Virtual Routing Engine System Description

Ericsson Core CLI

Description

5/1550-CRA 119 2047/8 Uen D2

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

Contents

1 Overview

1.1 Purpose

This document describes the Virtual Routing Engine (vRE) use, services, and architecture.

1.2 Scope

This document covers the overall architecture, functions and restrictions for vRE. It gives a description of the system's use cases and features. It also describes operation and management of the system.

1.3 Audience

This document is intended to present an overview of the vRE to network operators, network and service planners, and system engineers and administrators. The audience is expected to possess basic knowledge of telecommunications technology.

1.4 Disclaimer

This document contains vRE details, and is subject to change.

2 Introduction

The market trend towards Network Function Virtualization (NFV) is creating demand for virtualization and carrier services and applications moving to the cloud, namely Virtualized Network Functions (VNFs). Carriers want to achieve both OPEX and CAPEX savings in the long term by running VNFs on Commercial-Off-The-Shelf (COTS) hardware with cloud-based orchestration.

The carrier-grade features result from separate control and data plane architectures that can be deployed across multiple x86 platforms offering superior scaling and performance with High Availability. vRE provides cloud-based VNF using the same forwarding library of the Smart Services Router 8000 and Router 8801. It supports vanilla KVM, Mirantis OpenStack, Cloud Execution Environment (CEE), and VMware cloud systems.

vRE is designed for several deployments:

— Throughout the IP network.

— Within next-generation data centers.

— As part of NFV (Network Function Virtualization) architectures.

It is the first truly modular virtual router designed around a cloud-based architecture. vRE runs on Intel x86 based servers, which enable operators to choose their hardware platform and deploy various virtualized applications. Its modular architecture enables it to scale out seamlessly beyond the limitations of a single x86 socket or server. It has been designed to deliver industry-leading scale and resiliency for critical carrier applications. vRE has been built entirely for the cloud and NFV era.

Conceived and designed to increase velocity to virtualization, vRE represents a fundamental rethinking around how virtual network functions are deployed in next-generation networks and data centers. vRE is easy to deploy and integrate into existing networks. It is managed by a common management framework, shared with the broader Ericsson router portfolio. The vRE combines multiple functions into a single platform that provides Layer 2 and Layer 3 (IP) routing and advanced services for applications.

3 Use Case

The vRE use case and deployment options in the service provider and carrier networks.

3.1 Deployment Scenarios

vRE can be deployed in a variety of scenarios such as hosting network functions from the cloud and end-point router, among others as shown in Figure 1.

