

# پیاده‌سازی شبکه مجازی با Open vSwitch

معماری، قابلیت‌ها و عملکرد در محیط‌های ابری

ارائه دهنده: پویا کارآزمایان

نام درس: مجازی‌سازی شبکه / شبکه پیشرفته

# مجازی سازی شبکه با یک بررسی عمیق فنی معماری، قابلیت‌ها و پیاده‌سازی عملی

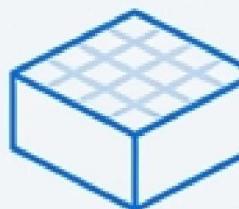
در این ارائه، به کالبدشکافی Open vSwitch به عنوان یک سویچ مجازی چندلایه و قابل برنامه‌ریزی می‌پردازیم. از معما ری بنیادین آن، شامل تفکیک فضای کرنل و کاربر، تا قابلیت‌های پیشرفته مانند تونل‌زنی، QoS و نقش آن در زیرساخت‌های مدرن ابری مانند OpenStack را بررسی خواهیم کرد.

# فهرست مطالب



## ۱. پیاده‌سازی سناریوهای عملی

- نصب و دستورات کلیدی DPDK
- سناریو عملی و نقش OpenStack
- جمع‌بندی و کاربرد در OpenStack



## ۲. مقدمه و مبانی

- تعریف مجازی‌سازی شبکه (NFV)
- چالش‌های سوییچ‌های سنتی

## ۳. معماری Open vSwitch

- معرفی OVS و اهداف آن
- تفکیک معماری (Kernel/User Space/DB)
- SDN و OpenFlow
- نقش پروتکل

## ۴. قابلیت‌های کلیدی و پیشرفته

- مدیریت VLAN و Flow Table
- تونل‌زنی Overlay و کیفیت سرویس (QoS)
- Linux Bridge
- مانیتورینگ و مقایسه با

## ۵. پیاده‌سازی و سناریوهای عملی

- نصب و دستورات کلیدی DPDK
- سناریو عملی و نقش OpenStack

## ۶. نتیجه‌گیری

- جمع‌بندی و کاربرد در OpenStack

# مجازی‌سازی توابع شبکه (NFV) چیست؟

تعریف NFV: «جداسازی توابع شبکه (مانند مسیریابی، فایروال، سوییچینگ) از سخت‌افزارهای اختصاصی و پیاده‌سازی آن‌ها به صورت نرم‌افزاری بر روی سرورهای استاندارد (SHVs).».

\***چابکی (Agility)**: ارائه سریع‌تر سرویس‌های جدید.

\***کاهش هزینه‌ها (Cost Reduction)**: استفاده از سخت‌افزارهای استاندارد و کاهش هزینه‌های عملیاتی.

\***جلوگیری از وابستگی به فروشنده (Vendor Lock-in)**: آزادی در انتخاب نرم‌افزار و سخت‌افزار.

زیرساخت مجازی‌شده NFV



زیرساخت سنتی



# محدودیت‌های سویچ‌های لایه ۲ سنتی

عدم کنترل برنامه‌پذیر  
پیکربندی‌های استاتیک و دشواری  
در اتوماسیون شبکه.



دید محدود به ترافیک مجازی  
دشواری در مانیتورینگ ترافیک  
بین ماشین‌های مجازی  
(East-West traffic) که روی یک  
هاست فیزیکی قرار دارند.



**مدیریت پیچیده VLAN**  
پیکربندی دستی، محدودیت  
مقیاس‌پذیری (سقف ۴۰۹۵) و عدم  
انعطاف‌پذیری در محیط‌های بزرگ.

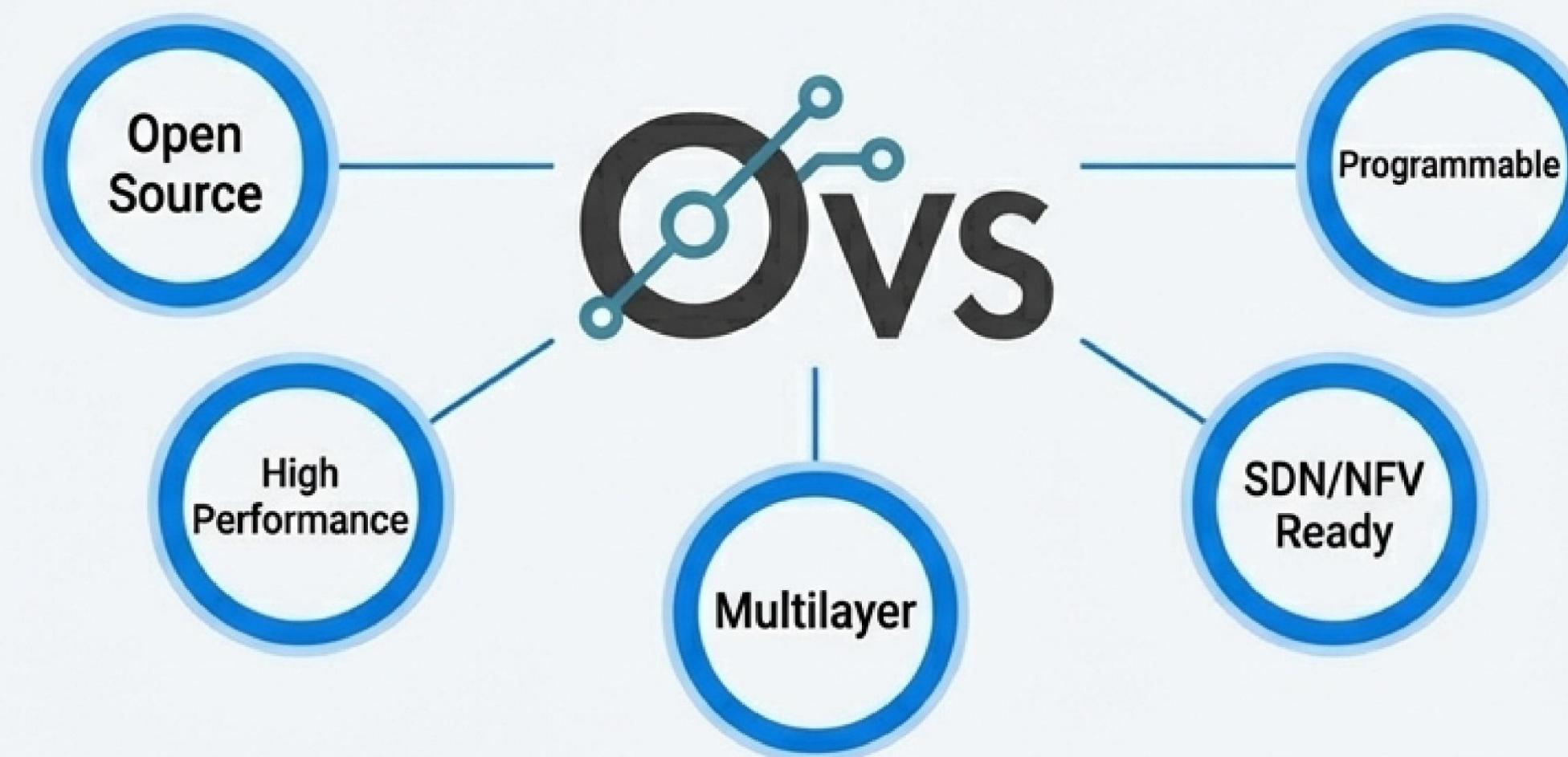


وابستگی به سخت‌افزار  
عدم امکان جداسازی منطق کنترل  
از سخت‌افزار و وابستگی به یک  
فروشنده خاص.



