TECHNISCHE UNIVERSITÄT DRESDEN

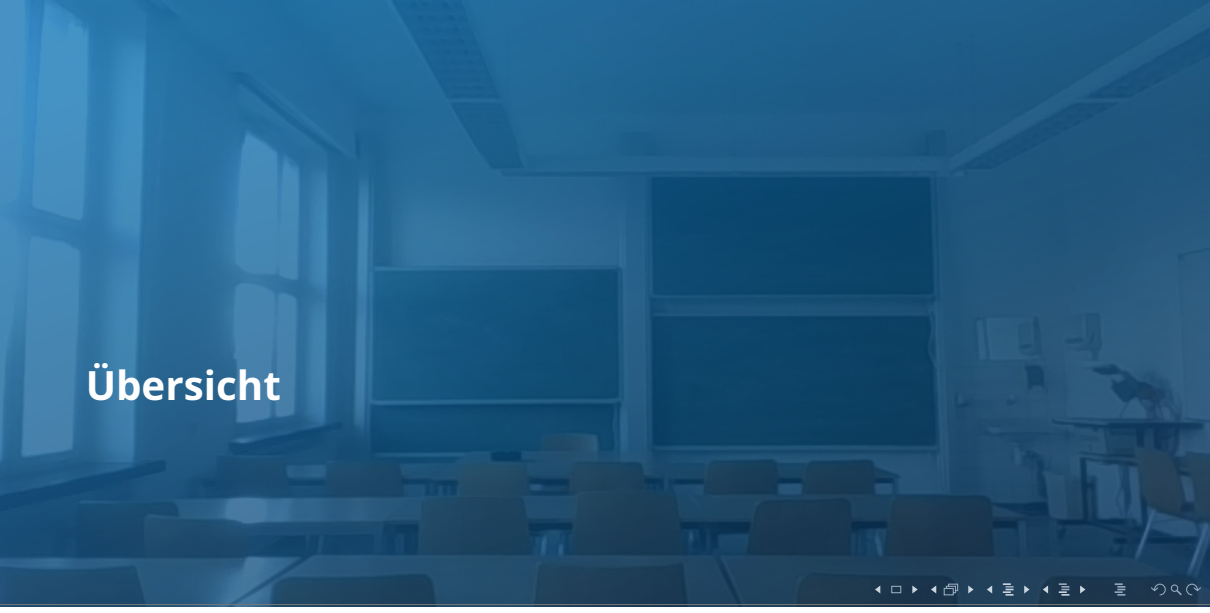
DRESDEN concept

Lucas Waclawczyk

TUN/TAP-Geräte
Übersicht, Funktionsweise und Implementierung im Linux-Kernel

Proseminar Rechnernetze // Dresden, 25. Mai 2020

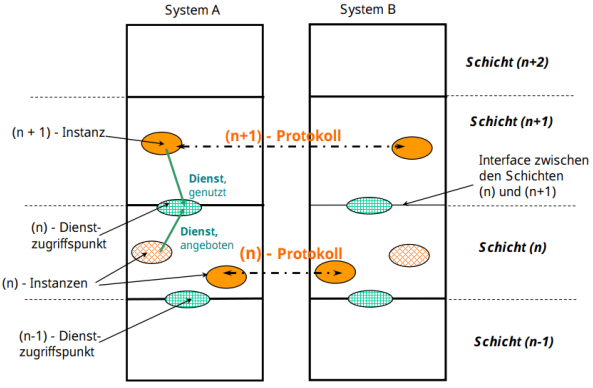
Inhalt



Übersicht

Rückblick Network Interfaces

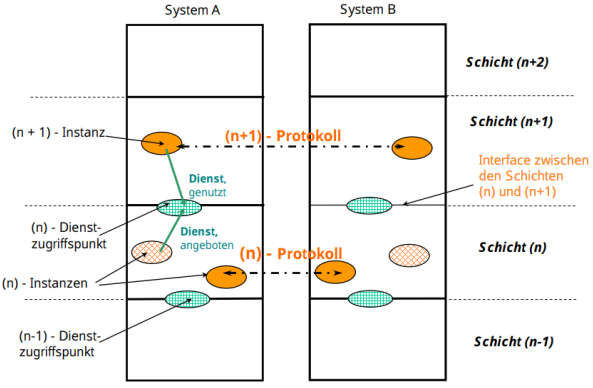
- Interface = Schnittstelle
- zwischen zwei Schichten im OSI-Modell



aus [1]

