7.2 TCP und UDP im Detail

TCP und UDP sind die wichtigsten Transportprotokolle auf OSI-Schicht 4.

Beide haben dieselbe Grundaufgabe:

```
Daten zwischen Anwendungen auf Endgeräten transportieren.
```

Der große Unterschied liegt darin, wie sie das machen.

TCP arbeitet:

```
verbindungsorientiert
zuverlässig
mit Bestätigungen
```

UDP arbeitet:

```
verbindungslos
schlanker
ohne eingebaute Zustellgarantie
```

Merksatz:

```
TCP = zuverlässig.
UDP = schnell und einfach.
```

Warum gibt es TCP und UDP?

Nicht jede Anwendung hat dieselben Anforderungen.

Manche Anwendungen brauchen sichere und vollständige Datenübertragung.

Beispiele:

```
- Webseite laden
- Datei übertragen
- E-Mail senden
```

- SSH-Verbindung
- Datenbankzugriff

Andere Anwendungen brauchen vor allem geringe Verzögerung.

Beispiele:

- VoIP
- Livestreaming
- Online-Gaming
- DNS-Anfragen
- Video-Konferenzen

Deshalb gibt es unterschiedliche Transportprotokolle.

Merksatz:

Unterschiedliche Anwendungen brauchen unterschiedliche Transportarten.

TCP-Grundidee

TCP steht für:

Transmission Control Protocol

TCP stellt eine Verbindung zwischen zwei Anwendungen her.

Dabei sorgt TCP dafür, dass Daten:

- vollständig ankommen
- in der richtigen Reihenfolge ankommen
- bestätigt werden
- bei Verlust erneut gesendet werden
- kontrolliert übertragen werden

Merksatz:

TCP kümmert sich um zuverlässige Datenübertragung.

UDP-Grundidee

UDP steht für:

```
User Datagram Protocol
```

UDP sendet einzelne Datagramme ohne vorherigen Verbindungsaufbau.

UDP kümmert sich nicht selbst darum, ob Daten:

```
- angekommen sind
- vollständig angekommen sind
- in richtiger Reihenfolge angekommen sind
- erneut gesendet werden müssen
```

Dafür ist UDP sehr schlank.

Merksatz:

```
UDP sendet einfach,
ohne Verbindung aufzubauen.
```

TCP und UDP direkt vergleichen

Merkmal	TCP	UDP
Verbindungsaufbau	ja	nein
Verbindung	verbindungsorientiert	verbindungslos
Zustellgarantie	ja, durch Mechanismen	nein
Reihenfolge	wird sichergestellt	nicht sichergestellt
Bestätigungen	ja	nein
erneute Übertragung	ja	nein
Geschwindigkeit	mehr Overhead	weniger Overhead
Datenform	Segment	Datagramm
typische Nutzung	HTTP, HTTPS, SSH, Mail	DNS, DHCP, VoIP, Streaming

Merksatz:

```
TCP kontrolliert mehr.
UDP ist einfacher.
```

Verbindungsorientiert bei TCP

Verbindungsorientiert bedeutet:

```
Vor der eigentlichen Datenübertragung wird eine Verbindung aufgebaut.
```

TCP prüft dabei, ob beide Seiten bereit sind.

Der Verbindungsaufbau erfolgt durch den Drei-Wege-Handshake:

```
SYN
SYN-ACK
ACK
```

Erst danach werden Nutzdaten übertragen.

Merksatz:

```
TCP baut vor der Datenübertragung eine Verbindung auf.
```

Verbindungslos bei UDP

Verbindungslos bedeutet:

```
Es gibt keinen Verbindungsaufbau wie bei TCP.
```

UDP sendet ein Datagramm direkt an Ziel-IP und Ziel-Port.

