ADMIN:DOCS

TCP/IP: Aufbau und Funktionsweise der Protokollfamilie

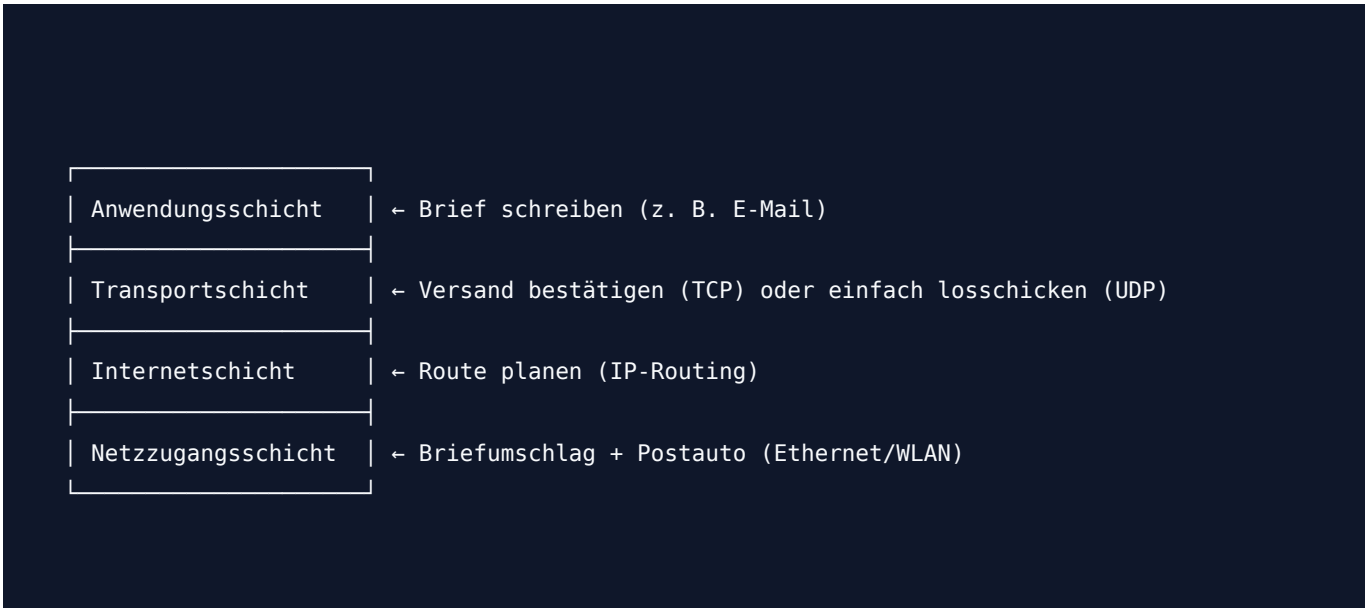
Eine umfassende Einführung in die TCP/IP-Protokollfamilie: Von den grundlegenden Konzepten über wichtige Protokolle bis zur praktischen Anwendung in modernen Netzwerken.

Kategorie: Netzwerk **Veröffentlicht:** 24.11.2024 **Zuletzt geprüft:** 06.03.2026

Die TCP/IP-Protokollfamilie ist das unsichtbare Fundament, auf dem alles läuft, was du als Linux-Administrator oder Fachinformatiker für Systemintegration täglich machst. Ob du einen Webserver erreichst, einen Docker-Container mit dem Internet verbindest oder ein Netzwerkproblem auf einem Ubuntu-Server löst – ohne TCP/IP funktioniert nichts.

Stell dir das Ganze wie ein riesiges, weltweites Postsystem vor. Du schreibst einen Brief (die `Daten`), gibst ihn am Schalter ab (deine `Anwendung`) und das System sorgt dafür, dass er sicher, in der richtigen Reihenfolge und ohne Verluste beim Empfänger ankommt. Genau das leistet TCP/IP – nur eben für Milliarden von Paketen pro Sekunde.

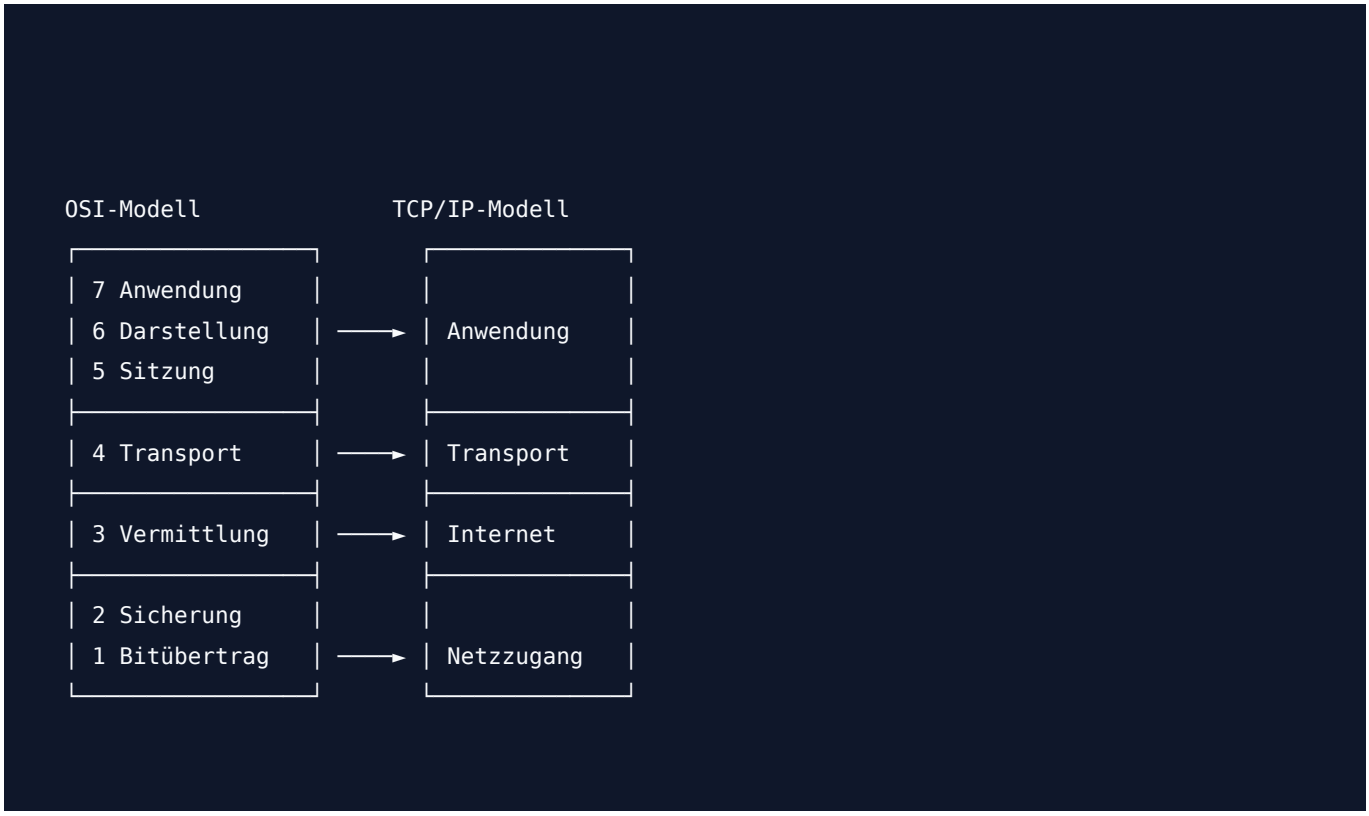
TCP/IP als Postsystem – der Überblick



Quelle: https://admindocs.de/de/netzwerk/tcp-ip-aufbau-und-funktionsweise-der-protokollfamilie

