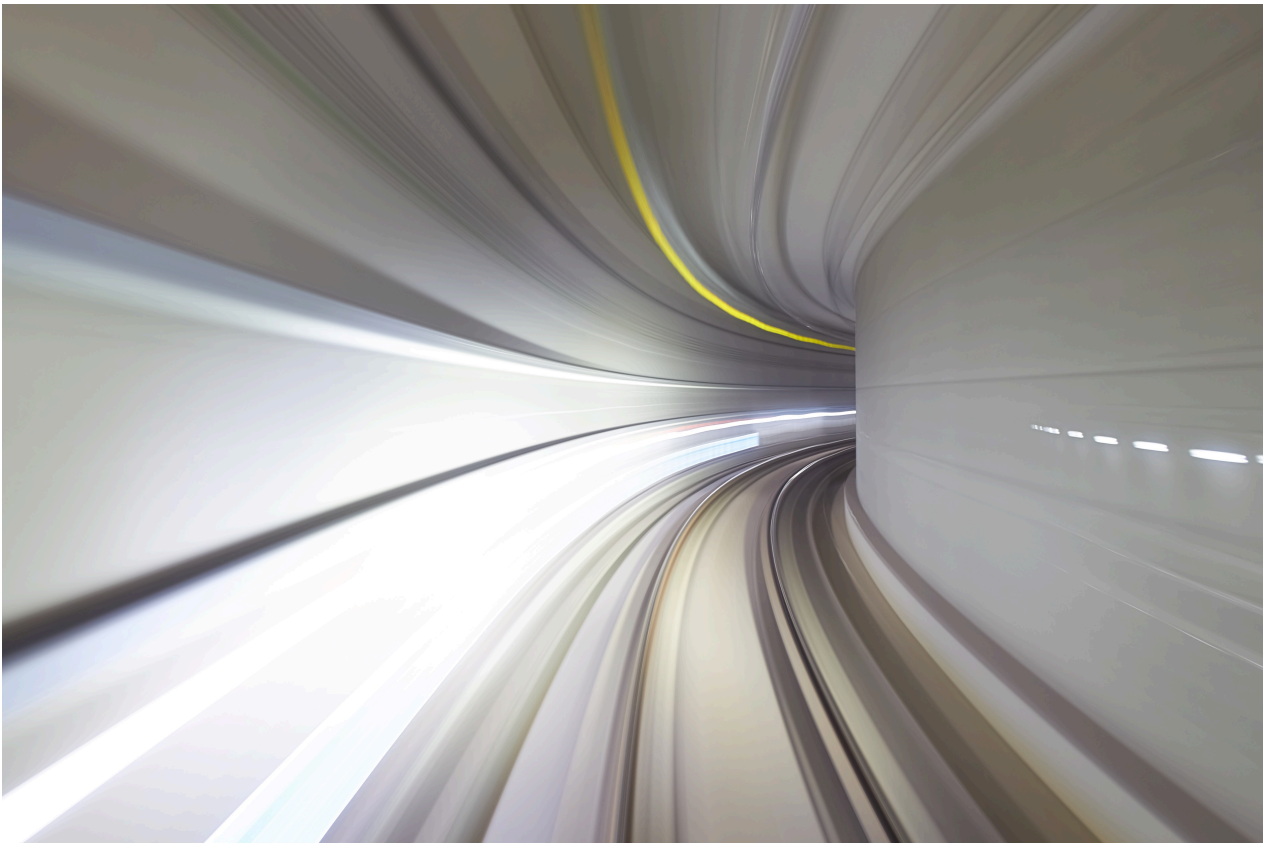


Linux 虚拟网卡：隧道

11月 3, 2019 11:00 · 2046 字 · 5 分钟阅读

[LINUX](#) [网络](#)



Linux 支持多种隧道，但是新用户可能搞不清它们之间的区别，无法因地制宜。本文简要介绍 Linux 内核中常用的隧道，没有代码。通过 `IPROUTE2` 命令 来查看隧道网卡的列表和某个隧道配置的帮助文档。 `ip link help`

常用的隧道：

- IPIP 隧道
- SIT 隧道
- IP6TNL 隧道
- GRE 和 GRETap
- 6GRE 和 IP6GRETap

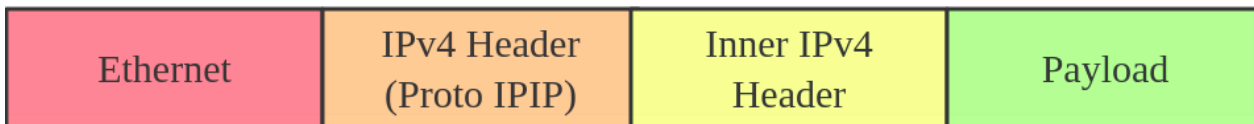


击

- GUE
- 日内瓦
- ERSPAN 和 IP6ERSPAN

IPIP 隧道

正如其名，IP over IP 的隧道，[RFC 2003](#)。



通常用于通过 IPv4 公网连接两个内部的 IPv4 子网。开销最小但是只能传输 IPv4 单播流量，也就是说 **无法** 通过 IPIP 隧道多播。

IPIP 隧道同时支持 IP over IP 和 MPLS over IP。

注意：当 模块被加载时，或者 IPIP 设备首次被创建，Linux 内核将在每个命名空间中创建一个属性 `且` 的默认设备。当接收到 IPIP 协议的数据包时，如果不能找到匹配的设备，内核默认将它们发送至 `。 ipip local=any remote=any tunl0 tunl0`