Der Sender erhält durch UDP selbst keine sichere Information darüber:

```
ob das Datagramm angekommen ist
ob es verworfen wurde
ob mehrere Datagramme in richtiger Reihenfolge ankamen
```

Merksatz:

```
UDP hat keinen Drei-Wege-Handshake.
```

TCP-Drei-Wege-Handshake

Der TCP-Verbindungsaufbau besteht aus drei Schritten.

Schritt	Richtung	Bedeutung
1	Client → Server	SYN
2	Server → Client	SYN-ACK
3	Client → Server	ACK

Vereinfacht:

```
Client fragt an.
Server bestätigt und fragt zurück.
Client bestätigt.
```

Danach ist die Verbindung aufgebaut.

Merksatz:

```
TCP-Handshake = SYN, SYN-ACK, ACK.
```

SYN

SYN steht für:

```
Synchronize
```

Ein SYN wird beim TCP-Verbindungsaufbau gesendet.

Der Client sagt damit vereinfacht:

```
Ich möchte eine TCP-Verbindung aufbauen.
```

Das SYN enthält unter anderem Startinformationen für die TCP-Kommunikation.

Merksatz:

SYN startet den TCP-Verbindungsaufbau.

SYN-ACK

SYN-ACK ist die Antwort des Servers auf das SYN des Clients.

Der Server sagt damit vereinfacht:

Ich habe deine Anfrage erhalten
und bin bereit für die Verbindung.

SYN-ACK enthält:

SYN des Servers
ACK als Bestätigung des Client-SYN

Merksatz:

SYN-ACK bestätigt die Anfrage und synchronisiert zurück.

ACK

ACK steht für:

Acknowledgement

ACK bedeutet:

Bestätigung

Im dritten Schritt bestätigt der Client das SYN-ACK des Servers.

Danach gilt die TCP-Verbindung als aufgebaut.

Merksatz:

ACK bestätigt empfangene TCP-Informationen.

TCP-Datenübertragung nach dem Handshake

Nach dem Drei-Wege-Handshake können Nutzdaten übertragen werden.

TCP nutzt dafür:

- Sequenznummern
- Bestätigungsnummern
- Prüfsummen
- Fenstergröße
- erneute Übertragung bei Verlust

Dadurch wird die Übertragung zuverlässiger.

Merksatz:

Nach dem Handshake sorgt TCP für geordnete Datenübertragung.

Sequenznummern

TCP nutzt Sequenznummern, um Daten zu ordnen.

Sie helfen dabei:

- Reihenfolge wiederherzustellen
- fehlende Daten zu erkennen
- Daten korrekt zu bestätigen

Beispiel:

Teil 1 kommt an.
Teil 3 kommt an.
Teil 2 fehlt.

TCP erkennt:

Es fehlt etwas.

Dann kann die fehlende Information erneut übertragen werden.

Merksatz:

```
Sequenznummern helfen TCP,
Daten richtig zu sortieren.
```

Bestätigungsnummern

Bestätigungsnummern zeigen dem Sender, welche Daten angekommen sind.

Der Empfänger bestätigt nicht nur:

```
Ich habe etwas bekommen.
```

Sondern sinngemäß:

```
Ich habe alles bis zu dieser Stelle bekommen
und erwarte als Nächstes diese Nummer.
```

Merksatz:

```
TCP bestätigt empfangene Daten mit ACKs.
```

Erneute Übertragung bei TCP

Wenn TCP erkennt, dass Daten fehlen, können sie erneut übertragen werden.

Mögliche Gründe für erneute Übertragung:

```
- Paketverlust
- beschädigte Daten
- keine Bestätigung erhalten
- Timeout
- Überlastung im Netz
```

Dadurch ist TCP zuverlässiger als UDP.

Merksatz:

TCP kann verlorene Daten erneut senden.

Reihenfolge bei TCP

IP-Pakete können im Netzwerk unterschiedliche Wege nehmen.

Dadurch können Daten in anderer Reihenfolge ankommen.

TCP kann sie wieder richtig zusammensetzen.

