UDP (User Datagram Protocol) Erklärung

UDP einfach erklärt

UDP steht für **User Datagram Protocol**.

UDP ist ein **verbindungsloses** Transportprotokoll. Es arbeitet auf der **Transportschicht** des TCP/IP-Modells und wird verwendet, wenn Daten schnell und mit möglichst wenig Verwaltungsaufwand übertragen werden sollen.

UDP ist einfacher aufgebaut als TCP. Dafür bietet UDP aber keine eingebaute Garantie, dass Daten ankommen, vollständig sind oder in der richtigen Reihenfolge eintreffen.

Typische Anwendungen mit UDP sind zum Beispiel:

```
DNS
DHCP
VoIP
Video-Streaming
Audio-Streaming
Online-Gaming
QUIC / HTTP/3
```

Grundidee von UDP

UDP wird verwendet, wenn eine schnelle und einfache Datenübertragung wichtiger ist als vollständige Kontrolle.

Das bedeutet:

```
- keine feste Verbindung vor der Datenübertragung
- kein 3-Wege-Handshake
- keine eingebaute Zustellgarantie
- keine eingebaute Reihenfolgekontrolle
- keine automatische erneute Übertragung verlorener Daten
- geringer Verwaltungsaufwand
```

UDP sendet Daten einfach los. Ob die Gegenseite erreichbar ist oder ob die Daten vollständig ankommen, wird von UDP selbst nicht zuverlässig kontrolliert.

Merksatz:

```
UDP = verbindungslos, einfach und schnell
TCP = verbindungsorientiert, zuverlässig und geordnet
```

UDP arbeitet mit Ports

Auch UDP verwendet Ports, genau wie TCP.

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:

```
192.168.0.50:53
```

Das bedeutet:

```
192.168.0.50 = Host / Gerät
53           = Dienst / Anwendung, hier DNS
```

Typische UDP-Ports:

```
53    = DNS
67    = DHCP Server
68    = DHCP Client
123   = NTP
500   = IKE / IPsec
1194  = OpenVPN
51820 = WireGuard
443   = QUIC / HTTP/3
```

Wichtig:

```
Ein Port kann bei TCP und UDP unterschiedlich verwendet werden.
```

Beispiel:

```
TCP 443 = HTTPS über TCP
UDP 443 = QUIC / HTTP/3
```

Merksatz:

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

Socket bei UDP

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

Beispiel:

```
192.168.0.50:53
```

Bei UDP kann ein Datenpaket von einem Quell-Socket zu einem Ziel-Socket gesendet werden.

Beispiel:

```
Client-Socket: 192.168.0.20:53000
Server-Socket: 192.168.0.1:53
```

Das bedeutet:

```
Client fragt von Port 53000 aus einen DNS-Server auf Port 53 an.
```

Eine UDP-Kommunikation wird also durch diese Informationen beschrieben:

```
Quell-IP
Quell-Port
Ziel-IP
Ziel-Port
Protokoll UDP
```

UDP hat keinen Verbindungsaufbau

Bei TCP gibt es vor der Datenübertragung einen 3-Wege-Handshake.

Bei UDP gibt es diesen Verbindungsaufbau nicht.

TCP:

```
Client                  Server

1. SYN  ---------------------->
2. SYN-ACK <------------------
3. ACK  ---------------------->

Verbindung steht
```

UDP:

```
Client                  Server

Datenpaket ------------------>

Keine vorherige Verbindung
Keine SYN-Nachricht
Kein SYN-ACK
Kein ACK durch UDP selbst
```

UDP sendet also direkt ein Datagramm an den Empfänger.

Merksatz:

```
TCP fragt vorher: Darf ich eine Verbindung aufbauen?
UDP sendet direkt los.
```

Was ist ein UDP-Datagramm?

Die Dateneinheit bei UDP nennt man **Datagramm**.

Ein UDP-Datagramm enthält unter anderem:

```
- Quellport
- Zielport
- Länge
- Prüfsumme
- Nutzdaten
```

UDP ist dadurch sehr schlank aufgebaut.

Wichtig:

```
UDP nummeriert die Daten nicht wie TCP mit Sequenznummern.
```

Deshalb kann UDP selbst nicht zuverlässig erkennen, ob ein Datagramm fehlt oder in falscher Reihenfolge angekommen ist.