Rückblick Network Interfaces

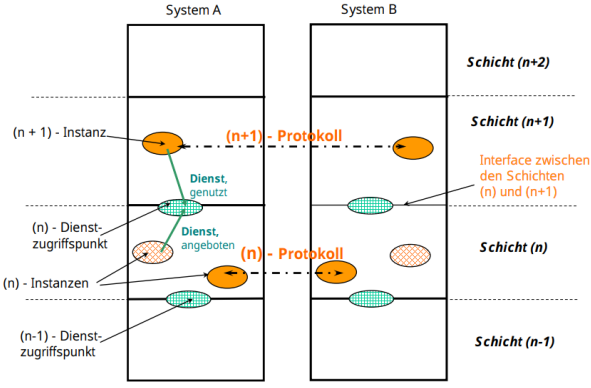
- Interface = Schnittstelle
- zwischen zwei Schichten im OSI-Modell



aus [1]

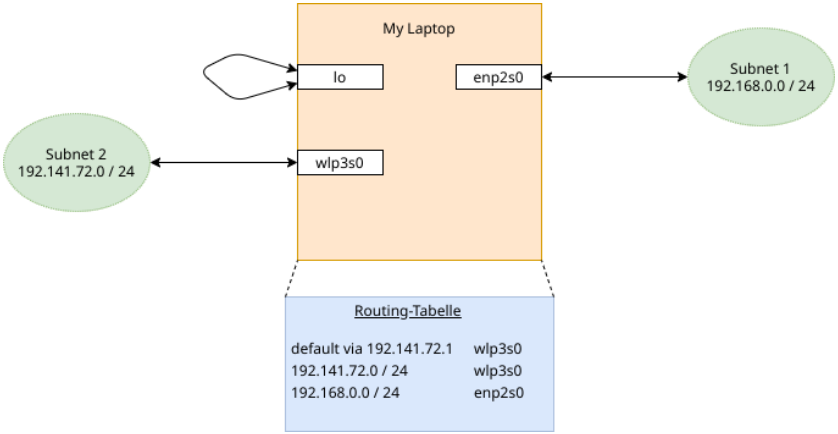
Rückblick Network Interfaces

- Interface = Schnittstelle
- zwischen zwei Schichten im OSI-Modell
- übernimmt Daten von Instanz eines $(n+1)$-Protokolls
- stellt Daten für Instanz eines (n)-Protokolls bereit



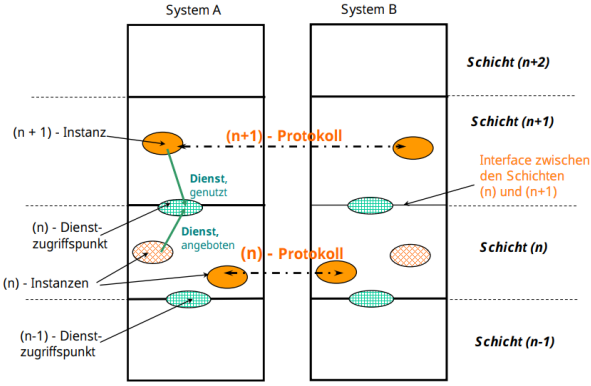
aus [1]

Rückblick Network Interfaces



Rückblick Network Interfaces

- Interface = Schnittstelle
- zwischen zwei Schichten im OSI-Modell
- übernimmt Daten von Instanz eines $(n+1)$-Protokolls
- stellt Daten für Instanz eines (n)-Protokolls bereit
- muss nicht physisch sein
  ➔ Virtual Network Interface



aus [1]

