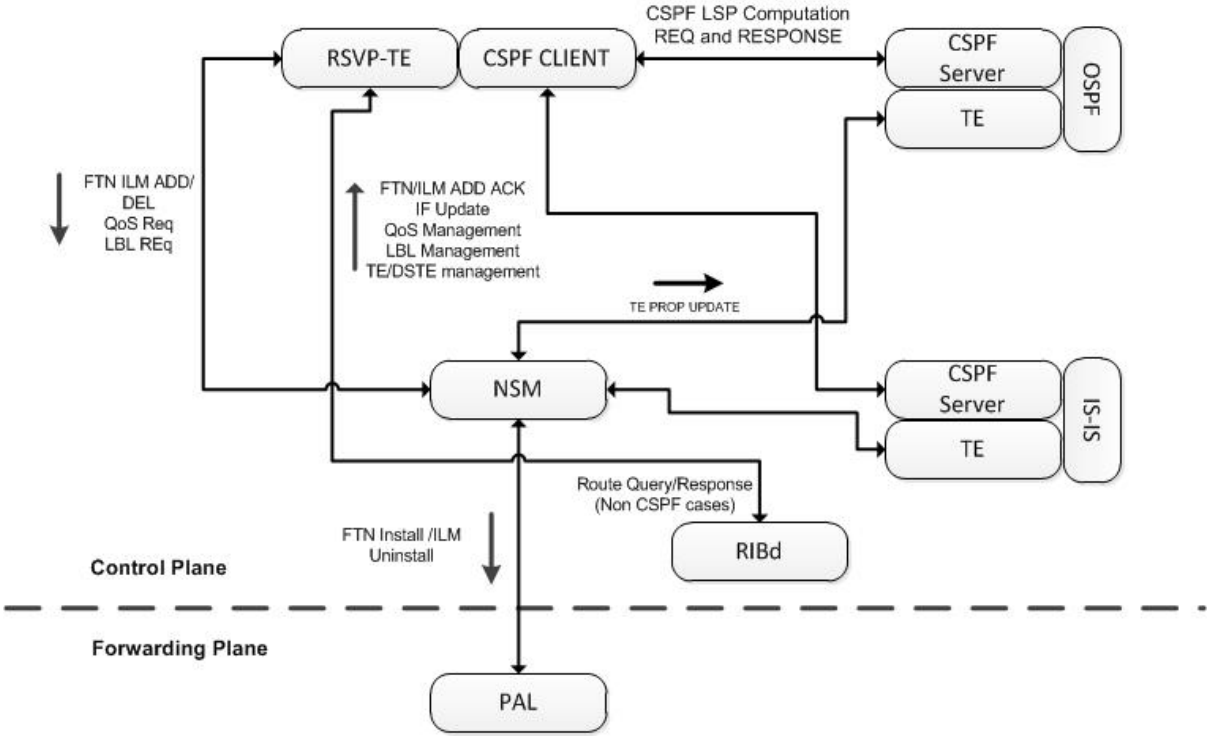

Figure 2-9: MPLS Traffic Engineering

Resource ReSerVation Protocol-Traffic Engineering

Resource ReSerVation Protocol-Traffic Engineering (RSVP-TE) communicates with the QoS module in NSM to:

- Find if the requested bandwidth is available
- Reserve bandwidth for an LSP
- Release a reserved resource

To manage labels, RSVP-TE communicates with the label pool manger in NSM. NSM provides a label block and RSVP-TE assigns a label for each LSP within the label range provided by NSM. If all labels in a label block are exhausted, RSVP queries and gets the next available label block from NSM.

- RSVP-TE gets interface up/down events from NSM
- RSVP-TE queries RIB for next hop information
- RSVP-TE sends FTN/ILM (obtained as result of LSP signaling) to NSM
- NSM maintains MPLS RIB information (FTN/ILM) and installs it in hardware

Differentiated Services

Differentiated Services (DiffServ) specifies a simple, scalable and coarse-grained mechanism to classify and manage network traffic and provides Quality of Service (QoS) guarantees for service providers. To provide a flexible DiffServe-over-MPLS solution, DiffServ extends the RSVP-TE module. The DiffServ module enables network traffic to be specified and prioritized by class, so that certain kinds of traffic, for example voice traffic, attain precedence over other types of traffic.

Because DiffServ by itself lacks the ability to efficiently use network transmission resources. The DiffServ-TE module performs traffic engineering per DiffServ class, rather than at an aggregate level. By combining DiffServ with MPLS-TE, routing devices can simultaneously classify and prioritize traffic and achieve fined-grained optimization of transmission resources.

The ZebOS-XP implementation of DiffServ supports RFC 3270 and RFC 4124.

Layer 2 Virtual Private Network

ZebOS-XP offers MPLS Layer 2 Virtual Private Network (VPN) and Virtual Private LAN Service (VPLS) protocol modules that enhance MPLS by providing transparent LAN access between VPN sites. The VPN infrastructure for provider-edge equipment uses the IP routing, traffic engineering, and MPLS switching features of ZebOS-XP.

Figure 2-10 shows the MPLS Layer 2 VPN high-level architecture.

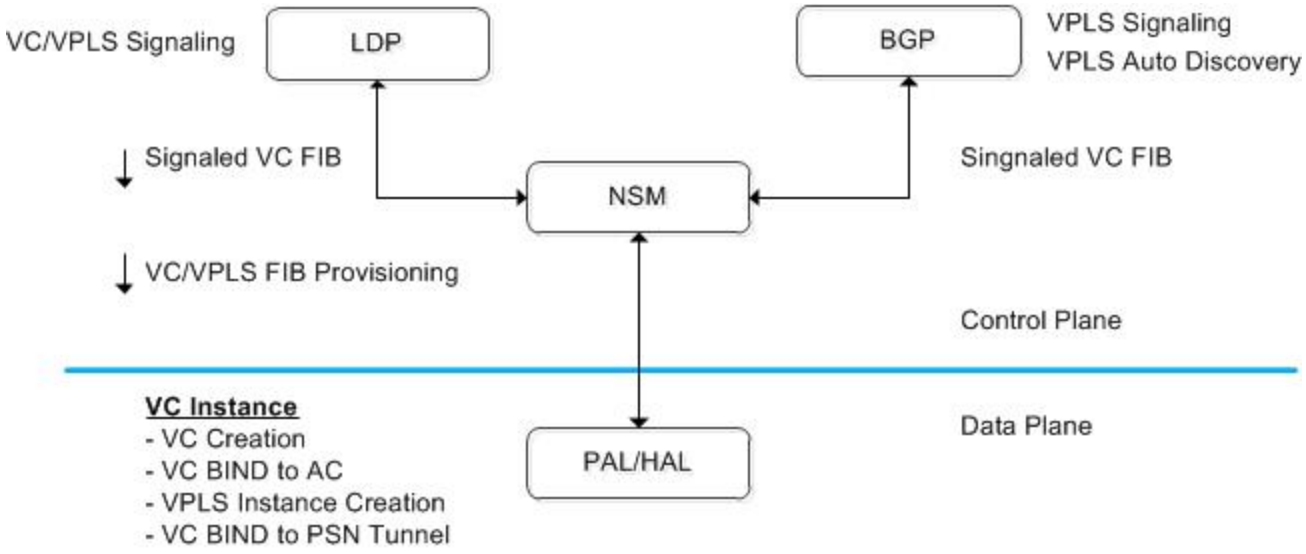

Figure 2-10: Layer 2 VPN

MPLS Layer 2 VPN provides end-to-end connection over a MPLS tunnel. Virtual Circuits (VC) addresses the problem of point-to-point VPN connections in MPLS VPNs. VPLS presents a solution for multipoint connectivity for the Layer 2 MPLS VPN.

In ZebOS-XP, MPLS-Layer 2VPN is achieved using signaling protocols such as LDP, BGP, or by static configuration. During VC/VPLS configuration, IMI relays all the VC/VPLS configuration information to NSM. NSM creates a VC/VPLS instance and bind this instance to the attachment circuit and PSN tunnel. Additionally, NSM sends this information to LDP and BGP which starts the signals:

- LDP provides signaling of VC/VPLS information and establishes targeted session and VC label exchange. The exchanged VC labels (FIB) are then sent to NSM.
- BGP uses auto-discovery to learn peers participating in a VPLS instances. Once the peers are discovered, VC labels are exchanged. The exchanged VC labels (FIB) are then sent to NSM.

NSM maintains the VC and PSN label information and installs the FIB information in the hardware.

Note: The ZebOS-XP Layer 2VPN solution has signaling extensions for both VPWS (Virtual Private Wire Service) and VPLS.

Layer 3 Virtual Private Network

The MPLS Layer 3 Virtual Private Network (Layer 3 VPN) is a PE-based technology for service provider VPN solutions. It uses BGP to advertise VPN routes and uses MPLS to forward VPN packets on service provider backbones. A CE first establishes adjacency with a directly connected PE and advertises its VPN routes to the PE and learns remote VPN routes from that PE. A CE and a PE use BGP/IGP to exchange routing information

The ZebOS-XP Layer 3 MPLS-VPN solution is supported by the ZebOS-XP MPLS forwarder and LDP modules by tightly integrating BGP-VPN extensions. It provides address space and routing separation through the use of per-VPN routing tables (VPN Routing and Forwarding) and MPLS switching in the MPLS core.