# معرفی یک سوییچ مجازی چندلایه (OVS) :Open vSwitch

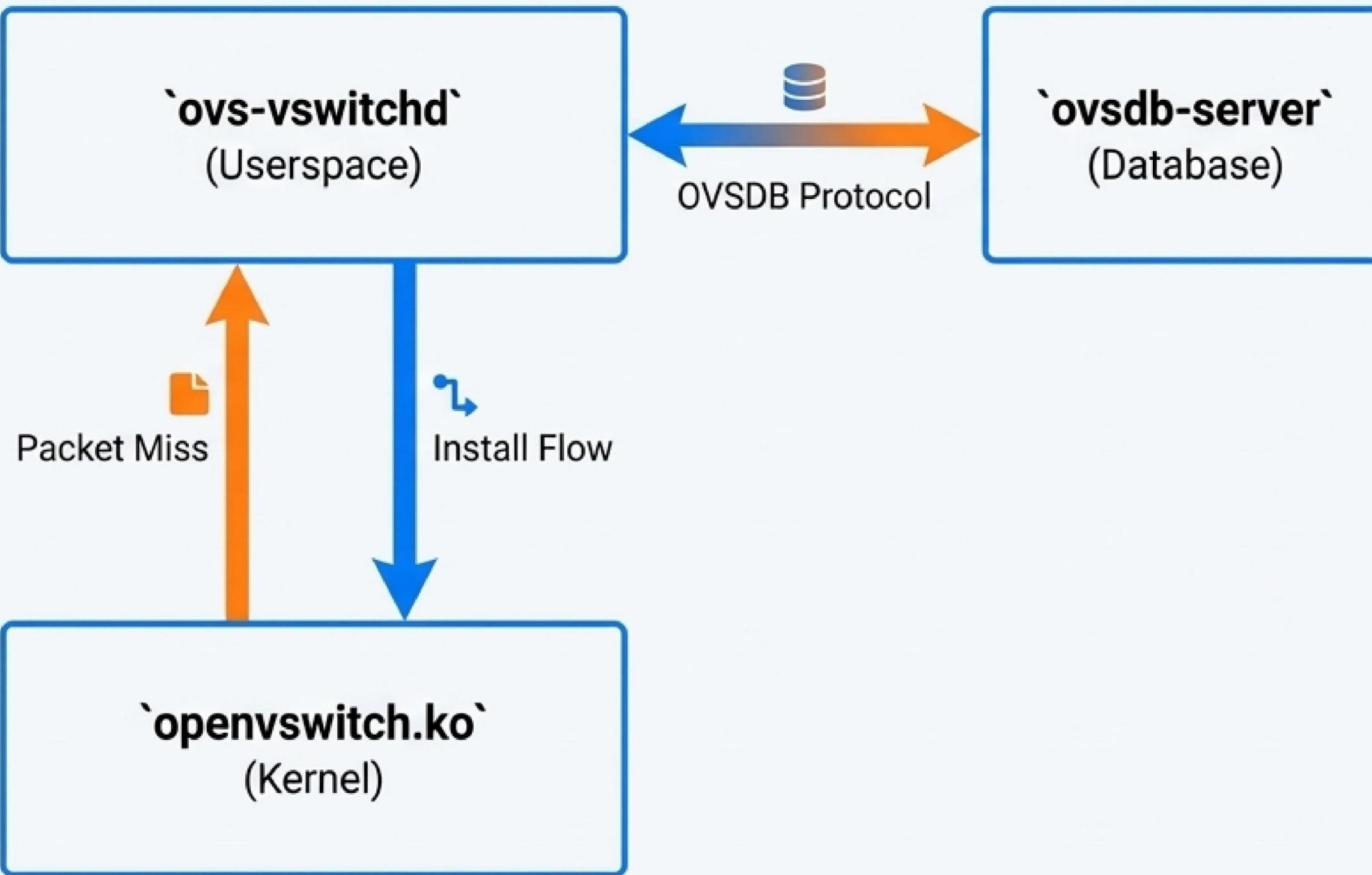
«OVS یک سوییچ مجازی متن-باز و چندلایه (multilayer) با کیفیت تولید است که برای اتوماسیون گستردگی شبکه از طریق کنترل برنامه‌پذیر طراحی شده است.»



اهداف اصلی OVS:

- فعالسازی اتوماسیون شبکه در مقیاس بزرگ.
- پشتیبانی از پروتکل‌های مدیریتی استاندارد (مانند NetFlow, sFlow).
- ارائه یک پلتفرم غنی برای پیاده‌سازی SDN و NFV.
- پشتیبانی از محیط‌های توزیع شده در چندین سرور فیزیکی.

# معماری OVS: تفکیک فضای کرنل، کاربر و دیتابیس



## Kernel Module (`openvswitch.ko`)

این بخش **Fast Path** یا **Datapath** است. جریان‌های (flows) شناخته‌شده را با سرعتی نزدیک به سخت‌افزار پردازش می‌کند.

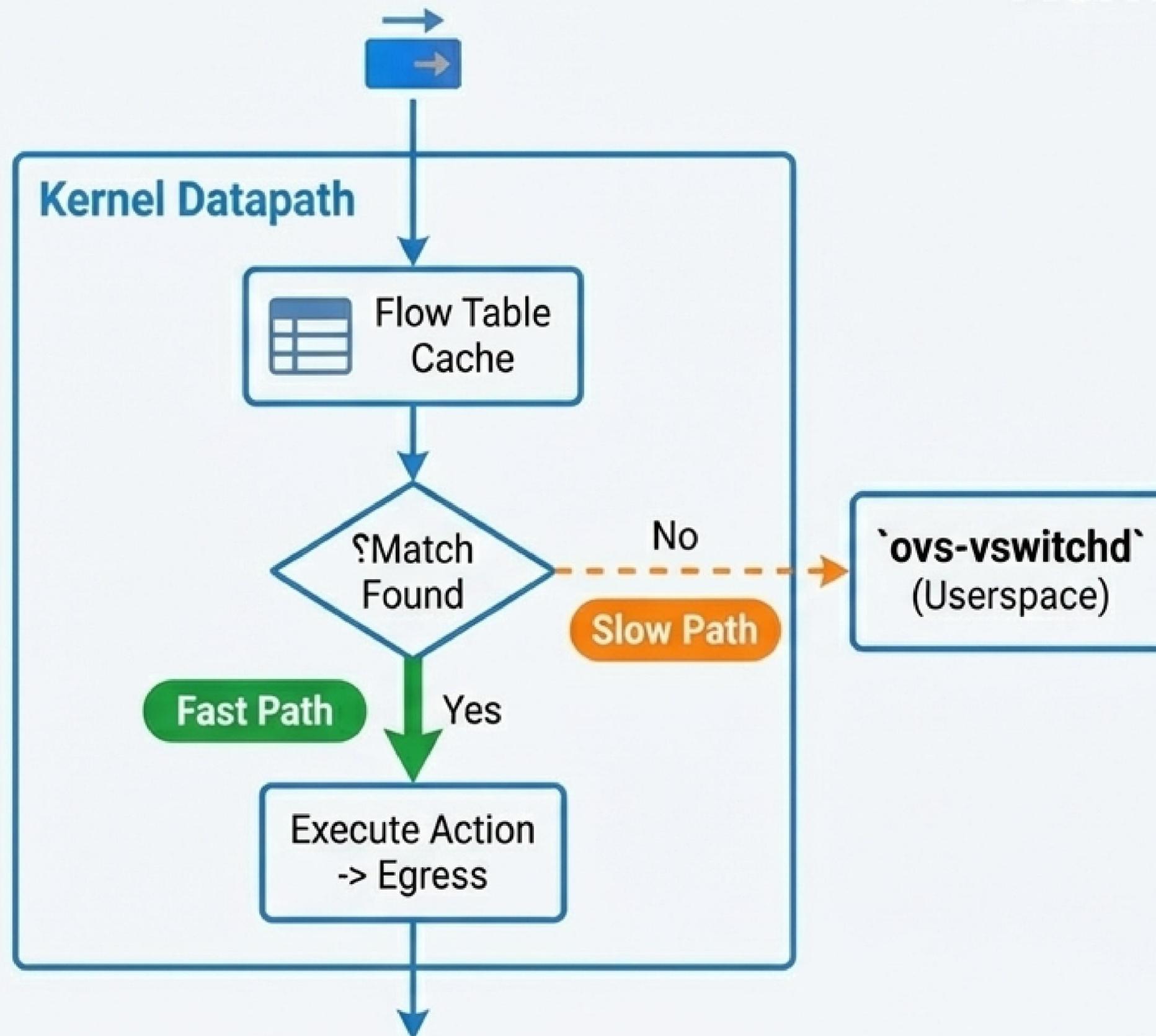
## Userspace Daemon (`ovs-vswitchd`)

این بخش **Slow Path** یا **Control Plane** است. بسته‌های جدید و ناشناخته (packet misses) را پردازش کرده، تصمیمات پیچیده مسیریابی را اتخاذ می‌کند و flow جدید را در مازول کرنل نصب می‌کند.