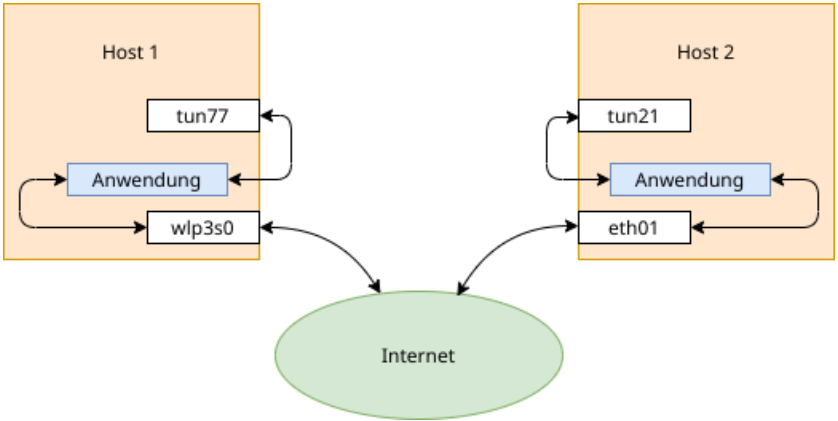
Generelles zu TUN und TAP

- Virtual Network Interfaces, d.h. man kann
  - … ihnen IP-Adressen zuweisen
  - … ihren Traffic analysieren
  - … Firewall-Regeln für sie konfigurieren
  - … uvm.

Generelles zu TUN und TAP

- Virtual Network Interfaces, d.h. man kann
  - … ihnen IP-Adressen zuweisen
  - … ihren Traffic analysieren
  - … Firewall-Regeln für sie konfigurieren
  - … uvm.
- *Interface* $\leftrightarrow$ *Anwendung* statt *Interface* $\leftrightarrow$ *physische Verbindung*

Generelles zu TUN und TAP



Generelles zu TUN und TAP

- Virtual Network Interfaces, d.h. man kann
  - … ihnen IP-Adressen zuweisen
  - … ihren Traffic analysieren
  - … Firewall-Regeln für sie konfigurieren
  - … uvm.
- *Interface* $\leftrightarrow$ *Anwendung* statt *Interface* $\leftrightarrow$ *physische Verbindung*
- u. A. Linux, Windows 2000 - 10, Mac OS X (nur TUN eingebaut)
- hier für Linux

Vor- und Nachteile [7]

TAP – Terminal Access Point
$\sim$ Ethernet $\sim$ Layer 2

TUN – Netzwerk-Tunnel
$\sim$ IP $\sim$ Layer 3

Vor- und Nachteile [7]

TAP – Terminal Access Point

$\sim$ Ethernet $\sim$ Layer 2

- $\oplus$ verhält sich wie echter Netzwerkadapter
- $\oplus$ flexible Protokollwahl
- $\oplus$ Bridging möglich

TUN – Netzwerk-Tunnel

$\sim$ IP $\sim$ Layer 3

Vor- und Nachteile [7]