Im Gegensatz zum OSI-Modell, das sieben theoretische Schichten beschreibt, arbeitet TCP/IP mit nur vier praktischen Schichten. Das macht es für den echten Einsatz im Rechenzentrum und auf deinem Linux-Server so erfolgreich.

OSI vs. TCP/IP – der direkte Vergleich



Warum TCP/IP für dich als Linux-Administrator unverzichtbar ist

Auf einem Linux-Server (`Ubuntu` , `Debian` , `Arch` oder `Rocky Linux`) kommuniziert alles über TCP/IP. Wenn du `docker run` ausführst und der Container ins Internet soll, wenn du `ssh` auf einen entfernten Server machst oder wenn `apt update` plötzlich hängt – immer steckt TCP/IP dahinter.

Du wirst lernen, warum ein `ping` funktioniert, warum Port 443 für HTTPS reserviert ist und wie du mit wenigen Befehlen herausfindest, warum dein Server plötzlich keine Verbindung mehr aufbaut.

Quelle: https://admindocs.de/de/netzwerk/tcp-ip-aufbau-und-funktionsweise-der-protokollfamilie

```
# Schnell-Check: Zeigt dir sofort deine TCP/IP-Konfiguration
ip addr show
ss -tuln
```

> **Tipp für Einsteiger**: Öffne jetzt ein Terminal auf deinem Linux-System und gib die beiden Befehle oben ein. Du siehst live, wie TCP/IP auf deinem Rechner arbeitet. Das ist der beste Weg, das Thema zu verstehen – nicht nur lesen, sondern sofort ausprobieren.

Was dich in diesem Artikel erwartet

In den kommenden vier Abschnitten nehmen wir dich Schritt für Schritt an die Hand:

- Die vier Schichten des TCP/IP-Modells und wie Daten durch sie wandern
- Die wichtigsten Protokolle (TCP, UDP, IP, ICMP, DNS, HTTP) mit allen Details
- Adressierung, Routing und moderne Entwicklungen (IPv4, IPv6, Dual-Stack)
- Praktische Anwendung und Troubleshooting auf echten Linux-Servern

Jeder Abschnitt enthält viele ASCII-Diagramme, Tabellen und echte Befehle, die du direkt kopieren kannst.

Für wen dieser Artikel gedacht ist

Dieser Beitrag richtet sich genau an dich, wenn du:

- Als Fachinformatiker Systemintegration die LPIC-1 oder CompTIA Network+ vorbereitest
- Als Linux-Administrator täglich Server, Docker-Container oder Netzwerke betreust
- Von Windows kommst und endlich verstehen willst, warum Linux-Netzwerke so anders (und besser) funktionieren
- Wert auf klare Erklärungen, Praxisbeispiele und keine Theorie-Floskeln legst

So holst du das meiste aus diesem Artikel heraus

Nimm dir ein Linux-System (Ubuntu 24.04 LTS oder eine VM) und arbeite parallel mit. Jedes Diagramm und jeder Befehl ist so aufgebaut, dass du ihn sofort nachmachen kannst.

> ⚠ **Wichtig**: Manche Befehle brauchen `sudo`. Ich markiere das immer klar.

> **Praktischer Tipp**: Lege dir ein Testverzeichnis an und speichere die wichtigsten Befehle in einer Datei `tcpip-cheatsheet.md`. So hast du später alles griffbereit.

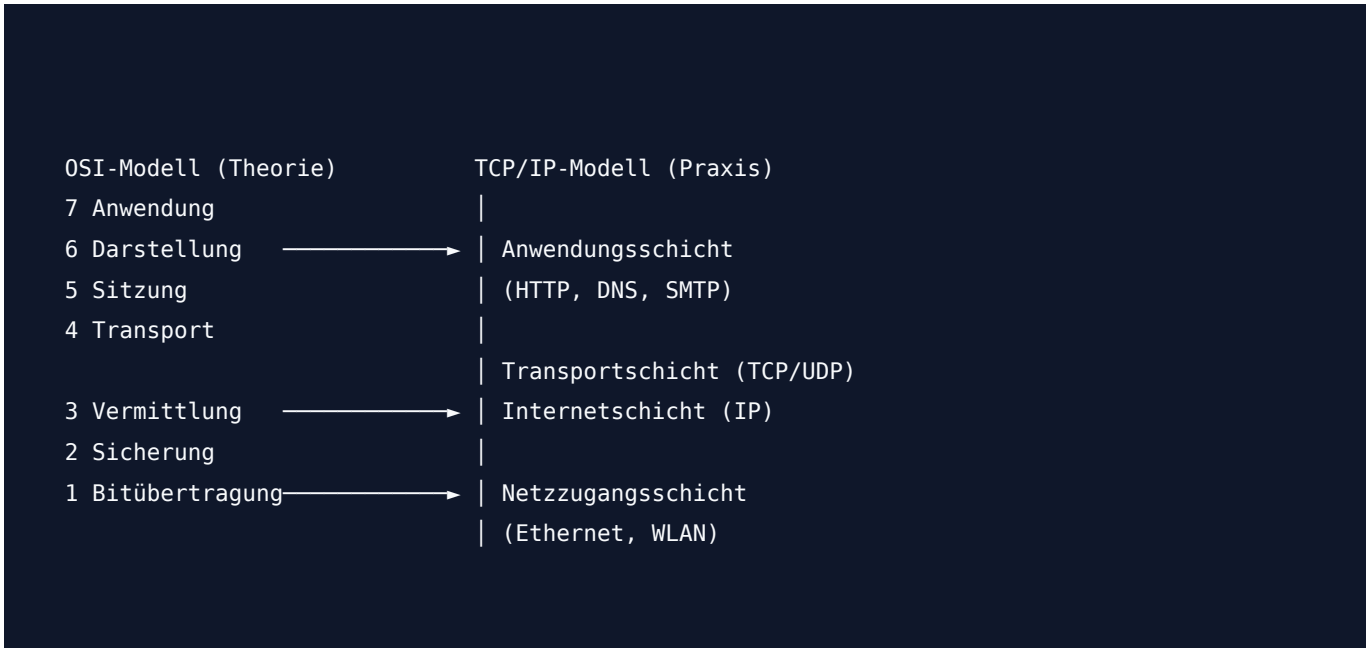
> **Stolperfalle**: Viele Einsteiger denken, TCP/IP sei „nur Theorie“. In Wirklichkeit löst du 80 % aller Server-Probleme, wenn du diese Protokollfamilie richtig verstehst.

