

## Praxisaufgabe: Aufbau eines grundlegenden TCP/IP-Netzwerks

Treiberentwicklung, Echtzeit- und Betriebssysteme · Sommersemester 2025 · Prof. Dr. Peter Gerwinski

Aufgabe: Vernetzen Sie Ihre Rechner mittels TCP/IP.

- Stellen Sie eine geeignete Hardware-Infrastruktur her.  
(Kabelgebundenes Netz, WLAN, Avian Carriers, ...)
- Wählen Sie geeignete IP-Adressen und prüfen Sie mittels `ping` die gegenseitige Erreichbarkeit.
- Stellen Sie mittels Netcat (`nc`) TCP-Verbindungen her (Ende-zu-Ende-Chat).
- Bieten Sie (z. B. mittels `nc -e`) in diesem Netz Dienste an und testen Sie diese.
- Untersuchen Sie den Netzwerkverkehr mittels `tcpdump` und/oder Wireshark.
- Testen Sie, wie sich externe Dienste (Webseiten, E-Mail) mittels `nc` nutzen lassen.

*Viel Erfolg!*

Stand: 10. April 2025

Copyright © 2025 Peter Gerwinski

Lizenz: CC BY-SA (Version 4.0) oder GNU GPL (Version 3 oder höher)

Sie können diese Praxisaufgabe einschließlich  $\LaTeX$ -Quelltext herunterladen unter:

<https://gitlab.cvh-server.de/pgerwinski/bs>

# Versuchsskizze

1. Für eine Verbindung zwischen mehreren Geräten gibt es diverse Möglichkeiten:

- Ethernet (IEEE 802.3), bei typischen aktuellen PCs max. 1-10 GBit/s möglich, bei Verbindung von zwei Geräten mittels Kabel (ab Gigabit ist hier kein Crossover-Kabel notwendig), bei mehr Geräten mit Hub oder Switch (im einfachsten Fall, sofern keine Segmentierung gewünscht). Unter Linux erscheinen die dann als Gerät *eth0*, *eth1*, ... oder *enp...* (für PCI-Geräte) oder *enx* (für Geräte mit USB-Anbindung), zur Benennung gibt es Details unter [https://systemd.io/PREDICTABLE\\_INTERFACE\\_NAMES/#what-precisely-has-changed-in-v197](https://systemd.io/PREDICTABLE_INTERFACE_NAMES/#what-precisely-has-changed-in-v197) und <https://www.freedesktop.org/software/systemd/man/latest/systemd.net-naming-scheme.html>
- Wireless-LAN (IEEE 802.11), theoretisch bei idealen Bedingungen Übertragung im Gigabit/s-Bereich möglich, aber ein geteiltes Medium und meist geringere Datenraten erreichbar. Hier wird typischerweise ein Access-Point zur Verbindung mehrerer Geräte benutzt, unter Linux erscheinen die Geräte z. B. als *wlan0* oder *wlp0s20f3*.
- Verbindung zweier Geräte per USB-C (USB3.2: maximal 20 GBit/s, USB4: maximal 80 GBit/s), erscheint unter Linux z. B. als *thunderbolt0*.
- Jeder Netzwerkadapter erhält vom Hersteller einer eindeutige Hardwareadresse, die normalerweise nicht verändert wird.

*Im Praxisversuch wurden Ethernet und USB-C verwendet.*

Damit Rechner (ohne Router oder über ein Point-to-Point-Protokoll) miteinander sprechen können, müssen sie sich im selben **Subnetz** befinden. Die Subnetzmaske bzw. Präfixlänge gibt an, wieviele Bits der IP-Adressen der Netzwerkteil sind, der Rest unterscheidet die einzelnen Hosts innerhalb eines Netzes, so ist z.B. bei der IPv4-Adresse *192.168.54.73/24* der Netzwerkteil 24 Bit lang und der Hostteil  $32-24=8$  Bit lang, die Hosts dürfen sich also nur in der letzten 8-Bit-Zahl unterscheiden. Zur Vergabe einer IP-Adresse gibt es mehrere Möglichkeiten:

- Manuelle Vergabe: Temporär z.B. mit `sudo ip addr add 198.51.100.1/30 dev enp0s31f6` oder dauerhaft mittels Network-Manager (entweder graphisch oder über

```
nmcli connection add con-name Beispielnetz type ethernet \
ifname enp0s31f6 ipv4.addresses 198.51.100.1/30
```

- Vergabe per **DHCP** (insb. bei IPv4 verbreitet): Ein oder mehrere Server vergeben auf Anfrage IP-Adressen und informieren über die Adresse des Default-Gateways, typischerweise werden auch Zusatzinformationen, wie z.B. die Subnetzmaske, DNS-Server und die Gültigkeitsdauer der DHCP-Adresse mitgeliefert. Eine IPv4-Adresse kann z.B. temporär per `dhclient -eth0` oder dauerhaft per

```
nmcli connection add con-name Beispielnetz type ethernet \
ifname enp0s31f6 ipv4.method auto ipv4.link-local disabled
```

(mit explizit deaktiviertem Fallback auf *link-local*-Adressen) konfiguriert werden.

- Automatische Auswahl einer lokalen Adresse (bei IPv6 üblich, bei IPv4 eher als Fallback, falls kein DHCP-Server erreichbar ist). Relevant sind hierfür insb. **Dynamic Configuration of IPv4 Link-Local Addresses** bzw. **SLAAC**. Der Rechner wählt selbstständig eine Adresse aus dem Bereich 169.254.0.0/16 (IPv4) bzw. ff02::/16 (IPv6) aus und überprüft mittels Address Resolution Protocol (IPv4), ob diese noch nicht verwendet wird (und wählt ggf. eine andere Adresse), bei IPv6 wird diese von der Hardware-Adresse abgeleitet und ist dadurch automatisch eindeutig. Explizite Konfiguration unter IPv4 z. B. mittels

```
nmcli connection add con-name Beispielnetz type ethernet \
ifname enp0s31f6 ipv4.method link-local
```

bei IPv6 wird eine lokale Adresse zwingend benötigt.

- Eigenständige Auswahl einer globalen IPv6-Adresse mittels SLAAC und des **Neighbor Discovery Protocol**: Übliche Vorgehensweise bei IPv6, hierfür müssen im Netzwerk entsprechende *Router Advertisements* verteilt werden, unter Linux mittels `radvd`.

Im Praxisversuch wurde zur Kommunikation vor Ort IPv4 mit link-local-Adresse verwendet.

Für die Anbindung der Systeme außerhalb der Hochschule wurde als Overlay-Netz [The onion router](#) verwendet. Unter Debian-basierten Linux-Distributionen ist die Installation mittels `apt install tor` möglich. Um hierüber eine Kommunikationsverbindung herzustellen, werden zwei TOR-Instanzen benötigt:

- Zunächst ist auf Serverseite die Konfigurationsdatei `/etc/tor/torrc` anzupassen (ggf. mit root-Rechten):

```
HiddenServiceDir /var/lib/tor/beispieldienst
HiddenServicePort 1234 127.0.0.1:22
```

Hiermit wird im TOR-Netzwerk ein sog. Hidden-Service auf dem Port 1234 geöffnet (dieser ist hidden, weil von außen nicht ohne weiteres feststellbar ist, auf welchem Gerät dieser Dienst gehostet wird. Sobald der Server mit `systemctl restart tor` neu gestartet wird, wird dafür ein Schlüsselpaar erzeugt, der daraus abgeleitete Hostname ist dann unter `/var/lib/tor/beispieldienst/hostname` abgespeichert.

- Auf dem Clientsystem kann nun mittels des SOCKS-Proxy-Protokolls eine Verbindung zu diesem Dienst aufgebaut werden, z.B. mittels  
`torsocks ssh -p 1234 benutzername@3u9wrgiv7...hvoqoqb52fflwpeyd.onion`  
(ggf. muss torsocks zuvor installiert werden).

