VyOS Networks

VyOS

VPP DATAPLANE

Accelerate critical traffic paths without redesigning your network.

2026

Product Overview

The evolution of network virtualization and cloud-native infrastructures is driving the need for routing platforms that deliver extreme performance, predictable latency, and flexible programmability. VyOS, a fully open-source network operating system, integrates Vector Packet Processing (VPP) as an optional high-speed dataplane to dramatically accelerate packet forwarding.

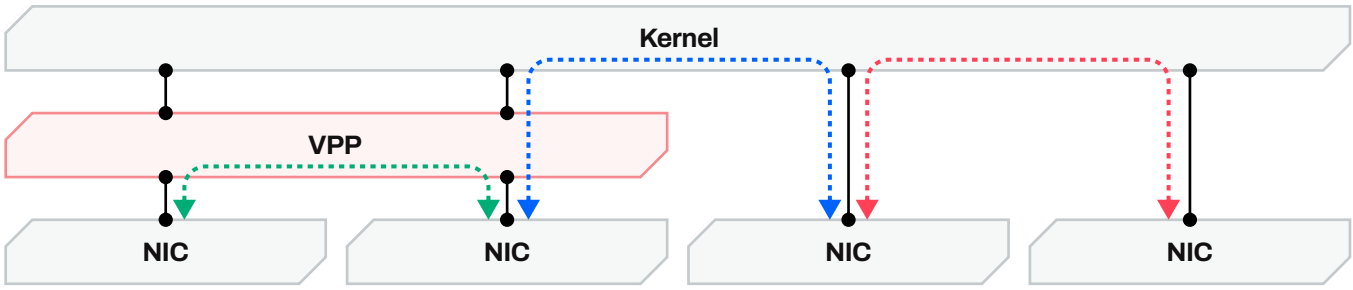
VyOS offers two forwarding architectures: the traditional Linux kernel dataplane and the VPP userspace dataplane. VPP dataplane enhances high-performance environments by enabling new solutions where high throughput is required or mandatory. This modern dataplane is ideal for demanding workloads and next-generation network infrastructures that require exceptional performance or new network virtualization functionality.

By leveraging VPP, VyOS becomes an even more powerful platform for next-generation routing, advanced firewalling, and service-chaining use cases, where performance, scalability, and deterministic behavior are critical.

Packets Processing Integration Details

VPP Dataplane integration is done in a way that minimizes configuration changes. Features that exist in the kernel dataplane are not removed but continue to operate in the kernel dataplane. VPP Dataplane only takes over packet forwarding for interfaces explicitly assigned to it.

Examples of traffic flow between interfaces connected to VPP and kernel dataplanes:



Green path

Traffic between two VPP interfaces is processed entirely within VPP for maximum performance. Packets that follow this path can use only features available inside the VPP dataplane.

Blue path

Traffic between a VPP interface and a kernel interface is processed by both dataplanes, with VPP handling the VPP side and the kernel handling the kernel side. Packets that follow this path can use features available in both VPP and kernel dataplanes at the same time.

Note: Because packets must follow both dataplanes, performance will be slower than with pure VPP or pure kernel forwarding. This path works well for low-speed control plane traffic or non-primary traffic flows that need to traverse both dataplanes.

Red path

Traffic between two kernel interfaces is processed entirely within the kernel dataplane. Packets that follow this path can use only features available inside the kernel dataplane and lack VPP acceleration.

This is the traditional VyOS dataplane operation.

When to Use VPP

VPP is an ideal choice for environments with:

- High throughput demands
- Latency-sensitive applications requiring consistent performance

When to Use the Kernel Dataplane

The kernel dataplane remains well-suited for scenarios involving:

- Flexibility with new architectures that require kernel involvement
- Custom kernel packages built and maintained by VyOS (set, patches, etc)
- Applications that depend on features not currently supported by the VPP dataplane

System Requirements and Compatibility

The VPP dataplane can only be enabled on VyOS-validated platforms and approved network interface cards. Hardware support is restricted to components that have been tested and certified by VyOS.

Area	Requirements / Notes
CPU	SSE4.2 support required; minimum 4 physical cores
Memory	Minimum 8 GB RAM; additional memory recommended for large routing tables or NAT-heavy workloads
NIC Support	Only validated NICs are supported; Compatibility is limited to specific models (Intel E810-2CQDA2, NVIDIA/Mellanox ConnectX-5/6/7, VirtIO; GVE; ENA; Hyper-V)

Note: The NIC compatibility list is subject to change and will be expanded with more tests done by VyOS. Reach out to us if you want to validate a NIC not presented in the current list.