UDP hat keine Sequenznummern wie TCP

TCP verwendet Sequenznummern, um die Position der Daten im Datenstrom festzulegen.

UDP macht das nicht.

TCP:

```
Segment 1: Sequenznummer 1000
Segment 2: Sequenznummer 1500
Segment 3: Sequenznummer 2000
```

UDP:

```
Datagramm 1 wird gesendet.
Datagramm 2 wird gesendet.
Datagramm 3 wird gesendet.

UDP selbst merkt sich keine Reihenfolge.
```

Wenn ein UDP-Datagramm verloren geht, wird es von UDP nicht automatisch erneut angefordert.

Wenn UDP-Datagramme in falscher Reihenfolge ankommen, sortiert UDP sie nicht automatisch.

Merksatz:

TCP kontrolliert die Reihenfolge.
UDP sendet einzelne Datagramme ohne Reihenfolgegarantie.

UDP hat keine eingebaute Zustellgarantie

UDP garantiert nicht, dass ein Datagramm beim Empfänger ankommt.

Ein UDP-Datagramm kann unterwegs verloren gehen, zum Beispiel durch:

- Netzwerküberlastung
- Paketverlust
- Routing-Probleme
- Firewall-Regeln
- fehlerhafte Übertragung

UDP selbst sendet verlorene Datagramme nicht automatisch erneut.

Das bedeutet:

Wenn ein UDP-Datagramm verloren geht,
merkt UDP selbst das nicht zuverlässig
und fordert es nicht automatisch erneut an.

Wenn eine Anwendung trotzdem Zuverlässigkeit braucht, muss sie diese selbst oberhalb von UDP einbauen.

Beispiele dafür sind:

- eigene Bestätigungen
- eigene Sequenznummern
- eigene Wiederholungen
- Fehlerkorrektur auf Anwendungsebene

UDP hat keine eingebaute Reihenfolgekontrolle

UDP garantiert nicht, dass Datagramme in der gesendeten Reihenfolge ankommen.

Beispiel:

```
Gesendet:
Datagramm 1
Datagramm 2
Datagramm 3

Angekommen:
Datagramm 1
Datagramm 3
Datagramm 2
```

UDP sortiert diese Datagramme nicht automatisch.

Wenn die Reihenfolge wichtig ist, muss die Anwendung selbst dafür sorgen.

Beispiel:

```
Eine Anwendung kann eigene Nummern in die Nutzdaten schreiben,
um die Reihenfolge später selbst zu prüfen.
```

Merksatz:

```
UDP liefert Datagramme so ab, wie sie ankommen.
UDP sortiert nicht automatisch.
```

UDP hat weniger Verwaltungsaufwand als TCP

UDP hat weniger Verwaltungsaufwand, weil es keinen Verbindungsaufbau, keine Sequenznummern, keine ACKs und keine automatische Wiederholung verlorener Daten gibt.

Dadurch ist UDP oft:

```
- einfacher
- schneller beim Start
- latenzärmer
- ressourcenschonender
```

Wichtig:

UDP ist nicht automatisch immer schneller im Sinne von höherer Datenrate.

Besser formuliert:

UDP hat weniger Verwaltungsaufwand als TCP und kann dadurch schneller bzw. latenzärmer sein.

Das ist besonders bei Anwendungen wichtig, bei denen Echtzeit wichtiger ist als perfekte Vollständigkeit.

Warum nutzt man UDP trotz fehlender Garantie?

UDP wird genutzt, weil bei manchen Anwendungen verlorene Daten weniger schlimm sind als Verzögerungen.

Beispiel VoIP:

Bei einem Telefonat ist es besser,
wenn ein kleines Audiostück kurz fehlt,
als wenn die Sprache stark verzögert ankommt.

Beispiel Online-Gaming:

Bei schnellen Positionsdaten ist es oft besser,
aktuelle Daten zu bekommen,
als alte verlorene Daten nachträglich zu übertragen.

Beispiel Streaming:

Bei Live-Video ist geringe Verzögerung wichtiger
als jedes einzelne Paket nachträglich zu retten.