Figure 2-11 shows the MPLS Layer 3 VPN high level architecture.

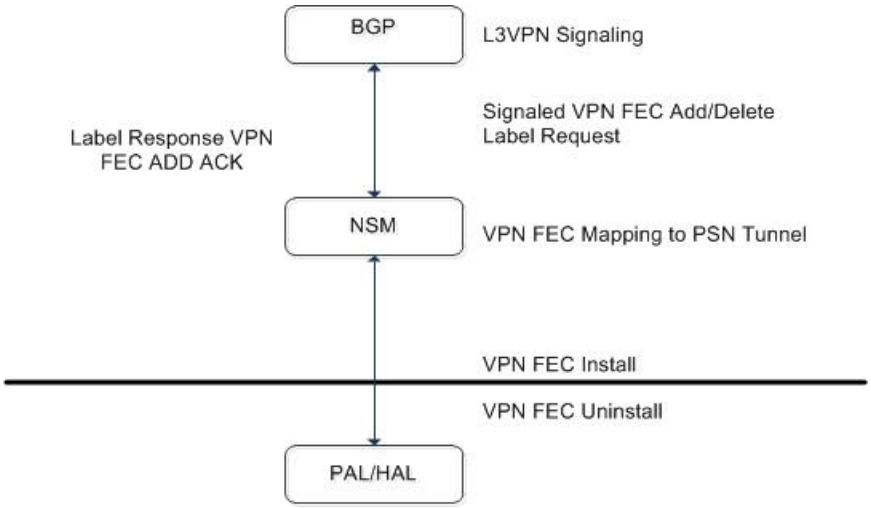

Figure 2-11: Layer 3 VPN

ZebOS-XP supports BGP-MPLS VPNs for both IPv4 and IPv6. Customer data packets are tunneled through the backbone, so that core routers are unaware of IPv4 and IPv6 VPN routes. NSM gives the routes it has learned from IGPs to BGP. BGP allocates the labels for these prefixes and then informs its peer about these labels. BGP then interacts with NSM to install the labels in the forwarder and to map the VPN prefix to the underlying MPLS tunnel. Data traffic is encapsulated with BGP labels and sent on the MPLS tunnel.

MPLS-TP Operation, Administration and Management

Multi-Protocol Label Switching Transport Profile Operation, Administration and Management (MPLS-TP OAM) provides remote monitoring, detection, and resolution of path errors on an MPLS network.

Figure 2-12 shows the high-level architecture of MPLS-TP OAM.

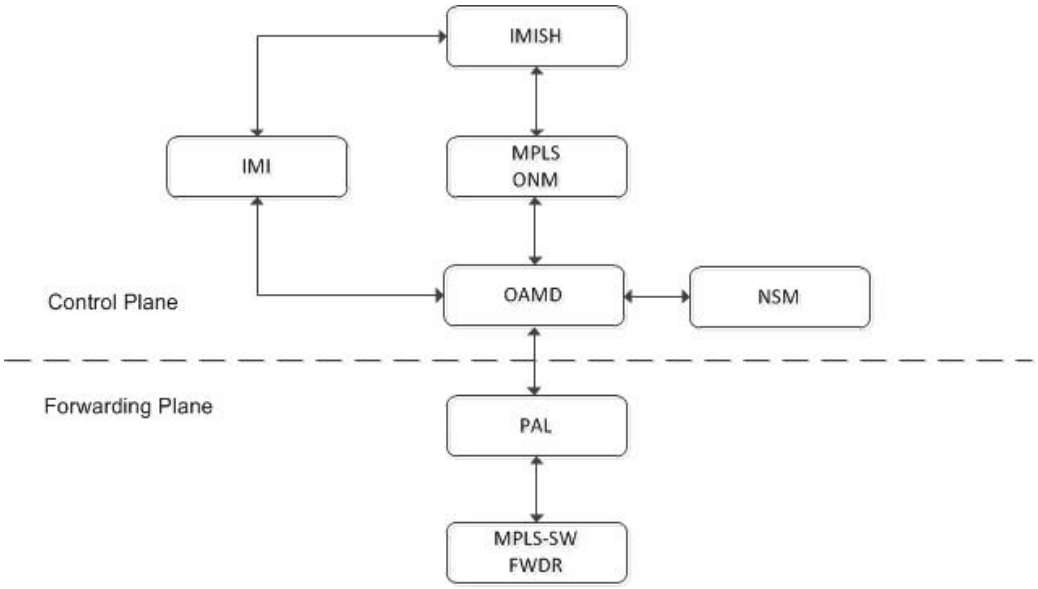

Figure 2-12: MPLS TP OAM

The ZebOS-XP implementation of MPLS-TP OAM supports the following:

- Detecting MPLS data plane failures
- Pseudowire Virtual Circuit Connectivity Verification (VCCV)
- Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)
- BFD For MPLS

MPLS-TP is a subset of MPLS with extensions. MPLS-TP has the advantages of the packet-based transport approach, is reliable, available, is OAM capable and manageable with features associated with traditional transport networks and it meets all the transport network requirements.

The extensions make MPLS more “transport like” by inheriting OAM, reliability, and operational simplicity from SONET/SDH networks.

MPLS-TP networks reduce the complexity of network operations linked with network performance monitoring and management, fault management, and protection switching.

MPLS-TP OAM provides tools to monitor and manage the network with the same attributes offered by legacy transport technologies. For example, the OAM travels on the exact same path that the data would take. In other words, MPLS-TP OAM monitors PWs or LSPs.

There are two approaches for MPLS-TP OAM:

- IETF solution based on the MPLS OAM
- ITUT solution based on Ethernet MAC layer OAM

IETF OAM

The IETF (Internet Engineering Task Force) MPLS-TP OAM enables an end-to-end fault, performance and protection-switching management along MPLS-TP Segments, LSPs (network infrastructure), and PseudoWires (PWs).

This module supports the following IETF OAM features:

- BFD (Bidirectional Forwarding Detection): Continuity check and connectivity verification protocol
- LPS: Linear protection switching using PSC (Protection State Co-ordination) protocol
- LI-LB: Lock instruct and loop-back protocol

- Ping and Trace over MPLS/MPLS-TP transport path
- Loss measurement (only control plane support)
- Delay measurement (only control plane support)

ITUT OAM

The ITUT (International Telecommunication Union Telecommunication) MPLS-TP OAM enables fault management and performance monitoring.

This module supports the following ITUT OAM features

- CC: Continuity check protocol
- LB: Loop back protocol synonymous to a basic ping protocol over MPLS/MPLS-TP transport path
- LPS: Linear protection switching using APS (Automatic Protection Switching) protocol
- Lock: Protocol to lock a transport path (only control plane support)
- Loss measurement (only control plane support)
- Delay measurement (only control plane support)
- AIS: Alarm indication signal to indicate loss of connectivity at the server layer
- Test: To perform diagnostic tests by verifying bit errors

Virtualization

ZebOS-XP supports different virtualization concepts like Virtual Router (VR) and Multi-Tenancy (MT).

Virtual Router (VR)

The Virtual Router (VR) feature logically subdivides a physical router into multiple virtual routers, with each virtual router independently monitored and managed by the network administrator. Each VR can have multiple VRFs (Virtual Routing and Forwarding instances) supporting multiple routing tables (RIBs) co-existing within the same router at same time. A single instance of ZebOS-XP manages all the VRs and its VRFs. ZebOS-XP offers VR support for both IPv4 and IPv6 routing protocols.

VR/VRF with Namespaces

Multiple Forward Information Base (FIB) tables in x86 platforms (using Linux kernel versions 3.0 and later) are supported using Linux namespaces. Each VRF is essentially mapped to one network namespace which has a Forwarding Information Base. ZebOS-XP supports the segregation of VRF routes at the control plane level which is extended to the FIB in the Linux IP stack using the namespace concept.

Multi-Tenancy

Multi-tenancy (MT) is a virtualization concept to create independent slices of a device, each of which can be individually managed and configured. Multi-tenancy is based on file system partitioning using `chroot` jails and Linux namespace support. This namespace allows the user of the Linux kernel to create multiple instances of a given resources. For example, the network namespace can be used to create multiple IP stacks.

In each tenant or logical partition, a complete instance of ZebOS-XP control plane (Layer 3 and Layer 2) can be executed.