Beispiel:

Gesendet:
1, 2, 3, 4

Angekommen:
1, 3, 2, 4

TCP übergibt der Anwendung die Daten wieder in korrekter Reihenfolge.

Merksatz:

TCP sorgt für richtige Reihenfolge.

Flusskontrolle

Flusskontrolle bedeutet:

Der Sender soll den Empfänger nicht überfordern.

TCP nutzt dafür unter anderem die Fenstergröße.

Die Fenstergröße gibt an, wie viele Daten gesendet werden dürfen, bevor weitere Bestätigungen nötig sind.

Vereinfacht:

Empfänger sagt:
So viel kann ich gerade aufnehmen.

Merksatz:

> Flusskontrolle schützt den Empfänger vor Überlastung.

Staukontrolle

Staukontrolle bedeutet:

> TCP versucht, Überlastung im Netzwerk zu vermeiden.

Wenn Paketverlust oder Verzögerung auftritt, kann TCP die Sendeleistung anpassen.

Ziel:

> Das Netzwerk soll nicht weiter überlastet werden.

Merksatz:

> Staukontrolle schützt das Netzwerk vor Überlastung.

TCP-Verbindungsabbau

TCP-Verbindungen werden geordnet beendet.

Dabei kommen häufig FIN- und ACK-Nachrichten vor.

Vereinfacht:

> Eine Seite sagt:
> Ich möchte die Verbindung beenden.
>
> Die andere Seite bestätigt.

Es gibt auch RST.

RST bedeutet:

> Verbindung sofort abbrechen.

Merksatz:

```
FIN beendet geordnet.
RST bricht ab.
```

TCP-Zustände

TCP kennt verschiedene Zustände.

Wichtige Beispiele:

Zustand	Bedeutung
LISTEN	Dienst wartet auf eingehende Verbindung
SYN-SENT	Client hat SYN gesendet
SYN-RECEIVED	Server hat SYN erhalten und SYN-ACK gesendet
ESTABLISHED	Verbindung besteht
FIN-WAIT	Verbindung wird beendet
TIME-WAIT	Wartephase nach Verbindungsende
CLOSED	Verbindung geschlossen

Merksatz:

```
ESTABLISHED bedeutet:
TCP-Verbindung steht.
```

UDP-Datagramm

UDP nutzt Datagramme.

Ein UDP-Datagramm enthält unter anderem:

```
- Quell-Port
- Ziel-Port
- Länge
- Prüfsumme
- Nutzdaten
```

UDP hat deutlich weniger Steuerinformationen als TCP.

Merksatz:

```
UDP-Datagramm = einfache Transport-Dateneinheit.
```

Warum UDP trotz fehlender Garantie sinnvoll ist

UDP ist sinnvoll, wenn Geschwindigkeit oder geringe Verzögerung wichtiger ist als perfekte Vollständigkeit.

Beispiel VoIP:

```
Wenn ein kleines Audiopaket verloren geht,
ist es oft besser,
einfach weiterzumachen.
```

Eine spätere erneute Übertragung wäre zu spät und würde nur stören.

Merksatz:

```
Bei Echtzeitdiensten ist spät manchmal schlimmer als verloren.
```

UDP bei DNS

DNS nutzt häufig UDP Port 53.

Warum?

```
DNS-Anfragen sind oft kurz.
DNS-Antworten sind oft klein.
Es soll schnell gehen.
```

DNS kann aber auch TCP Port 53 verwenden.

Beispiele:

```
- große Antworten
- Zonentransfers
- bestimmte DNSSEC-Fälle
```

Merksatz:

> DNS nutzt häufig UDP 53,
> kann aber auch TCP 53 nutzen.

UDP bei DHCP

DHCP nutzt UDP.

Wichtige Ports:

Richtung	Port
DHCP-Server	UDP 67
DHCP-Client	UDP 68

Warum UDP?