Lass uns jetzt gemeinsam eintauchen. Nach diesem Artikel wirst du nicht nur wissen, wie TCP/IP funktioniert – du wirst es auf deinem Linux-Server sehen, messen und reparieren können. Bereit?

Grundlagen und die vier Schichten

Jetzt wird es konkret. Bevor wir uns die einzelnen Protokolle anschauen, musst du zuerst verstehen, wie TCP/IP die Daten eigentlich durch das Netzwerk schickt. Das TCP/IP-Modell ist bewusst einfach gehalten – nur vier Schichten statt der sieben theoretischen des OSI-Modells. Diese Vereinfachung ist genau der Grund, warum TCP/IP seit Jahrzehnten das Internet am Laufen hält und warum du als Linux-Administrator damit so gut arbeiten kannst.

Warum vier Schichten statt sieben?



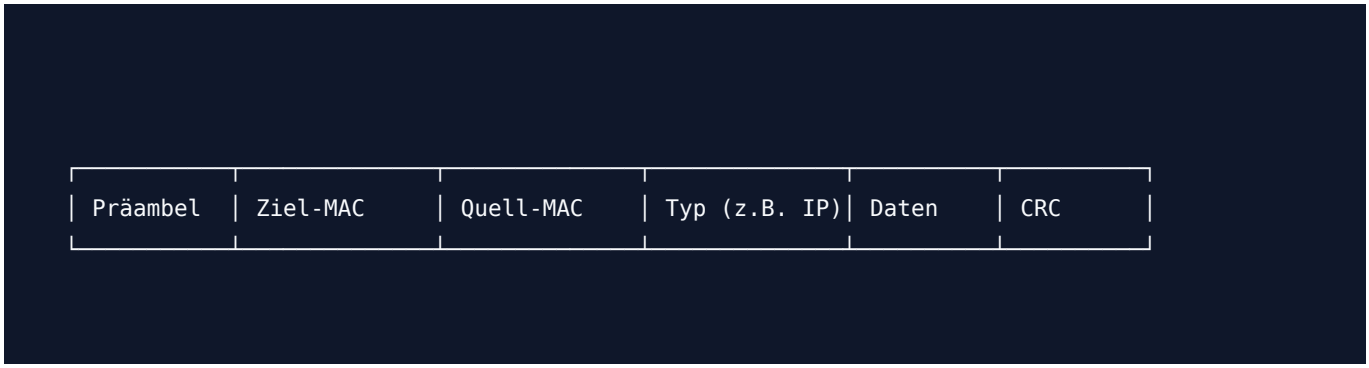
Jede Schicht hat eine klare Aufgabe und übergibt die Daten nur an die Schicht direkt darunter oder darüber. Das nennt man Encapsulation – die Daten werden auf dem Weg nach unten immer weiter „eingepackt“ und auf dem Weg nach oben wieder ausgepackt.

1. Netzzugangsschicht

Der physische Transport

Diese unterste Schicht ist verantwortlich für den tatsächlichen Transport der Bits über `Kabel`, `WLAN` oder `Glasfaser`. Hier kommen `Ethernet-Frames`, `MAC-Adressen` und `WLAN-Frames` ins Spiel.

Ethernet-Frame (vereinfacht)



MAC-Adresse Jede Netzwerkkarte hat eine weltweit eindeutige 48-Bit-MAC-Adresse (z. B. `00:1A:2B:3C:4D:5E`). Sie ist hardwareseitig und unveränderlich – im Gegensatz zur IP-Adresse, die du frei vergeben kannst.

```
# Zeigt dir alle Netzwerkinterfaces und deren MAC-Adressen
ip link show

# Beispiel-Ausgabe auf einem Ubuntu-Server:
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:ab:cd:ef brd ff:ff:ff:ff:ff:ff
```

> □□ **Tipp**: Die ersten drei Bytes der MAC-Adresse verraten dir den Hersteller (OUI). Mit `ip link` siehst du sofort, ob du eine Intel-, Realtek- oder Broadcom-Karte hast.

WLAN vs. Ethernet Bei WLAN übernimmt die gleiche Schicht die Aufgabe, nur dass statt elektrischer Signale Radiowellen verwendet werden. Die Frame-Struktur bleibt fast identisch.

> ⚠ **Wichtig für Einsteiger**: Die Netzzugangsschicht kümmert sich **nicht** darum, wohin das Paket geht – das macht erst die Internetschicht. Sie weiß nur: „Dieses Frame muss zum nächsten Gerät mit dieser MAC-Adresse“.

2. Internetschicht

Das globale Navigationssystem

Hier kommt das Internet Protocol (IP) ins Spiel. Diese Schicht ist für die logische Adressierung und das Routing zuständig. Jedes Paket erhält eine Quell- und Ziel-IP-Adresse.

IP-Paket (IPv4 Header stark vereinfacht)



Routing in der Praxis Dein Linux-Server hat eine Routing-Tabelle. Wenn du `ping google.com` machst, schaut der Kernel nach: „Welche Schnittstelle und welches Gateway benutze ich für `8.8.8.8`?“

```
# Zeigt die komplette Routing-Tabelle
ip route show

# Typische Ausgabe:
default via 192.168.1.1 dev enp0s3 proto dhcp metric 100
192.168.1.0/24 dev enp0s3 proto kernel scope link src 192.168.1.42
```

Praktisches Beispiel: Stell dir vor, du hast einen Docker-Container, der nicht ins Internet kommt. Mit `ip route` siehst du sofort, ob die Default-Route fehlt.

Fragmentierung Wenn ein Paket zu groß für das nächste Netz ist (MTU), wird es in kleinere Fragmente zerlegt. Das passiert meist bei VPNs oder alten Geräten.

> **Häufiger Fehler**: MTU-Probleme führen zu „Black-Hole“-Verbindungen (Pakete verschwinden). Mit `ping -M do -s 1472 8.8.8.8` kannst du die maximale MTU testen.

Quelle: https://admindocs.de/de/netzwerk/tcp-ip-aufbau-und-funktionsweise-der-protokollfamilie

3. Transportschicht

Zuverlässig oder schnell?

Diese Schicht sorgt für die Ende-zu-Ende-Kommunikation zwischen zwei Anwendungen. Hier triffst du die wichtigste Entscheidung: TCP oder UDP?