TAP – Terminal Access Point

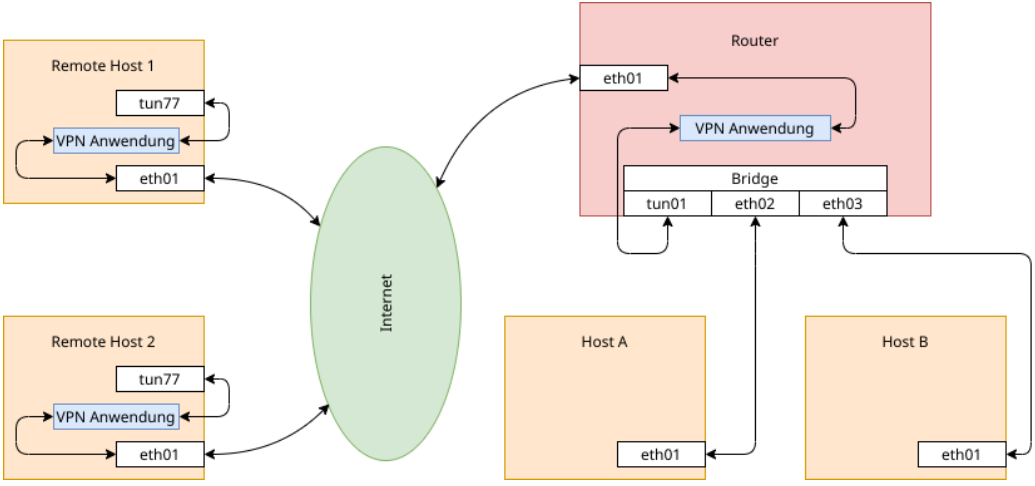
∼ Ethernet ∼ Layer 2

- ⊕ verhält sich wie echter Netzwerkadapter
- ⊕ flexible Protokollwahl
- ⊕ Bridging möglich
- ⊖ viel Overhead (Ethernet-Header, Broadcast)
- ⊖ skaliert schlecht
- ⊖ kein Support bei Android, iOS

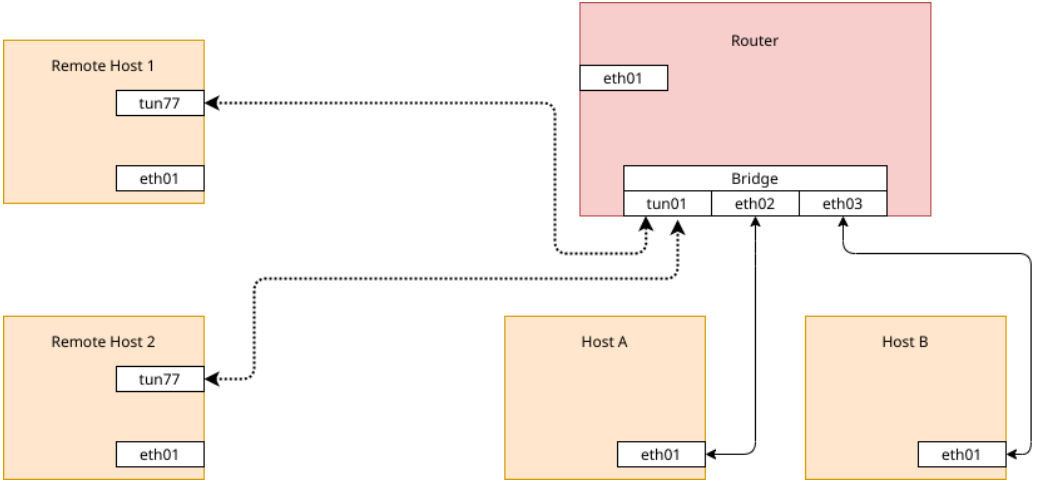
TUN – Netzwerk-Tunnel

∼ IP ∼ Layer 3

Vor- und Nachteile [7]



Vor- und Nachteile [7]



Vor- und Nachteile [7]

TAP – Terminal Access Point

∼ Ethernet ∼ Layer 2

- ⊕ verhält sich wie echter Netzwerkadapter
- ⊕ flexible Protokollwahl
- ⊕ Bridging möglich
- ⊖ viel Overhead (Ethernet-Header, Broadcast)
- ⊖ skaliert schlecht
- ⊖ kein Support bei Android, iOS

TUN – Netzwerk-Tunnel

∼ IP ∼ Layer 3

- ⊕ weniger Overhead (kein Ethernet-Header, kein Broadcast)
- ⊕ nur Layer-3-Pakete

Vor- und Nachteile [7]