> Ein Client hat am Anfang oft noch keine vollständige IP-Konfiguration.
> DHCP muss früh im Netzwerkprozess funktionieren.

Merksatz:

> DHCP nutzt UDP 67 und 68.

UDP bei VoIP und Streaming

VoIP, Videokonferenzen und Streaming nutzen häufig UDP.

Grund:

> geringe Verzögerung ist wichtig.

Wenn ein Paket verloren geht, ist eine spätere Wiederholung oft nicht sinnvoll.

Beispiel:

> Ein verlorener Tonfetzen bei Telefonie ist kurz störend.
> Eine verspätete Wiederholung würde das Gespräch stärker stören.

Merksatz:

```
Echtzeitkommunikation nutzt häufig UDP.
```

TCP bei Web und Dateiübertragung

Webseiten, Dateiübertragungen und viele Verwaltungsdienste nutzen TCP.

Beispiele:

```
- HTTP
- HTTPS
- SSH
- FTP
- SMTP
- IMAP
- POP3
```

Warum?

```
Daten sollen vollständig und korrekt ankommen.
```

Bei einer Datei wäre es schlecht, wenn Teile fehlen oder vertauscht sind.

Merksatz:

```
Wenn Vollständigkeit wichtig ist,
ist TCP häufig geeignet.
```

TCP und HTTPS

HTTPS nutzt typischerweise TCP Port 443.

Dabei kommen mehrere Dinge zusammen:

```
IP-Adresse für das Zielgerät
TCP-Port 443 für den Dienst
TCP-Verbindung für Transport
```

TLS für Verschlüsselung
HTTP für die Webanwendung

Merksatz:

HTTPS nutzt TCP,
ist aber selbst eine höhere Anwendung mit TLS.

UDP und QUIC

QUIC ist ein modernes Transportprotokoll, das auf UDP basiert.

HTTP/3 nutzt QUIC.

Warum UDP als Grundlage?

QUIC bringt eigene Mechanismen für Verbindungen, Sicherheit und Zuverlässigkeit mit,
nutzt aber UDP als Transportbasis.

Für AP1/AP2 reicht meistens:

HTTP/3 nutzt QUIC über UDP.

Merksatz:

UDP bedeutet nicht automatisch,
dass keine Zuverlässigkeit möglich ist;
sie kann in höheren Protokollen umgesetzt werden.

TCP-Overhead

TCP hat mehr Overhead als UDP.

Warum?

- Verbindungsaufbau
- Bestätigungen
- Sequenznummern

- erneute Übertragung
- Flusskontrolle
- Staukontrolle
- Verbindungsabbau

Dieser Overhead lohnt sich, wenn zuverlässige Übertragung wichtig ist.

Merksatz:

TCP hat mehr Kontrolle,
aber auch mehr Aufwand.

UDP-Overhead

UDP hat wenig Overhead.

Warum?

- kein Handshake
- keine TCP-Bestätigungen
- keine TCP-Sequenzsteuerung
- kein TCP-Verbindungszustand

Das macht UDP schlank und schnell.

Aber:

Die Anwendung muss selbst entscheiden,
ob sie zusätzliche Zuverlässigkeit braucht.

Merksatz:

UDP ist schlank,
aber einfacher.

Ports bei TCP und UDP

TCP und UDP haben jeweils eigene Portbereiche.

Das bedeutet:

```
TCP 53 und UDP 53 sind nicht derselbe Kommunikationskanal.
```

Beispiel DNS:

```
UDP 53 für typische Abfragen
TCP 53 für bestimmte Fälle
```

Eine Firewall-Regel für TCP 53 erlaubt nicht automatisch UDP 53.

Merksatz:

```
TCP-Port und UDP-Port sind getrennt zu betrachten.
```

Gleiche Portnummer, anderes Protokoll

Die gleiche Portnummer kann bei TCP und UDP unterschiedliche Bedeutung haben.