TCP vs. UDP – direkter Vergleich

Merkmal	TCP	UDP
Verbindung	Ja (Three-Way-Handshake)	Nein
Zuverlässigkeit	Ja (ACK, Retransmit)	Nein
Reihenfolge	Garantiert	Nicht garantiert
Overhead	Hoch	Sehr niedrig
Typische Ports	22 (SSH), 80/443 (Web)	53 (DNS), 123 (NTP)

TCP Three-Way-Handshake Schritt für Schritt

1. Client sendet SYN
2. Server antwortet mit SYN-ACK
3. Client sendet ACK



Ports Jede Anwendung „hört“ auf einem Port (0–65535). Die ersten 1023 sind privileged (nur root).

```
# Zeigt alle lauschenden TCP/UDP-Ports
ss -tuln

# Beispiel: Webserver + SSH + Docker
Netid  State   Recv-Q  Send-Q  Local Address:Port  Peer Address:Port
tcp    LISTEN  0       128     0.0.0.0:22         0.0.0.0:*
tcp    LISTEN  0       128     0.0.0.0:80         0.0.0.0:*
udp    UNCONN  0       0       0.0.0.0:53         0.0.0.0:*
```

4. Anwendungsschicht

Wo du als Admin arbeitest

Hier laufen die Protokolle, die du direkt nutzt: HTTP, HTTPS, SMTP, DNS, SSH. Diese Schicht spricht direkt mit deinen Programmen.

```
Datenfluss von oben nach unten (Encapsulation)
Anwendung:   GET /index.html HTTP/1.1
             |
Transport:   TCP-Paket (Port 443 → 443)
             |
Internet:    IP-Paket (192.168.1.42 → 8.8.8.8)
             |
Netzzugang:  Ethernet-Frame (MAC zu MAC)
```

Live mit tcpdump anschauen Du kannst den gesamten Weg sehen:

```
# Fang nur HTTP-Verkehr auf Port 80 ab (als root)
sudo tcpdump -i enp0s3 -nn port 80 -c 10
```

Warum das alles für Linux-Admins wichtig ist Auf einem echten Server siehst du diese Schichten ständig. Ein `docker ps` zeigt dir Container-Ports, `ip addr` zeigt IP und MAC, `ss -tuln` zeigt TCP/UDP und `tcpdump` zeigt dir jedes einzelne Paket.

□□ Dein erstes Praxis-Experiment

1. Öffne zwei Terminals
2. Terminal 1: `sudo tcpdump -i any -nn port 80`
3. Terminal 2: `curl -I https://admindocs.de`

Du siehst live, wie die vier Schichten zusammenarbeiten.

Nachdem du jetzt die vier Schichten verstanden hast, gehen wir im nächsten Abschnitt tief in die wichtigsten Protokolle hinein – `TCP`, `UDP`, `IP`, `ICMP` und `DNS`. Du wirst danach jeden Befehl auf deinem Linux-Server mit ganz anderen Augen sehen.

Die wichtigsten Protokolle im Detail

Jetzt wird es richtig spannend. Du kennst die vier Schichten – nun schauen wir uns die Protokolle an, die in diesen Schichten tatsächlich die Arbeit erledigen. Als Linux-Administrator triffst du diese Protokolle jeden Tag: beim SSHen auf einen Server, beim Auflösen von Domains oder wenn ein Docker-Container plötzlich keine Verbindung mehr hat. Wir gehen schichtweise durch und du lernst genau, welches Protokoll wann welche Aufgabe übernimmt.

Transportprotokolle

TCP und UDP – die beiden Gesichter der Zuverlässigkeit

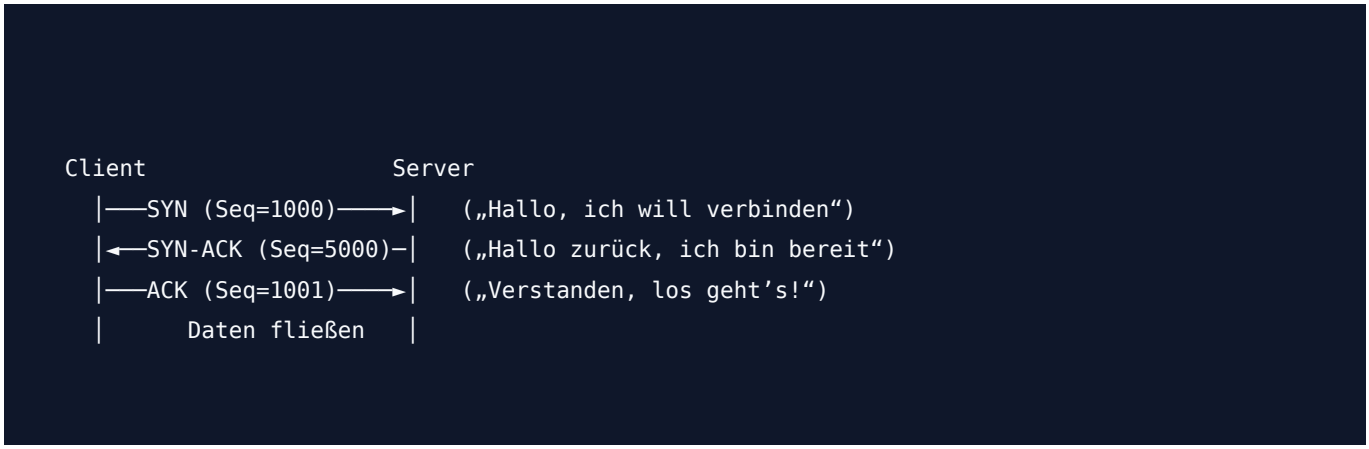
Die Transportschicht ist die erste Stelle, an der du als Admin eine echte Entscheidung triffst: brauche ich absolute Zuverlässigkeit oder maximale Geschwindigkeit?

TCP vs. UDP – der direkte Praxis-Vergleich

Eigenschaft	TCP	UDP
Verbindungsaufbau	Three-Way-Handshake	Keiner
Fehlerkorrektur	Automatisch (ACK + Retransmit)	Keine
Reihenfolge	Garantiert	Nicht garantiert
Geschwindigkeit	Langsamer	Blitzschnell
Overhead	Hoch (Header ~20 Byte)	Niedrig (Header 8 Byte)
Typische Anwendungen	SSH, HTTP, SMTP, Datenbanken	DNS, NTP, VoIP, Streaming, Gaming

TCP – das zuverlässige Arbeitspferd TCP baut zuerst eine Verbindung auf. Das passiert immer in genau drei Schritten – dem berühmten Three-Way-Handshake.

Three-Way-Handshake im Detail



Nach dem Handshake bestätigt jeder Paket-Empfang mit einem ACK. Geht etwas verloren, wird es automatisch neu gesendet. Genau deshalb ist TCP perfekt für alles, wo kein Bit fehlen darf.