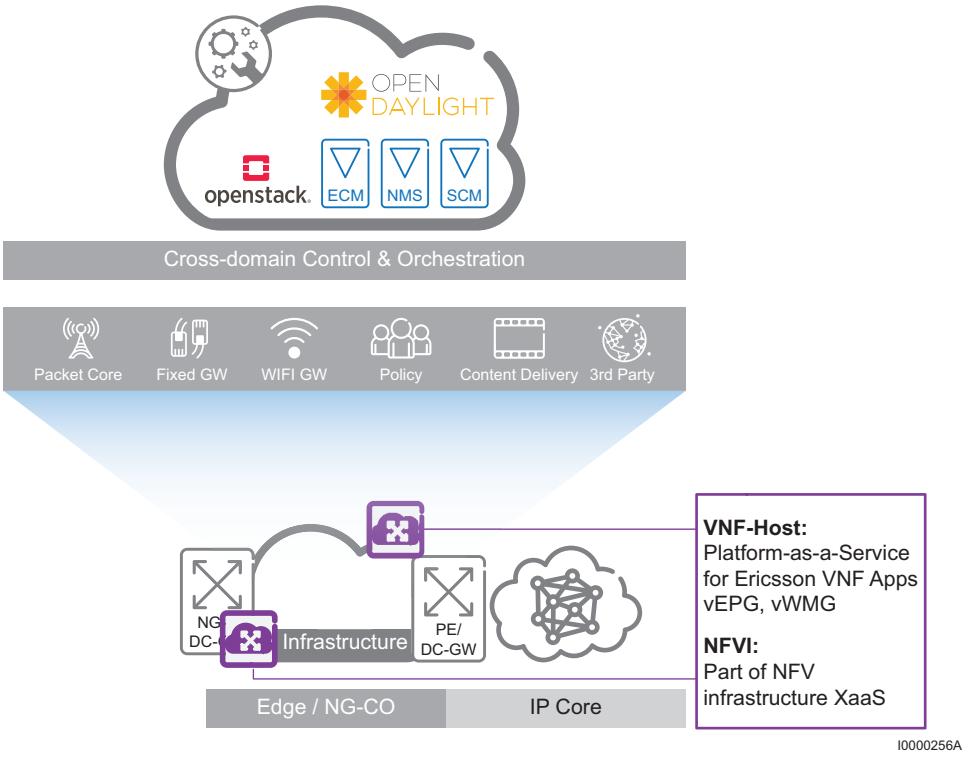

Figure 1 Overview of Possible Deployment Scenarios

3.2 NFVI

vRE can be integrated into the NFV Infrastructure (NFVI). In so doing, vRE can provide various services to other VNFs hosted in the NFV environment, such as layer 3 inter-VM routing (L3aaS). In this mode of operations, a special OpenStack Neutron plug-in is used to connect OpenStack to vRE to expose these capabilities.

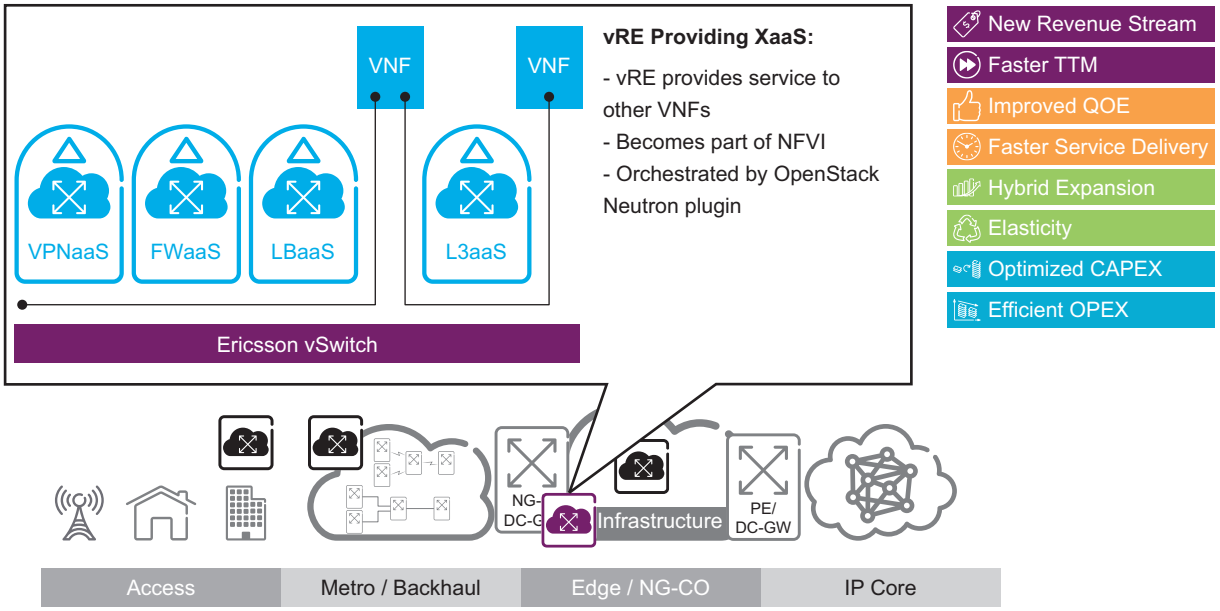

Figure 2 NFVI Scenario

4 Features

4.1 Layer 2 Solutions

The vRE platform supports Ethernet ports.

The following diagram illustrates the router in a basic Layer 2 network.

