TCP (Transmission Control Protocol) Erklärung

TCP einfach erklärt

TCP steht für **Transmission Control Protocol**.

TCP ist ein **verbindungsorientiertes** und **zuverlässiges** Transportprotokoll. Es arbeitet auf der **Transportschicht** des TCP/IP-Modells und sorgt dafür, dass Daten möglichst vollständig, geordnet und fehlerfrei beim Empfänger ankommen.

Typische Anwendungen mit TCP sind zum Beispiel:

```
HTTP/HTTPS
SSH
E-Mail
Dateiübertragung
Remote Desktop
```

Grundidee von TCP

TCP wird verwendet, wenn Daten zuverlässig übertragen werden sollen.

Das bedeutet:

```
- vor der Datenübertragung wird eine Verbindung aufgebaut
- Daten werden nummeriert
- empfangene Daten werden bestätigt
- fehlende Daten können erneut übertragen werden
- Daten werden in der richtigen Reihenfolge an die Anwendung weitergegeben
```

TCP ist dadurch zuverlässiger als UDP, hat aber mehr Verwaltungsaufwand.

Merksatz:

```
TCP = verbindungsorientiert, zuverlässig und geordnet
UDP = verbindungslos, schnell und mit weniger Kontrolle
```

TCP arbeitet mit Ports

Eine IP-Adresse zeigt auf einen Host im Netzwerk.

Beispiel:

```
192.168.0.50
```

Ein Port zeigt auf einen bestimmten Dienst oder eine Anwendung auf diesem Host.

Beispiel:

```
192.168.0.50:443
```

Das bedeutet:

```
192.168.0.50 = Host / Gerät
443          = Dienst / Anwendung, hier HTTPS
```

Typische TCP-Ports:

```
80   = HTTP
443  = HTTPS
22   = SSH
25   = SMTP
110  = POP3
143  = IMAP
3389 = Remote Desktop
```

Merksatz:

```
IP-Adresse = welcher Host?
Port       = welcher Dienst?
Socket     = IP-Adresse + Port
```

Socket bei TCP

Ein Socket ist die Kombination aus IP-Adresse und Port.

Beispiel:

```
192.168.0.50:443
```

Bei einer TCP-Verbindung gibt es immer zwei Seiten:

```
Client-Socket: 192.168.0.20:51544
Server-Socket: 93.184.216.34:443
```

Der Client verwendet oft einen zufälligen hohen Quellport. Der Server verwendet meistens einen bekannten festen Zielport.

Eine vollständige TCP-Verbindung wird also durch diese Informationen beschrieben:

```
Quell-IP
Quell-Port
Ziel-IP
Ziel-Port
Protokoll TCP
```

TCP-Verbindungsaufbau: 3-Wege-Handshake

Bevor TCP Daten überträgt, wird eine Verbindung aufgebaut. Dieser Verbindungsaufbau heißt **3-Wege-Handshake**.

Dabei tauschen Client und Server Kontrollinformationen aus.

Ablauf:

```
Client                    Server

1. SYN  ---------------------->
   Verbindungswunsch

2. SYN-ACK <------------------
   Verbindungswunsch bestätigt

3. ACK  ---------------------->
   Bestätigung zurück

Verbindung steht
```

Was bedeutet SYN?

SYN ist die Abkürzung für:

```
Synchronize
```

Auf Deutsch:

```
synchronisieren
```

Beim TCP-Verbindungsaufbau bedeutet SYN:

```
Der Client möchte eine neue TCP-Verbindung starten
und seine Start-Sequenznummer mit dem Server synchronisieren.
```

SYN ist also nicht einfach nur „Hallo“, sondern enthält auch technische Informationen für den Start der Verbindung.

Was bedeutet SYN-ACK?

SYN-ACK besteht aus zwei Teilen:

```
SYN = Server möchte ebenfalls seine Start-Sequenznummer synchronisieren
ACK = Server bestätigt den SYN des Clients
```

Der Server sagt damit vereinfacht:

```
Ich habe deinen Verbindungswunsch erhalten.
Ich bin bereit.
Hier ist meine eigene Start-Sequenznummer.
```

Was bedeutet ACK?

ACK steht für:

```
Acknowledgement
```

Auf Deutsch:

```
Bestätigung
```

Beim dritten Schritt bestätigt der Client die Antwort des Servers.

Danach gilt:

```
Die TCP-Verbindung ist aufgebaut.
Daten können übertragen werden.
```

Merksatz:

```
SYN     = Verbindungswunsch + Start-Sequenznummer synchronisieren
SYN-ACK = Verbindungswunsch bestätigen + eigene Start-Sequenznummer senden
ACK     = Bestätigung zurücksenden
```

Sequenznummer ist keine Portnummer

Eine Sequenznummer ist nicht dasselbe wie eine Portnummer.