UDP – das schnelle Sprinter-Protokoll UDP schickt einfach los – ohne Handshake, ohne Bestätigung. Wenn ein Paket verloren geht, ist es weg. Das klingt riskant, ist aber genau richtig für Echtzeit-Anwendungen.

```
# Zeigt dir live, welche Ports TCP und UDP auf deinem Server nutzen
ss -tuln | grep -E ':(22|53|80|443)'
```

Wichtige Well-Known-Ports (die ersten 1023)

- 22 → `SSH`
- 53 → `DNS` (`UDP` + `TCP`)
- 80 → `HTTP`
- 443 → `HTTPS`
- 25 → `SMTP`

```
# Schnell-Check: Welche Ports lauschen gerade?
sudo ss -tulp | head -10
```

> **LPIC-1-Prüfungs-Tipp**: Du musst die Ports 20/21 (`FTP`), 53 (`DNS`), 80/443 (`Web`) und 22 (`SSH`) auswendig kennen. Die Prüfung fragt immer wieder danach.

IP – das Herzstück der Internetschicht

Das Internet Protocol ist der Briefumschlag jedes Pakets. Es gibt zwei Versionen, die du heute parallel betreibst.

IPv4 vs. IPv6 – Kopf-an-Kopf-Vergleich

```
IPv4                          IPv6
192.168.1.42                  2001:db8::1:42
32 Bit (4,3 Milliarden)       128 Bit (340 Undezillionen)
Dezimal mit Punkten           Hexadezimal mit Doppelpunkten
Header 20–60 Byte             Header 40 Byte (fix)
NAT meist nötig               Kein NAT mehr nötig
```

Warum IPv6? IPv4 ist seit Jahren erschöpft. Jeder Provider gibt dir deshalb oft nur noch eine einzige öffentliche IPv4-Adresse und versteckt alles dahinter mit NAT. IPv6 löst das endgültig – jedes Gerät bekommt eine eigene, globale Adresse.

```
# Zeigt dir beide Adressen auf einmal
ip -4 addr show
ip -6 addr show
```

IPv6 in der Praxis auf Linux Moderne Ubuntu/Debian-Systeme haben IPv6 standardmäßig aktiviert. Du siehst es sofort:

```
# Test: Ping6 zu Google
ping -6 google.com
```

Hilfsprotokolle

ICMP und ARP – die stillen Helfer

ICMP – das Diagnose-Protokoll ICMP ist der Postbote mit Rückmeldung. Ping und Traceroute basieren komplett darauf.

```
Typische ICMP-Nachrichten
Echo Request     → „Bist du da?“
Echo Reply       → „Ja, hier bin ich!“
Destination Unreachable → „Weg blockiert!“
Time Exceeded    → „TTL abgelaufen“ (Traceroute nutzt das)
```

```
# Klassischer Ping
ping -c 4 8.8.8.8

# Traceroute mit ICMP
traceroute -I google.com
```

ARP – der MAC-IP-Übersetzer Im lokalen Netz muss IP in MAC übersetzt werden. Genau das macht ARP.

ARP-Prozess Schritt für Schritt

```
PC: „Wer hat 192.168.1.1?“
     |
     ▼ Broadcast an alle
Router: „Ich! Meine MAC ist 00:1A:2B:3C:4D:5E“
     |
     ▼ ARP-Tabelle wird gefüllt
```

Quelle: https://admindocs.de/de/netzwerk/tcp-ip-aufbau-und-funktionsweise-der-protokollfamilie

```
# Zeigt die aktuelle ARP-Tabelle
ip neigh show
```

Anwendungsprotokolle

DNS, HTTP/HTTPS und mehr

DNS – das Telefonbuch des Internets Ohne DNS müsstest du dir jede IP-Adresse merken. DNS übersetzt `admindocs.de` in eine IP.

DNS-Auflösung live

```
Browser → „Wo ist admindocs.de?“
     |
     ▼ Lokaler DNS-Server (meist dein Router)
     |
     ▼ Root-Server → .de-Server → admindocs.de-Server
     |
     ▼ Antwort: 185.199.108.153
```

```
# DNS-Abfrage direkt auf der Kommandozeile
dig admindocs.de
# oder kurz:
host admindocs.de
```

HTTP/HTTPS – das Web-Protokoll Jede Webseite läuft darüber. HTTPS ist einfach HTTP + TLS-Verschlüsselung.

```
Häufige HTTP-Statuscodes (unbedingt kennen!)
200 OK
301/302 Moved Permanently / Found
404 Not Found
500 Internal Server Error
```

Moderne Ergänzung: QUIC und HTTP/3 HTTP/3 läuft nicht mehr über `TCP`, sondern direkt über `UDP` mit dem `QUIC`-Protokoll. Dadurch sind Verbindungen schneller aufgebaut und Paketverluste stören weniger. Viele moderne Browser und Server (inklusive nginx) nutzen es bereits standardmäßig.

```
# Prüfen, ob dein Server HTTP/3 unterstützt
curl -I --http3 https://admindocs.de
```

Dein Praxis-Beispiel Öffne zwei Terminals und starte:

Terminal 1:

```
sudo tcpdump -i any -nn port 53 or port 80 or port 443 -c 20
```

Terminal 2:

```
curl -I https://admindocs.de
```

Du siehst live, wie DNS (`UDP 53`), dann TCP-Handshake (`Port 443`) und anschließend die Daten fließen.

> **Prüfungsrelevantes für LPIC-1:** Du wirst oft gefragt: Welches Protokoll nutzt `Port 53` ? (Antwort: `DNS` – `UDP` primär, `TCP` bei großen Antworten). **Oder:** Welches Protokoll ist verbindungslos? (`UDP`).

Nachdem du jetzt die wichtigsten Protokolle im Detail kennst, gehen wir im nächsten Abschnitt zur Adressierung und zum Routing über – und schauen uns an, wie Pakete wirklich durch das Internet reisen und welche modernen Entwicklungen du als Admin heute schon nutzen kannst.

Adressierung, Routing und moderne Entwicklungen

Du weißt jetzt, welche Protokolle in welcher Schicht arbeiten. Jetzt kommt der Teil, der für jeden Linux-Administrator tägliche Realität ist: wie Pakete wirklich ihren Weg finden und wie du mit IP-Adressen, Subnetzen und Routing umgehst. Hier lernst du nicht nur Theorie, sondern genau die Befehle, mit denen du auf einem echten Server Probleme löst.

IP-Adressierung: IPv4 und IPv6 im Detail

Jedes Gerät im Netzwerk braucht eine eindeutige Adresse. `IPv4` ist das, was du wahrscheinlich am besten kennst – vier Zahlen zwischen `0` und `255` , getrennt durch Punkte.