TAP – Terminal Access Point

∼ Ethernet ∼ Layer 2

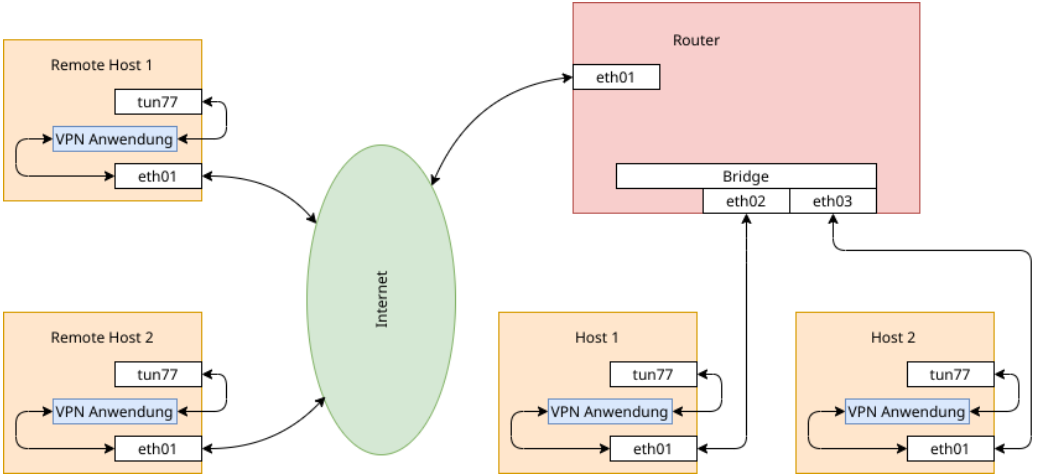
- ⊕ verhält sich wie echter Netzwerkadapter
- ⊕ flexible Protokollwahl
- ⊕ Bridging möglich
- ⊖ viel Overhead (Ethernet-Header, Broadcast)
- ⊖ skaliert schlecht
- ⊖ kein Support bei Android, iOS