VyOS 1.5 VPP Dataplane Implementation

Features supported:

List of feature supported on VyOS VPP Dataplane:

Interface and L2/L3 services

Feature	Capabilities
Bonding interface	Hash policy: layer2, layer2+3, layer3+4; Custom MAC address; Mode: 802.3ad, active-backup, broadcast, round-robin, xor-hash; Compatible with services in kernel dataplane
Bridge domain	Only for untagged traffic; L3 features (IP addresses)
GRE tunnel	Modes: point-to-point or point-to-multipoint; Tunnel type: erspan, l3, teb; External tunnel address: IPv4/IPv6; Compatible with services in kernel dataplane
IPIP tunnel	External tunnel address: IPv4/IPv6; Compatible with services in kernel dataplane
VXLAN	External tunnel address: IPv4/IPv6; VNI; Compatible with services in kernel dataplane
L2 cross-connect	Direct connection between two interfaces
Interfaces management via Kernel	IPv4/IPv6 static and DHCP addresses; MTU; VLAN subinterfaces;

NAT44 and CGNAT

Feature	Capabilities
NAT44 interfaces	Inside interface list; Outside interface list
NAT44 address pools	Translation pools: address range or interface; Twice-NAT pools: address range or interface
NAT44 static mappings	External: address + port; Local: address + port; Protocol selection; Options: twice-nat, self-twice-nat, out-to-in-only, twice-nat-address
NAT44 exclude rules	Local: address + port; Protocol selection; External interface

Feature	Capabilities
NAT44 global settings	Timeouts: icmp, tcp-established, tcp-transitory, udp; Session limit per thread; Worker list/range
CGNAT interfaces	Inside interface list; Outside interface list
CGNAT rules	Inside IPv4 prefix; Outside IPv4 prefix
CGNAT timeouts	icmp, tcp-established, tcp-transitory, udp

IP and MAC ACLs

Feature	Capabilities
IP ACLs	Action: permit, deny, permit-reflect; Source: prefix + port range; Destination: prefix + port range; Protocol selection; TCP flags match
IP ACL interface binding	Apply ACL tag-name to the interface input or output direction
MACIP ACLs	Action: permit or deny; Source IP prefix; Source MAC address + MAC mask
MACIP ACL interface binding	Apply ACL tag-name to the interface

Telemetry and flow export

Feature	Capabilities
IPFIX	Collector: IPv4/IPv6 destination + port (default 4739); Path MTU; Source address: IPv4/IPv6; Template interval (seconds); UDP checksum; Interface probes: selective interface, direction (rx/tx/both), flow-variant (l2/ipv4/ipv6); Flowprobe record layers (l2/l3/l4, multi); Active/inactive timeouts
VPP sFlow	Integrated with kernel; Interface list; Header-bytes: 64/96/128/160/192/224/256

Integration and lifecycle nuances

Area	Details
Linux control plane pairing	VPP interfaces can be paired with kernel interfaces for address management, visibility, and transparent connectivity; LCP synchronization is configurable via netlink. Routes from the kernel can be installed or ignored in VPP

Area	Details
PPPoE and DHCP integration	PPPoE server and DHCP detection integrate with VPP LCP pairs to keep control-plane services in Linux while accelerating data-plane interfaces
Dynamic routing protocols and Redundancy/HA integration	Dynamic routing protocols such as BGP, OSPF, and VRRP (for Redundancy/HA functionality) integrate with VPP through LCP pairs. This architecture keeps control-plane services running in Linux while leveraging VPP to accelerate dataplane interfaces.

Observability and operational visibility (op-mode)

Operational View	Capabilities
VPP interfaces	Interface inventory, hardware counters, dataplane statistics, LCP pair visibility
ACL	IP, MACIP; Interfaces lists; Applied tags
NAT44	Summary, sessions, interfaces, address pools, static mappings
CGNAT	Summary, sessions, interfaces, rules, and prefixes
IPFIX	Collectors, interfaces, flowprobe configuration, and status
LACP and bonding	LACP state for VPP bonds
Bridge domains	Bridge domain and member status