IPv4-Adressaufbau am Beispiel `192.168.1.42` :

```
┌─────────────────┬─────────────────┐
│ Netzwerkteil  │ Hostteil      │
│ 192.168.1     │ 42            │
└─────────────────┴─────────────────┘
Subnetzmaske /24 = 255.255.255.0
```

Subnetting einfach erklärt Eine `/24` bedeutet: die ersten `24 Bit` sind fest (Netzwerk), die letzten `8 Bit` gehören dir (`256` mögliche Hosts).

Für ein typisches Heim- oder Firmennetz reicht das meist.

```
# Deine aktuelle IP und Subnetzmaske anzeigen
ip addr show dev enp0s3 | grep inet
```

IPv6 – die Zukunft, die schon da ist IPv6-Adressen sind `128 Bit` lang und sehen erstmal kompliziert aus.

In der Praxis schreibst du sie oft abgekürzt.

```
IPv6-Beispiel: 2001:db8:85a3::8a2e:370:7334

Global Routing Prefix | Subnet-ID | Interface-ID
2001:db8:85a3         | 0000      | 0000:8a2e:0370:7334
```

Viele Provider verteilen heute automatisch ein `/56`- oder `/64`-Prefix. **Das bedeutet:** Du hast Millionen von Subnetzen zur Verfügung – kein NAT mehr nötig.

```
# IPv6-Adressen und -Routes auf deinem System
ip -6 addr show
ip -6 route show
```

Dual-Stack in der Praxis Fast alle modernen Linux-Systeme laufen parallel mit `IPv4` und `IPv6` . Du kannst beide gleichzeitig nutzen.

> ⚠ **Wichtige Warnung**: Viele Anfänger deaktivieren IPv6 komplett, weil „es kompliziert“ ist. Das führt oft zu Problemen bei Google, Netflix oder Cloud-Diensten, die IPv6 bevorzugen.

Routing

Wie Pakete wirklich durch das Internet reisen

Dein Linux-Kernel hat eine Routing-Tabelle. Jedes Paket wird dort nachgeschlagen: „Wohin damit?“

Routing-Tabelle (vereinfacht)

```
Ziel            Gateway         Interface
0.0.0.0/0       192.168.1.1     enp0s3     ← Default-Route
192.168.1.0/24  -               enp0s3     ← Lokales Netz
10.0.0.0/8      10.0.0.1        docker0    ← Docker-Netz
```

```
# Die komplette Routing-Tabelle anzeigen
ip route show
```

Wie Routing in mehreren Schritten funktioniert

- Dein PC schickt das Paket zum Default-Gateway (meist dein Router).
- Der Router schaut in seiner Tabelle nach und schickt es weiter.
- Das wiederholt sich, bis das Paket beim Ziel ankommt.

```
# Zeigt dir den kompletten Weg zu einem Ziel
traceroute 8.8.8.8
```

Fragmentierung und MTU Wenn ein Paket zu groß für das nächste Netz ist, wird es zerlegt. Das kostet Performance. Deshalb ist die richtige MTU-Einstellung wichtig.

```
# MTU deines Interfaces prüfen
ip link show enp0s3 | grep mtu
```

Praktisches Docker-Beispiel Docker erstellt automatisch ein eigenes Netzwerk mit `Bridging`.

Du siehst es sofort:

```
docker network ls
ip addr show docker0
```

Moderne Entwicklungen

Hohe IPv6-Adoption Immer mehr Provider und Cloud-Anbieter setzen voll auf `IPv6`. Dein Ubuntu-Server bekommt oft schon eine öffentliche IPv6-Adresse ohne zusätzliche Konfiguration.

5G und zukünftige Mobilfunknetze Mobilfunk basiert komplett auf `TCP/IP`. Die niedrige Latenz und hohe Bandbreite von `5G` machen neue Anwendungen wie `Edge-Computing` möglich – und du als Admin betreibst die Server dahinter.

Software Defined Networking (SDN) In großen Umgebungen (`Kubernetes`, `OpenStack`) wird Routing nicht mehr nur in Hardware-Routern gemacht, sondern per Software gesteuert. Tools wie `Cilium` oder `Calico` nutzen `eBPF` im `Linux-Kernel`, um extrem schnelles Routing direkt im Kernel zu erledigen.

```
# eBPF-basierte Netzwerk-Tools prüfen (z. B. in Kubernetes)
kubectl get pods -n kube-system | grep cilium
```

Sicherheit: IPsec und WireGuard Moderne VPNs wie `WireGuard` laufen über `UDP` und sind extrem einfach zu konfigurieren. Du kannst in unter 5 Minuten einen sicheren Tunnel aufbauen.

```
# Schneller Check, ob WireGuard läuft
ip link show wg0
```

> **Häufiger Stolperstein**: Viele denken, IPv6 sei „unsicherer“. In Wirklichkeit ist es sicherer, weil keine NAT-Tricks mehr nötig sind und du saubere Firewall-Regeln schreiben kannst.

Dein sofortiger Test Führe diese Befehle nacheinander aus:

```
ip route get 8.8.8.8
ip route get 2001:4860:4860::8888
traceroute -4 8.8.8.8
traceroute -6 google.com
```

Du siehst live, wie dein System für `IPv4` und `IPv6` unterschiedliche Wege wählt.

Nachdem du jetzt Adressierung und Routing vollständig verstanden hast, schauen wir im letzten Abschnitt, wie du all das in der Praxis als Linux-Administrator einsetzt – mit echten Szenarien, Troubleshooting und den Befehlen, die dir in der Prüfung und im Job das Leben retten.

Praktische Anwendung

Troubleshooting als Linux-Administrator

Du hast jetzt die Schichten, die Protokolle, die Adressierung und das Routing verstanden. Jetzt wird es richtig spannend: Wie setzt du das alles auf einem echten Linux-Server ein? Als Fachinformatiker Systemintegration oder Linux-Admin löst du 80 % aller Netzwerkprobleme genau hier – mit den Befehlen, die du jetzt lernst.

Wir gehen drei Alltags-Szenarien durch, verfolgen den kompletten Datenfluss durch alle vier Schichten und zeigen dir danach die komplette Troubleshooting-Toolbox, mit der du jedes Problem in Minuten findest.

Beispiel 1: Webseitenaufruf – der Klassiker

Du tippst `curl https://admindocs.de` oder öffnest den Browser. Was passiert wirklich?

Datenfluss Schritt für Schritt durch alle Schichten

- Anwendungsschicht: Browser/DNS-Anfrage (HTTP/HTTPS)
- Transportschicht: TCP-Verbindung aufbauen (Port 443)
- Internetschicht: IP-Routing zur Ziel-IP
- Netzzugangsschicht: Ethernet-Frame zum Router

Zuerst fragt der Browser den DNS-Server (`UDP Port 53`). Sobald die IP da ist, baut TCP den Three-Way-Handshake auf.