The high level modules in the system are:

- Logical Switch Router (LSR): A LSR is one partition of a physical switch and provides virtualization for management plane, control plane and data plane. Each LSR has an instance of the control plane processes (OSPF, BGP, RIP, STP, RIB, MRIB, NSM), management plane processes (SNMPD, IMI) and system services processes (SYSLOGD) running in the LSR under a `chroot` jail.
- Default LSR/ Privileged LSR: The default/ privileged LSR (LSR 0) is always active and enabled and can never be deleted. This LSR comprises the LSM which manages the multi-LSR eco-system.
- Logical System Manager (LSM): This module is a separate Linux process that is responsible for LSR management:
  - Creating and deleting LSRs
  - Setting up the environment for the LSR such as the root file system
  - Loading/unloading individual processes in a LSR

  LSM is only present in the default/privileged LSR. When the LSR is created, all processes needed for the the LSR are loaded (imi, nsm, rib, mrib, syslogd, vlogd, snmpd).
- `syslogd`: Linux logger daemon. One instance of `syslogd` runs per LSR.

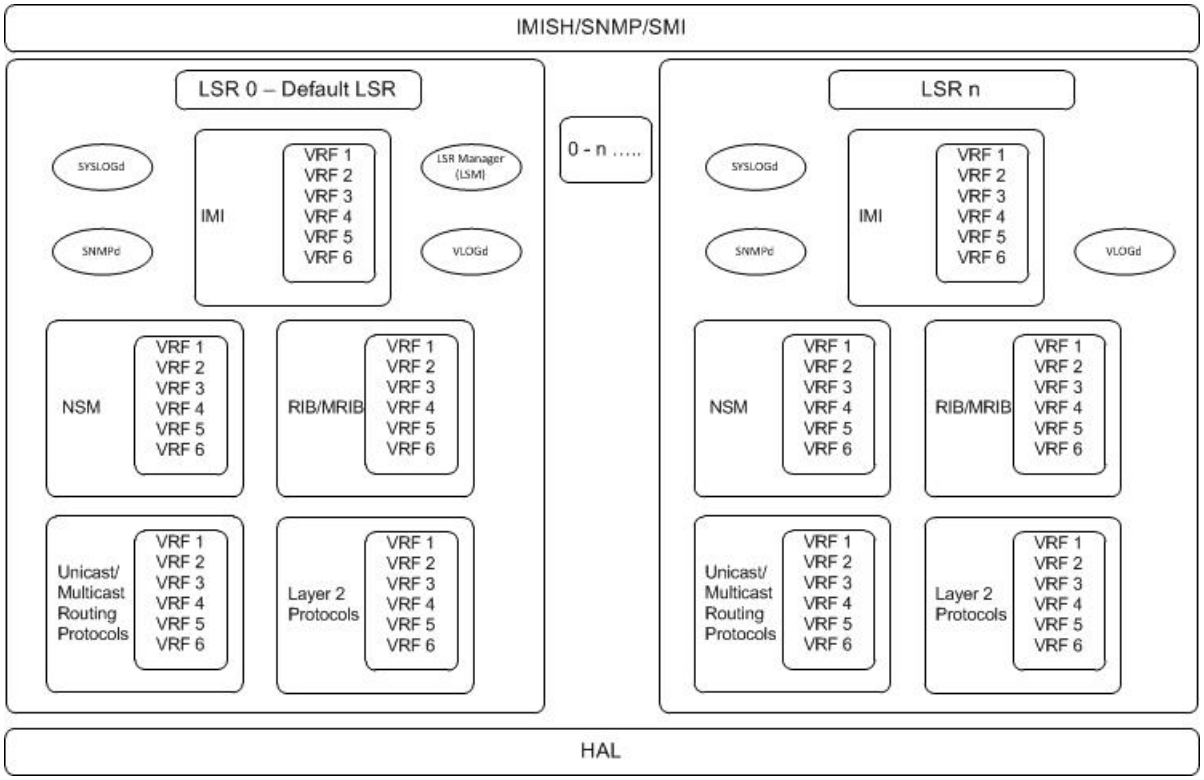

Figure 2-13: Multi-Tenancy

Software Defined Networks (SDN)

ZebIC supports Open-Flow(OFL) feature integration.

Open-Flow (OFL)

SDN is a Software Defined Network where Control plane and Data plane are seprated out. Control plane runs on Controller and Data plane is with any commodity switches. Openflow is a SDN protocol, which is to make communitcation between SDN Controller and Openflow-enabled switches.

Openflow-enabled switches pass Openflow bridge addition or deletion, port addition or deletion, Interface up or down information to the controller through Openflow message so that the controller will have network topology. Based on the application request, controller passes flow addition or deletion, group addition or deletion messages to switches through Openflow messages.

In case, flow entry is expired on the switch or Admin deletes flow entry or group entry then the switch should inform the Controller to update its topology map. Statistics of flow or port or group would also be carried by openflow messages.

With current approach, flow or group entries could be created for processing Layer2 and Layer3 traffic based on match criterias, instruction and actions provided by ONF specification.

All flow or group would be created or deleted on Broadcom provided TCAM.

Note: The Open-Flow testing has been done using ODL (Open-Day-Light) HYDROGEN controller which can be downloaded using the below link.

https://www.opendaylight.org/software/release-archives/https://www.opendaylight.org/software/downloads/hydrogen-base-10

ZebOS-XP Thread Mechanism

ZebOS-XP uses a thread-queuing mechanism to manage a variety of asynchronous events. The threading mechanism can be characterized as non-preemptive, execute-to-completion. For the host operating system, it is a single thread. Therefore, there are no contention problems because all code execution is serialized.

Each protocol module has a master thread structure that holds several thread linked-lists. Each linked list is dedicated to an executable type: network I/O read, network I/O write, event handling (high and low), timer, and so on. The thread linked-list is built of threading elements that contain pointers to the actual executable task. Each thread element has an executable function associated with it. For every task that needs to be executed, a thread element is added to the linked-list for its type.

CHAPTER 3 ZebOS-XP Abstraction Layers

This chapter describes the abstraction layers in ZebOS-XP that enable hardware and software portability:

- The Platform Abstraction Layer (PAL) enables ZebOS-XP to be ported to different operating systems and CPUs
- The Hardware Abstraction Layer (HAL) decouples the ZebOS-XP protocol modules from the underlying data plane, enabling portability across multiple silicon switching chipsets

These abstraction layers isolate hardware and operating system specific interactions to a small set of well-defined function calls for the control plane. The abstraction layers provide a unified interface for the control plane to interact with the data plane for all Layer 2, Layer 3, multicast, and MPLS forwarding needs. The functions calls above the abstraction layer run unmodified for any switching and routing hardware platform.

Additionally, this chapter describes the Hardware Services Layer (HSL) which implements the hardware system-related functions.

Platform Abstraction Layer

ZebOS-XP provides platform independence with the Platform Abstraction Layer (PAL) functions. The PAL functions call the system services specific to the platform to isolate ZebOS-XP daemons from the operating system.

PAL allows ZebOS-XP to support multiple operating systems. The complexity of each component is handled by a PAL module for each OS. This allows a ZebOS-XP component to make an OS-independent call and then the compile-time options link the appropriate PAL module that fulfills that call.

ZebOS-XP comes with a set of production-quality, fully-tested PAL modules for the most common operating systems. Any porting efforts to non-supported platforms can be modelled on these PAL modules.

Features

The features of PAL are as follows.

Memory Management

A memory management in PAL gives developers more control over memory, including setting the class and size of the requested storage.

Sockets

Sockets services are used to create TCP sockets. PAL sockets handle all socket functionality. For example, creating and deleting, reading and writing; and synchronizing. In addition, there are extensions to the socket calls that handle layer 2 and raw sockets.

String

The string abstraction includes string operations such as copying, scanning, parsing, and comparing. It interacts with the memory abstraction to create or move memory cells.

Standard Library

The standard library (`stdlib`) abstraction takes advantage of the most effective functions on the system. If there is an enhanced or particular functionality available for a feature, that functionality can be used in preference to the standard function.

Configuration Storage

PAL abstracts storage of configuration files. This allows platforms without a file system to store configuration information in any available memory model, including linear memory or flash memory.

Logging

Logging facilities are abstracted so that the system can log to any available facility, for example syslog, file, or console. These outputs can also run in parallel so that each output can be set independently to different log trap levels.

Interfaces and Routes

PAL abstracts kernel routes and interfaces. This allows ZebOS-XP to not directly perceive how routes are manipulated.

Daemons

Process daemons are abstracted so that they are automatically handled on systems with this functionality.

Source

The source tree layout puts each protocol in its own directory. There is no platform-specific source in these directories. All calls, data types, and other components that are operating-system-specific or platform-specific are in subdirectories of the `pal` directory. All included modules either document other required included modules or automatically include the necessary modules.

Dummy Build Environment

A “dummy” build environment helps determine if the module include any system-dependent functions. The dummy environment excludes the system header files, and each module that builds with the dummy environment does not include any system-dependent functions. This ensures that all modules run in any PAL supported environment, even if that environment does not support the standard library feature.

Module Compatibility

Changes, additional features, and enhancements to protocols are restricted to the source for that module. Changes made necessary by a platform are restricted to the PAL module for that platform. A PAL API can change to support an operating system feature. This ensures backward compatibility for protocol modules with later versions of an API.