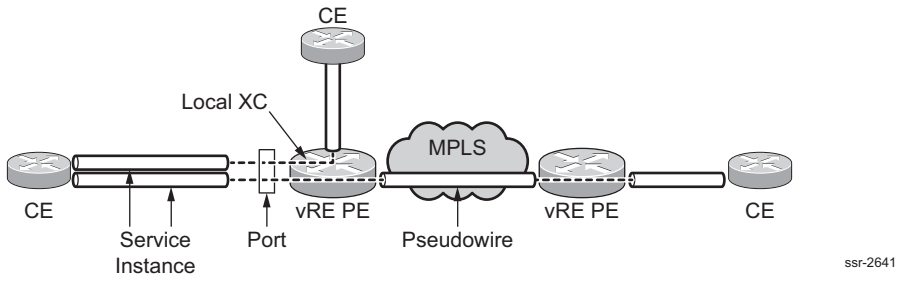

Figure 3 Router in a Layer 2 Network

4.1.1 Ethernet Transport Services

You can use the router to provide services for Ethernet traffic. For example:

— Layer 2 VPNs (L2VPNs) based on virtual private wire service (VPWS)—Provides end-to-end Layer 2 cross-connected circuits over IP/Multiprotocol Label Switching (MPLS) core networks.

— Ethernet to Ethernet Layer 2 Cross-Connects (XCs)

— Cross-Connect VPWS-based transport, including tagged and untagged frames as part of the VPWS (does not support MAC learning).

— Link Aggregation Groups (LAGs) provide increased bandwidth and availability. The Ericsson IP Operating System implementation of LAG includes both packet-based hashing, such as a trunk link group, and circuit-based hashing, such as an access-facing link group. It is referred to as Unified LAG with access LAG internals in the control plane and supports a full feature set that can be configured for either core-facing or access-facing applications. Link groups support fast failover and QoS policing, metering, and queuing features. Although you set the QoS configuration for link groups at the linkgroup level in link group configuration mode, policing, metering, and queuing are performed internally per constituent port.

  The system supports 802.1AX link groups. The number of supported ports per link group depends on the card type.

Table 1 lists the features that you can configure for Layer 2 solutions.

Note: Not all features described in this section are supported on all cards. For a list of supported features for each card, consult the *vRE Card Feature Matrix* in the `Planning` folder of this library.

Table 1 Configurable Features for Ethernet-Based Layer 2 Solution

Business Application	Layer 2 Transport Method	Routing and Label Distribution Options	Services
L2VPN (business VPN)	VPWS	LSP: LDP, T-LDP, RSVP, or RSVP-TE IGP: OSPF or IS-IS	Port Pseudowire —IPoE over PWE —IPoE over VLAN over PWE —IPoE over QinQ over PWE Layer 2 ACL (policy, filtering, metering, QoS) QoS
L2VPN (business VPN)	XC	Ethernet-Ethernet Local XC	Layer 2 ACL (policy, filtering, metering, QoS) QoS

4.2 MPLS-Based Solutions

The router supports solutions using MPLS networks in which customer connectivity among multiple remote sites is deployed across a shared central infrastructure and still provides the same access or security as a private network. For example, it supports L2VPNs, and VPWS in MPLS network topologies.

— **VPWS**

A VPWS cross-connects the local SI between the local CE and your router to a pseudowire that crosses the MPLS backbone network to the remote PE router. A VPWS is based on L2VPN in which CE routers send Layer 2 traffic to PE routers over Layer 2 circuits, that are configured between the PE and the CE routers.

The router serves as a PE router and supports these Layer 2 circuits: Ethernet port and 802.1Q VLAN.

You can configure the L2VPN on PE routers and use it to cross-connect a local Layer 2 circuit with a corresponding remote Layer 2 circuit through an LSP tunnel that crosses the network backbone. For more information, see VPWS (L2VPN).

— **BGP/MPLS VPNs**

Layer 3 BGP/MPLS VPNs are a collection of policies that control connectivity among a set of sites. A customer site is connected to the service provider network, often called a backbone, by one or more ports. The service provider associates each port with a VPN context.

