

# What is the "destination address" for a TAP/TUN device?

Asked 5 years, 5 months ago   Active 1 year, 2 months ago   Viewed 9k times

What is the purpose of the "destination address" for a TAP/TUN device?

15 [Pytun](#) lets you easily set parameters of a tap/tun device:

```
tun = TapTunDevice(name='mytun')
tun.addr = '10.66.66.1'
tun.dstaddr = '10.66.66.2'
tun.netmask = '255.255.255.0'
tun.up()
```

Doing this will result in a device configured as such:

```
$ ifconfig mytun
mytun: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.66.66.1 netmask 255.255.255.0 destination 10.66.66.2
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500
    (UNSPEC)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

I understand that the system now has a virtual interface with IP 10.66.66.1. And it's presumable that in this scenario, the TUN device would be "connected" to a (e.g. VPN gateway) device whose IP address is 10.66.66.2.

But what purpose specifically, does it serve for the kernel to know that this is a "point-to-point" interface, and the IP address of the destination? Does it impact routing in some way that simply configuring the route table would not achieve?

Setting the `dstaddr` property results in a `SIOCSIFDSTADDR` ioctl.

The [netdevice\(7\)](#) man page simply says:

```
SIOCGIFDSTADDR, SIOCSIFDSTADDR
    Get or set the destination address of a point-to-point device
    using ifr_dstaddr. For compatibility, only AF_INET addresses
    are accepted or returned. Setting the destination address is
    a privileged operation.
```

tun

Share Improve this question Follow

asked Apr 2 '16 at 16:03



[Jonathon Reinhart](#)

119k ● 27 ● 230 ● 306

2 Answers

Active Oldest Votes

## I don't care about all this I want to configure my interface

18

You don't need to set a destination address. If you want to configure `10.66.66.1/24` on the interface, you can do:

```
tun = TapTunDevice(name='mytun')
tun.addr = '10.66.66.1'
tun.netmask = '255.255.255.0'
tun.up()
```

This interface only connects two hosts, so you don't actually need a whole `/24`. You can only say that `10.66.66.1` is connected to `10.66.66.2` (`10.66.66.1 peer 10.66.66.2`):

```
tun = TapTunDevice(name='mytun')
tun.addr = '10.66.66.1'
tun.dstaddr = '10.66.66.2'
tun.netmask = '255.255.255.255'
tun.up()
```

In this setup, the two IP addresses do not need to be in the same range at all.

Alternatively, you could use a `/31`, [RFC3021](#):

```
tun = TapTunDevice(name='mytun')
tun.addr = '10.66.66.2'
tun.dstaddr = '10.66.66.3'
tun.netmask = '255.255.255.254'
tun.up()
```

Notice, how I had to change the IP addresses in order for them to be in the same `/31`.

## What is a POINTOPOINT device?

The `POINTOPOINT` means that on this interface there is no Layer 2 addressing (no MAC address) on this interface:

- no ARP requests (IPv4);
- no NDP requests (IPv6);
- the neighbour table is useless for this interface (`ip neighbour`);
- in routing table entries for this interface the `via` directive is ignored;
- packets on this interface are always send to the same (only) next-hop.

Examples of `POINTOPOINT` devices

- PPP interfaces: There is not Layer 2 address for PPP as this type of interface connects a single host to another host (hence the name "point-to-point protocol")
- TUN interfaces: They are IP only interfaces without a Layer 2.

`POINTOPOINT` means that this is a point-to-point interface (surprise!) which means that there can be only one peer connected at the other-side of the interface: you have one neighbor on this interface and you do not need to use ARP/NDP for mapping IP address to link-layer address (and you do not have link layer address at all).

In contrast, an Ethernet device is not a point-to-point interface because multiple hosts can be directly reachable via this interface. When you send an IP packet to such a device, the network stack has to find a layer 2 identifier (using ARP, NDP) for the intended IP address and send the message to this link-layer address.

Say this, is your routing table (in Ethernet):

```
default via 192.0.2.1 dev eth0 proto static metric 100
192.0.2.0/24 dev eth0 proto kernel scope link src 192.0.2.2 metric 100
```

Multiple hosts can be directly connected to you via the eth0 interface. If you want to send a packet to 198.51.100.1, this route is selected:

```
default via 192.0.2.1 dev eth0 proto static metric 100
```

which means that among all your neighbors on the `eth0` device, you have to send the packet to 192.0.2.1. In order to do that, your network stack has to find the MAC address of 192.0.2.1 by using ARP.

On a `POINTOPOINT` device, there is always only one neighbor so you don't need to do ARP, you only need to send the packet.

TUN and PPP interfaces are `POINTOPOINT` devices. Ethernet, Ethernet TAP devices and Wifi interfaces are not `POINTOPOINT`.

## What is the destination (peer) address?

Usually the IP configuration of an interface is in the form: `192.0.2.1/24`. This means that the IP address of this interface is `192.0.2.1` and that all IP in the `192.0.2.0/24` subnet are *directly* reachable via this interface: this adds a routing rule `192.0.2.0/24 dev tun0`.

The Linux kernel supports another type of configuration when the local IP address and the peer address does not belong to the same IP subnet: `192.0.2.1 peer 198.51.100.1`. This means that the IP address of this interface is `192.0.2.1` and that the IP address of the peer is `198.51.100.1`: this adds a routing rule `198.51.100.1 dev tun0`. A more general form can be used: `192.0.2.1 peer 198.51.100.1/24`.

```
$ ip address show tun0
14: tun0: mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
```

```
inet 192.0.2.1 198.51.100.1/24 scope global tun0
    valid_lft forever preferred_lft forever
```

The `dstaddr` parameter (and the `SIOCSIFDSTADDR`) can be used to set such as destination address.

This is useful if you don't want to allocate a common subnet for the two peers. You don't have to use a special destination address with point to point interface. You could use a standard IP subnet. Or you could allocate [a /31](#). Using the destination address/`peer` configuration, you can avoid allocating a subnet for this point-to-point link.

## What is the relation between the peer/destination address and `POINTOPOINT` devices?

These are independant. You don't have to set a destination address on a `POINTOPOINT` interface. You can set a destination address on a `POINTOPOINT` and you can do it on a normal one as well.

However, using a peer destination address is especially/mostly useful for `POINTOPOINT` interfaces.

Share Improve this answer Follow

edited Jun 25 '20 at 21:53

answered May 19 '16 at 14:59



ysdx

7,893

1 32 44

IMO this answer isn't very clear -- i have exactly the same question as the OP and i don't feel like you answered it particularly clearly. ./ – [horseyguy](#) Jul 11 '18 at 6:00

2 @horseyguy: added a TLDR on top. – [ysdx](#) Jul 11 '18 at 9:07

Thx! But I'm still confused, why is a dest address even required? And if your answer is "because it's a layer 3 p2p interface", would you be able to explain why it's a p2p? I asked someone else this and they just got into a loop saying "that's just how layer 3 interfaces work" until they finally admit they don't really understand, and figure the destination address is just a dummy address... 😊 – [horseyguy](#) Jul 11 '18 at 11:24

@horseyguy: What is required is to configure a IP+netmask to the interface. The dtsaddress is just a convenient extensions which really is independent from the POINTTOPOINT thing. AFAIU, you can use this feature without a POINTTOPOINT interface. And you don't need to use it with a POINTTOPOINT interface. – [ysdx](#) Jul 11 '18 at 11:29 ✎

thanks but what do you mean by "convenient extensions" ? i still dont understand why it's necessary – [horseyguy](#) Jul 11 '18 at 13:14

@horseyguy: it's *not* necessary. If you don't need it/don't care about it you can just set `dstaddr` to `addr` and it will work as intended. You can use it if you want a local IP address and the remote/peer one to be in different IP ranges. – [ysdx](#) Jul 11 '18 at 16:17

Actually I think, if you omit to set it will just work (address and netmak). – [ysdx](#) Jul 11 '18 at 16:28

You switch to `PEERTOPEER` in the second section - do you mean `POINTTOPOINT` ? Also, some of the IP addresses in the first section seem incorrect - can you give them a careful check? – [Jonathon Reinhart](#) Aug 14 '18 at 4:35

@JonathonReinhart, Thanks I fixed that. Indeed there is no `PEERTOPEER` flag. – [ysdx](#) Jun 25 '20 at

21:54



If you add an interface with

0

```
inet 10.66.66.1 netmask 255.255.255.0
```



No matter if you create it as point to point, or not- a new routing entry will be added to the kernel for 10.66.66.1/24 with destination of the new interface.

So I don't think that there is a difference there.

[Share](#) [Improve this answer](#) [Follow](#)

answered Apr 6 '16 at 16:58



[dream-weaver](#)

111 ● 1 ● 6