创建 IPIP 隧道：

On Server A:

```
# ip link add name ipip0 type ipip local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR
# ip link set ipip0 up
# ip addr add INTERNAL_IPV4_ADDR/24 dev ipip0
Add a remote internal subnet route if the endpoints don't belong to the same subnet
# ip route add REMOTE_INTERNAL_SUBNET/24 dev ipip0
```

On Server B:

```
# ip link add name ipip0 type ipip local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR
# ip link set ipip0 up
# ip addr add INTERNAL_IPV4_ADDR/24 dev ipip0
# ip route add REMOTE_INTERNAL_SUBNET/24 dev ipip0
```

注意：根据实际环境替换 `LOCAL_IPv4_ADDR`、`REMOTE_IPv4_ADDR`、`INTERNAL_IPV4_ADDR`、`REMOTE_INTERNAL_SUBNET`。

SIT 隧道

SIT 即简单网络传输（Simple Internet Transition）。主要为了互连全球 IPv4 互联网中的隔离 IPv6 网络。

最初，它只有 IPv6 over IPv4 隧道模式。经过多年开发终于支持了几种不同的模式（和 IPIP 隧道一样）、和。其中 模式用来同时接收 IPv4 和 IPv6 流量，在开发时很有用。SIT 隧道也支持 ISATA，这里有个用例。 `ipip ip6ip mplsip any any`

SIT 隧道头：



当 模块被加载时，Linux 内核将创建一个名为 的默认设备。 `sit sit0`

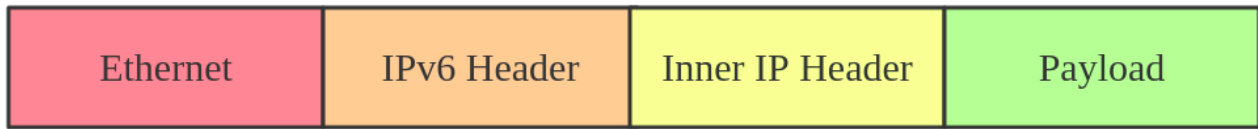
创建一个 SIT 隧道：

```
On Server A:
# ip link add name sit1 type sit local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR mode any
# ip link set sit1 up
# ip addr add INTERNAL_IPv4_ADDR/24 dev sit1
```

然后在远端再来一把。

IP6TNL 隧道

ip6tnl 是 IPv4/IPv6 over IPv6 的隧道，看起来有点像 SIT 隧道的 IPv6 版本。



IP6TNL 支持 、。模式是 IPv4 over IPv6，是 IPv6 over IPv6，模式同时支持 IPv4/IPv6 over IPv6。 `ip6ip6 ipip6 any ipip6 ip6ip6 any`

当 模块被加载时，Linux 内核会创建名为 的默认设备。 `ip6tnl ip6tnl0`

创建一个 IP6TNL 隧道：

```
# ip link add name ipip6 type ip6tnl local LOCAL_IPv6_ADDR remote REMOTE_IPv6_ADDR mode any
```



GRE 和 GRE TAP

通用路由封装（Generic Routing Encapsulation），也称为 GRE，RFC 2784。

GRE 隧道在内外的 IP 头之间添加了一个额外的 GRE 头。理论上，GRE 可以封装任何三层协议，而 IPIP 只能封装 IP 包。



注意，可以通过 GRE 隧道传输多播流量和 IPv6。

当 模块被加载时，Linux 内核将创建名为 的默认设备。 gre gre0

创建 GRE 隧道：

```
# ip link add name gre1 type gre local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR [seq] key KEY
```



当 GRE 隧道在 OSI 模型三层工作，GRETAP 在 OSI 模型二层工作，意味着在内部头中有个 Ethernet 头。



创建 GRETAP 隧道：

```
# ip link add name gretap1 type gretap local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR
```

IP6GRE 和 IP6GRETAP

IP6GRE 等价于 IPv6 版 GRE，这让我们可以通过 IPv6 封装任何第三层的协议。



IP6GRETAP，就像 GRETAP，在内部头中有一个 Ethernet 头：



创建 GRE 隧道：

```
# ip link add name gre1 type ip6gre local LOCAL_IPv6_ADDR remote REMOTE_IPv6_ADDR
# ip link add name gretap1 type ip6gretap local LOCAL_IPv6_ADDR remote REMOTE_IPv6_ADDR
```



FOU

隧道可以在网络栈的多个层级上。IPIP、SIT、GRE 隧道都在 IP 层，而 FOU（foo over UDP）是传输层的隧道。

使用 UDP 隧道的优势在于 UDP 可以与现有的硬件基础设置一起工作，像 NIC 中的 RSS，交换机中的 ECMP。开发者的补丁集显示 SIT 和 IPIP 协议的性能显著提高。