TUN – Netzwerk-Tunnel

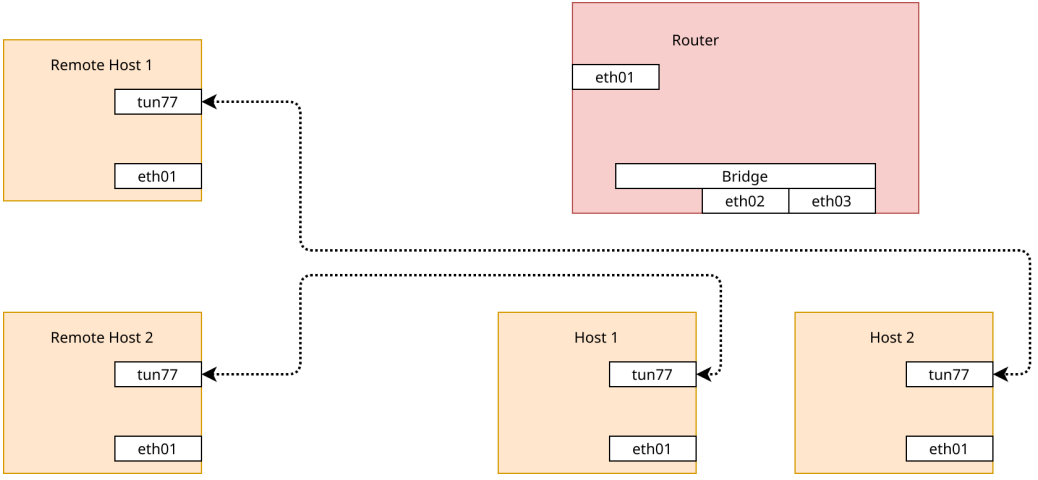
∼ IP ∼ Layer 3

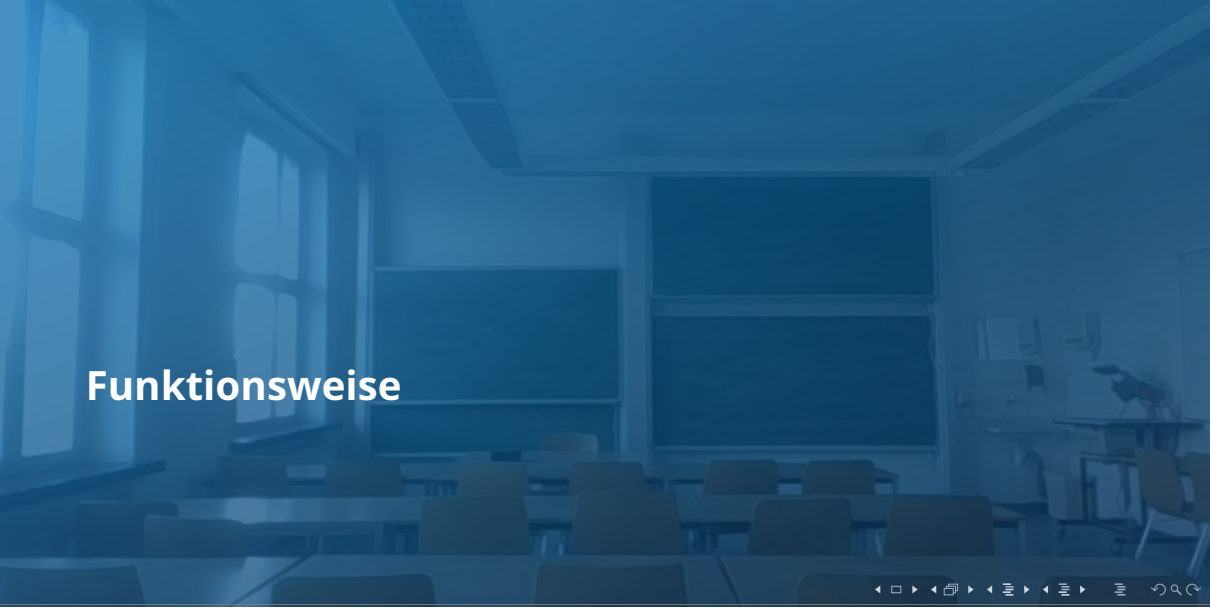
- ⊕ weniger Overhead (kein Ethernet-Header, kein Broadcast)
- ⊕ nur Layer-3-Pakete
- ⊖ nur Layer-3-Pakete
- ⊖ keine Broadcasts
- ⊖ kein Bridging möglich

Vor- und Nachteile [7]



Vor- und Nachteile [7]