Beispiel:

```
TCP 53
UDP 53
```

Beide gehören zu DNS, aber technisch sind es verschiedene Transportprotokolle.

Beispiel:

```
TCP 443
UDP 443
```

TCP 443 ist klassisches HTTPS.

UDP 443 kann bei QUIC / HTTP/3 relevant sein.

Merksatz:

```
Portnummer immer zusammen mit TCP oder UDP betrachten.
```

Quell-Port und Ziel-Port bei TCP

Beispiel HTTPS:

```
Client:
192.168.10.20:52344

Server:
93.184.216.34:443
```

Dabei ist:

```
52344 = temporärer Quell-Port des Clients
443 = Ziel-Port des HTTPS-Dienstes
```

Die Antwort kommt zurück an:

```
192.168.10.20:52344
```

Merksatz:

```
Ziel-Port zeigt den Dienst.
Quell-Port hilft beim Rückweg zur Verbindung.
```

Quell-Port und Ziel-Port bei UDP

Auch UDP nutzt Quell-Port und Ziel-Port.

Beispiel DNS:

```
Client:
192.168.10.20:54001

DNS-Server:
192.168.10.1:53
```

Der Client sendet von einem temporären Port an UDP 53.

Die Antwort kommt zurück an den temporären Client-Port.

Merksatz:

```
Auch UDP nutzt Ports für Dienst und Rückzuordnung.
```

Ephemeral Ports

Ephemeral Ports sind temporäre Client-Ports.

Sie werden für ausgehende Verbindungen genutzt.

Typischer Bereich:

```
49152 bis 65535
```

Beispiel:

```
Client nutzt 53210 als Quell-Port
für Verbindung zu Server TCP 443.
```

Nach Ende der Verbindung kann der Port später wiederverwendet werden.

Merksatz:

```
Ephemeral Port = temporärer Quell-Port eines Clients.
```

Well-Known Ports

Well-Known Ports liegen im Bereich:

```
0 bis 1023
```

Sie sind bekannten Standarddiensten zugeordnet.

Beispiele:

Dienst	Protokoll	Port
FTP	TCP	21
SSH	TCP	22

Dienst	Protokoll	Port
Telnet	TCP	23
SMTP	TCP	25
DNS	TCP / UDP	53
HTTP	TCP	80
HTTPS	TCP	443

Merksatz:

> Well-Known Ports sollte man für die Prüfung sicher kennen.

TCP-Fehlersuche

Typische TCP-Fragen:

> Ist der Zielhost erreichbar?
> Ist der Ziel-Port offen?
> Lauscht der Dienst auf dem Port?
> Kommt ein SYN beim Server an?
> Antwortet der Server mit SYN-ACK?
> Blockiert eine Firewall?
> Wird die Verbindung zurückgesetzt?
> Gibt es NAT- oder Portweiterleitungsfehler?

Merksatz:

> Bei TCP prüft man besonders den Verbindungsaufbau.

UDP-Fehlersuche

UDP-Fehlersuche ist oft schwieriger.

Warum?

> Es gibt keinen Handshake.

Wenn keine Antwort kommt, ist nicht sofort klar:

```
Paket verloren?
Dienst nicht erreichbar?
Firewall blockiert?
Anfrage ungültig?
Antwortweg gestört?
NAT-Problem?
```

Deshalb braucht man oft Logs, Mitschnitte oder passende Testwerkzeuge.

Merksatz:

```
UDP-Fehler sind schwerer eindeutig zu erkennen als TCP-Fehler.
```

TCP-Handshake scheitert

Wenn ein TCP-Handshake scheitert, kann die Verbindung nicht aufgebaut werden.

Mögliche Ursachen:

```
- Server nicht erreichbar
- Dienst hört nicht auf dem Port
- Firewall blockiert SYN
- Firewall blockiert SYN-ACK
- NAT oder Portweiterleitung falsch
- Server lehnt Verbindung ab
- Rückroute fehlt
```

Merksatz:

```
Kein TCP-Handshake = keine TCP-Verbindung.
```

TCP Reset

Ein TCP Reset wird mit RST gekennzeichnet.