A BGP/MPLS VPN allows you to implement a wide range of policies. For example, within a VPN, you can allow every site to have a direct route to

every other site (full mesh), or you can restrict certain pairs of sites from having direct routes to each other (partial mesh). For more information, see BGP/MPLS VPN.

4.3 SR-IOV VLAN and SR-IOV Flat

In an SR-IOV VLAN mode, all VLAN configuration is done on a host OS such as assigning the VLAN tag to the packets for virtual functions. In the SR-IOV flat mode, all VLAN configuration is initiated by guest OS, and the host OS does not assign VLAN tag to the packets for virtual functions.

5 System Architecture

vRE consists of a set of virtual machines and virtual interconnects (such as a virtual switch) that provide Ethernet, IP switching, and routing functions along with para-virtualized compute, network, or storage resources together with application hosting functions to other applications. As shown in the figure vRE consists of control plane Virtual Route Processor (vRP), data plane Virtual Forwarder (vFRWD), Virtual Service (vSRVC), and virtual backplane.

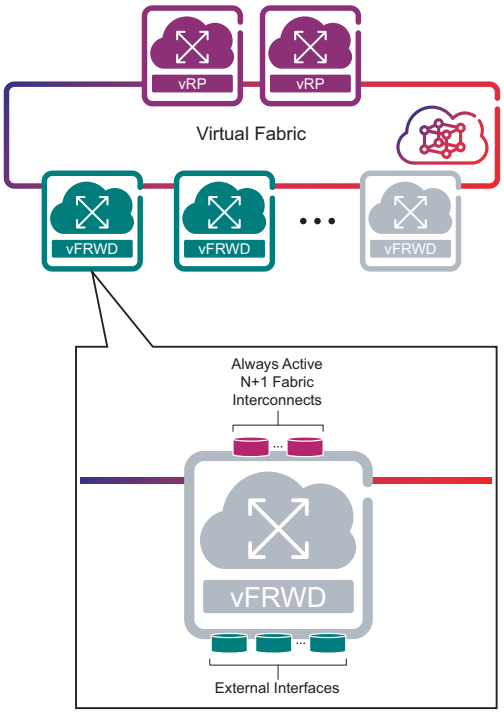

Figure 4 System Architecture

— **vRP**: Represents the control plane component which is instantiated in a independent Virtual Machines (VMs).

— **vFRWD**: Represents forwarding plane components which are instantiated in independent VM(s).

— **vSRVC**: Represents virtual machines upon which other applications can be hosted.

— **vSFO:** Represents virtual machines that offer the forwarding plane component features of vFRWD VMs while also hosting applications like vSRVC VMs.

— **Virtual Backplane**: Two interconnects providing control connectivity and data packet interconnect. The control connectivity is between the vRP and vFRWD for management and configuration. The data packet interconnect is a virtual fabric (vFAB) between vFRWD cards linking ingress and egress interfaces.

  Secure backplane capability improves the confidentiality of vRE deployments involving multiple Virtual Machines (VMs), by encrypting internal communication messages. The scope of confidentiality includes Inter Process Communication (IPC) messages, that way ensure the availability, integrity and privacy of vRE sensitive internal control plane communication. The encrypted IPC traffic includes inter vRP Virtual Route Processor (vRP) communication as well as IPC between vRPs and Virtual Forwarder (vFRWD) , Virtual Service (vSRVC), and Virtual Service-Forwarder (vSFO) components.

System utilities consist of a set of tools, scripts, templates, and guidelines provided with the vRE system components to allow easier creation or customization of virtual router instances (such as building custom OVF XML schema as a function of vRP and vFRWD and vSRVC required scale for instantiation or scale-out purposes).

System dependencies include a set of conditions, rules, and requirements that are not considered as part of the system, yet needed for successful deployment. System dependencies are grouped into two major groups:

— Underlay dependencies against cloud execution environments, host OS, hardware, and cloud underlay or overlay fabric.