## Database Server (`ovsdb-server`)

این بخش پیکربندی سویچ (bridge)، پورت‌ها، اینترفیس‌ها) را در خود ذخیره می‌کند. ابزار `ovs-vsctl` با این دیتابیس از طریق پروتکل OVSDB ارتباط برقرار می‌کند.

# مسیر سریع: بررسی Kernel Datapath



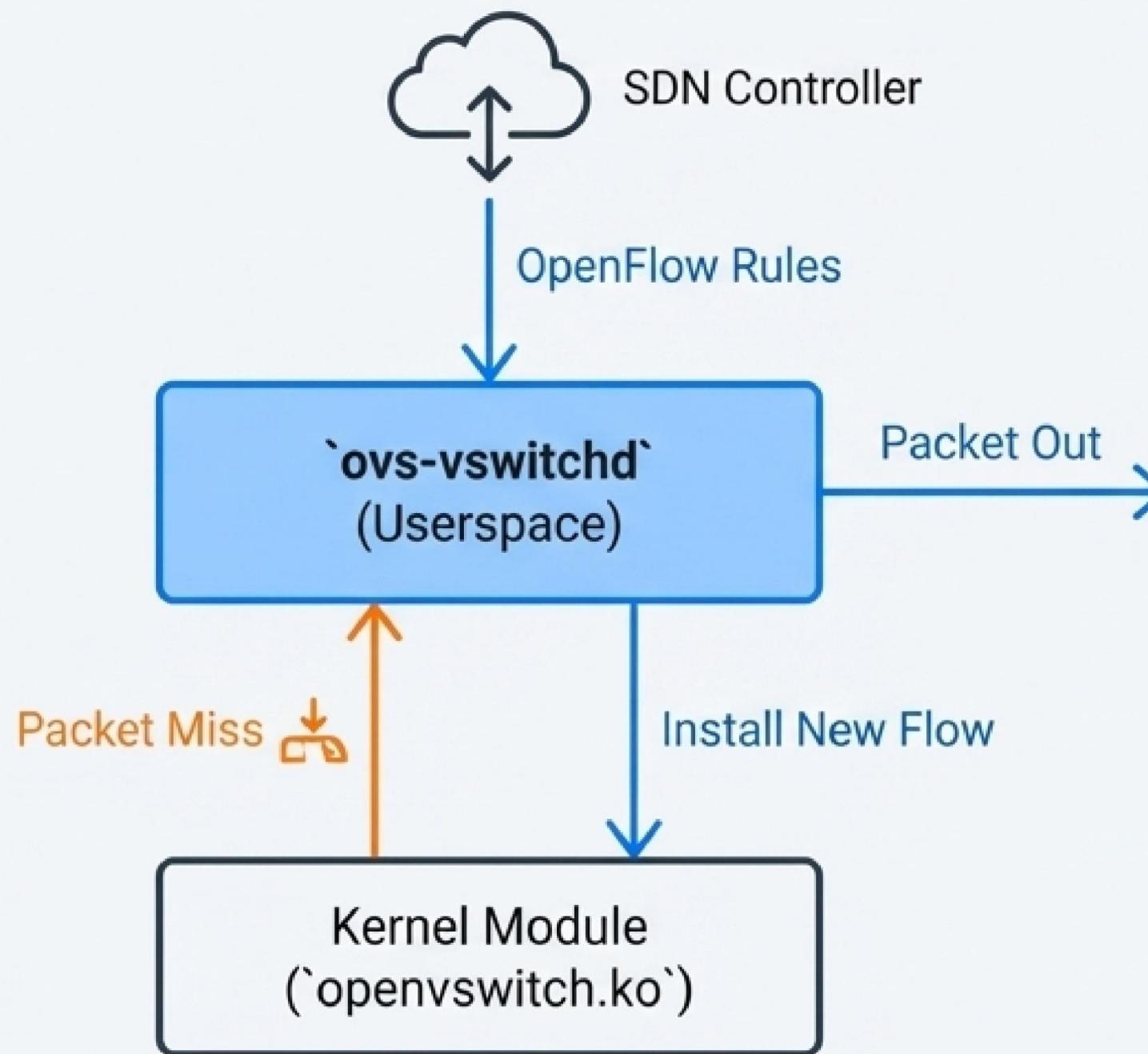
عملکرد

کرنل شامل یک جدول جریان (flow table) ساده برای جستجوی سریع است.

فرآیند

۱. یک بسته وارد مازول کرنل می‌شود.
۲. جدول جریان برای یافتن یک rule منطبق جستجو می‌شود.
۳. اگر مطابقت پیدا شود (Flow Hit): اکشن مربوطه (مثلاً Fast Path به یک پورت) فوراً اجرا می‌شود. این مسیر فوراً اجرا می‌شود.
۴. اگر مطابقت پیدا نشود (Flow Miss): بسته برای تصمیم‌گیری گیری به 'ovs-vswitchd' در فضای کاربر ارسال (punt) می‌شود. این مکانیزم کلید عملکرد بالای OVS برای ارتباطات برقرار شده است.

# vswitchd Daemon: نقش



## نقش اصلی

دریافت بسته‌هایی که کرنل آن‌ها را تشخیص نمی‌دهد (packet misses).

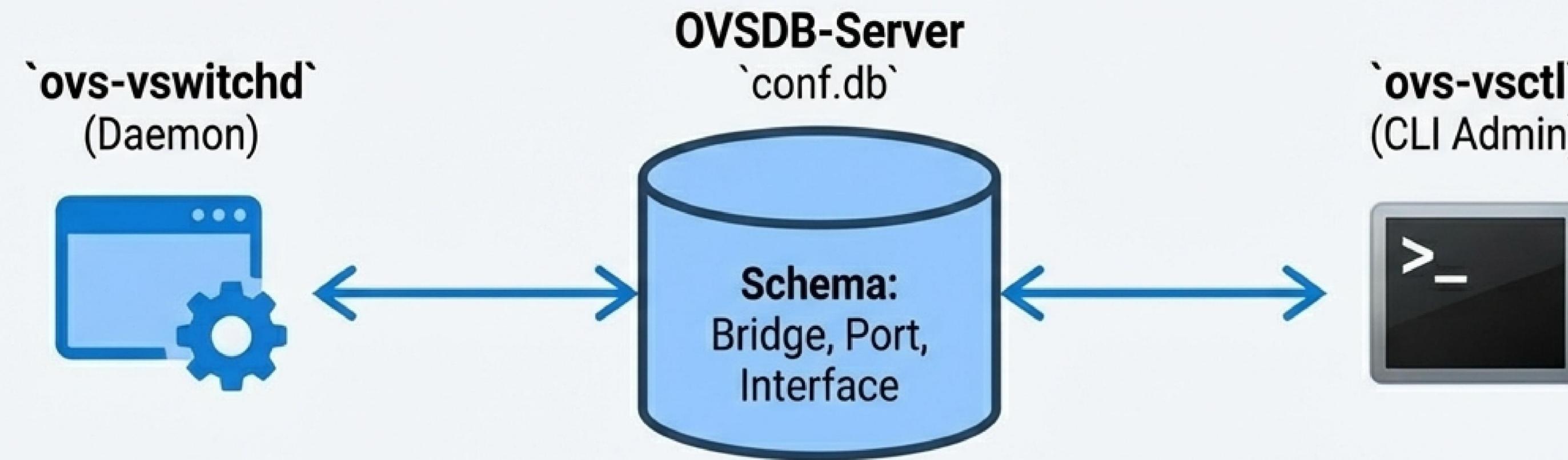
## منطق تصمیم‌گیری

منطق پیچیده‌ای را برای تصمیم‌گیری در مورد سرنوشت بسته اعمال می‌کند. این منطق اغلب توسط یک کنترلر SDN از طریق پروتکل OpenFlow دیکته می‌شود.