RST bedeutet:

```
Verbindung sofort abbrechen.
```

Mögliche Ursachen:

```
- Port geschlossen
- Dienst lehnt Verbindung ab
- Anwendung beendet Verbindung
- Firewall sendet Reset
- fehlerhafte Verbindung
```

Merksatz:

```
RST = TCP-Verbindung wird sofort zurückgesetzt.
```

TCP Timeout

Ein Timeout bedeutet:

```
Eine erwartete Antwort kommt nicht rechtzeitig.
```

Mögliche Ursachen:

```
- Paket geht verloren
- Firewall verwirft still
- Server antwortet nicht
- Rückweg fehlt
- Routing-Problem
- NAT-Problem
```

Merksatz:

```
Timeout bedeutet:
keine rechtzeitige Antwort.
```

Gefilterter Port

Ein Port wirkt gefiltert, wenn eine Firewall Pakete blockiert oder verwirft.

Unterschied:

geschlossen:
Ziel antwortet aktiv, dass kein Dienst vorhanden ist.

gefiltert:
keine oder keine klare Antwort.

Merksatz:

Gefiltert bedeutet oft:
Firewall im Weg.

Offener Port

Ein Port ist offen, wenn ein Dienst auf diesem Port Verbindungen annimmt oder Anfragen beantwortet.

Beispiel:

TCP 443 offen

bedeutet:

Ein Dienst akzeptiert HTTPS-Verbindungen oder zumindest TCP-Verbindungen auf Port 443.

Aber:

Das sagt noch nicht,
dass die Webanwendung korrekt funktioniert.

Merksatz:

Offener Port bedeutet:
Dienst ist grundsätzlich erreichbar.

Geschlossener Port

Ein Port ist geschlossen, wenn kein Dienst auf diesem Port lauscht.

Beispiel:

```
Server erreichbar,
aber TCP 22 geschlossen
```

Das bedeutet:

```
Der Server antwortet,
aber SSH läuft dort nicht oder nicht auf Port 22.
```

Merksatz:

```
Geschlossener Port heißt:
Host kann erreichbar sein,
Dienst aber nicht.
```

TCP und Paketmitschnitt

In einem Mitschnitt kann man TCP gut erkennen.

Wichtige Hinweise:

```
SYN
SYN-ACK
ACK
RST
FIN
Retransmission
Window Size
Duplicate ACK
```

Damit kann man sehen:

```
Wird die Verbindung aufgebaut?
Wo bleibt die Antwort aus?
Wird zurückgesetzt?
Gibt es Paketverlust?
```

Merksatz:

```
TCP lässt sich im Mitschnitt gut analysieren.
```

UDP und Paketmitschnitt

Bei UDP sieht man im Mitschnitt:

```
Quell-Port
Ziel-Port
Länge
Prüfsumme
Nutzdaten
```

Aber man sieht keinen Handshake.

Deshalb muss man prüfen:

```
Geht die Anfrage raus?
Kommt eine Antwort zurück?
Ist die Antwort korrekt?
Wird etwas blockiert?
Ist der Port richtig?
```

Merksatz:

```
Bei UDP vergleicht man Anfrage und Antwort.
```

TCP oder UDP bei Firewall-Regeln

Firewall-Regeln müssen das richtige Protokoll berücksichtigen.

Beispiel:

```
DNS erlauben
```

Nicht ausreichend, wenn nur steht:

```
Port 53 erlauben
```

Genauer:

```
UDP 53 erlauben
TCP 53 bei Bedarf ebenfalls erlauben
```

Beispiel HTTPS:

```
TCP 443 erlauben
```

Bei HTTP/3 kann zusätzlich relevant sein:

```
UDP 443 erlauben
```

Merksatz:

```
Firewall-Regeln brauchen Protokoll und Port.
```

TCP oder UDP bei Portweiterleitungen

Auch Portweiterleitungen müssen das richtige Protokoll nutzen.