— Northbound dependencies against Management and Orchestration layer including (but not limited to) Virtual Infrastructure Manager (VIM), Orchestration Manager (NFV-O), Software Defined Network (SDN) Controllers (when applicable), and Network or Service activation and management systems (such as ECM, EMS, NMS, SCM)

Cloud execution environment includes a software package that provides the entire host environment to run on the servers in a cloud deployment. The software package includes the hypervisor, host OS, and orchestration agent.

NFV-O software packages orchestrate cloud resources, create virtual machines and networks as needed, and manage VNF scaling.

Certified Systems are a set of components (hardware, cloud systems) on which vRE can be run and on which Ericsson has tested and can assure end-to-end functionality and performance. For more information, on Certified Systems refer to *Deployment and Scaling Guide*.

System scalability allows adding or removing capacity in either of the methods:

— **Scale out/in**: Data plane and IP services capacity is adjusted by adding or removing the entire vRP or vFRWD and vSRVC VMs to an existing virtual router instance.

— **Scale up/down**: Data plane or control plane capacity is adjusted by adding or removing resources (memory) to the existing VMs of a virtual router instance.

  - Scaling up/down during run-time is not supported in vRE independent of cloud platforms.
  - Scale up/down is possible in Mirantis/CEE/VMware cloud Platforms but only with downtime.

5.1 Deployment Scenarios

The vRP and vFRWD VMs can be used in different deployment scenarios: VNF (Virtualized Network Function) is run independently, while in an NFVI (Network Function Virtualization Infrastructure) deployment scenario, Openstack Neutron instantiates vRE. In NFVI deployments, specific functionality (eg. Layer 3 routing) is provided as a service (XaaS).

vRE forwarding and control planes are made up of virtual machines. Some of the main components and configuration options found in vRE deployment are shown in the following figure.