## ایجاد Flow جدید

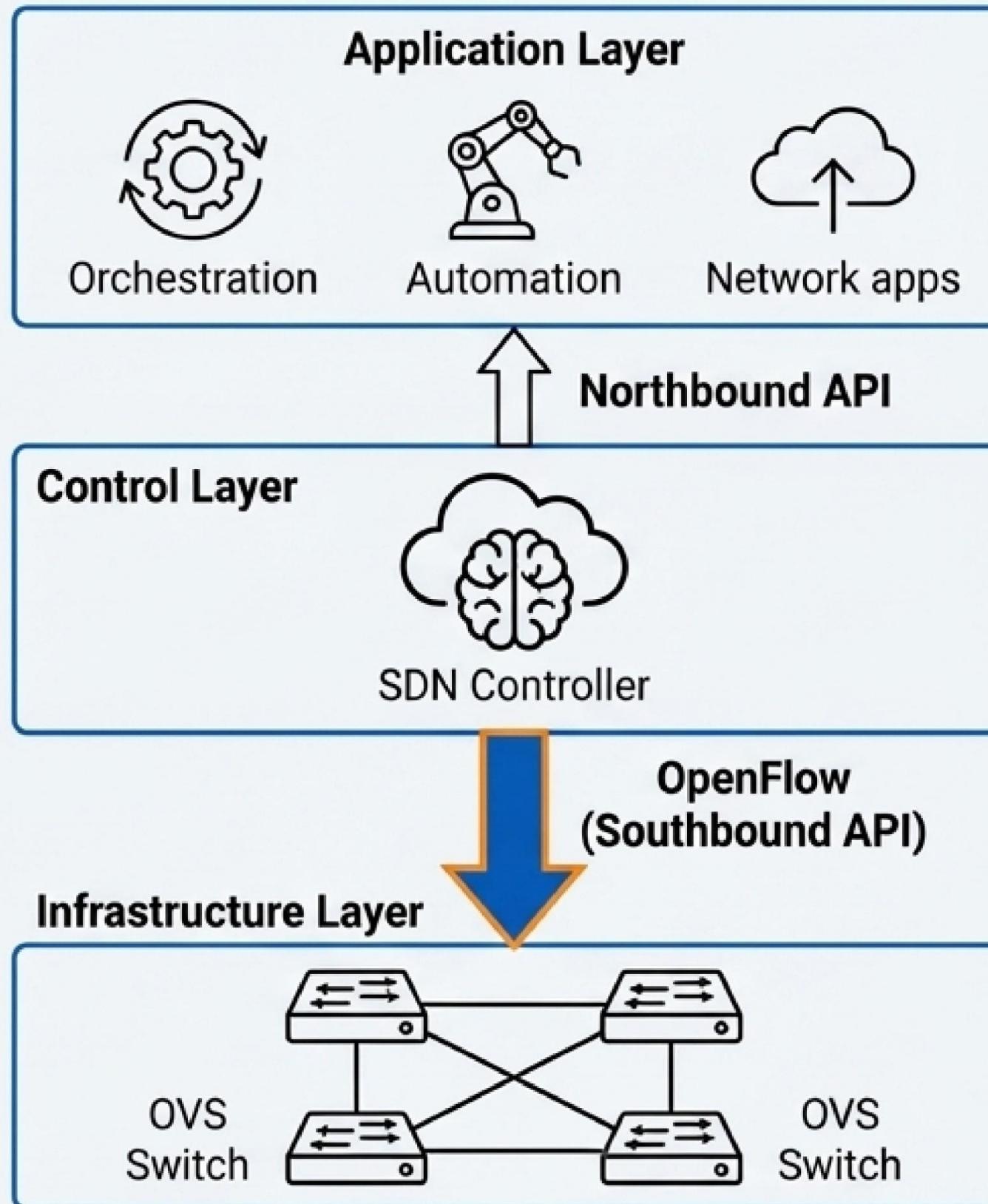
پس از تصمیم‌گیری، 'ovs-vswitchd' یک flow جدید ایجاد کرده و آن را در Datapath کرنل نصب می‌کند. این کار باعث می‌شود بسته‌های بعدی در همان جریان، از طریق Fast Path پردازش شوند و دیگر نیازی به ارسال به فضای کاربر نداشته باشند.

# لایه پیکربندی: نقش دیتابیس OVSDB-Server



- `ovsdb-server` میزبان دیتابیس پیکربندی OVS (معمولًا `conf.db`) است.
- تمام وضعیت‌های پایدار سوییچ شامل bridgeها، پورت‌ها، اینترفیس‌ها، تگ‌های VLAN، و تنظیمات تونل در این دیتابیس ذخیره می‌شوند.
- ابزارهایی مانند `ovs-vsctl` برای پیکربندی سوییچ با این دیتابیس تعامل می‌کنند.
- `ovs-vswitchd` نیز اطلاعات پیکربندی را از این دیتابیس می‌خواند تا بداند چه ساختاری را باید پیاده‌سازی کند.

# قدرت برنامه‌پذیری با SDN و OpenFlow



## Software-Defined Networking (SDN)

یک معماری شبکه که در آن صفحه کنترل (Control Plane) از صفحه داده (Data Plane) جدا شده و به صورت مرکزی مدیریت می‌شود.

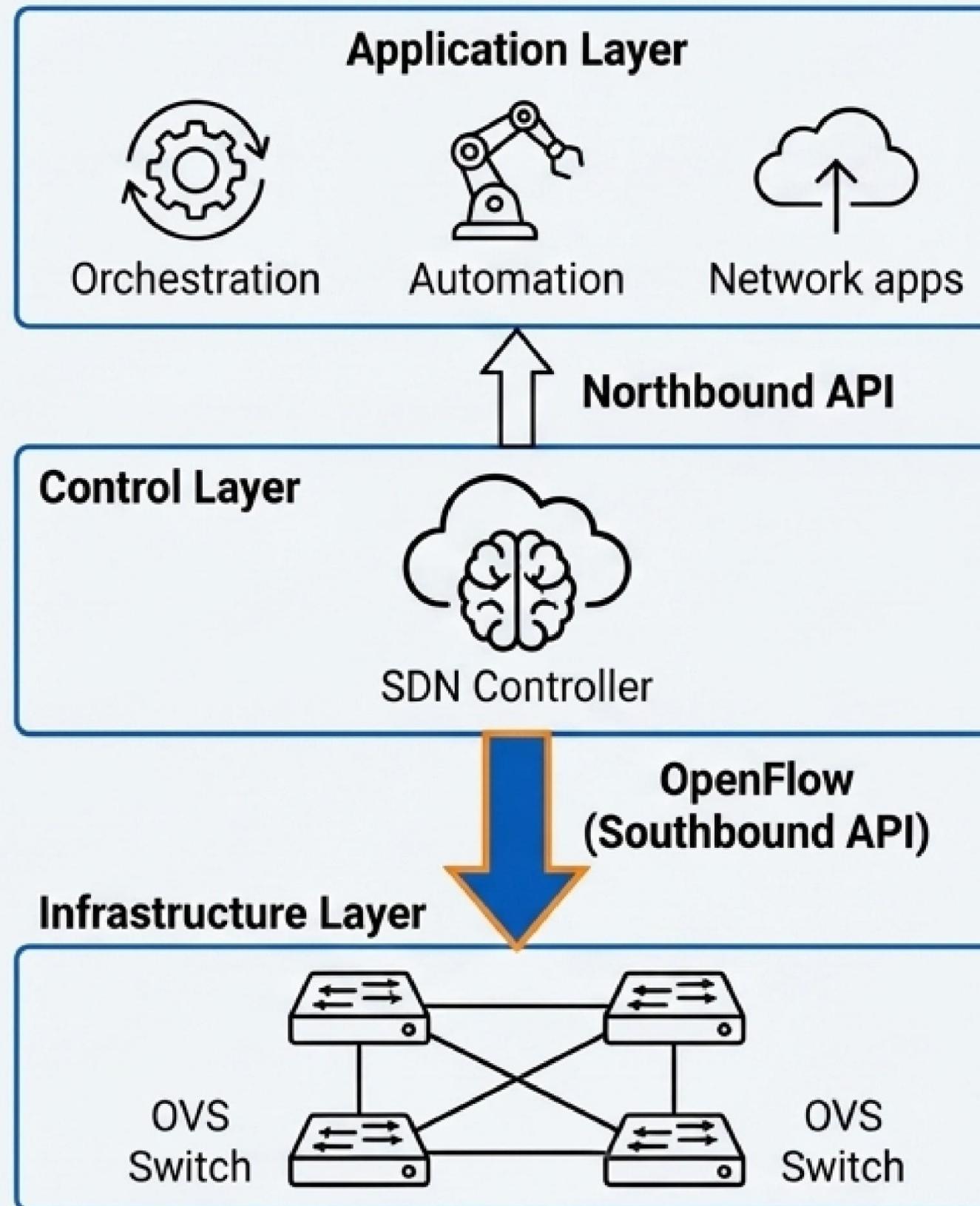
## OpenFlow

پروتکل ارتباطی استاندارد بین کنترلر SDN (مغز شبکه) و سویچ‌ها (مانند OVS).