Dann fließen die HTTP-Daten verschlüsselt über TLS.

```
# Den kompletten Webaufruf live mitverfolgen
sudo tcpdump -i enp0s3 -nn 'port 53 or port 443' -c 30
```

Praktisches Docker-Beispiel Du startest einen Container und willst testen, ob er ins Internet kommt:

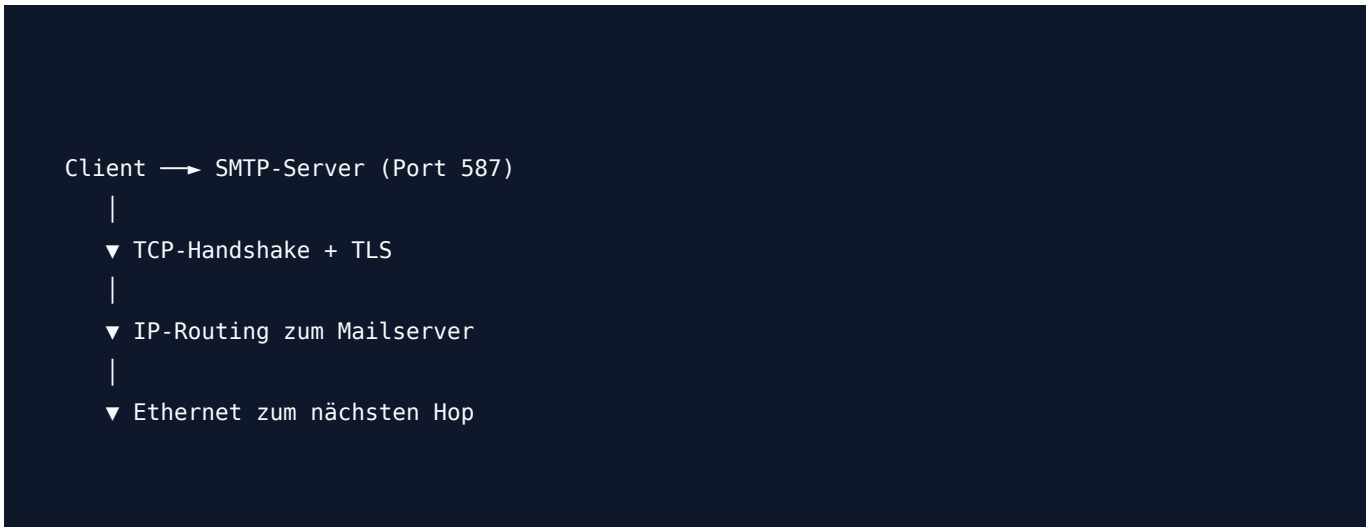
```
docker run --rm -it alpine curl -I https://admindocs.de
```

Der Container nutzt das Docker-Bridge-Netzwerk – du siehst die NAT-Übersetzung in der Routing-Tabelle.

Beispiel 2: E-Mail-Versand mit SMTP

Du schickst eine E-Mail über `mail` oder deinen Mail-Client. SMTP (Port 25 oder 587) läuft komplett über TCP.

SMTP-Datenfluss




```
# SMTP-Verbindung testen
telnet mail.admindocs.de 587
# oder sicherer:
openssl s_client -connect mail.admindocs.de:587 -starttls smtp
```

Warum TCP hier Pflicht ist Eine E-Mail darf kein einziges Byte verlieren – deshalb `ACKs` und `Retransmits` bei jedem Paket.

Beispiel 3: Video-Streaming mit UDP und QUIC

Netflix oder YouTube läuft heute oft über `UDP` (`Port 443` mit `QUIC/HTTP/3`). Kein Handshake, dafür extrem schnelle Pakete.

Streaming-Datenfluss

```
Browser ──► UDP-Pakete (QUIC)
   │
   ▼ Kein ACK – bei Verlust einfach nächstes Paket
   │
   ▼ IP-Routing + Ethernet
```

```
# QUIC-Verkehr erkennen
sudo tcpdump -i enp0s3 -nn udp port 443
```

Der Browser puffert ein paar Sekunden – deshalb merkst du kaum Verluste.

Die komplette Linux-Troubleshooting-Toolbox

Grundlegende Checks

```
# 1. IP-Konfiguration prüfen
ip addr show
ip route show

# 2. Verbindungen ansehen
ss -tuln
ss -tulp | grep ssh
```

Diagnose-Tools

```
# Ping (ICMP)
ping -c 4 8.8.8.8
ping -6 google.com

# Route verfolgen
traceroute 8.8.8.8
traceroute -6 google.com

# Pakete mitschneiden
sudo tcpdump -i any -nn -c 20 port 80 or port 443
```

Wireshark für Fortgeschrittene Installiere mit `sudo apt install wireshark` und starte die grafische Oberfläche.

Filter: `http` or `dns` or `tcp.port == 22`.

Typische Fehler und ihre Lösungen

 „No route to host“ → Routing-Tabelle fehlt oder Gateway down

```
ip route add default via 192.168.1.1
```

 DNS funktioniert nicht → `/etc/resolv.conf` falsch

```
cat /etc/resolv.conf
systemctl restart systemd-resolved
```

 Port ist belegt → `ss -tlnp | grep :80` zeigt den Prozess

```
sudo kill $(sudo ss -tlnp | grep :80 | awk '{print $6}' | cut -d, -f2)
```

Docker-spezifische Probleme

```
docker network inspect bridge
ip addr show docker0
```

Best Practices für sichere TCP/IP-Konfiguration

- Immer `ufw` oder `firewalld` nutzen und nur notwendige Ports freigeben
- IPv6 nie komplett deaktivieren
- `sysctl` für SYN-Flood-Schutz:

```
sudo sysctl -w net.ipv4.tcp_syncookies=1
```

 Tipp für den Alltag Lege dir ein Skript `netcheck.sh` an:

```
#!/bin/bash
```

```
echo "=== IP ==="; ip addr show
echo "=== Routes ==="; ip route show
echo "=== Listening Ports ==="; ss -tuln
echo "=== DNS Test ==="; host admindocs.de
```

Mach es ausführbar mit `chmod +x netcheck.sh` und starte es bei jedem Problem.

> ⚠ **Wichtigster Hinweis:** Die meisten Ausfälle kommen nicht vom Internet, sondern von falscher Konfiguration auf deinem Linux-Server. Mit den Befehlen oben findest du die Ursache in unter 2 Minuten.

Nachdem du diese Szenarien und Tools beherrschst, kannst du jedes Netzwerkproblem auf deinem Server systematisch lösen. Du bist jetzt kein Anfänger mehr – du bist der Admin, der genau weiß, was in den vier Schichten passiert.

Weiterführende Ressourcen

Du hast jetzt die komplette TCP/IP-Welt von den vier Schichten bis zum praktischen Troubleshooting durchgearbeitet. Damit du das Gelernte sofort vertiefen und langfristig parat hast, hier die besten Ressourcen, die ich als Linux-Admin täglich nutze. Alles ist praxisnah sortiert, damit du direkt loslegen kannst – ohne Umwege.