Hardware Abstraction Layer

As shown in Figure 3-1, the Hardware Abstraction Layer (HAL) enables the control plane to run on different hardware platforms, regardless of the type of chip set or operating system being used. Additionally, HAL makes the control plane fully independent of the specific TCP/IP stack implementation used with the operating system.

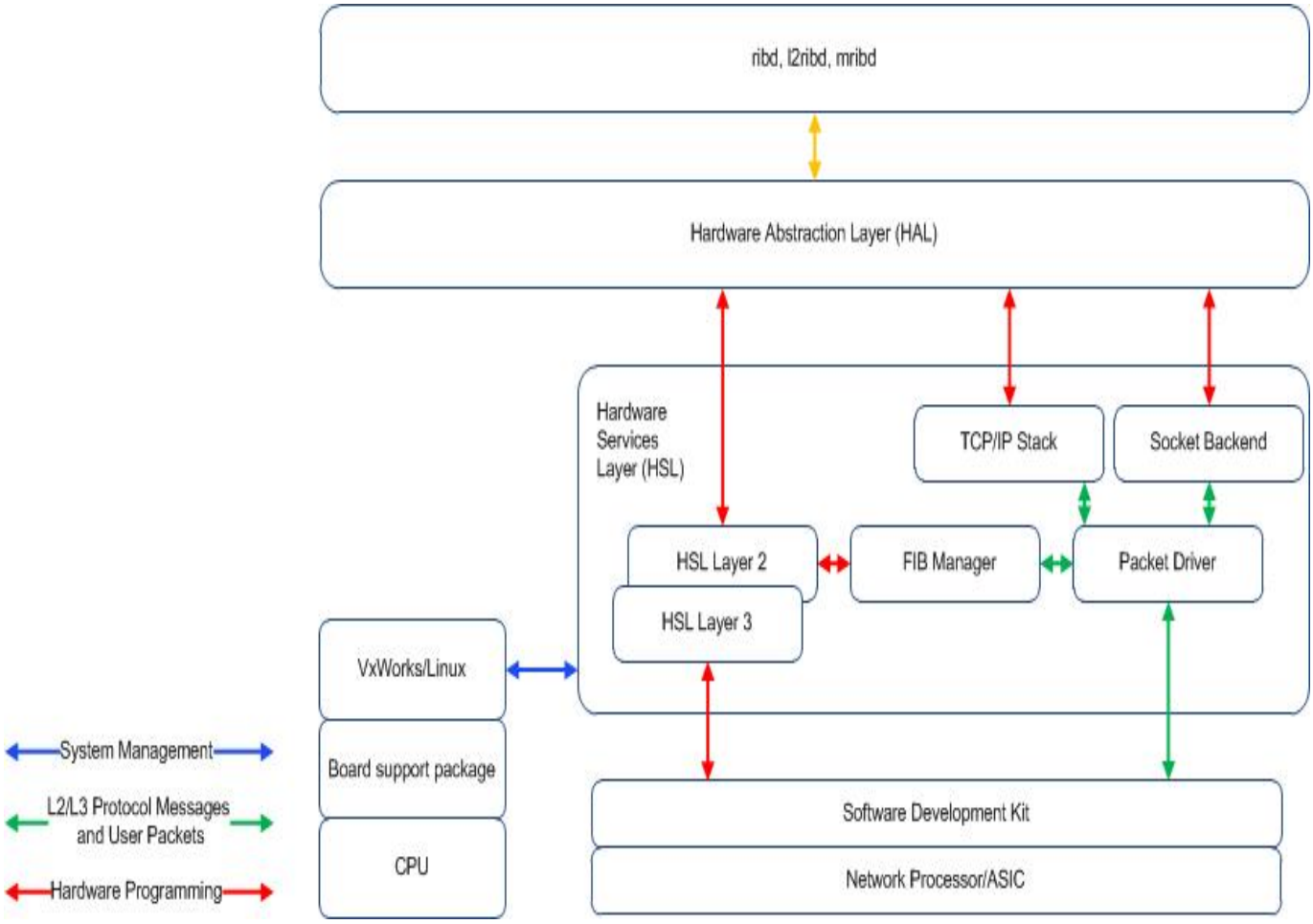

Figure 3-1: HAL and HSL

HAL has two major functions:

- Encapsulating and sending control messages: These messages originate from `ribd`, `l2ribd`, and `mribd`.
- Processing System Responses: HAL processes system responses and events, and calls the appropriate module in the control plane based on the type of event. The processing of system layer responses includes parsing or decoding of the message, and then notifying the appropriate module in the control plane, if required.

To accomplish these tasks, HAL has the following components:

- Socket communication layer
- Hardware Services Layer (HSL)
- Packet driver
- FIB manager
- Ethernet driver

Each component is described in the following sections.

Socket Communication Layer

HAL communicates with the Hardware Services Layer (HSL) through the socket communication layer. In other words, the control plane communicates with the forwarding plane through the socket communication layer. The communication tasks are:

- Deliver hardware and operating system configuration messages
- Handle event messages generated by the forwarding plane
- Poll forwarding plane information (such as interface information)

The communication is based on a client/server model:

- The control plane implements the client side
- The forwarding plane implements the server side

During hardware initialization, the forwarding plane creates a server socket to which control plane clients connect once they start. All communication between the control plane and the forwarding plane is message-based.

ZebOS-XP provides one proprietary `AF_HSL` family socket mechanism which is registered with the operating system.

Each message contains the following:

- Message type
- Message length
- Message data in the TLV encoded (length/field) format

The sender is responsible for preparing the message header and encoding the data. The receiver is responsible for decoding messages, acknowledging receipt of the message to the sender, and performing an integrity check.

Bulk Messaging and Asynchronous Messaging

With bulk messaging, several requests are sent in a single group or a few groups, relieving the burden on the IPC channel and optimizing performance. Bulk messaging is deployed across applications such as NSM, `mstpd`, `ribd`, and `mribd` that have intensive message transactions.

There is a mechanism to bulk information when programming the hardware in the HAL layer. This mechanism can be used by an application layer to bulk information when programming the hardware. However, this feature is not available on x86 platforms, but only for specific silicon vendor hardware.

With asynchronous messaging, the application layer can program the hardware to respond quickly and process the success or failure of a request later. Asynchronous messaging is implemented using message sequencing, a success/failure callback mechanism, and timers. However, this feature is not available on x86 platforms, but only for specific silicon vendor hardware.

Hardware Services Layer

The Hardware Services Layer (HSL) implements the hardware system-related functions.

Note: Depending of the platform, portions of HSL run in user space or kernel space.

HSL has four main interfaces to the system which are described below:

- Socket Interface to control plane
- SDK interface to hardware
- Operating system Interface
- Event interface

Socket Interface to Control Plane

The socket interface receives control plane messages (such as configuration messages) and sends the response and notification.

SDK Interface to Hardware

The SDK (software development kit) interface configures the hardware chip set (fast path) according to the control plane messages. It is also retrieves counters and statistics.

Operating System Interface

The operating system interface configures the slow path according to control plane messages. The operating system interface also maintains a full forwarding information base (FIB), which is not restricted by hardware limitations.

Event Interface

The event interface registers callbacks with the hardware and the operating system to receive and process system events. Examples of callbacks are:

- Hardware module attachment/detachment
- Link/auto-negotiation status update,
- New address addition/notification from the operating system

HSL processes these events and notifies the control modules and system modules of the change.

For example, to detach an interface, a notification to the control plane is sent to stop all the protocols running on the detached interface. This also notifies the interface manager to delete the detached interface and any associated configuration.

A socket interface receives configuration updates and then sends responses and notifications. Message encapsulation and sending are similar to HAL; however, HSL does not require a response or acknowledgement message. Moreover, HSL acknowledges or sends a response to all messages.

HSL Operating System Functions

HSL uses abstracted operating function calls to provide services for the control plane.These functions are broadly abstracted for the following system services:

- Logging
- Memory allocation
- Synchronization mechanisms (semaphores)
- Thread/process management
- Inter-process communication (IPC)

FIB Manager

Networking devices store forwarding information in two databases:

- RIB (Routing Information Database) which represents a collection of all forwarding possibilities known to a device, learned from different sources, including identical and overlapping entries
- FIB (Forwarding Information Database) which represents an active selection of forwarding entries, used by a device to forward data

Traditionally, there are two paths for traffic forwarding:

- Fast Path where traffic moves via pre-programmed entries on a network processor or ASIC chip

- Slow Path forwarding: the network processor or ASIC chip is unable to forward the traffic and data is sent to the CPU/OS before being processed

The FIB manager database contains all forwarding entries, learned dynamically and statically by the system:

- The FIB manager stores and programs forwarding entries selected for forwarding to the network processor or ASIC and operating system.
- The FIB manager removes forwarding entries from the network processor or ASIC and operating system if an entry was withdrawn from the FIB.