Merksatz:

UDP wird häufig verwendet,
wenn Aktualität wichtiger ist als vollständige Nachlieferung.

Beispiel: DNS mit UDP

DNS verwendet sehr häufig UDP auf Port 53.

Ablauf vereinfacht:

```
1. Client möchte wissen, welche IP-Adresse zu einer Domain gehört.
2. Client sendet eine DNS-Anfrage per UDP an Port 53.
3. DNS-Server antwortet per UDP.
4. Die Antwort enthält die passende IP-Adresse.
```

Beispiel:

```
Client: 192.168.0.20:53000
DNS-Server: 192.168.0.1:53
Protokoll: UDP
```

Warum UDP hier sinnvoll ist:

```
- DNS-Anfragen sind meist klein
- eine Verbindung vorher aufzubauen wäre zusätzlicher Aufwand
- bei Verlust kann die Anfrage einfach erneut gestellt werden
```

Wichtig:

```
DNS kann auch TCP verwenden,
zum Beispiel bei größeren Antworten oder Zonentransfers.
```

Beispiel: VoIP mit UDP

Bei VoIP werden Sprachdaten in kleinen Paketen übertragen.

UDP eignet sich hier gut, weil geringe Verzögerung besonders wichtig ist.

Beispiel:

```
Sprecher → Sprachpakete → Netzwerk → Empfänger
```

Wenn ein kleines Sprachpaket verloren geht, ist das meist weniger schlimm als eine große Verzögerung.

Deshalb ist bei VoIP oft wichtiger:

```
geringe Latenz statt vollständige Nachlieferung
```

Beispiel: Online-Gaming mit UDP

Online-Spiele übertragen häufig Positionsdaten, Bewegungen und Zustände.

Beispiel:

```
Spielerposition
Blickrichtung
Bewegung
Aktionen
```

Wenn ein altes Positionspaket verloren geht, ist es oft nicht sinnvoll, es später noch nachzuliefern.

Wichtiger ist:

```
Der aktuelle Zustand soll möglichst schnell ankommen.
```

Deshalb wird für solche Echtzeitdaten häufig UDP genutzt.

QUIC und HTTP/3 nutzen UDP

QUIC ist ein modernes Transportprotokoll, das auf UDP basiert.

HTTP/3 verwendet QUIC.

Vereinfacht:

```
HTTP/3 läuft über QUIC.
QUIC läuft über UDP.
UDP läuft über IP.
```

Wichtig:

```
QUIC nutzt UDP als Grundlage,
baut aber eigene Funktionen für Zuverlässigkeit,
```

Verschlüsselung und Verbindungssteuerung ein.

Das bedeutet:

UDP selbst ist einfach und verbindungslos.
QUIC ergänzt darauf zusätzliche moderne Funktionen.

UDP und Firewall

Firewalls können UDP-Verkehr filtern, genau wie TCP-Verkehr.

Dabei werden zum Beispiel geprüft:

- Quell-IP
- Ziel-IP
- Quellport
- Zielport
- Protokoll UDP

Da UDP keine feste Verbindung wie TCP aufbaut, ist die Zustandsverfolgung schwieriger.

Eine Stateful Firewall kann sich aber trotzdem merken, dass ein interner Client ein UDP-Datagramm nach außen gesendet hat.

Beispiel:

Client sendet DNS-Anfrage an DNS-Server.
Firewall merkt sich diese Anfrage kurzzeitig.
DNS-Antwort darf zurück.
Unerwartete UDP-Pakete von außen werden blockiert.

Merksatz:

Auch UDP kann von einer Stateful Firewall verfolgt werden,
aber ohne echten TCP-Verbindungszustand.

UDP und NAT/PAT

Auch UDP kann über NAT/PAT ins Internet gehen.