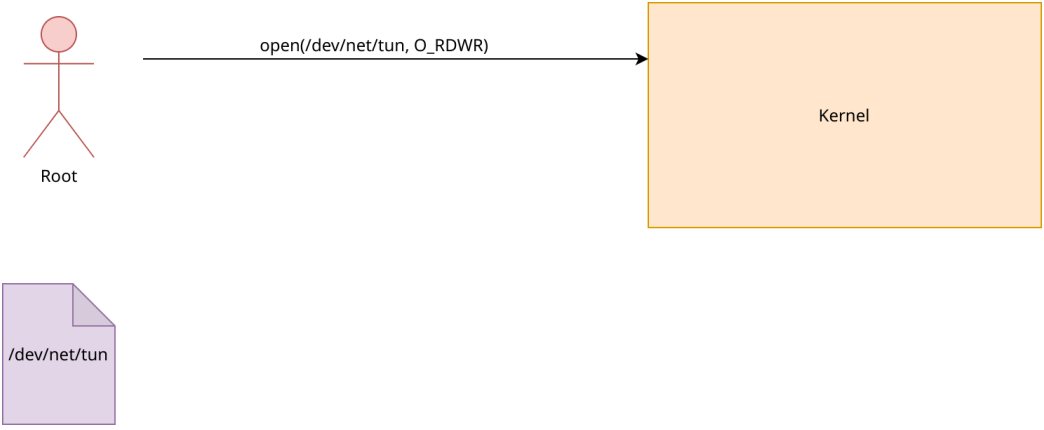
Funktionsweise

Set Up

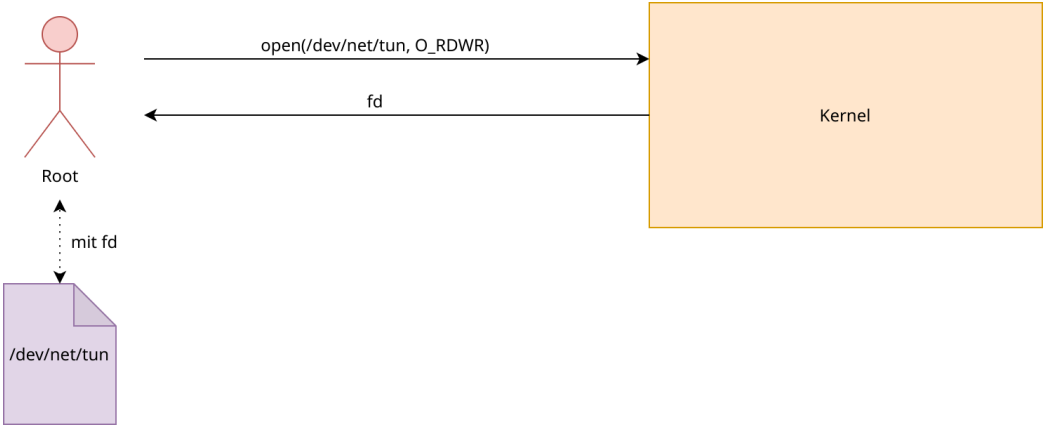
Interface erzeugen (erfordert CAP_NET_ADMIN capability):

1. `/dev/net/tun` (Clone Device) öffnen (r, w)
2. Systemaufruf: `ioctl(fd, TUNSETIFF, options)`
3. ggf. persistent einrichten

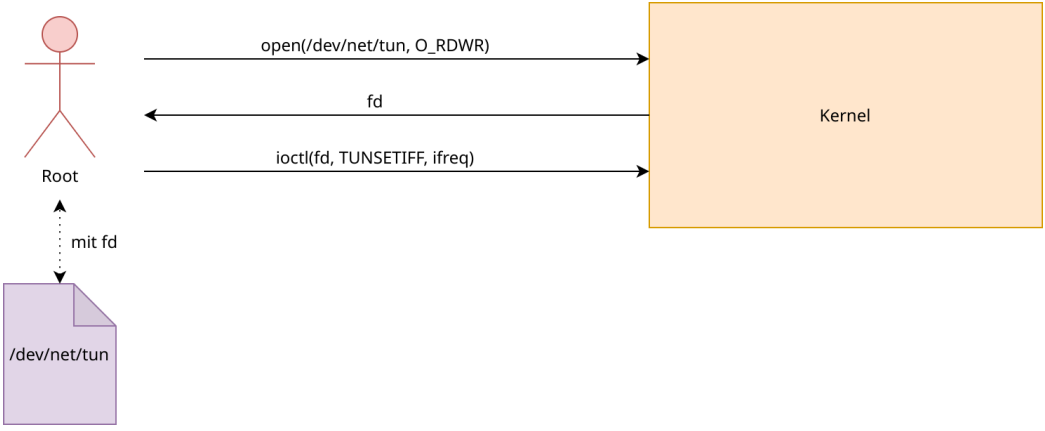
Set Up



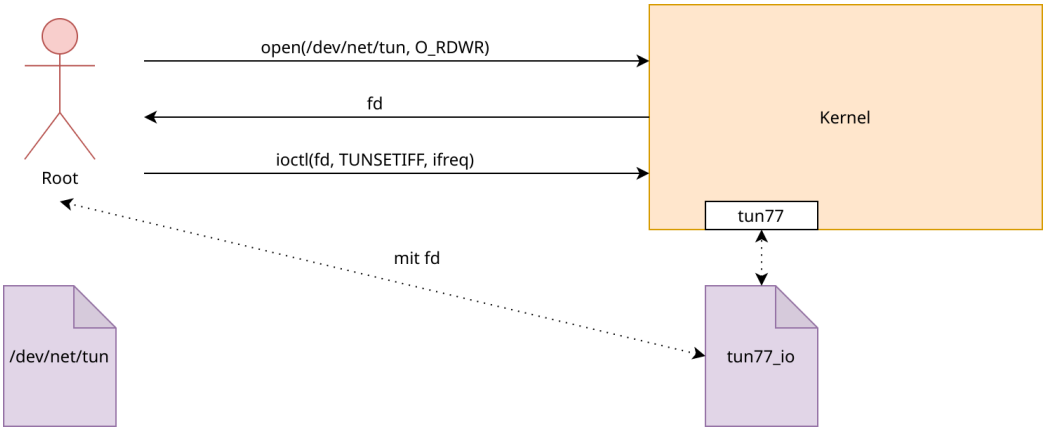
Set Up



Set Up


open(/dev/net/tun, O_RDWR)
fd
ioctl(fd, TUNSETIFF, ifreq)
Kernel
Root
mit fd
/dev/net/tun