目前，FOU 隧道支持基于协议 IPIP，SIT 和 GRE 封装。



创建 FOU 隧道：

```
# ip fou add port 5555 ipproto 4
# ip link add name tun1 type ipip remote 192.168.1.1 local 192.168.1.2 ttl 225 encap fou enca
```

第一行命令设置 FOU 从 5555 端口接收 IPIP 数据包；GRE 的话就要设置 。第二行命令创建了一个新的用于 FOU 封装的 IPIP 虚拟网卡（tun1），目标端口是 5555。ipproto 47

注意：Red Hat Enterprise Linux 不支持 FOU。

GUE

Generic UDP Encapsulation (GUE) 是另一种 UDP 隧道。GUE 和 FOU 不同之处在于有着自己的封装头，包含了协议信息和一些其他数据。

目前，GUE 隧道支持内部封装 IPIP、SIT、GRE。



创建 GUE 隧道：

```
# ip fou add port 5555 gue
# ip link add name tun1 type ipip remote 192.168.1.1 local 192.168.1.2 ttl 225 encap gue enca
```

创建一个 CUE 从 5555 端口接收 IPIP 数据包，并为 GUE 封装配置一个 IPIP 隧道。

注意：Red Hat Enterprise Linux 不支持 GUE。

GENEVE

Generic Network Virtualization Encapsulation (GENEVE) 支持 VXLAN、NVGRE 和 STT，旨在克服它们的局限性。很多人认为 GENEVE 最终能够完全代替这些早期的格式。



看起来与 VXLAN 非常相似。主要的区别在于 GENEVE 头是弹性的。通过扩展新的 Type-Length-Value (TLV) 字段，可以轻松添加新功能。

Open Virtual Network (OVN) 使用 GENEVE 作为默认的封装。

创建 GENEVE 隧道：

```
# ip link add name geneve0 type geneve id VNI remote REMOTE_IPv4_ADDR
```

ERSPAN 与 IP6ERSPAN

Encapsulated Remote Switched Port Analyzer (ERSPAN) 使用 GRE 将基础端口镜像功能从二层扩展到三层，允许通过可路由的 IP 网络发送镜像流量。



ERSPAN 隧道允许 Linux 主机充当 ERSPAN 流量源，并将 ERSPAN 镜像流量发送到远程主机或 ERSPAN 目标，后者接收并解析从 Cisco 或其他有 ERSPAN 功能的交换机生成的 ERSPAN 数据包。此设置可以用于分析，诊断和探测恶意流量。

Linux 当前支持两个 ERSPAN 版本的绝大多数功能。

创建 ERSPAN 隧道：

```
# ip link add dev erspan1 type erspan local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR seq key K
or
# ip link add dev erspan1 type erspan local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR seq key K

Add tc filter to monitor traffic
# tc qdisc add dev MONITOR_DEV handle ffff: ingress
# tc filter add dev MONITOR_DEV parent ffff: matchall skip_hw action mirred egress mirror dev
```



总结

隧道/链路类型	外部标头	Encapsulate 标头	内部标题
IPIP （国际知识产权）	IPv4 协议	没有	IPv4 协议
坐	IPv4 协议	没有	IPv4/IPv6 协议
IP6TNL 防护等级	IPv6 协议	没有	IPv4/IPv6 协议
VTI	IPv4 协议	IPsec	IPv4 协议
VTI6	IPv6 协议	IPsec	IPv6 协议
螺 柱	IPv4 协议	螺 柱	IPv4/IPv6 协议
格雷塔普	IPv4 协议	螺 柱	以太 + IPv4/IPv6
IP6GRE 防护等级	IPv6 协议	螺 柱	IPv4/IPv6 协议
IP6格雷塔普	IPv6 协议	螺 柱	以太 + IPv4/IPv6
缶	IPv4/IPv6 协议	UDP 协议	IPv4/IPv6/GRE
格	IPv4/IPv6 协议	UDP + GUE	IPv4/IPv6/GRE
日内瓦	IPv4/IPv6 协议	UDP + 日内瓦	以太 + IPv4/IPv6
Erspar	IPv4 协议	GRE + ERSPAN	IPv4/IPv6 协议
IP6erspar	IPv6 协议	GRE + ERSPAN	IPv4/IPv6 协议

查看更多

- [Uber 是如何优化 LLM 训练的](#)

11月 23 2024
- [fake-dcgm-exporter （伪造 DCGM 导出器）](#)

11月 19 2024
- [Golang 别名](#)

10月 5， 2024
- [农场](#)

9月 10， 2024
- [Kubernetes Pod 网络初始化原理](#)

7月 11， 2024
- [Uber 如何支持万亿级索引](#)

7月 3， 2024
- [KubeVirt Domain Informer 原理](#)

6月 27， 2024
- [中央处理器 ID](#)


6月 20， 2024

自定义 JSON 反序列化搞定 KubeVirt 向前兼容

6月 16， 2024

eBPF 抓取 TLS 明文

6月 14， 2024

 由 HF 提供♥