As shown in Figure 3-2, the FIB manager database maintains two radix trees:

- A prefix tree that contains all active routes
- A nexthop tree that maintains a tree of attached hosts

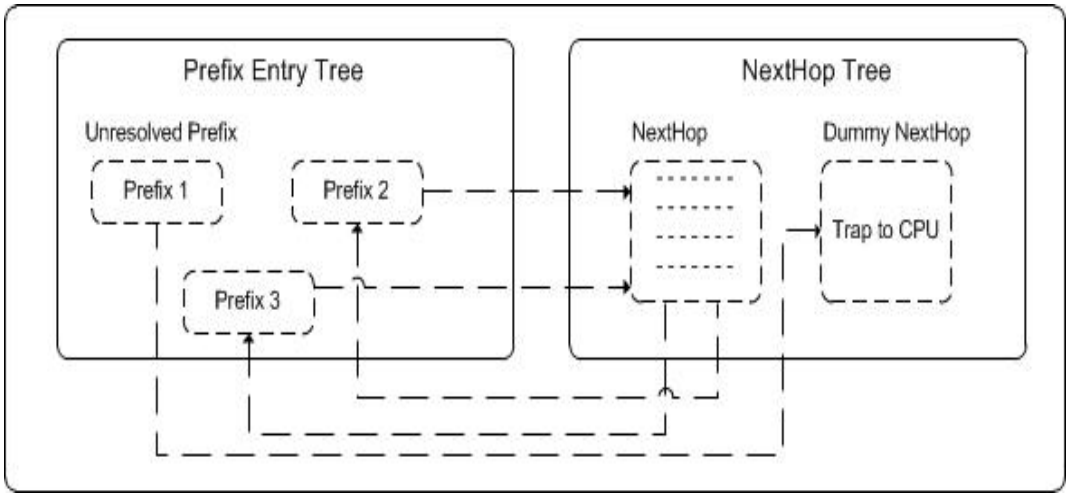

Figure 3-2: FIB manager database

Each prefix with a resolved next-hop points to the next-hop entry in the nexthop tree. Each next-hop points to all prefixes pointing to it. Prefixes with an unresolved next-hop point to a dummy next-hop entry, which indicates to send that traffic to the CPU. Once the CPU receives the traffic, the operating system stack attempts to resolve the next-hop and update the hardware.

The FIB manager performs these major tasks:

- Operating system programing: All prefixes and resolved next-hops are stored in the operating system.
- Hardware programming: Every prefix is installed in the hardware, based on the resolved or dummy nexthop.
- Keeping a shadow of all programmed and un-programmed prefixes.

Every time a new hop is resolved, or an existing nexthop is updated, the nexthop tree is updated, and all prefixes pointing to it are updated with relevant information.

For un-programmed prefix storage:

- The FIB manager stores prefixes and nexthops that cannot be programmed to the hardware due to hardware FIB table size limitations. If the hardware FIB is full, only entries used by traffic are stored; and others are removed.
- The FIB manager monitors prefix usage in hardware to identify and remove unused entries. The FIB Manager uses the ARP aging mechanism to remove unused nexthop entries.
- The FIB Manager handles nested routes to ensure proper forwarding.
- Each time an exception packet is received, the FIB Manager attempts to program a relevant entry to the hardware, making sure that the next time, forwarding occurs through the fast path.

Typically, both the operating system and chip are identically programmed, and only control traffic goes to the CPU. In addition, there is a memory limitation on processors for the number of prefixes a chip can accommodate. Modules keep a shadow of installed forwarding entries in both the chip and operating system. The FIB manager database has additional fields specific to the hardware, since the processor and ASIC programming require additional information.

Packet Driver

The packet driver module processes packets. Packet receipt occurs in the hardware-interrupt context, when the packet descriptor is passed to the user-callback function. To avoid a packet drop in case of bulk, the packet driver does not immediately start parsing and processing packets. Instead, the packet driver implements a first-in, first out (FIFO) queue for packets to be processed. During hardware interrupt, the packet descriptor is stored in the queue, and the packet-ready semaphore is released. If the queue is full, or the system is not ready for traffic, packets are dropped. Packet processing is based on the hardware CPU error code.

The packet driver is responsible for receiving packets. Packet receipt usually occurs in the hardware-interrupt context. As shown in Figure 3-3, the packet driver implements a first-in, first out (FIFO) queue for packets to be processed.

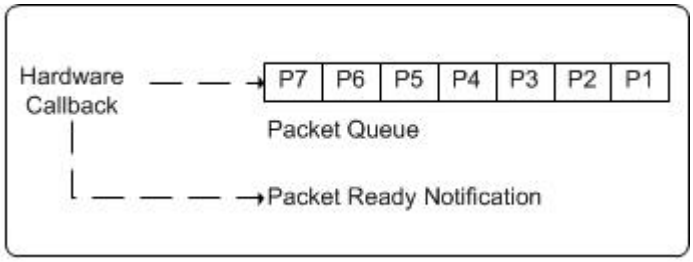

Figure 3-3: Packet driver

When the packet queue is not empty, the packet driver does the following:

1. Gets the packet from the head.
2. Resolves the receiving interface based on the port and the VLAN in the packet.
3. Verifies the packet validity for the interface based on the CPU error code or the packet Ether type.
4. If the packet is a Layer 2 control type (BPDU), it duplicates the packet descriptor, and sends it to a protocol socket.
5. Verifies whether learning is required. If required, it attempts to learn the new neighbor hosts.
6. Verifies if a valid forwarding entry exists. If it exists, the packet driver attempts to program the fast-path FIB.
7. Prepares the packet (CRC/VLAN tag).
8. Uploads the packet to the TCP/IP stack.

Note the following:

- Layer 2 control messages are passed to the protocol modules directly through the sockets.
- Layer 3 data traffic and control messages are passed to the operating system TCP/IP stack.
- Before passing the traffic to the socket or the operating system, the packet driver translates the hardware receive port to the operating system interface, based on the received port and VLAN.
- If the packet is not specific to the interface, the packet driver drops the packet.
- The packet driver removes the VLAN tag if the TCP/IP stack does not support the VLAN interfaces.
- The packet driver removes the packet CRC, if needed.

- The packet driver learns about the neighbor hosts and local FIB updates from the ARP and neighbor solicitation messages.
- The packet driver tries to program the fast path for packets not handled by the hardware.

Ethernet Driver

The Ethernet driver has these functions:

- Handling interfaces in the operating system
- Passing traffic between the hardware and the operating system stack
- Managing network buffer pools

The Ethernet driver registers interfaces in the operating system based on the interface type. Each interface type is represented as a separate device driver. Each device registers a set of driver functions for device handling (start-stop and send-receive). Interfaces of the same type are treated as multiple instances of the device. The Ethernet driver manages the network buffer pool to pass traffic between the hardware SDK and the operating system stack.

The Ethernet driver resolves the destination hardware logical port and modifies packets, if necessary. Possible modifications include VLAN tag insertion and CRC recalculation. Based on the operating system transmitting interface, the Ethernet driver resolves destination hardware ports, updates the packets based on (destination port and VLAN) pair and transmits the packets on the interface.

If the destination interface is the VLAN termination interface including multiple ports, the following logic is applied:

The Ethernet Driver performs a lookup in the Layer 3 FIB:

1. If the destination interface has multiple ports:
    - If the destination port is found in the packet, it is transmitted to that port
    - If the destination port is not found in the packet, the packet is transmitted to all participating ports
2. If the destination interface is an aggregator, the Ethernet driver finds the first participating port where it transmits the packet.

CHAPTER 4 ZebIC

This chapter describes the ZebIC module which integrates the ZebOS-XP control plane with the reference designs of networking chipsets.

ZebIC provides a scalable, hardware-independent architecture that enables you to efficiently incorporate networking capabilities into your networking product.

The transition to SDN (Software Defined Networking) requires a control plane architecture that can isolate hardware development from software development and that allows rapid portability of new networking services across a wide-range of silicon. ZebIC enables accelerated transition to SDN by enabling developers to develop, integrate, and test the target platform while the actual hardware system is still under development.

ZebIC separates hardware development from software development through the ZebOS-XP abstraction layer, which isolates all of the hardware and operating system specific interactions into a small set of well-defined function calls for the control plane. This enables you to easily move from one switching chipset and/or operating system to another, reducing both time to market and development cost

The abstraction layer provides a unified interface for the control plane to interact with the forwarding plane for all Layer 2, Layer 3, multicast, and MPLS forwarding needs. The function calls above the abstraction layer run unmodified for any switching and routing hardware platform. The result is that you have the full flexibility to select the required protocol modules in the most cost and code space effective way.

As shown in Figure 4-1, ZebIC integrates the control plane with the reference designs of the popular chipsets in the networking industry. The software integration is built using a vendor-specific chipset SDK and reference operating system recommended by the vendor.