Eine **Portnummer** sagt:

```
Welcher Dienst oder welches Programm ist gemeint?
```

Eine **Sequenznummer** sagt:

```
An welcher Stelle im TCP-Datenstrom befinden sich diese Daten?
```

Merksatz:

```
Portnummer     = welcher Dienst?
Sequenznummer = welche Stelle im Datenstrom?
```

Oder einfacher:

Portnummer ist wie die Türnummer eines Dienstes.

Sequenznummer ist wie die Position der Daten im Datenstrom.

Was macht die Sequenznummer bei TCP?

TCP nummeriert nicht einfach nur einzelne Pakete wie „Paket 1, Paket 2, Paket 3“.

Technisch genauer:

TCP nummeriert die Bytes im Datenstrom.

Dadurch weiß der Empfänger:

- wo ein empfangenes TCP-Segment im Datenstrom beginnt
- ob Daten in der richtigen Reihenfolge angekommen sind
- ob Daten fehlen
- ob Daten doppelt angekommen sind

Ein TCP-Segment enthält eine Sequenznummer. Diese Sequenznummer zeigt, an welcher Position im Datenstrom die enthaltenen Daten beginnen.

Einfaches Beispiel für Sequenznummern

Segment 1: Sequenznummer 1000, enthält 500 Byte

Segment 2: Sequenznummer 1500, enthält 500 Byte

Segment 3: Sequenznummer 2000, enthält 500 Byte

Das bedeutet:

Segment 1 beginnt bei Byte 1000.

Segment 2 beginnt bei Byte 1500.

Segment 3 beginnt bei Byte 2000.

Wenn alles korrekt ankommt, kann der Empfänger die Daten in der richtigen Reihenfolge zusammensetzen.

Was passiert, wenn Daten verloren gehen?

Angenommen, Segment 2 geht verloren:

```
Segment 1 kommt an: Sequenznummer 1000
Segment 2 fehlt:    Sequenznummer 1500
Segment 3 kommt an: Sequenznummer 2000
```

Der Empfänger merkt:

```
Ich habe Daten ab 1000 bekommen.
Danach müsste eigentlich 1500 kommen.
Jetzt kommt aber schon 2000.
Also fehlt der Bereich ab 1500.
```

TCP erkennt dadurch, dass ein Teil der Daten fehlt.

Die fehlenden Daten können dann erneut übertragen werden.

ACK = Bestätigung empfangener Daten

Mit einem ACK bestätigt der Empfänger, welche Daten korrekt angekommen sind.

Vereinfacht gesagt:

```
ACK 1500 = Ich habe alles bis vor 1500 korrekt erhalten.
           Bitte sende als Nächstes die Daten ab 1500.
```

Wenn Daten fehlen, bestätigt der Empfänger nicht einfach alles als vollständig.

Stattdessen signalisiert er sinngemäß:

```
Ich erwarte weiterhin die Daten ab dieser Sequenznummer.
```

Dadurch erkennt der Sender, dass Daten erneut übertragen werden müssen.

Muss TCP komplett warten, wenn ein Segment fehlt?

Nicht unbedingt.

Später angekommene Daten können zwischengespeichert werden.

Beispiel:

```
Segment 1 kommt an.
Segment 2 fehlt.
Segment 3 kommt schon an.
```

TCP kann Segment 3 intern speichern.

Wichtig ist aber:

```
An die Anwendung werden die Daten erst in der richtigen Reihenfolge weitergegeben.
```

Das bedeutet:

```
Die Anwendung bekommt Segment 3 nicht vor Segment 2.
Erst wenn die fehlenden Daten angekommen sind,
werden die Daten geordnet weitergegeben.
```

Dadurch sieht die Anwendung einen zuverlässigen und geordneten Datenstrom.

Warum ist TCP zuverlässig?

TCP gilt als zuverlässig, weil es mehrere Kontrollmechanismen verwendet.

Dazu gehören:

```
- Verbindungsaufbau durch 3-Wege-Handshake
- Sequenznummern
- ACK-Bestätigungen
- erneute Übertragung verlorener Daten
- Reihenfolgekontrolle
- Erkennung doppelt empfangener Daten
- Flusskontrolle
```

Dadurch kann TCP sicherstellen, dass Daten vollständig und in der richtigen Reihenfolge beim Empfänger ankommen.

