



An introduction to Linux virtual interfaces: Tunnels

May 17, 2019

Hangbin Liu

Related topics: [Linux](#)

Related products: [Red Hat Enterprise Linux](#)

Share:    

 [Table of contents:](#)



[Linux](#) has supported many kinds of tunnels, but new users may be confused by their differences and unsure which one is best suited for a given use case. In this article, I will give a brief introduction for commonly used tunnel interfaces in the Linux kernel. There is no code analysis, only a brief introduction to the interfaces and their usage on Linux. Anyone with a network background might be interested in this information. A list of tunnel interfaces, as well as help on specific tunnel configuration, can be obtained by issuing the `iproute2` command `ip link help`.

This post covers the following frequently used interfaces:

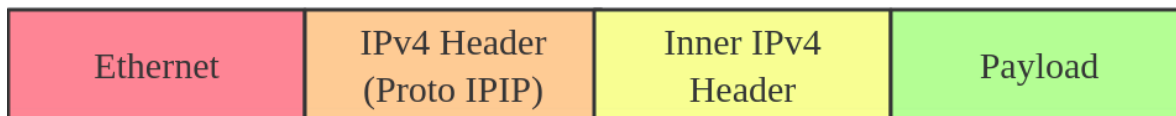
- [IPIP Tunnel](#)
- [SIT Tunnel](#)
- [ip6tnl Tunnel](#)
- [VTI and VTI6](#)
- [GRE and GRETap](#)
- [IP6GRE and IP6GRETap](#)
- [FOU](#)
- [GUE](#)
- [GENEVE](#)

- [ERSPAN and IP6ERSPAN](#)

After reading this article, you will know what these interfaces are, the differences between them, when to use them, and how to create them.

IPIP Tunnel

IPIP tunnel, just as the name suggests, is an IP over IP tunnel, defined in [RFC 2003](#). The IPIP tunnel header looks like:



It's typically used to connect two internal IPv4 subnets through public IPv4 internet. It has the lowest overhead but can only transmit IPv4 unicast traffic. That means you **cannot** send multicast via IPIP tunnel.

IPIP tunnel supports both IP over IP and [MPLS](#) over IP.

Note: When the `ipip` module is loaded, or an IPIP device is created for the first time, the Linux kernel will create a `tunl0` default device in each namespace, with attributes `local=any` and `remote=any`. When receiving IPIP protocol packets, the kernel will forward them to `tunl0` as a fallback device if it can't find another device whose local/remote attributes match their source or destination address more closely.

Here is how to create an IPIP tunnel:

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 network
# 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
```

 [Copy snippet](#)

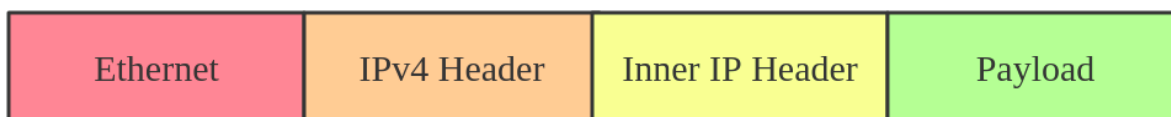
Note: Please replace `LOCAL_IPV4_ADDR`, `REMOTE_IPV4_ADDR`, `INTERNAL_IPV4_ADDR`, `REMOTE_INTERNAL_SUBNET` to the addresses based on your testing environment. The same with following example configs.

SIT Tunnel

SIT stands for Simple Internet Transition. The main purpose is to interconnect isolated IPv6 networks, located in global IPv4 internet.

Initially, it only had an IPv6 over IPv4 tunneling mode. After years of development, however, it acquired support for several different modes, such as `ipip` (the same with IPIP tunnel), `ip6ip`, `mplsip`, and `any`. Mode `any` is used to accept both IP and IPv6 traffic, which may prove useful in some deployments. SIT tunnel also supports [ISATA](#), and here is a [usage example](#).

The SIT tunnel header looks like:



When the `sit` module is loaded, the Linux kernel will create a default device, named `sit0`.

Here is how to create a SIT tunnel:

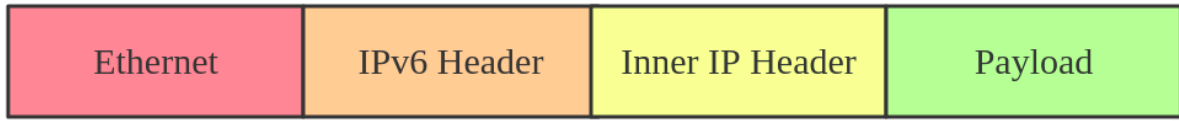
```
On Server A:
# ip link add name sit1 type sit local LOCAL_IPV4_ADDR remote REMOTE_IPV4
# ip link set sit1 up
# ip addr add INTERNAL_IPV4_ADDR/24 dev sit1
```

 [Copy snippet](#)

Then, perform the same steps on the remote side.

ip6tnl Tunnel

ip6tnl is an IPv4/IPv6 over IPv6 tunnel interface, which looks like an IPv6 version of the SIT tunnel. The tunnel header looks like:



ip6tnl supports modes `ip6ip6`, `ipip6`, `any`. Mode `ipip6` is IPv4 over IPv6, and mode `ip6ip6` is IPv6 over IPv6, and mode `any` supports both IPv4/IPv6 over IPv6.

When the `ip6tnl` module is loaded, the Linux kernel will create a default device, named `ip6tnl0`.

Here is how to create an ip6tnl tunnel:

```
# ip link add name ipip6 type ip6tnl local LOCAL_IPv6_ADDR remote REMOTE_
```

 [Copy snippet](#)

VTI and VTI6

Virtual Tunnel Interface (VTI) on Linux is similar to Cisco's VTI and Juniper's implementation of secure tunnel (st.xx).

This particular tunneling driver implements IP encapsulations, which can be used with xfrm to give the notion of a secure tunnel and then use kernel routing on top.

In general, VTI tunnels operate in almost the same way as ipip or sit tunnels, except that they add a fwmark and IPsec encapsulation/decapsulation.

VTI6 is the IPv6 equivalent of VTI.

Here is how to create a VTI tunnel:

```
# ip link add name vtil type vti key VTI_KEY local LOCAL_IPv4_ADDR remote
# ip link set vtil up
# ip addr add LOCAL_VIRTUAL_ADDR/24 dev vtil
```

```
# ip xfrm state add src LOCAL_IPv4_ADDR dst REMOTE_IPv4_ADDR spi SPI PROT
# ip xfrm state add src REMOTE_IPv4_ADDR dst LOCAL_IPv4_ADDR spi SPI PROT
# ip xfrm policy add dir in tmpl src REMOTE_IPv4_ADDR dst LOCAL_IPv4_ADDR
# ip xfrm policy add dir out tmpl src LOCAL_IPv4_ADDR dst REMOTE_IPv4_ADDR
```

 Copy snippet

GRE and GRETAP

Generic Routing Encapsulation, also known as GRE, is defined in [RFC 2784](#)

GRE tunneling adds an additional GRE header between the inside and outside IP headers. In theory, GRE could encapsulate any Layer 3 protocol with a valid Ethernet type, unlike IPIP, which can only encapsulate IP. The GRE header looks like:



Note that you can transport multicast traffic and IPv6 through a GRE tunnel.

When the `gre` module is loaded, the Linux kernel will create a default device, named `gre0`.

Here is how to create a GRE tunnel:

```
# ip link add name gre1 type gre local LOCAL_IPv4_ADDR remote REMOTE_IPv4_ADDR
```

 Copy snippet

While GRE tunnels operate at OSI Layer 3, GRETAP works at OSI Layer 2, which means there is an Ethernet header in the inner header.



Here is how to create a GRETAP tunnel:

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

 [Copy snippet](#)

IP6GRE and IP6GRETAP

IP6GRE is the IPv6 equivalent of GRE, which allows us to encapsulate any Layer 3 protocol over IPv6. The tunnel header looks like:



IP6GRETAP, just like GRETAP, has an Ethernet header in the inner header:



Here is how to create a GRE tunnel:

```
# 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
```

 [Copy snippet](#)

FOU

Tunneling can happen at multiple levels in the networking stack. IPIP, SIT, GRE tunnels are at the IP level, while FOU (foo over UDP) is UDP-level tunneling.

There are some advantages of using UDP tunneling as UDP works with existing HW infrastructure, like [RSS](#) in NICs, [ECMP](#) in switches, and checksum offload. The developer's [patch set](#) shows significant performance increases for the SIT and IPIP protocols.

Currently, the FOU tunnel supports encapsulation protocol based on IPIP, SIT, GRE. An example FOU header looks like:



Here is how to create a FOU tunnel:

```
# ip fou add port 5555 ipproto 4
# ip link add name tunl type ipip remote 192.168.1.1 local 192.168.1.2 tt
```

 [Copy snippet](#)

The first command configured a FOU receive port for IPIP bound to 5555; for GRE, you need to set `ipproto 47`. The second command set up a new IPIP virtual interface (tun1) configured for FOU encapsulation, with dest port 5555.

Note: FOU is not supported in [Red Hat Enterprise Linux](#).

GUE

[Generic UDP Encapsulation](#) (GUE) is another kind of UDP tunneling. The difference between FOU and GUE is that GUE has its own encapsulation header, which contains the protocol info and other data.

Currently, GUE tunnel supports inner IPIP, SIT, GRE encapsulation. An example GUE header looks like:



Here is how to create a GUE tunnel:

```
# ip fou add port 5555 gue
# ip link add name tunl type ipip remote 192.168.1.1 local 192.168.1.2 tt
```

 [Copy snippet](#)

This will set up a GUE receive port for IPIP bound to 5555, and an IPIP tunnel configured for GUE encapsulation.

Note: GUE is not supported in Red Hat Enterprise Linux.

GENEVE


Generic Network Virtualization Encapsulation (GENEVE) supports all of the capabilities of VXLAN, NVGRE, and STT and was designed to overcome their perceived limitations. Many believe GENEVE could eventually replace these earlier formats entirely. The tunnel header looks like:



which looks very similar to [VXLAN](#). The main difference is that the GENEVE header is flexible. It's very easy to add new features by extending the header with a new Type-Length-Value (TLV) field. For more details, you can see the latest [geneve ietf draft](#) or refer to this [What is GENEVE?](#) article.

[Open Virtual Network \(OVN\)](#) uses GENEVE as default encapsulation. Here is how to create a GENEVE tunnel:

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

 [Copy snippet](#)

ERSPAN and IP6ERSPAN

Encapsulated Remote Switched Port Analyzer (ERSPAN) uses GRE encapsulation to extend the basic port mirroring capability from Layer 2 to Layer 3, which allows the mirrored traffic to be sent through a routable IP network. The ERSPAN header looks like:




The ERSPAN tunnel allows a Linux host to act as an ERSPAN traffic source and send the ERSPAN mirrored traffic to either a remote host or to an ERSPAN destination, which receives and parses the ERSPAN packets generated from Cisco or other ERSPAN-capable switches. This setup could be used to analyze, diagnose, and detect malicious traffic.

Linux currently supports most features of two ERSPAN versions: v1 (type II) and v2 (type III).

Here is how to create an ERSPAN tunnel:

```
# ip link add dev erspan1 type erspan local LOCAL_IPv4_ADDR remote REMOTE_ADDR
or
# ip link add dev erspan1 type erspan local LOCAL_IPv4_ADDR remote REMOTE_ADDR

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 mirror
```

 Copy snippet

Summary

Here is a summary of all the tunnels we introduced.

Tunnel/Link Type	Outer Header	Encapsulate Header	Inner Header
ipip	IPv4	None	IPv4
sit	IPv4	None	IPv4/IPv6
ip6tnl	IPv6	None	IPv4/IPv6
vti	IPv4	IPsec	IPv4
vti6	IPv6	IPsec	IPv6
gre	IPv4	GRE	IPv4/IPv6

gretap	IPv4	GRE	Ether + IPv4/IPv6
ip6gre	IPv6	GRE	IPv4/IPv6
ip6gretap	IPv6	GRE	Ether + IPv4/IPv6
fou	IPv4/IPv6	UDP	IPv4/IPv6/GRE
gue	IPv4/IPv6	UDP + GUE	IPv4/IPv6/GRE
geneve	IPv4/IPv6	UDP + Geneve	Ether + IPv4/IPv6
erspan	IPv4	GRE + ERSPAN	IPv4/IPv6
ip6erspan	IPv6	GRE + ERSPAN	IPv4/IPv6

Note: All configurations in this tutorial are volatile and won't survive a server reboot. If you want to make the configuration persistent across reboots, consider using a networking configuration daemon, such as [NetworkManager](#), or distribution-specific mechanisms.

If you want to learn more, read this article: [Introduction to Linux interfaces for virtual networking](#)

Last updated: August 14, 2023

Related Posts

[Introduction to Linux interfaces for virtual networking](#)

[An introduction to Linux bridging commands and features](#)

[Get started with XDP](#)

Recent Posts

[Step-by-step guide to configuring alerts in Cluster Observability Operator](#)

[Introducing Builds for OpenShift 1.2](#)

[Our top Linux articles of 2024](#)

[Monitoring OpenShift Virtualization at scale with Red Hat Advanced Cluster Management: Part 2](#)

[Brief overview of Cluster Observability Operator](#)

What's up next?



Download the Advanced Linux Cheat Sheet for a collection of Linux commands and executables for developers who are using the Linux operating system in advanced programming scenarios.

[Get the cheat sheet](#) →



Products

Build

Quicklinks

Communicate

RED HAT DEVELOPER

Build here. Go anywhere.

We serve the builders. The problem solvers who create careers with code.

Join us if you're a developer, software engineer, web designer, front-end designer, UX designer, computer scientist, architect, tester, product manager, project manager or team lead.

[Sign me up →](#)



[About Red Hat](#)

[Jobs](#)

[Events](#)

[Locations](#)

[Contact Red Hat](#)

[Red Hat Blog](#)

[Diversity, equity, and inclusion](#)

[Cool Stuff Store](#)

[Red Hat Summit](#)

© 2024 Red Hat, Inc.

[Privacy statement](#)

[Terms of use](#)

[All policies and guidelines](#)

[Digital accessibility](#)

[Cookie 喜好设置](#)