2. Die per Kabel verbundenen Rechner konnten sich untereinander per `ping`-Kommando kontaktieren, wobei ein Rechner auf Grund einer aktiven Firewall zunächst nicht antwortete. Mittels `ip neighbour` und `wireshark` konnte aber dennoch überprüft werden, dass dieser Rechner tatsächlich auf `arp`-Requests antwortete (per IPv6 wäre stattdessen NDP benutzt worden). Auch lies sich per `wireshark` beobachten, wie mittels `arp` überprüft wurde, ob die gewählte IPv4-Adresse noch frei war.
3. Mittels `nc -l -p 1234` wurde ein Chat-Dienst bereit gestellt und mittels `nc 169.254.4.2 1234` wurde eine Verbindung zu diesem aufgebaut.

Wenn in der `torrc` die Zeile, die mit `HiddenServicePort` beginnt zu

```
HiddenServicePort 1234 127.0.0.1:1234
```

geändert wird (und TOR neugestartet wird), kann im Anschluss auch mittels

```
torsocks nc 3u9wrgiv7...hvoqoqb52fflwpeyd.onion 1234
```

 ein Zugriff per TOR auf den Chatserver gestartet werden (je nach Netcat-Version wird ggf. auch von diesem selbst ein Zugriff per SOCKS über den TOR-Server auf `localhost:9050`).

4. Als Serverdienst wurde ein SSH-Server eingesetzt, über `ssh 169.254.4.2` konnte ein Verbindungsaufbau durchgeführt werden, alternativ ist auch mittels `nc 169.254.4.2 22` eine Verbindung aufgebaut werden, welche mit einer Meldung der Art `SSH-2.0-OpenSSH_9.9p2 Debian-2` beantwortet wurde und fortgeführt wurde, wenn über den Netcat-Client eine entsprechend aufgebaute Nachricht übermittelt wurde (getestet durch Copy-and-Paste der initialen Nachricht).
5. Geeignete Dienste für einen Test wären `HTTP` oder `SMTP`, entsprechende Dialoge sehen folgendermaßen aus:

Webseitenabruf per HTTP:

```
nc -C www.cvh-server.de 80
GET / HTTP/1.1
Host: www.cvh-server.de
```

```
HTTP/1.1 302 Found
Date: Thu, 10 Apr 2025 08:51:22 GMT
Server: Apache/2.4.62 (Debian)
Strict-Transport-Security: max-age=15768000; includeSubDomains
Location: https://www.cvh-server.de/
Content-Length: 293
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD_HTML_2.0//EN">
<html><head>
<title>302 Found</title>
```

```
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="https://www.cvh-server.de/">here</a>.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at www.cvh-server.de Port 80</address>
</body></html>
```

E-Mailversand per (E)SMTP (Hinweis: Auch wenn Sie hier früher quasi beliebige Absenderdaten eintragen konnten, überprüfen einige Mailserver inzwischen, ob Ihre Absenderdaten zu Ihrem E-Mailaccount passen, auch gibt es Verfahren wie [DKIM](#), [SPF](#) und [DMARC](#) mit denen überprüft werden kann, ob ein Mailserver berechtigt ist, Mails für eine bestimmte Domain zu erzeugen bzw. weiterzuleiten.)

```
$ nc -C localhost 25
220 rechnername ESMTP Exim 4.98.2 Thu, 10 Apr 2025 11:06:28 +0200
EHLO localhost
250-rechnername Hello localhost [::1]
250-SIZE 52428800
250-LIMITS MAILMAX=1000 RCPTMAX=50000
250-8BITMIME
250-PIPELINING
250-PIPECONNECT
250-CHUNKING
250-STARTTLS
250-PRDR
250 HELP
MAIL FROM: praxisversuch@localhost.cvh-server.de
250 OK
RCPT TO: bwildenhain@cvh-server.de
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
Subject: Bitte schicken Sie uns Ihre PIN zu
From: support@badbank.example.org
To: bwildenhain@cvh-server.de
```

Bitte Senden Sie uns umgehendend Ihre PIN fuer Ihr Online-Banking zu um Ihr Konto zu verifizieren.

```
.
250 OK id=1u2nsR-000000006Ck-3mZP
quit
221 rechnername closing connection
```