Beispiel:

```
Webserver klassisch:
TCP 80
TCP 443
```

Beispiel DNS-Server:

```
UDP 53
TCP 53
```

Beispiel Spielserver:

je nach Spiel oft UDP oder TCP unterschiedlich

Prüfungsfalle:

TCP-Portweiterleitung hilft nicht,
wenn der Dienst UDP nutzt.

Merksatz:

Portweiterleitung muss TCP oder UDP richtig treffen.

NAT/PAT und TCP/UDP

PAT nutzt Port-Zuordnungen.

Bei TCP kann ein NAT-Gerät Verbindungszustände gut verfolgen.

Bei UDP gibt es keinen echten Verbindungszustand.

Deshalb nutzt NAT bei UDP Zeitfenster und Zuordnungstabellen.

Wenn die Zuordnung abläuft, können Antworten verloren gehen.

Merksatz:

NAT verfolgt TCP leichter als UDP.

Typische Anwendungen und Transportprotokolle

Anwendung / Dienst	typisches Transportprotokoll
HTTP	TCP
HTTPS	TCP
HTTP/3	UDP
SSH	TCP
Telnet	TCP
SMTP	TCP
IMAP	TCP

Anwendung / Dienst	typisches Transportprotokoll
POP3	TCP
DNS	UDP und TCP
DHCP	UDP
NTP	UDP
VoIP	häufig UDP
Streaming	häufig UDP oder TCP je nach Technik
VPN	je nach Lösung TCP oder UDP

Merksatz:

> Dienst immer mit Protokoll und Port betrachten.

TCP und UDP in der Praxisentscheidung

TCP ist sinnvoll, wenn wichtig ist:

> - Vollständigkeit
> - Reihenfolge
> - Zuverlässigkeit
> - fehlerfreie Dateiübertragung
> - stabile Verbindung

UDP ist sinnvoll, wenn wichtig ist:

> - geringe Latenz
> - wenig Overhead
> - Echtzeitverhalten
> - kurze Anfragen
> - Anwendung übernimmt eigene Steuerung

Merksatz:

> TCP für Genauigkeit.
> UDP für Geschwindigkeit und Echtzeit.

Einordnung in das OSI-Modell

Thema	Schicht
IP-Adresse	3
Routing	3
ICMP	3
TCP	4
UDP	4
Port	4
TCP-Handshake	4
Sequenznummer	4
ACK	4
DNS-Dienst	7, nutzt TCP/UDP 53
HTTP	7, nutzt TCP
HTTPS	7, nutzt TCP und TLS
HTTP/3	7, nutzt QUIC über UDP

Merksatz:

> TCP und UDP gehören zur Transportschicht.

Typische IHK-Fragen

In AP1 und AP2 kann zum Beispiel gefragt werden:

- Was ist TCP?
- Was ist UDP?
- Worin unterscheiden sich TCP und UDP?
- Was bedeutet verbindungsorientiert?
- Was bedeutet verbindungslos?
- Wie funktioniert der TCP-Drei-Wege-Handshake?
- Was bedeuten SYN, SYN-ACK und ACK?
- Warum ist TCP zuverlässig?
- Warum ist UDP schneller beziehungsweise schlanker?
- Wann eignet sich TCP?
- Wann eignet sich UDP?

- Warum nutzt DNS UDP und TCP?
- Warum nutzt DHCP UDP?
- Warum nutzen Echtzeitdienste häufig UDP?
- Warum kann Ping funktionieren, aber TCP 443 nicht?
- Warum muss eine Firewall zwischen TCP und UDP unterscheiden?

Typische Prüfungsfallen

TCP ist verbindungsorientiert.