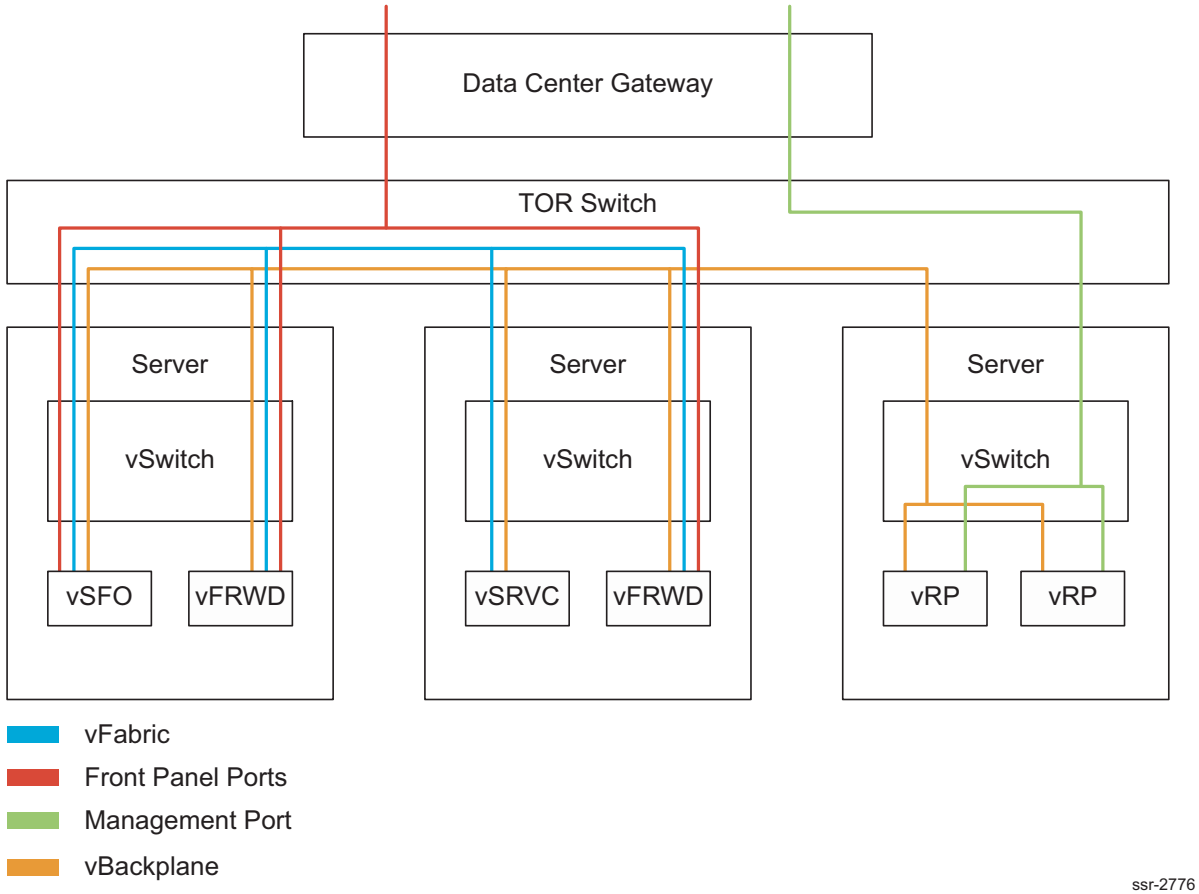

Figure 5 System Components in a PV setup

vRP is the VM that handles the control plane on the system. Each deployment supports two vRPs for redundancy purposes.

vFRWD instances make up the virtual forwarding plane. The total number of vFRWD instances supported per virtual chassis is 30. vSFO/vFRWD/vSRVC is designed to work in Paravirtulization mode (PV) or Single Root I/O Virtulization (SR-IOV) or Passthrough mode or FLAT SR-IOV.

In virtual router environments, the capacity of the virtual fabric is a limiting factor in terms of line rate traffic. If the egress and ingress PFEs would be the same, egress optimization allows the packet to loop back and be processed within the same routing context. While in some special cases it may be beneficial to span traffic across multiple cards, providing application hosting and egress processing within a single VM yields a significant performance increase by eliminating the overhead of an extra hop to a vFRWD. Egress optimization also works for upstream subscriber traffic. However, for traffic in downstream direction, traffic may still need to go across vFAB to reach vFRWD that hosts the circuit.

vSRVC VMs enable functional consolidation and delivery of IP services. They can be deployed in the vRE to run various applications, providing services beyond the scope of the terminating and forwarding capabilities provided by the vFRWD

VMs. vSRVC VMs allow VNF-Host (VNF-H) deployment scenarios; in these scenarios applications such as EPG are run on top of vRE.

vSFO VMs implement egress optimization by allowing applications to send egress packets out of the same VM without requiring an additional hop to a vFRWD for egress processing. Doing so avoids I/O bottlenecks incurred by forwarding packets between VMs across the virtual fabric. In addition to the increase in performance, they allow more efficient utilization of the 30 vRE slots by co-locating vFRWD and vSRVC functions in a single VM that occupies a single slot ID instead of two separate ones.

In PV mode, all the vFRWD vNICs connect to a vSwitch, which then provides external network connectivity. The vSwitch is an OVS for Mirantis deployments, Cloud SDN Switch (CSS) for ECS deployments, or a vNetwork Distributed Switch (vDS) for VMware deployments. When using Virtio (PV) mode without VMware, a CSS virtual switch is preferred to an OVS for its better performance.

SR-IOV mode enables VMs to directly access the x86 host networking cards to provide optimal networking performance (up to linerate). However, SR-IOV mode places restrictions on what physical hardware may be used.

Passthrough mode allows a complete NIC to be passed to the guest VM from the host. In OpenStack and CEE, Passthrough is supported only by using a YAML file; in VMware, this feature is supported through manual setting on VDS. For more details, see *Deployment and Scaling Guide*

Two major types of networks are defined: virtual backplane (that handles traffic between components) and externally accessible networks (for data traffic and access by customers).