## رابطه

کنترلر SDN تصمیمات سطح بالا را می‌گیرد و با استفاده از OpenFlow rule، آنها را به مشخصی ترجمه می‌کند که آنها را در datapath ovs-vswitchd نصب می‌کند.

# قدرت برنامه‌پذیری با SDN و OpenFlow



## Software-Defined Networking (SDN)

یک معماری شبکه که در آن صفحه کنترل (Control Plane) از صفحه داده (Data Plane) جدا شده و به صورت مرکزی مدیریت می‌شود.

## OpenFlow

پروتکل ارتباطی استاندارد بین کنترلر SDN (مغز شبکه) و سویچ‌ها (مانند OVS).

## رابطه

کنترلر SDN تصمیمات سطح بالا را می‌گیرد و با استفاده از OpenFlow rule، آنها را به مشخصی ترجمه می‌کند که آنها را در datapath ovs-vswitchd نصب می‌کند.

# مدیریت Flow Table: منطبقسازی و اجرا

اولویت	فیلدهای تطبیق	اکشن‌ها			شمارنده‌ها
		→ Output	✎ Modify	刪 Drop	
100	in_port=1 dl_src=00:11:22:33:44:55 nw_dst=10.0.0.1	→ output:2 ✎ mod_dl_dst=66:77:88:99:aa:bb			packets=150, bytes=12345
10	nw_dst=192.168.1.1	刪 drop			packets=5, bytes=450

هر ورودی در Flow Table شامل سه بخش اصلی است:

- (فیلدهای تطبیق): معیارهایی برای شناسایی بسته‌ها (مانند پورت ورودی، آدرس‌های MAC و IP، پورت‌های TCP/UDP).
- (اکشن‌ها): عملیاتی که باید روی بسته‌های منطبق انجام شود (مانند ارسال به پورت، تغییر هدر، حذف بسته).
- (شمارنده‌ها): آمار مربوط به تعداد بسته‌ها و بایت‌های منطبق شده با rule.
- مثال از یک rule ساده با استفاده از `ovs-ofctl dump-flows` :

```
cookie=0x0, duration=3600.000s, table=0, n_packets=50, n_bytes=5000, priority=32768, in_port=1
actions=output:2
```

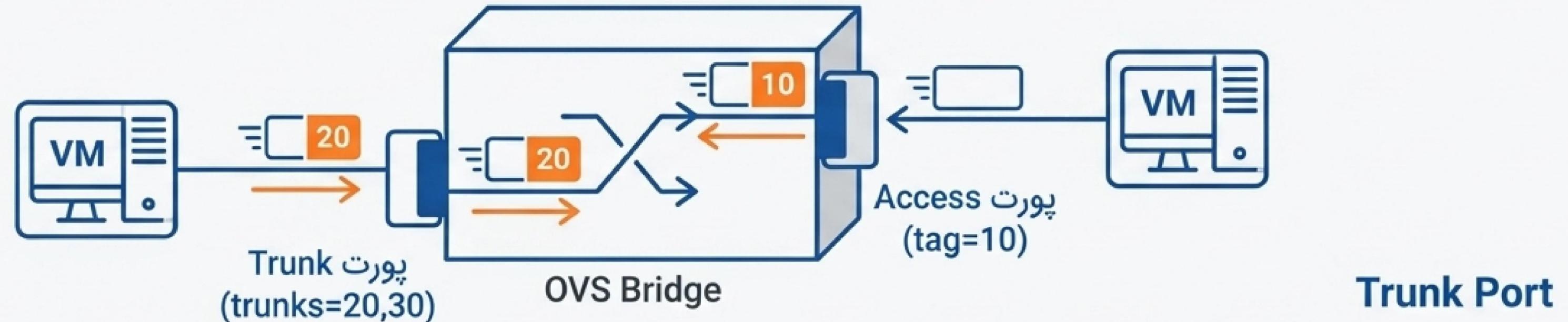
# (802.1Q) VLAN پیاده‌سازی

## Access Port

ترافیک بدون تگ را از یک ماشین مجازی دریافت کرده و تگ VLAN مشخصی به آن اضافه می‌کند.

```
ovs-vsctl set port <port_name> tag=<vlan_id>
```

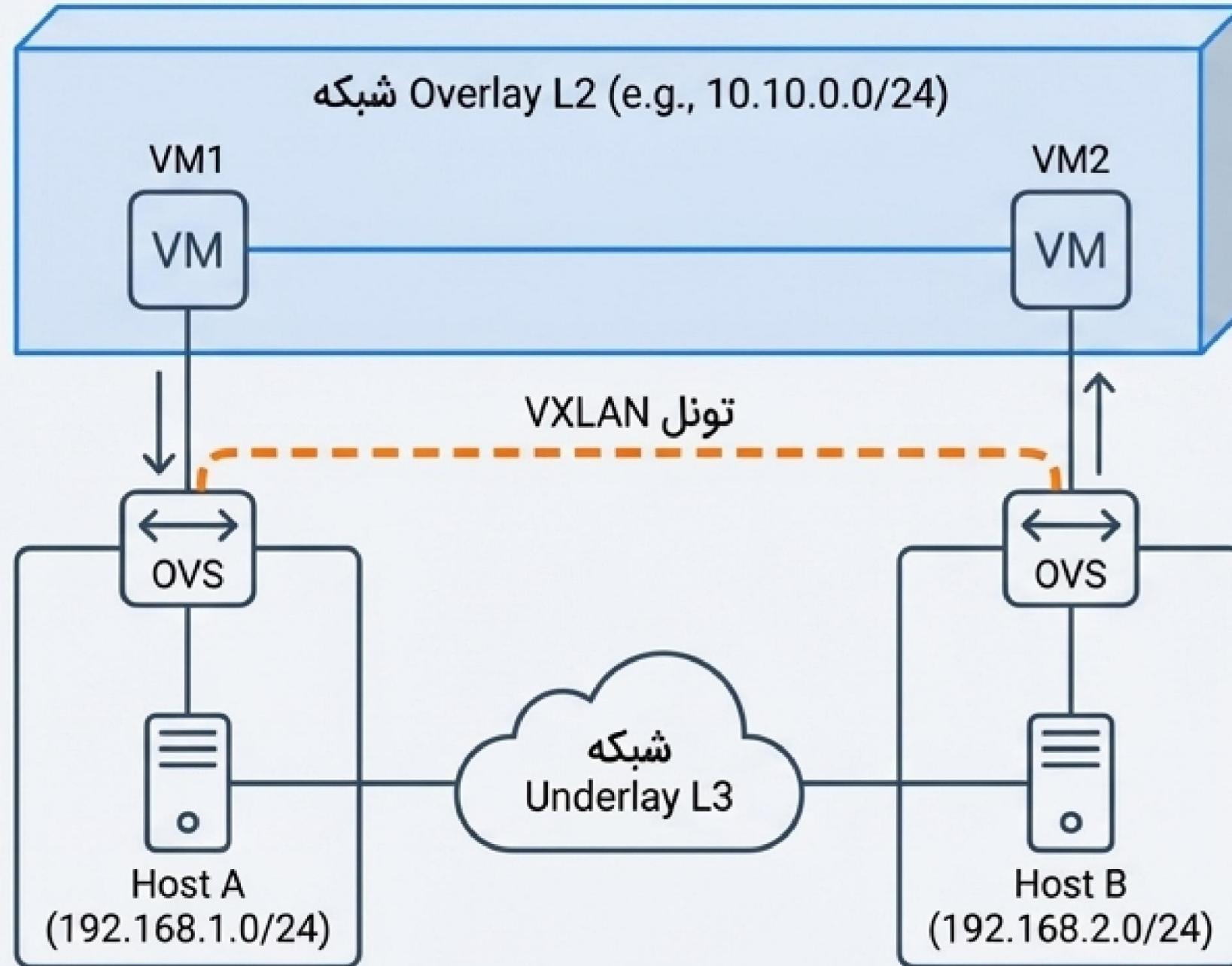
دو نوع پورت VLAN را پشتیبانی می‌کند:



ترافیک تگدار را برای چندین VLAN عبور می‌دهد.

```
ovs-vsctl set port <port_name> trunks=<vlan_id1>,<vlan_id2>,...
```

# شبکه‌های Overlay با تونل‌زنی (VXLAN, GRE)



## شبکه Overlay

یک شبکه مجازی که بر روی یک شبکه فیزیکی موجود (Underlay) ساخته می‌شود.