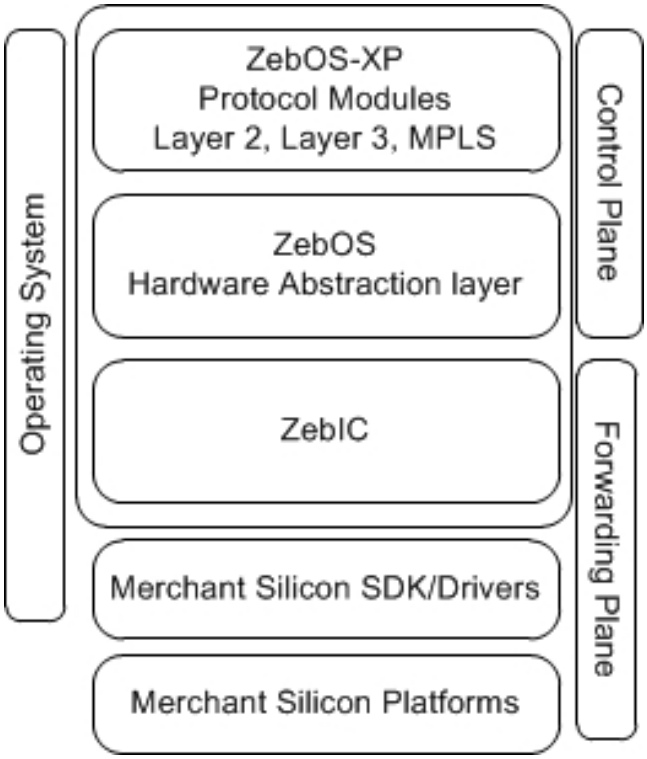

Figure 4-1: Chip set and operating system integration

ZebIC provides pre-integrated software on industry-leading silicon switching chipsets provided by:

- Broadcom
- Marvell
- Wintegra
- PMC WinPath3

ZebIC supports a wide range of network system designs configurations, including pizza box, chassis, or stacking devices.

CHAPTER 5 ZebHA

This chapter describes the ZebHA (High Availability) module which minimizes system downtime by operating redundant nodes when a node fails.

Overview

The ZebHA (High Availability) module supports protocol modules in Layer 2 and Layer 3. ZebHA provides:

- Control plane redundancy necessary to meet the needs of network operators for high network service availability
- Reliable handling of operational, application, system and component failures;
- Strict Service Level Agreements (SLA) requirements of network operator customers

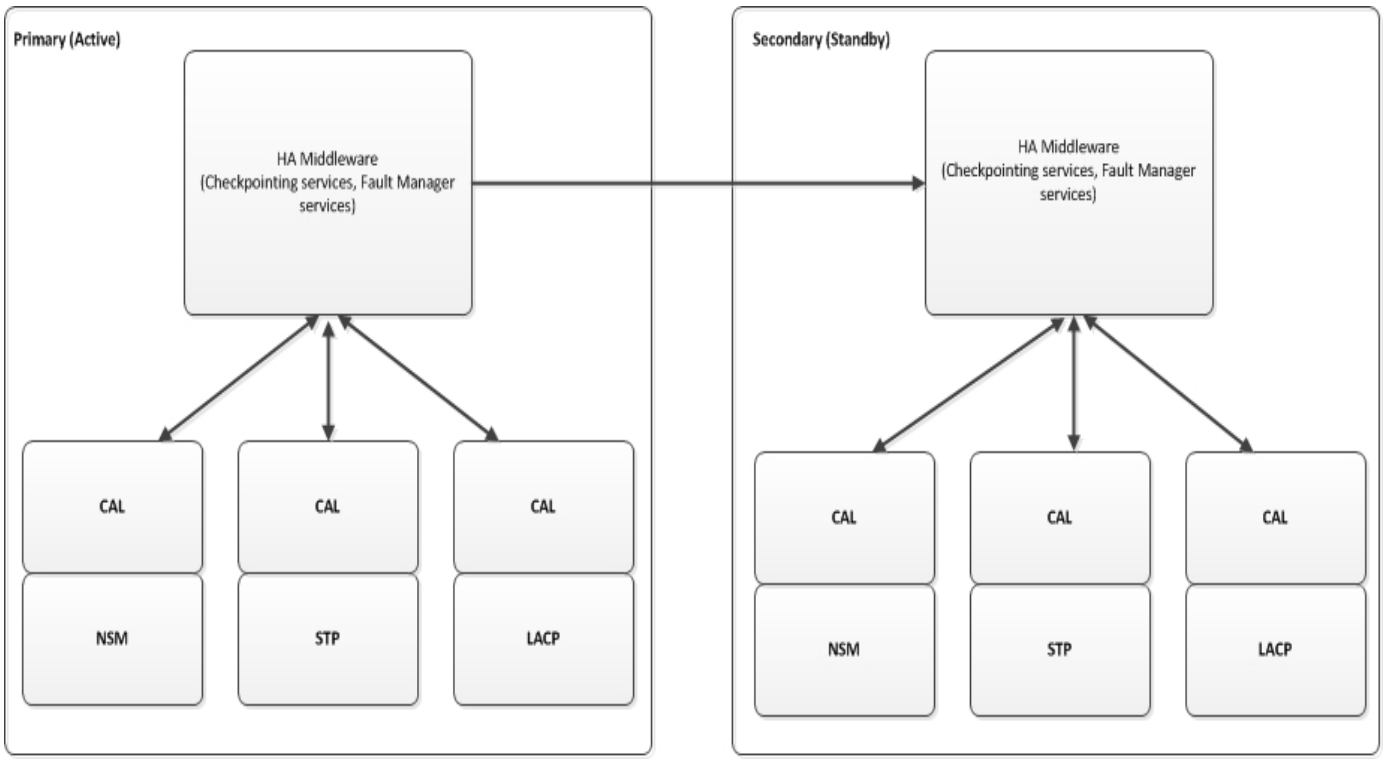

Figure 5-1: High Availability Overview

ZebHA ensures a pre-agreed level of operational performance by minimizing system downtime. A ZebHA system operates redundant nodes that can provide continued service when a node fails. High availability does not mean that components will never fail, but it ensures that the system is available when the user needs it even if components fail.

ZebHA allows Simplex-Active or Active-Standby (1+1) control plane CPU redundancy. There are two types of protocol recovery after a redundancy switchover:

- Stateful switchover (SSO) recovers those protocols for which a complete checkpointing of all protocol states is performed. Link Aggregation Control Protocol (LACP), Multiple Spanning Tree Protocol (MSTP), and the RIB daemon and Interface Manager, are completely checkpointed between the Active and the Standby controllers.

- Graceful restart recovery allows rebuilding the routing database and protocol states of Layer 3 routing protocols such as OSPF, IS-IS, and BGP without withdrawing its routes from the ZebOS-XP global Routing Information Base (RIB).

During the post-switchover recovery process, the Layer 2 and Layer 3 forwarding planes continue uninterrupted forwarding of data packets, while the control plane is being switched to the new Active controller.

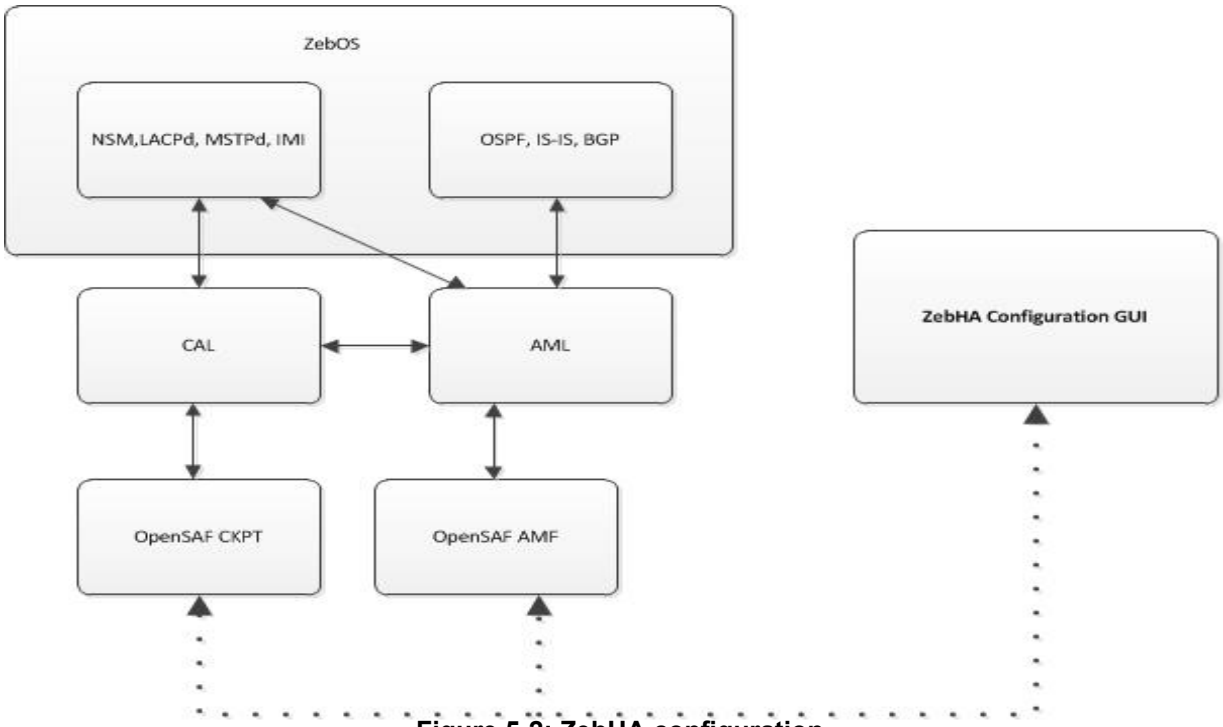

Figure 5-2: ZebHA configuration

For more information about ZebHA, see the *High Availability Developer Guide*

Components

Checkpoint Abstraction Layer

The Checkpoint Abstraction Layer (CAL) is an independent library which provides an abstraction of the checkpoint service. This can be integrated with a middleware like OpenSAF to achieve the application level checkpoint.