5.2 vRE Networking

5.2.1 Virtual Backplane

The virtual backplane (vBackplane) interconnects all the components of the chassis and carries control messages to maintain and operate the chassis. The different interface descriptions are:

— BP-1: PAKIO network, vFRWD and vRP use this network to send and receive control packets and PAKIO on host packets. IPC network, vFRWD and vRP use this network to pass IPC traffic with high priority.

— MATE: RP mate-to-mate network: vRPs use this network to communicate with each other for redundancy.

The vBackplane interfaces uses custom IPv4 addresses based on the parameter `backplane_ip_net` set by the `template_generator` input configuration file.

BP-1 is the only interface on layer 2 networks, however, it is separated as two interfaces on layer 3 networks. In other words, cloud infra orchestrates only the BP-1 Ethernet port for each VM.

5.2.2 Virtual Fabric (vFAB)

The virtual fabric (vFAB) interface is the fabric-side data path forwarding used to send packets between all other cards (vFRWD, vSRVC and vSFO). All vFRWD, vSRVC and vSFO VMs must have the same number of vFAB Virtual Network Interface Cards (vNICs).

vFAB packets are a custom internal format and use a fixed MAC address (02:00:00:04:<slot_number>:02) to address the destination vFAB on each slot. A separate virtual network must be employed for the vFAB, which is inserted by the SR-IOV or virtio layer. NOTE: vFAB is not supported on PCI passthrough.

5.2.3 Externally Accessible Networks

Network for vRP management interfaces:

— This network manages the redundant vRPs from external management systems or the CLI.

— It is used with a floating IP address between the two vRPs.

— The active vRP is responsible to manage this IP.

Externally accessible network for vFRWD front panel ports (FPP) can operate in paravirtualized (PV), SR-IOV or PCI pass through mode.

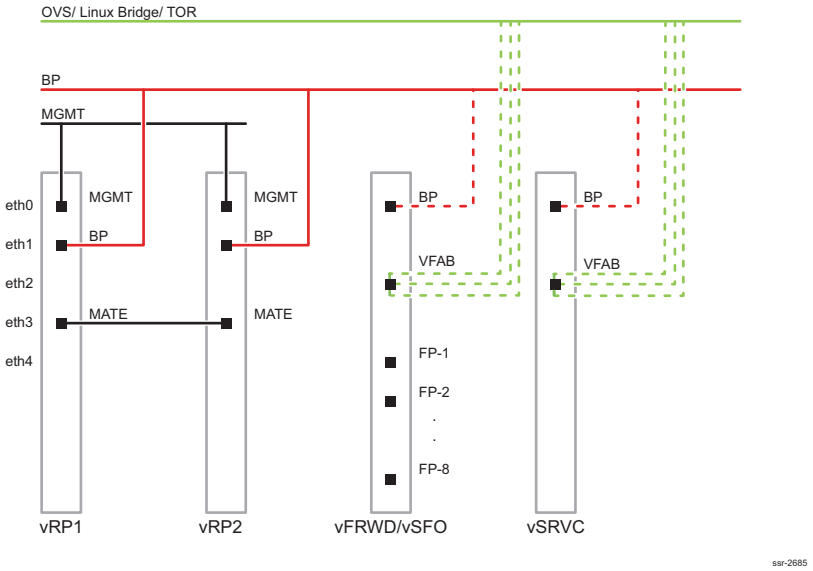

Figure 6 vRE Networks and Interfaces

5.3 System Dependencies

Execution environments provide the entire host environment to run on the servers in a cloud deployment including host OS, hypervisor, and orchestration agent.

vRE also makes use of a Network Function Virtualization Orchestrator. Supported orchestrators include ECM and VMware vCenter. With ECS or Mirantis OpenStack, ECM communicates with OpenStack to create the VMs and provides functionality to scale out running instances. With VMware, the vSphere Web Client provides a user interface to vCenter Server for provisioning and monitoring functions.

Certified execution environments use the instantiation time scale up to support a flexible vCPU footprint for vFRWD . During vFRWD instantiation time, vCPUs and memory are specified based on the scale and performance requirements. Asymmetric vCPU configurations between vFRWD VMs is supported within the same virtual chassis.