## تونل‌زنی

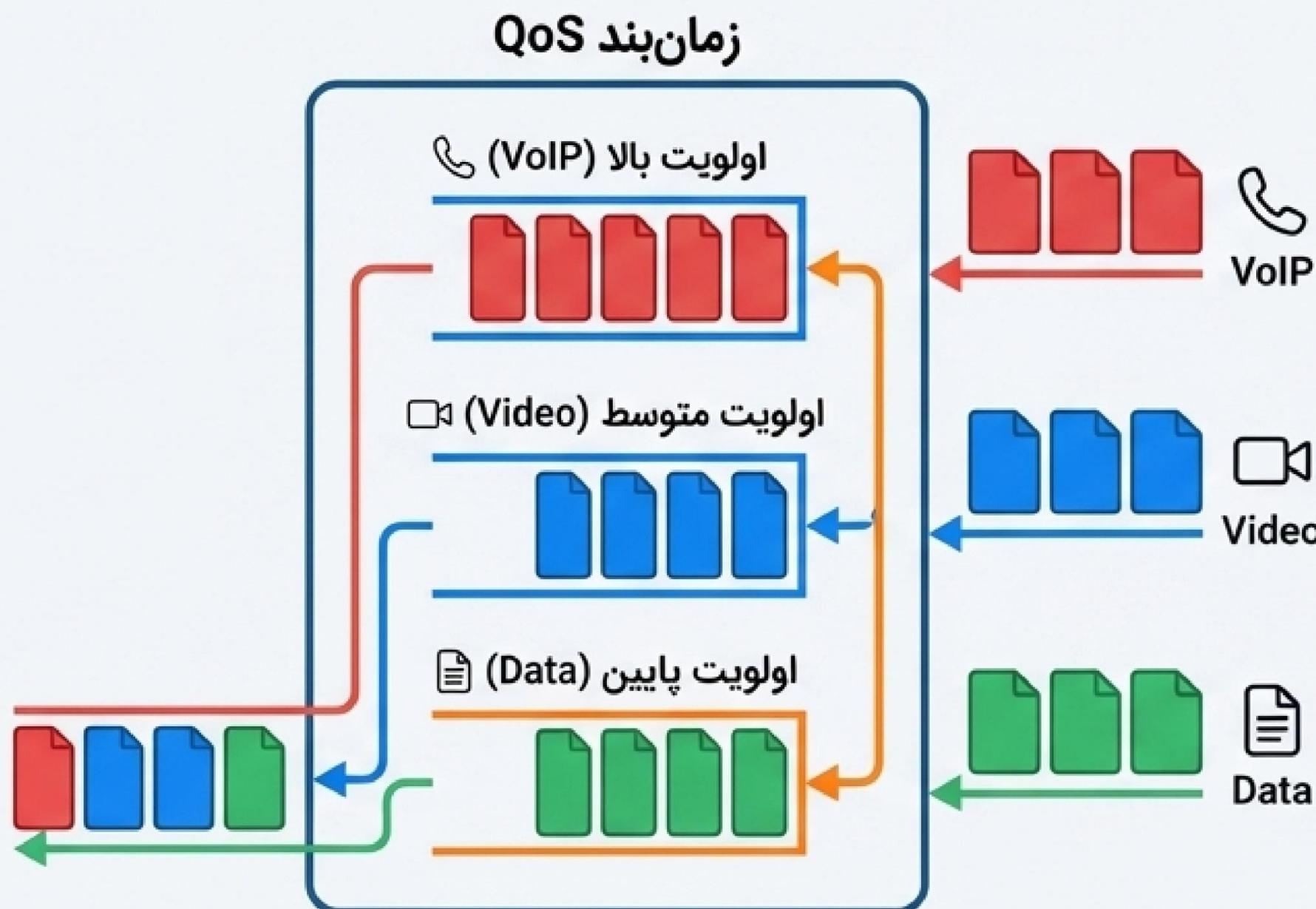
پروتکل‌هایی مانند VXLAN و GRE فریم‌های لایه ۲ را درون بسته‌های لایه ۳ (IP) کپسوله می‌کنند.

## مزیت

این تکنیک اجازه می‌دهد تا سگمنت‌های لایه ۲ مجازی و ایزوله، بین هاستهایی که در شبکه‌های فیزیکی لایه ۳ متفاوتی قرار دارند، ایجاد شود.

این قابلیت برای پلتفرم‌های ابری مانند Kubernetes و OpenStack حیاتی است.

# کیفیت سرویس (QoS) و شکل دهی ترافیک



## Rate Limiting (Policing)

تعیین حد اکثر نرخ ترافیک برای یک پورت یا صف. ترافیک مازاد حذف می‌شود.

## Traffic Shaping (Queuing)

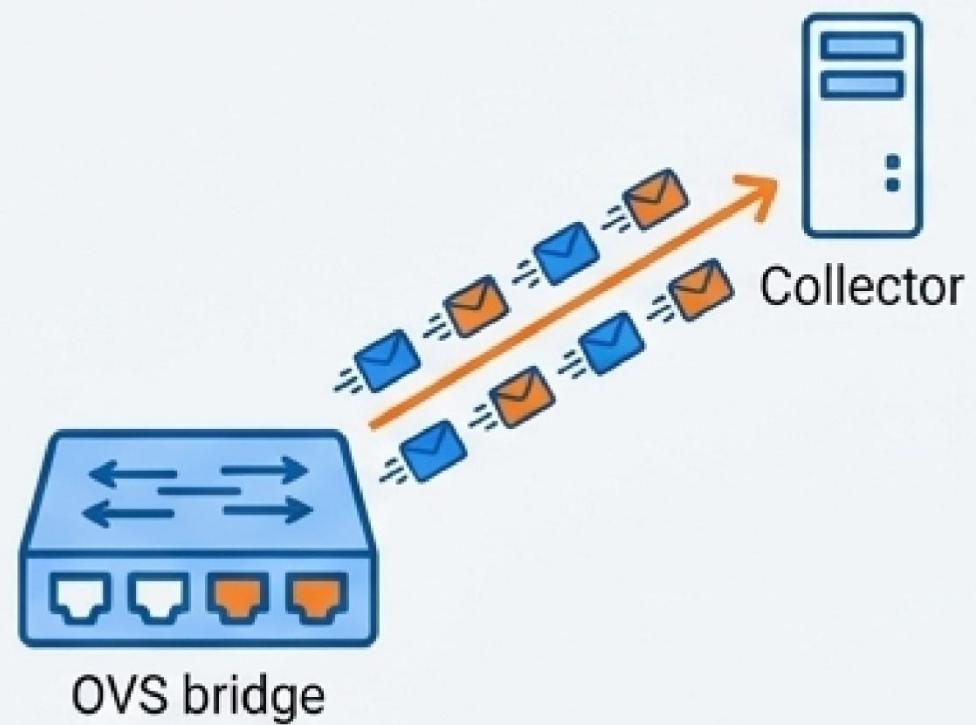
مدیریت نرخ ترافیک خروجی با استفاده از صفحه‌های بافر. ترافیک مازاد به جای حذف، با تاخیر ارسال می‌شود.