Flusskontrolle bei TCP

TCP verwendet Flusskontrolle, damit ein schneller Sender einen langsameren Empfänger nicht überfordert.

Der Empfänger kann dem Sender mitteilen, wie viele Daten er aktuell aufnehmen kann.

Das nennt man vereinfacht:

```
Empfangsfenster
```

Wenn der Empfänger nur wenig Speicher frei hat, kann er dem Sender signalisieren:

```
Sende langsamer oder warte kurz.
```

Dadurch wird verhindert, dass der Empfänger mit zu vielen Daten überlastet wird.

TCP-Verbindungsabbau

Eine TCP-Verbindung wird nicht einfach abrupt beendet, sondern kontrolliert geschlossen.

Dafür werden unter anderem FIN- und ACK-Nachrichten verwendet.

Vereinfacht:

```
Eine Seite sagt: Ich möchte die Verbindung beenden.
Die andere Seite bestätigt das.
Danach kann auch die andere Seite ihre Richtung schließen.
```

Typische Steuerbits beim Verbindungsabbau:

```
FIN = Verbindung geordnet beenden
ACK = Bestätigung
```

Merksatz:

```
SYN = Verbindung aufbauen
FIN = Verbindung beenden
ACK = bestätigen
```

TCP im Vergleich zu UDP

TCP:

```
- verbindungsorientiert
- zuverlässig
- geordnete Datenübertragung
- Bestätigungen durch ACKs
- erneute Übertragung verlorener Daten
- mehr Verwaltungsaufwand
```

UDP:

```
- verbindungslos
- geringer Verwaltungsaufwand
- keine eingebaute Zustellgarantie
- keine eingebaute Reihenfolgekontrolle
- keine eingebaute erneute Übertragung
- oft latenzärmer als TCP
```

TCP wird verwendet, wenn Zuverlässigkeit wichtig ist.

UDP wird häufig verwendet, wenn geringe Verzögerung wichtiger ist als vollständige Kontrolle.

Beispiel: TCP beim Webseitenaufruf

Wenn ein Client eine HTTPS-Webseite aufruft, passiert vereinfacht Folgendes:

```
1. Client möchte Website aufrufen.
2. Client baut TCP-Verbindung zum Server-Port 443 auf.
3. TCP führt den 3-Wege-Handshake aus.
4. Danach werden Daten übertragen.
5. TCP nummeriert die Daten mit Sequenznummern.
6. Der Empfänger bestätigt empfangene Daten mit ACKs.
7. Fehlende Daten werden erneut übertragen.
8. Die Anwendung erhält die Daten in der richtigen Reihenfolge.
```

Beispiel:

```
Client: 192.168.0.20:51544
Server: 93.184.216.34:443
```

```
Protokoll: TCP
```

Wichtige TCP-Begriffe

```
TCP = Transmission Control Protocol
SYN = Synchronize
ACK = Acknowledgement
FIN = Finish
Port = Dienstadresse auf einem Host
Socket = IP-Adresse + Port
Sequenznummer = Position der Daten im TCP-Datenstrom
3-Wege-Handshake = Verbindungsaufbau bei TCP
```

IHK-sichere Kurzformulierung

TCP ist ein verbindungsorientiertes und zuverlässiges Transportprotokoll. Vor der Datenübertragung wird durch den 3-Wege-Handshake eine Verbindung aufgebaut. TCP verwendet Sequenznummern, um die Reihenfolge der übertragenen Daten im Datenstrom festzulegen. Mit ACKs bestätigt der Empfänger korrekt erhaltene Daten. Fehlende Daten können erkannt und erneut übertragen werden. Dadurch stellt TCP eine geordnete und zuverlässige Datenübertragung bereit.

Merksätze

```
TCP = verbindungsorientiert, zuverlässig und geordnet
```

```
SYN     = Verbindung starten
SYN-ACK = Verbindung bestätigen und eigene Startnummer senden
ACK     = Bestätigung
```

```
Portnummer     = welcher Dienst?
Sequenznummer = welche Stelle im Datenstrom?
```

```
Sequenznummern helfen TCP zu erkennen,
ob Daten fehlen, doppelt angekommen sind
oder in falscher Reihenfolge eintreffen.
```

ACK bestätigt,

bis wohin Daten korrekt empfangen wurden.

Fehlende Daten werden erneut übertragen.

Später angekommene Daten können zwischengespeichert werden.

An die Anwendung geht alles erst in der richtigen Reihenfolge.

SYN = Verbindung aufbauen

FIN = Verbindung beenden

ACK = bestätigen

Revision #2
Created 1 June 2026 07:30:06 by Admin
Updated 3 June 2026 06:18:12 by Admin