Beispiel:

```
Interner Client: 192.168.0.20:53000
Öffentliche IP: 84.10.20.30:40001
DNS-Server: 1.1.1.1:53
```

Die Firewall bzw. der Router merkt sich die Zuordnung:

```
192.168.0.20:53000 → 84.10.20.30:40001
```

Wenn die Antwort vom DNS-Server zurückkommt, weiß der Router:

```
Diese Antwort gehört zurück an 192.168.0.20:53000.
```

Wichtig:

```
Bei UDP sind solche NAT-Zuordnungen meist zeitlich begrenzt,
weil es keine dauerhaft aufgebaute Verbindung wie bei TCP gibt.
```

UDP im Vergleich zu TCP

TCP:

```
- verbindungsorientiert
- 3-Wege-Handshake
- zuverlässig
- Sequenznummern
- ACK-Bestätigungen
- erneute Übertragung verlorener Daten
- Reihenfolgekontrolle
- mehr Verwaltungsaufwand
```

UDP:

```
- verbindungslos
- kein 3-Wege-Handshake
- keine eingebaute Zustellgarantie
```

- keine eingebaute Reihenfolgekontrolle
- keine automatische erneute Übertragung
- weniger Verwaltungsaufwand
- oft latenzärmer

Merksatz:

TCP kontrolliert stärker.
UDP ist schlanker und direkter.

Wann verwendet man TCP?

TCP verwendet man, wenn Daten zuverlässig und vollständig ankommen müssen.

Beispiele:

- Webseiten über HTTP/HTTPS
- Dateiübertragung
- SSH
- E-Mail
- Remote Desktop

Hier wäre es schlecht, wenn Daten fehlen oder in falscher Reihenfolge bei der Anwendung ankommen.

Wann verwendet man UDP?

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

Beispiele:

- DNS
- DHCP
- VoIP
- Live-Streaming
- Online-Gaming
- NTP
- VPN-Protokolle wie WireGuard
- QUIC / HTTP/3

Hier ist es oft besser, schnell weiterzumachen, statt verlorene alte Daten nachzuliefern.

Ist UDP unsicherer als TCP?

UDP ist nicht automatisch „unsicherer“ als TCP.

UDP hat nur weniger eingebaute Kontrollfunktionen.

Sicherheit hängt vor allem davon ab:

```
- welches Anwendungsprotokoll verwendet wird
- ob Verschlüsselung eingesetzt wird
- wie die Firewall konfiguriert ist
- ob der Dienst korrekt abgesichert ist
```

Beispiel:

```
QUIC nutzt UDP,
kann aber trotzdem verschlüsselte Kommunikation ermöglichen.
```

UDP bedeutet also nicht automatisch unsicher. Es bedeutet nur:

```
UDP selbst garantiert weniger als TCP.
```

Wichtige UDP-Begriffe

```
UDP = User Datagram Protocol
Datagramm = einzelne UDP-Dateneinheit
Port = Dienstadresse auf einem Host
Socket = IP-Adresse + Port
Quellport = Port des sendenden Systems
Zielport = Port des empfangenden Dienstes
Prüfsumme = einfache Fehlererkennung im UDP-Datagramm
```

IHK-sichere Kurzformulierung

UDP ist ein verbindungsloses Transportprotokoll mit geringem Verwaltungsaufwand. Im Gegensatz zu TCP baut UDP vor der Datenübertragung keine Verbindung auf und bietet keine eingebaute

Garantie für Zustellung, Reihenfolge oder erneute Übertragung verlorener Daten. Dadurch ist UDP besonders für Anwendungen geeignet, bei denen geringe Verzögerung wichtiger ist als vollständige Kontrolle, zum Beispiel DNS, VoIP, Streaming oder Online-Gaming.

Merksätze

UDP = verbindungslos, einfach und mit wenig Verwaltungsaufwand

UDP sendet direkt los.
TCP baut vorher eine Verbindung auf.

UDP garantiert nicht,
dass Datagramme ankommen,
in richtiger Reihenfolge ankommen
oder erneut übertragen werden.

UDP ist oft latenzärmer als TCP,
aber nicht automatisch immer schneller in jeder Situation.

TCP = Zuverlässigkeit und Kontrolle
UDP = Geschwindigkeit und geringer Verwaltungsaufwand

Wenn UDP Zuverlässigkeit braucht,
muss die Anwendung diese selbst einbauen.

DNS, VoIP, Streaming und Online-Gaming
sind typische Beispiele für UDP.

Revision #2
Created 1 June 2026 07:36:29 by Admin
Updated 3 June 2026 06:18:12 by Admin