Set Up


open(/dev/net/tun, O_RDWR)
fd
ioctl(fd, TUNSETIFF, ifreq)
Root
Kernel
tun77
mit fd
/dev/net/tun
tun77_io

Set Up

Interface erzeugen (erfordert CAP_NET_ADMIN capability):

1. `/dev/net/tun` (Clone Device) öffnen (r, w)
2. Systemaufruf: `ioctl(fd, TUNSETIFF, options)`
3. ggf. persistent einrichten

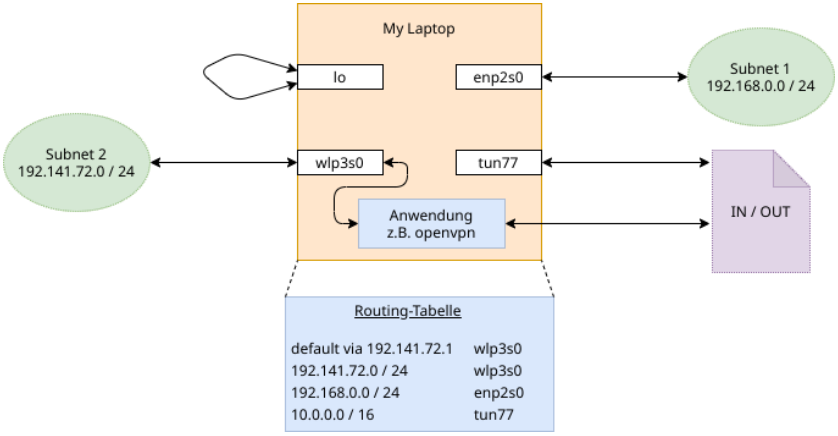
VPN Anwendung anbinden:

- wiederhole obiges als beliebiger Nutzer
- nutze bestehenden Interface-Namen

Workflow

-
- Vorteile
  - gut konfigurierbar
  - verhält sich wie echter Netzwerkadapter
- Nachteile
  - ineffizient
  - langsamer als Heimnetz (schwächstes Glied)

Workflow


My Laptop
lo
enp2s0
Subnet 1
192.168.0.0 / 24
Subnet 2
192.141.72.0 / 24
wlp3s0
tun77
IN / OUT
Anwendung
z.B. openvpn
Routing-Tabelle
default via 192.141.72.1 wlp3s0
192.141.72.0 / 24 wlp3s0
192.168.0.0 / 24 enp2s0
10.0.0.0 / 16 tun77

Tear Down

- *transientes* Gerät verschwindet bei Beenden der angebundenen Anwendung
- *persistentes* Gerät muss aktiv abgebaut werden

Implementierung

Wo findet man das?

- Kernel-Code: `https://github.com/torvalds/linux.git`
- TUN: `/linux/drivers/net/tun.c`
- Clone Device: `/dev/net/tun`

Code

etwa drei Folien

Quellen

Quellen

[1] Skript und Übungsaufgaben der Vorlesung Rechnernetze, TU Dresden 2019 (präzise genug?)

[2] https://backreference.org/2010/03/26/tuntap-interface-tutorial/

[3] https://www.elektronik-kompendium.de/sites/net/0811011.htm

[4] https://en.wikipedia.org/wiki/TUN/TAP

[5] https://www.thomas-krenn.com/de/wiki/OpenVPN_Grundlagen

[6] https://floating.io/2016/05/tuntap-demystified/3/

[7] https://community.openvpn.net/openvpn/wiki/BridgingAndRouting