## پیکربندی

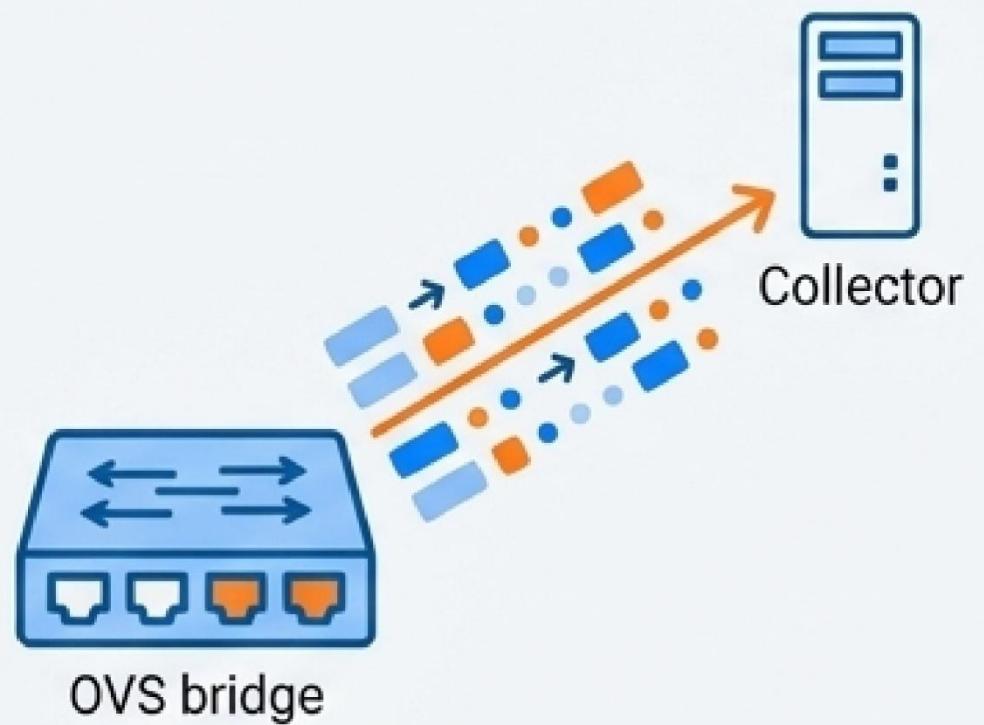
با استفاده از `ovs-vsctl` می‌توان برای هر پورت، یک سیاست QoS و صفحه‌های مختلف با ویژگی‌هایی مانند `max-rate` و `min-rate` تعریف کرد.

# مانیتورینگ و دیدپذیری شبکه

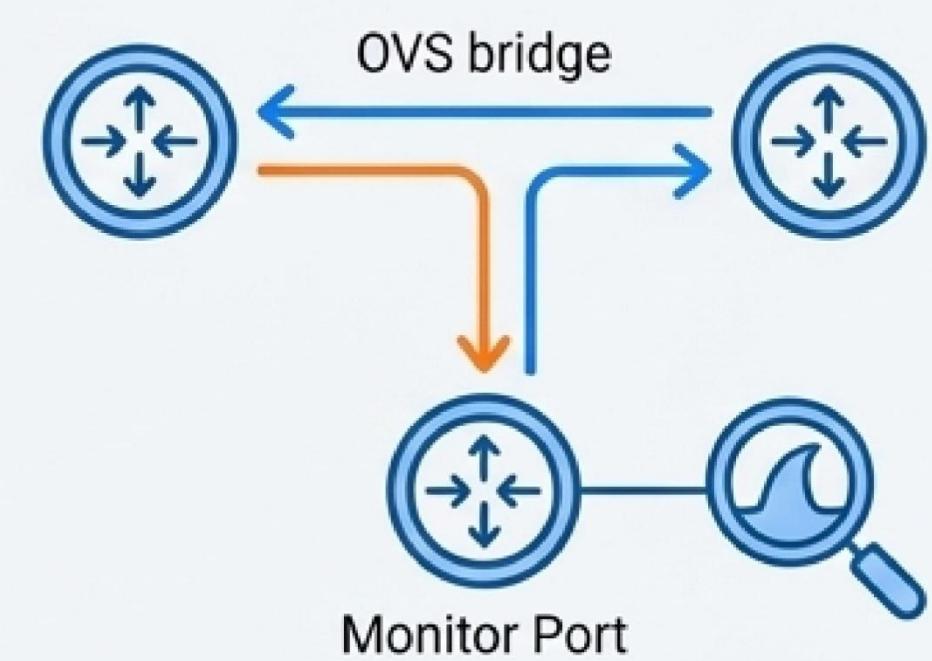
## sFlow



## NetFlow/IPFIX



## Port Mirroring (SPAN/RSPAN)



ارسال نمونه‌هایی از بسته‌ها (samples) و شمارنده‌های اینترفیس (packet Counter) به یک Collector.

ارسال متادیتای مربوط به جریان‌های ترافیکی (IP‌ها، پورت‌ها، بایت‌ها) به یک Collector برای تحلیل‌های آماری و امنیتی.

کپی کردن ترافیک یک یا چند پورت به یک پورت دیگر برای تحلیل با ابزارهایی مانند Wireshark.

# مقایسه فنی: Linux Bridge در برابر Open vSwitch

Open vSwitch	Linux Bridge	قابلیت
سوییچینگ پیشرفته لایه ۲ و ۳	سوییچینگ ساده لایه ۲	عملکرد اصلی
(VXLAN, GRE, Geneve) ✓ پشتیبانی گسترده	محدود (فقط VXLAN با دستورات ip)	تونل زنی
✓ قابلیت های غنی Shaping و Policing	بسیار محدود	کیفیت سرویس (QoS)
✓ پشتیبانی کامل و بومی	✗ پشتیبانی نمی کند	کنترل SDN/OpenFlow
ovs-vsctl / ovs-ofctl / OVSDDB ✓	brctl / ip	ابزار مدیریت
✓ محیط های ابری، NFV و شبکه های پیچیده	سناریوهای ساده و ایستا	کاربرد
✓ شتاب دهنی با DPDK در فضای کاربر	محدود به کرنل	شتاب دهنی

# نصب در لینوکس: پکیج‌ها و سرویس‌ها

نصب OVS از طریق مدیر بسته‌های استاندارد توزیع‌های مختلف لینوکس امکان‌پذیر است.

RHEL/CentOS/Fedora در  

```
sudo dnf install openvswitch
```

Debian/Ubuntu در  

```
sudo apt update  
sudo apt install openvswitch-switch  
openvswitch-common
```

## فعال‌سازی و اجرای سرویس

```
sudo systemctl enable --now openvswitch.service
```

# دستورات کلیدی: کار با `ovs-vsctl`

دستورات زیر برای عملیات پایه‌ای ساخت پل و افزودن پورت استفاده می‌شوند:

# نمایش وضعیت کل OVS

```
ovs-vsctl show
```

# ایجاد یک Bridge جدید به نام

```
ovs-vsctl add-br br-int
```

# Bridge اینترفیس فیزیکی به

```
ovs-vsctl add-port br-int eth0
```

# Bridge یک اینترفیس مجازی به

```
ovs-vsctl add-port br-int tap0
```

# مشاهده وضعیت جدید

```
ovs-vsctl show
```

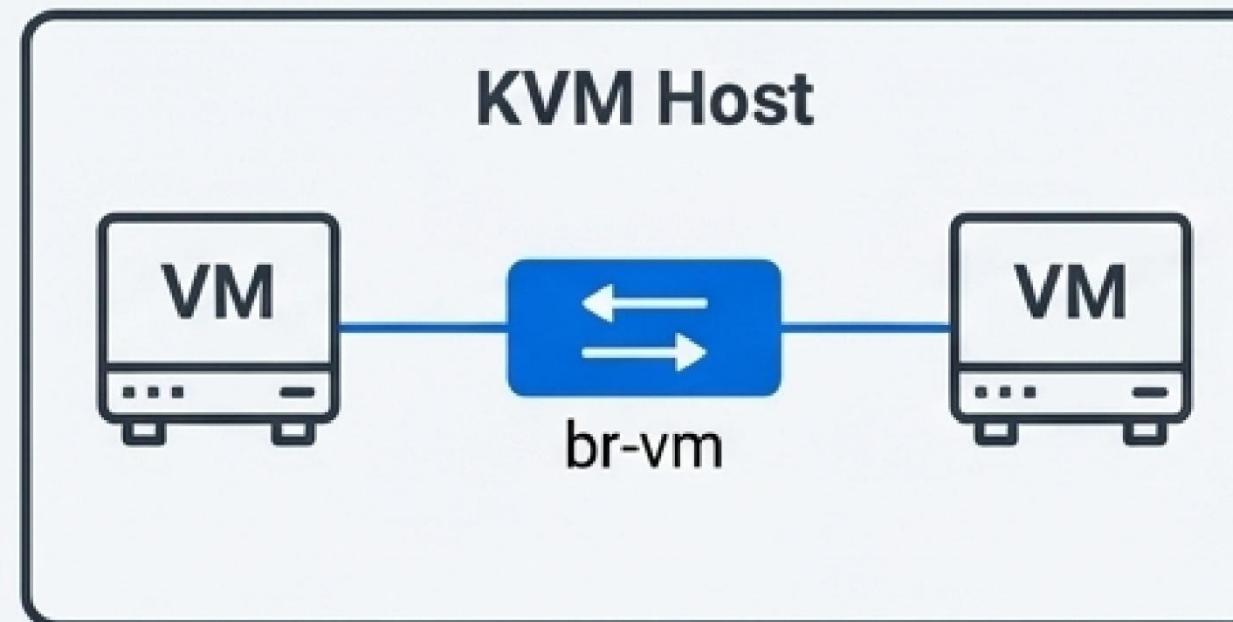
# Bridge از حذف پورت

```
ovs-vsctl del-port br-int tap0
```