AML

The Availability Management Layer (AML) enables existing or newly implemented applications with minimal effort to become SA-Aware.

Middleware (OpenSAF)

ZebOS-XP supports the Open Service Availability Forum (OpenSAF) checkpoint service which provides full-process atomicity at the process level with the following functionality:

- Responsible for the allocation of records within a section of checkpoint data that maps to the CAL. Each checkpoint session record is represented by a state, which indicates whether a record buffer's space within a section is taken or empty.
- Explicit record create, modify or delete operations are not required by CAL. The only requirement is that the opcode accompany the record data and be stored in the checkpoint data, along with the checkpoint record data.

- When memory allocates for a section by the checkpoint service, it stays allocated until the section is overwritten.
- Checkpoint actions that belong to a transaction are accumulated by the OpenSAF and forwarded to the checkpoint library upon reception of the `cal_txn_commit` request.
- A write operation returns only after the checkpoint replica on the local node is updated, so blocking write actions to the local checkpoint replica. The update of remote replicas is carried out asynchronously.
- Live checkpoint updates are not accompanied by checkpoint data. The standby node must explicitly retrieve these updates from the local checkpoint replica.

Audit

Audit refers to the process of reconciliation between the control plane and data plane to synchronize their states after a restart. When NSM restarts (in simplex mode of operation), the entries programmed into the hardware are removed. After a restart, NSM recovers its state from the Checkpoint Database (CDB). Then the hardware is reinitialized and entries are reprogrammed into the hardware. This is against the principle of non-stop forwarding because there is a disruption of data forwarding during the interval between removal and reprogramming.

Also when NSM restarts, there is no audit mechanism between NSM and HSL. Therefore, NSM is not aware of the state of the hardware and ends up reprogramming all entries. Again the audit mechanism is needed to process any events (say a link down) that happened during NSM restart interval.

Features

Stateful Recovery

Stateful recovery refers to the process of active/standby controllers running in synchronized mode with respect to state information. Redundant node is in hot-standby mode and active/standby controllers synchronize configuration and dynamic protocol state information.

Graceful Restart

The control plane restarts can disrupt data forwarding in the following cases,

- Due to software upgrades: “planned restart”
- Due to software bugs: “unplanned restart”

Graceful restart is a way to rebuild forwarding information in routing protocols when the control plane has recovered from a failure

The steps for graceful restart are:

- Separate the control and data plane
- If the control plane fails, the data plane keeps forwarding data
- Neighbors of the restarting node hides failure from all other nodes
- When the node recovers, neighbors synchronize without disrupting service
- Graceful restart is protocol dependent which means it is handled by each protocol separately

Configuration Checkpointing

Configurations which are applied successfully on the active node, should be saved and check-pointed to the standby node. The saved configurations, in the CDB, should be used at the time of the restarting of the active node (in case of simplex mode) and the check pointed commands should keep the configurations of the standby synchronized with that of the active node (in case of 2N redundancy mode).

Planned Upgrade/Maintenance

For software upgrade the manual switch-over can be used to facilitate the process. If the active node is required to be upgraded it is made the standby manually. It is then upgraded and then switched back as active.

Heart-Beat Failure Handling

When ZebHA is deployed in the active-standby redundancy model, one of the controllers assumes an active role and another assumes a standby role. Internal data and states of the ZebOS-XP protocols running on the active node are continuously synchronized with those on the standby node. AMF keeps monitoring the redundancy state of the nodes using heart-beat mechanism between active and standby controllers.

It is possible that the physical link on which heart-beat messages are exchanged between active and standby controllers becomes faulty. In such a case, the heart-beat messages cannot be exchanged between controllers. The active controller assumes that a fault has occurred in the standby but it continues as active and the standby controller assumes that the active node has become faulty and therefore it changes its state to active. This “split-brain” situation is not desirable.

Normally, in chassis architecture, both the controllers are connected via the backplane which has redundant physical links. If this is supported by the hardware platform, ZebHA need not deploy any additional mechanism to handle the above situation. Even if one of links becomes faulty, the heart-beat messages are exchanged on the redundant link and there will never arise a situation wherein both controllers are active.

A heart beat failure handling solution is proposed for deployments where physical link redundancy is not available in hardware.

CHAPTER 6 Management Modules

This chapter describes these management modules for ZebOS-XP:

- ZebSBI: a C language API you use to write applications to configure and manage ZebOS-XP routing and switching
- ZebHPI: a system management module that supports common host protocols

ZebSBI

ZebSBI (South-Bound Interface) is a C language API that you use to write applications that configure and manage the ZebOS-XP routing and switching protocols. ZebSBI is also called Simple Management Interface (SMI).

Every ZebOS-XP protocol module provides a C language API for configuring properties of the protocol. For example, the `ospf_api.h` include file defines an API for configuring the OSPF protocol. However, you must call the functions in a ZebOS-XP C language API in the same process space as ZebOS-XP (that is, from a ZebOS-XP daemon). ZebSBI provides a mechanism to call the equivalent functions from a *different* process space.

ZebSBI provides get and set operations that duplicate operations you can perform in `imish`. In this respect, ZebSBI operations are similar to SNMP get and set operations. ZebSBI also supports asynchronous event notification that is similar to the SNMP trap mechanism.

ZebSBI uses a client-server architecture. The ZebSBI client is a dynamically linked object library. A ZebSBI server runs as part of each ZebOS-XP protocol module daemon. A ZebSBI server starts automatically whenever its daemon starts. A ZebSBI client uses socket-based inter-process communication (IPC) to exchange messages with the ZebSBI server.

Your application makes calls to functions in the ZebSBI client that internally creates a message and sends it to the ZebSBI server. The ZebSBI server calls functions in the ZebOS-XP API and returns the result to the client. Figure 6-1 shows the high-level architecture of the ZebSBI framework.

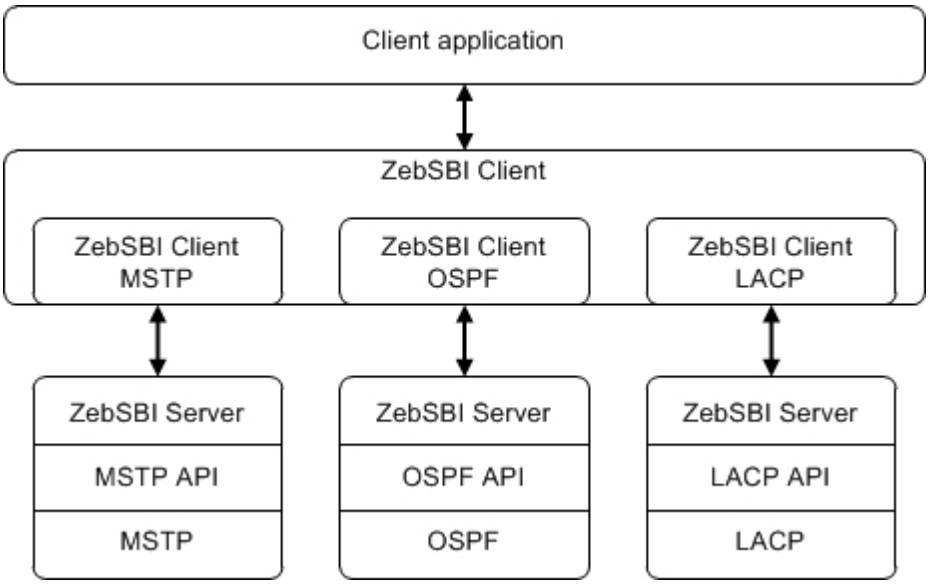

Figure 6-1: ZebSBI

Use the *Simple Management Interface Developer Guide* along with the corresponding SMI API reference for the protocols/features for which you are writing client applications. Each SMI reference manual has a file name with this format:

- ZebOS-XP-SMI-Reference-*xxx*.pdf

where *xxx* is the name of the protocol or feature. For example, the SMI reference manual for BGP is:

- ZebOS-XP-SMI-Reference-BGP.pdf

ZebHPI

The ZebHPI (Host Protocol Interface) is a system management module that supports these host protocols:

- Authentication, Authorization, and Accounting
- Dynamic Host Configuration Protocol Client
- Dynamic Host Configuration Protocol Relay
- Domain Name System
- Lightweight Directory Access Protocol
- Network Time Protocol
- Remote Authentication Dial In User Service
- Secure Shell
- Simple Network Management Protocol
- Syslog
- TACACS+
- Telnet
- User Management

Note: ZebHPI is a standalone package that is not integrated with ZebOS-XP.