UDP ist verbindungslos.

TCP nutzt den Drei-Wege-Handshake.

UDP nutzt keinen Drei-Wege-Handshake.

TCP bietet Zuverlässigkeit durch Bestätigungen und erneute Übertragung.

UDP bietet keine eingebaute Zustellgarantie.

TCP stellt Reihenfolge sicher.

UDP stellt Reihenfolge nicht selbst sicher.

TCP nutzt Segmente.

UDP nutzt Datagramme.

DNS nutzt UDP 53 und TCP 53.

DHCP nutzt UDP 67 und 68.

HTTP/3 nutzt QUIC über UDP.

Gleiche Portnummer bei TCP und UDP ist technisch nicht dasselbe.

Firewall-Regeln müssen TCP oder UDP korrekt angeben.

Portweiterleitungen müssen TCP oder UDP korrekt treffen.

Ping nutzt ICMP,
nicht TCP oder UDP.

Wichtige Begriffe kurz erklärt

Begriff	Kurze Erklärung
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
verbindungsorientiert	Verbindung wird vor Datenübertragung aufgebaut
verbindungslos	Daten werden ohne vorherige Verbindung gesendet
SYN	Start einer TCP-Verbindung
SYN-ACK	Antwort auf SYN mit Bestätigung
ACK	Bestätigung
Drei-Wege-Handshake	TCP-Verbindungsaufbau mit SYN, SYN-ACK, ACK
Sequenznummer	Position von Daten im TCP-Strom
Bestätigungsnummer	bestätigt empfangene Daten
Retransmission	erneute Übertragung
Flusskontrolle	schützt Empfänger vor Überlastung
Staukontrolle	schützt Netzwerk vor Überlastung
FIN	geordneter TCP-Verbindungsabbau
RST	sofortiger TCP-Abbruch
Timeout	erwartete Antwort bleibt zu lange aus
Segment	TCP-Dateneinheit
Datagramm	UDP-Dateneinheit

IHK-sichere Kurzformulierung

TCP und UDP sind Transportprotokolle auf OSI-Schicht 4. TCP ist verbindungsorientiert und baut vor der Datenübertragung mit SYN, SYN-ACK und ACK eine Verbindung auf. Es bietet Zuverlässigkeit durch Bestätigungen, Sequenznummern, erneute Übertragung und Reihenfolgesicherung. UDP ist verbindungslos und sendet Datagramme ohne Verbindungsaufbau und ohne eingebaute Garantie für Zustellung oder Reihenfolge. Dadurch ist UDP schlanker und eignet sich gut für Echtzeitdienste oder kurze Anfragen. TCP- und UDP-Ports sind getrennt zu betrachten, auch wenn dieselbe Portnummer verwendet wird.

Merksätze

TCP = Transmission Control Protocol.

UDP = User Datagram Protocol.

Beide gehören zu Schicht 4.

TCP ist verbindungsorientiert.

UDP ist verbindungslos.

TCP baut Verbindung auf.

UDP sendet ohne Handshake.

TCP-Handshake = SYN, SYN-ACK, ACK.

TCP nutzt ACKs.

TCP nutzt Sequenznummern.

TCP kann erneut übertragen.

TCP stellt Reihenfolge sicher.

UDP bietet keine eingebaute Zustellgarantie.

UDP ist schlanker als TCP.

TCP = Segment.

UDP = Datagramm.

DNS nutzt UDP und TCP 53.

DHCP nutzt UDP 67 und 68.

HTTP/3 nutzt QUIC über UDP.

Gleiche Portnummer bei TCP und UDP ist nicht dasselbe.

Firewall-Regel braucht Protokoll und Port.

TCP-Portweiterleitung hilft nicht bei UDP-Dienst.

Ping ist ICMP,
nicht TCP oder UDP.

TCP für Zuverlässigkeit.

UDP für geringe Verzögerung.