6 Operation, Administration, and Management

vRE includes the Ericsson IP Operating System that provides a set of Operation, Administration, and Management (OAM) facilities including:

— Ericsson Core CLI

— NETCONF for YANG

— SNMP

— Scripting Interface

Dynamic scale-out of VNF capacities are supported by adding vFRWD to an active instance.

6.1 Using the CLI

The CLI is the primary administrative interface for the router, which provides the following two types of commands for configuring and monitoring the system:

— Modeled commands

  The system is configured through a series of modeled commands and modes, which are rendered from YANG data models. The syntax and semantics of the modeled commands are present as the data node hierarchy and statements defined in YANG data models. For details, see YANG Data Model.

— Non-modeled commands

  Some system features are configured using non-modeled commands, for example, transaction control and system monitor. For details, see Action Commands for the vRE and Show Commands for the vRE.

You can access the CLI in the following ways:

— Ethernet management port connection to a local management workstation

  Requires a PC-type workstation using a Telnet or Secure Shell (SSH) session. Requires a shielded Ethernet crossover cable for a local workstation.

— Ethernet management port connection to a remote management workstation

  Requires a PC-type workstation using a Telnet or SSH session. Requires a shielded Ethernet straight cable (shipped with the system) or a router or bridge.

— Console port connection to a remote console terminal

  Requires either an ASCII or VT100 console terminal or equivalent that runs at 9600 baud, 8 data bits, no parity, 1 stop bit, or a PC-type workstation with a terminal emulator in the same configuration as the ASCII or VT100 terminal.

Note: Log on using the console and configure an IP address before logging on remotely.

It is advisable to have two access methods available, such as a remote workstation connected to the Ethernet management port and a console port connected to a terminal server (a console cable is shipped with the system). Several administrative tasks are performed with the CLI through a terminal server, because some processes, such as reloading or upgrading the software, interrupt an Ethernet management port connection.

The system provides default settings for local console session. You can customize the settings in the duration of console session through the following commands:

— **screen-length**: sets the screen length of the CLI output in operational mode.

— **screen-width**: sets the screen width of the CLI output in operational mode.

— **timestamp**: displays a time stamp in the UTC+00:00 timezone after a command has been entered.

— **terminal monitor**: displays system events on a remote session continuously as they are logged.

For more information about command modes and prompts, the command hierarchy, user groups for administrators, see Using the CLI.

6.2 Using the NETCONF for YANG

The NETCONF protocol provides mechanisms to modify the configuration of network devices, and to monitor status and statistics. It uses an XML- based data encoding for the configuration data and the protocol messages. Configuration and state data is represented in Ericsson IP Operating System NETCONF-specific XML format in the NETCONF messages. The NETCONF protocol operations are realized as Remote Procedure Calls (RPCs).

The NETCONF supports order-independent configuration in one transaction. The commands on the nodes of one data model do not require input in strict node dependency in a transaction.

YANG models are adopted by the system to model configuration and state data manipulated by the NETCONF, NETCONF remote procedure calls, and NETCONF notifications as defined in RFC 6020 and RFC 6991. YANG is used to model the operations and content layers of NETCONF.

For details on NETCONF for YANG, see NETCONF Interface for YANG.

6.3 Scripting Interface

The scripting interface provides an interface to execute a set of pre-defined scripting functions, which are python-based. The scripting functions support:

— Monitoring and logging system health over time.

— Detecting system problems in advance, when possible, and as they happen and taking pre-configured actions.

— Pre-empting fatal system crashes, when possible, by notifying operators and executing programmable actions.

— Reacting to specified events such as link down by executing programmable actions.

— Simplifying system troubleshooting.

Scripting functions include the following types:

— Event scheduler

— Object tracker

— Troubleshooter

The scripting interface supports runtime updates for executable scripts. Therefore, root user is able to upload, update, and delete the executable scripts at runtime, without restarting healthd process.