Authentication, Authorization, and Accounting

The authentication, authorization, and accounting (AAA) commands provide these functions:

- *Authentication* identifies users by challenging them to provide a username and password. This information can be encrypted if required, depending on the underlying protocol.
- *Authorization* provides a method of authorizing commands and services on a per user profile basis.
- *Accounting* collects detailed system and command information and stores it on a central server where it can be used for security and quality assurance purposes.

The AAA feature allows you to verify the identity of, grant access to, and track the actions of users managing devices. The AAA feature works with these access control protocols as described in this section:

- Lightweight Directory Access Protocol
- Remote Authentication Dial In User Service
- TACACS+

Dynamic Host Configuration Protocol Client

The Dynamic Host Configuration Protocol (DHCP) client is used to configure devices that are connected to a network so they can communicate on that network using the Internet Protocol (IP). DHCP is implemented in a client-server model where DHCP clients request configuration data, such as an IP address, a default route, or DNS server addresses from a DHCP server.

Dynamic Host Configuration Protocol Relay

The Dynamic Host Configuration Protocol (DHCP) relay allows DHCP clients to communicate directly with DHCP servers in small networks with only one IP subnet. To allow DHCP clients on subnets not directly served by DHCP servers to communicate with DHCP servers, DHCP relay agents can be installed on these subnets. The DHCP client broadcasts on the local link and the relay agent receives the broadcast and transmits it to one or more DHCP servers using unicast. The DHCP server replies to the client and the relay agent then retransmits the response on the local network.

Domain Name System

The Domain Name System (DNS) translates easily-to-remember domain names into numeric IP addresses needed to locate computer services and devices. By providing a worldwide, distributed keyword-based redirection service, DNS is an essential component of the Internet.

The DNS database is hierarchical. When a client such as a Web browser gives a request that specifies a host name, the DNS resolver on the client first contacts a DNS server to determine the server's IP address. If the DNS server does not contain the needed mapping, it forwards the request to a different DNS server at the next higher level in the hierarchy. After potentially several forwarding and delegation messages are sent within the DNS hierarchy, the IP address for the given host eventually arrives at the resolver, that in turn completes the request over Internet Protocol (IP).

Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) accesses and maintains distributed directory information services over an IP network. LDAP is specified is in RFC 4511.

For ZebOS-XP, LDAP provides authentication services.

Network Time Protocol

The Network Time Protocol (NTP) is used to synchronize computer clock times in a network of computers. NTP uses Coordinated Universal Time (UTC) to synchronize computer clock times to a millisecond, and sometimes to a fraction of a millisecond.

Remote Authentication Dial In User Service

Remote Authentication Dial In User Service (RADIUS) provides centralized Authentication, Authorization, and Accounting (AAA) management for users that connect to and use a network service. RADIUS is specified in RFC 2865.

Secure Shell

Secure Shell (SSH) is a cryptographic protocol for secure data communication, remote login, remote command execution, and other secure network services between two networked computers.

Simple Network Management Protocol

SNMP provides a standardized framework and a common language for monitoring and managing devices in a network. The SNMP framework consists of three parts:

- An SNMP manager: The system used to control and monitor the activities of network devices. This is sometimes called a Network Management System (NMS).
- An SNMP agent: The component within a managed device that maintains the data for the device and reports these data SNMP managers.
- Management Information Base (MIB): SNMP exposes management data in the form of variables on the managed device which network management agents can extract from the ZebOS-XP protocols for all standard defined MIBs

In SNMP, administration groups are known as *communities*. SNMP communities consist of one agent and one or more SNMP managers. You can assign groups of hosts to SNMP communities for limited security checking of agents and management systems or for administrative purposes. Defining communities provides security by allowing only management systems and agents within the same community to communicate.

Communities are identified by community names that you assign. A host can belong to multiple communities at the same time, but an agent does not accept a request from a management system outside its list of acceptable community names. All GET and SET requests from an SNMP manager to an agent need to specify the community name of which the SNMP manager is a member so that the agent can perform access permission checking.

SNMP access rights are organized by groups. Each group is defined with three accesses: read access, write access, and notification access. Each access can be enabled or disabled within each group.

A user can begin communicating with an agent once their user name is created, their groups are set up, and they are added to the groups.

The SNMP v3 security level determines if an SNMP message needs to be protected from disclosure and if the message needs to be authenticated. The security levels are:

- noAuthNoPriv: Ne authentication or encryption
- authNoPriv: Authentication but no encryption
- authPriv: Both authentication and encryption

SNMP is defined in RFCs 3411-3418.

Syslog

Linux applications use the `syslog` utility to collect, identify, time-stamp, filter, store, alert, and forward logging data. The `syslog` utility can track and log all manner of system messages from informational to extremely critical. Each system message sent to a `syslog` server has two descriptive labels associated with it:

- The function (facility) of the application that generated it. For example, applications such as `mail` and `cron` generate messages with facilities named mail and cron.
- Eight degrees of severity (numbered 0-7) of the message which are listed in Table 6-1:.

Table 6-1: Syslog severities

Severity Level	Keyword	Description
0	emergency	System unusable
1	alert	Immediate action required
2	critical	Critical condition

Table 6-1: Syslog severities

Severity Level	Keyword	Description
3	error	Error conditions
4	warning	Warning conditions
5	notification	Normal but significant conditions
6	informational	Informational messages
7	debugging	Debugging messages

TACACS+

Terminal Access Controller Access-Control System Plus (TACACS+, usually pronounced like tack-axe) is an access control network protocol for network devices. There is no RFC protocol specification for TACACS.

Unlike RADIUS, TACACS+ provides separate authentication, authorization and accounting services. Like RADIUS, TACACS+ is an open, publicly documented protocol. TACACS+ uses the TCP protocol and encrypts the entire packet (except the header).

The differences between RADIUS and TACACS+ can be summarized as follows:

- RADIUS combines authentication and authorization in a user profile, while TACACS+ separates the two activities.
- RADIUS encrypts only the password in the access-request packet sent from the client to the server. The remainder of the packet is unencrypted. TACACS+ encrypts the entire body of the packet but leaves a standard TACACS+ header.
- RADIUS uses UDP, while TACACS+ uses TCP.
- RADIUS is based on an open standard (RFC 2865), while TACACS+ is proprietary to Cisco.

Telnet

Telnet is a client/server protocol that establishes a session between a user terminal and a remote host:

- The telnet client software takes input from the user and sends it to the server’s operating system
- The telnet server takes output from the host and sends it to the client to display to the user

While telnet is most often used to implement remote login capability, the protocol is general enough to allow it to be used for a variety of functions.

User Management

The user management feature lets you create roles that contain rules that define the operations allowed for any user who is assigned the role.

A rule is the basic element of a role. A rule defines what operations the role allows the user to perform. You can apply rules for the following:

- Command: A command or group of commands
- Feature: Commands that apply to a function provided by the device
- Feature group: A user-defined feature group.

These item have a hierarchical relationship:

- The most basic level is the command
- The next level is the feature which represents all commands associated with the feature
- The last level is the feature group which combines related features and allows you to easily manage the rules

CHAPTER 7 ZebOS-XP Versus ZebOS 7.10.x

This chapter summarizes the changes in architecture between ZebOS-XP and ZebOS 7.10.x.

NSM Architecture Changes

In ZebOS 7.10.x, NSM is the core of the architecture. NSM is the base module that simultaneously and independently communicates with every routing and switching process. NSM is a backbone of software modules, and supports IPv4, IPv6, MPLS, multicast, Layer 2 switching, and Layer 3 routing protocols. NSM also supports Routing Redundancy, Virtual Routing, plus management and services for all protocol modules. NSM communicates directly with each ZebOS routing and switching modules to manage the route tables and to perform route conversion and route redistribution. NSM also interfaces with the Platform Abstraction Layer (PAL) and the Hardware Abstraction Layer (HAL) to communicate with the underlying operating system or network processor for forwarding table updates. This puts a processing burden on NSM and causes processing delays.

ZebOS-XP enhances performance by addressing these bottlenecks. NSM is made more lightweight to minimize the load and remove bottlenecks which helps in terms of scaling and performance. The following components are moved out of NSM and into separate daemons:

- Unicast Routing Information Base Daemon
- Virtual Router Redundancy Protocol
- Layer 2 Multicast Routing Information Base Daemon

NSM is now restricted to handle limited functionality such as:

- Interface management
- Layer 2 functionality (VLAN, DCB)
- MPLS

In addition, applications such as NSM, `mstpd`, `ribd`, and `mribd` use bulk messaging. For more, see Bulk Messaging and Asynchronous Messaging

Index