Interne Wiki-Artikel auf admindocs.de

Diese Beiträge bauen perfekt aufeinander auf und sind genau auf deine Zielgruppe (Fachinformatiker Systemintegration und Linux-Admins) zugeschnitten:

- Das OSI-Modell: Eine detaillierte Einführung in die 7 Schichten.
- Ultra Ethernet erklärt – Grundlagen, Einsatzbereiche und Praxisbeispiel.
- Glasfaser: Von der Technologie bis zum Hausanschluss.

Empfohlene Lesereihenfolge

1. OSI-Modell
2. TCP/IP (dieser Artikel)
3. Ultra Ethernet
4. Glasfaser

Offizielle Dokumentationen

Die echten Quellen – kurz, präzise und immer aktuell:

- `RFC 791` – Internet Protocol (IPv4)
- `RFC 793` – Transmission Control Protocol
- `RFC 768` – User Datagram Protocol
- `RFC 2460` – Internet Protocol, Version 6 (IPv6)
- `RFC 1035` – Domain Names (DNS)

```
# Schnell alle RFCs lokal speichern (für Offline-Nachschlage)
mkdir -p ~/rfc && cd ~/rfc
for rfc in 791 793 768 2460 1035; do
  curl -O https://www.rfc-editor.org/rfc/rfc$rfc.txt
done
```

Praktische Tools – deine tägliche Werkzeugkiste

Diese Programme gehören auf jeden Linux-Server und jede Admin-Workstation:

- `Wireshark` – der Grafik-Analyzer für alle Schichten
- `tcpdump` – der Kommandozeilen-Klassiker (immer vorinstalliert)
- `nmap` – Port-Scanner und Netzwerk-Entdecker
- `ss` und `ip` – die modernen Nachfolger von netstat und ifconfig

```
# Alles in einem Rutsch installieren (Debian/Ubuntu)
sudo apt update && sudo apt install -y wireshark tshark nmap tcpdump

# Auf Arch Linux
sudo pacman -S wireshark-cli nmap tcpdump
```

Tool-Vergleich auf einen Blick

```
Tool         | Grafisch | Schnell | Für Einsteiger | Live-Capture
Wireshark    | Ja       | Mittel  | Sehr gut       | Perfekt
tcpdump      | Nein     | Blitz   | Gut            | Perfekt
ss           | Nein     | Blitz   | Sehr gut       | Ports nur
nmap         | Optional | Mittel  | Gut            | Discovery
```

Lernressourcen für den nächsten Schritt

- `WCisco Networking Academy` – kostenlose Kurse mit praktischen Labs
- `TCP/IP Guide` (tcpipguide.com) – das Nachschlagewerk schlechthin
- `Buch: „Computer Networks: A Systems Approach“` (kostenlos online verfügbar)
- `LPIC-1 Vorbereitung`: Kapitel `109.1` und `109.2` – genau diese TCP/IP-Inhalte

```
# Schnell-Start für Wireshark-Lernen
wireshark &  # GUI starten
# oder im Terminal:
tshark -i enp0s3 -f "port 80 or port 443" -c 50
```

> ⚠ **Hinweis**: Die Protokolle und Standards entwickeln sich ständig weiter. Achte immer darauf, dass du die aktuellsten Versionen der Tools und RFCs verwendest. Mit den oben genannten Ressourcen bist du jedoch für die nächsten Jahre bestens gerüstet und kannst jedes Netzwerkproblem auf deinem Linux-Server souverän angehen.

Jetzt bist du bereit. Speichere diesen Artikel, lege deine `netcheck.sh` an und fang an, die Befehle täglich zu nutzen – dann wird TCP/IP für dich bald so selbstverständlich wie `ls` und `cd`.

Fazit

Die TCP/IP-Protokollfamilie ist das Fundament der modernen Internetkommunikation. Wie wir in diesem Artikel Schritt für Schritt durchgegangen sind, arbeiten die vier Schichten – `Netzzugangsschicht`, `Internetschicht`, `Transportschicht` und `Anwendungsschicht` – perfekt zusammen, um Daten sicher, zuverlässig und effizient von einem Gerät zum anderen zu bringen. Du hast gesehen, wie Ethernet-Frames auf der untersten Ebene die Bits transportieren, wie IP-Pakete ihren Weg durch Router finden, wie `TCP` mit seinem Three-Way-Handshake für absolute Zuverlässigkeit sorgt und wie `UDP` bei Streaming und DNS für Geschwindigkeit sorgt.

Du kennst jetzt die wichtigsten Protokolle im Detail: `TCP` und `UDP` mit allen Vor- und Nachteilen, `IP` in `IPv4` und `IPv6`, `ICMP` für Diagnosen, `ARP` für die lokale Adressauflösung und `DNS`, `HTTP/HTTPS` sowie `QUIC` als moderne Ergänzung. Du verstehst, warum Ports wie 22, 53, 80 und 443 so wichtig sind und wie du mit `ip`, `ss`, `tcpdump` und `traceroute` auf deinem Linux-Server live verfolgen kannst, was wirklich passiert.

Besonders wertvoll für dich als angehender Fachinformatiker Systemintegration oder Linux-Administrator ist das praktische Wissen: Du kannst jetzt einen Webseitenaufruf, einen E-Mail-Versand oder ein Video-Streaming durch alle Schichten verfolgen, Docker-Container-Netzwerke debuggen und typische Fehler wie `No route to host`, DNS-Probleme oder MTU-Fehler in Minuten beheben. Mit deiner eigenen `netcheck.sh` und den Troubleshooting-Befehlen hast du ein echtes Werkzeug in der Hand, das dir im Job und in der LPIC-1-Prüfung sofort weiterhilft.

> ⚠ Die meisten Netzwerkprobleme entstehen nicht irgendwo im Internet, sondern direkt auf deinem Server – durch falsche Routen, blockierte Ports oder IPv6-Konfigurationsfehler. Mit dem Verständnis von TCP/IP siehst du diese Ursachen jetzt klar vor dir.

Nimm dir die Zeit und arbeite die drei Praxis-Beispiele aktiv nach. Starte `tcpdump`, rufe eine Webseite auf, sende eine Test-E-Mail und analysiere den Traffic. Je öfter du die Befehle anwendest, desto selbstverständlicher wird das gesamte System für dich.

Die TCP/IP-Protokollfamilie ist seit Jahrzehnten das Rückgrat des Internets – und sie wird es auch bleiben. Du hast jetzt das solide Fundament, um sie nicht nur zu verstehen, sondern aktiv zu beherrschen. Nutze dieses Wissen täglich auf deinen Servern, in Docker-Umgebungen und bei der Fehlersuche. Dann wirst du nicht nur die Prüfung mit Bravour bestehen, sondern auch in der echten Administration der Admin sein, der Probleme löst, bevor andere sie überhaupt bemerken.