# Bridge کامل حذف

```
ovs-vsctl del-br br-int
```

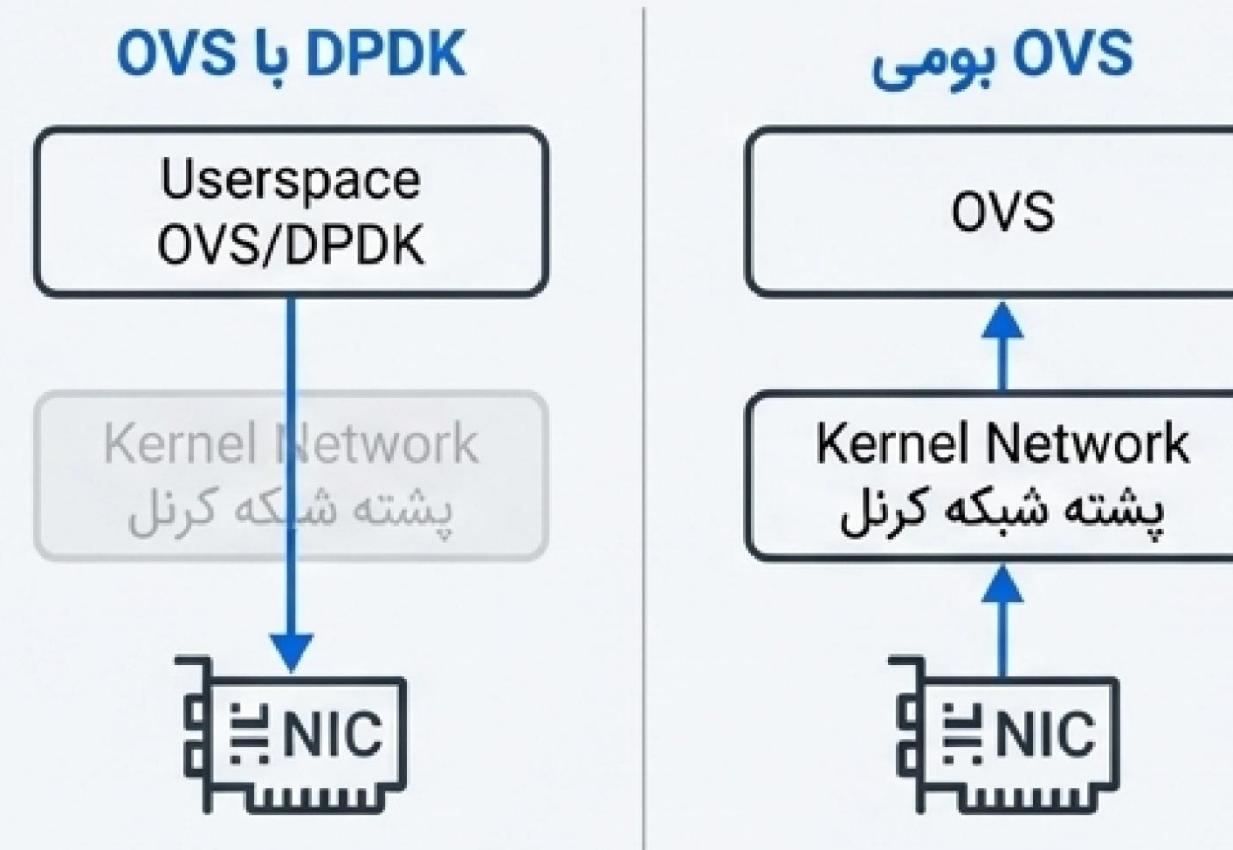
# سناریو عملی: اتصال دو ماشین مجازی و شتابدهی با DPDK



## بخش ۱: اتصال ماشین‌های مجازی (KVM)

۱. یک OVS bridge ایجاد کنید:
۲. دو ماشین مجازی KVM را با استفاده از `virt-install` راهاندازی کرده و کارت شبکه آنها را به `br-vm` متصل کنید:

```
--network=bridge:br-vm,virtualport_type=openvswitch
```



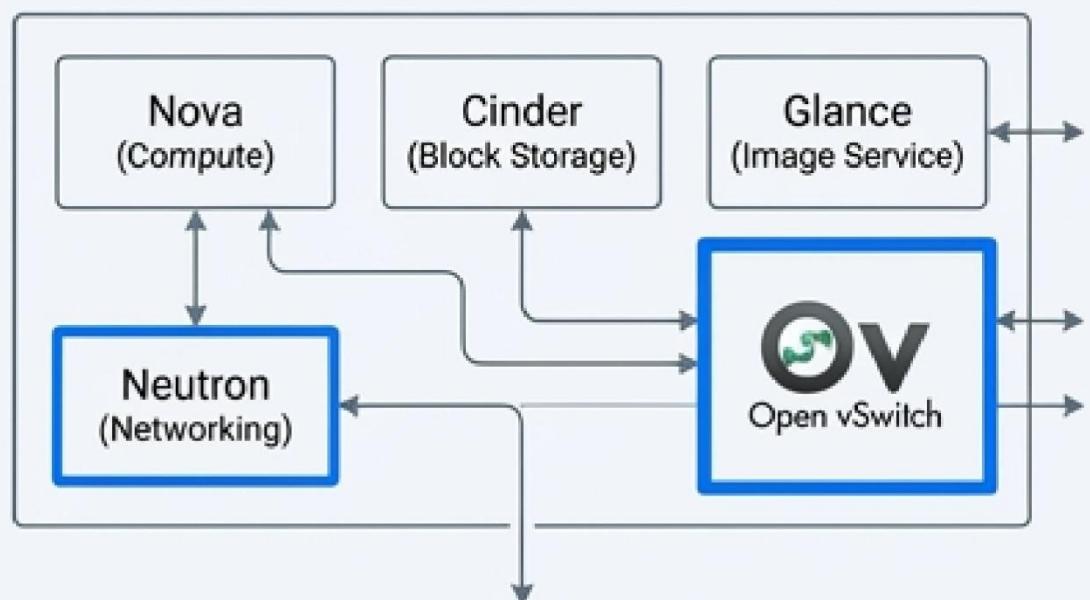
## بخش ۲: شتابدهی با (Data Plane Development Kit) DPDK

DPDK چیست؟: مجموعه‌ای از کتابخانه‌ها برای شتابدهی پردازش بسته‌ها در فضای کاربر (userspace) است که با دور زدن کامل پشتۀ شبکه کرنل لینوکس (kernel bypass)، به عملکرد بسیار بالایی دست می‌یابد.

افزایش عملکرد: نزدیک به ۱۲ برابر سریع‌تر در سناریوهای physical-to-physical و بیش از ۷ برابر سریع‌تر در سناریوهای physical-to-virtual.

# نتیجه‌گیری: OVS به عنوان ستون فقرات شبکه در OpenStack

## کاربرد در OpenStack



تمام شبکه‌های مجازی self-service و provider که برای ایزوله‌سازی ترافیک مستاجرین (tenants) در محیط‌های ابری بزرگ ضروری هستند، توسط OVS پیاده‌سازی می‌شوند.

## جمع‌بندی مزایای OVS

**برنامه‌پذیری و اتوماسیون:** کنترل کامل از طریق .OVSDB و OpenFlow ✓

**عملکرد بالا:** به خصوص با شتابدهی DPDK ✓

**قابلیت‌های غنی:** پشتیبانی از تونل‌زنی، QoS، مانیتورینگ پیشرفته. ✓

**اکوسیستم باز:** یک پروژه متن-باز با پشتیبانی گستردگی جامعه. ✓