

Technische Grundlagen der Informatik 2 – Teil 3: Layer 4

Sebastian Harnau

Block 3/9

Transportschicht (Layer 4)
Einführung, Multiplexing,
UDP

Funktionen der Transportschicht

Die Transportschicht:

- **ermöglicht transparente Übertragung von Daten zwischen Endbenutzern (Hosts)**
- **stellt den übergeordneten Schichten zuverlässige Datenübertragungsdienste zur Verfügung.**

Die Transportschicht steuert die Zuverlässigkeit einer gegebenen Leitung mithilfe von:

- **Flusskontrolle**
- **Segmentierung/Desegmentierung**
- **Fehlerkontrolle**

Einige Protokolle sind zustands- und verbindungsorientiert, das heißt, die Transportschicht kann Segmente nachverfolgen und alle jene neu übertragen, deren Versand fehlgeschlagen ist.

Zweck der Transportschicht

Die Transportschicht ermöglicht Anwendungen auf Geräten die Kommunikation. Die folgenden Aufgaben fallen im Wesentlichen in die Zuständigkeit der Transportschicht:

- **Initiieren einer Sitzung**
- **Nachverfolgen der einzelnen Kommunikationsvorgänge zwischen Anwendungen auf den Absender - und Zielhosts**
- **Segmentieren der Daten und Verwalten jeder Dateneinheit**
- **Neuzusammensetzen der Segmente zu Anwendungsdatenströmen**
- **Erkennungen der verschiedenen Anwendungen**
- **Flusskontrolle zwischen Endbenutzern**
- **Wiederherstellen im Fehlerfall**

Einzelne Kommunikationsvorgänge nachverfolgen

Auf jedem Host können stets mehrere Anwendungen ausgeführt werden, die über das Netzwerk mit einer oder mehreren Anwendungen auf entfernten Hosts kommunizieren.

Die Transportschicht ist dafür zuständig, die verschiedenen Kommunikationsströme zwischen diesen Anwendungen aufrechtzuerhalten.

Betrachten Sie einen Computer, der an ein Netzwerk angeschlossen ist und auf dem gleichzeitig eMail und Instant Messaging-Nachrichten gesendet und empfangen und Webseiten angezeigt werden und zudem ein VoIP - Anruf durchgeführt wird. Jede dieser Anwendungen sendet und empfängt zur selben Zeit Daten über das Netzwerk. Allerdings werden Daten des Telefongesprächs weder an den Webbrowser weitergeleitet noch erscheint der Text der Instant Messaging-Nachricht in einer eMail. Die Transportschicht steuert das korrekt aus.

Segmentierung

- **Übergabe großer Datenmengen von der Anwendungsschicht an die Transportschicht.**
- **Unterteilung in kleinere Einheiten, die für die Übertragung besser geeignet sind. Diese Einheiten heißen Segmente.**
- **Jedes Segment benötigt Header, in denen angegeben wird, zu welcher Anwendung es gehört.**
- **Mehrere verschiedene Anwendungen, die gleichzeitig auf dem Computer ausgeführt werden, können gleichzeitig senden und empfangen. (Muxing/Demuxing)**
- **Ohne eine Segmentierung könnte nur eine einzige Anwendung zur Zeit Daten senden oder empfangen.**

Multiplexing und Demultiplexing

- Erweiterung des von der Netzwerkschicht angebotenen Host-zu-Host-Zustelldienstes zu einem Prozess-zu-Prozess-Zustelldienst für Anwendungen.
- Erinnerung: Sockets stellen Türen zwischen Netzwerk und Prozess dar.
- Das Einsammeln von Datenblöcken bei verschiedenen Sockets, das Einfügen von Header-Informationen und die Weiterleitung der Pakete an die Netzwerkschicht wird **Multiplexing** genannt.
- Das Abliefern empfangener Daten am richtigen Socket wird **Demultiplexing** genannt.

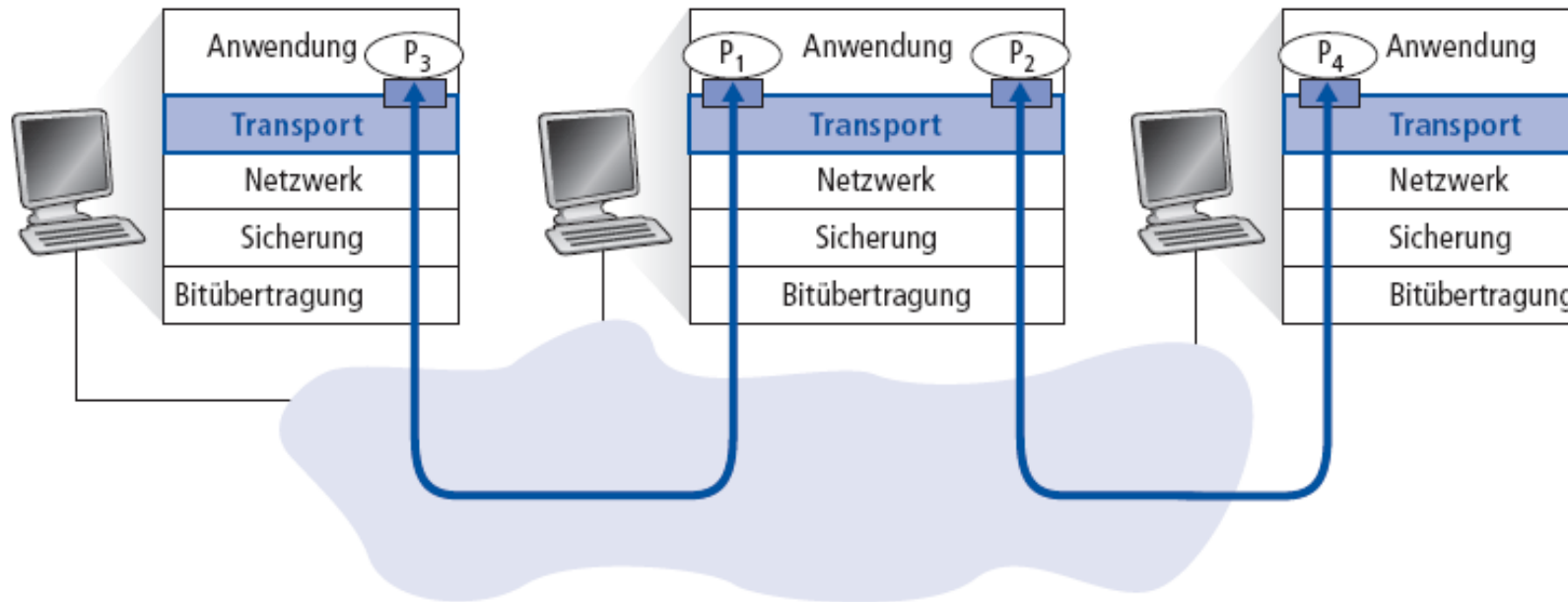
Multiplexing/Demultiplexing

Multiplexing beim Sender:

Daten von mehreren Sockets einsammeln,
Daten mit einem Header versehen (der später
für das Demultiplexing verwendet wird)

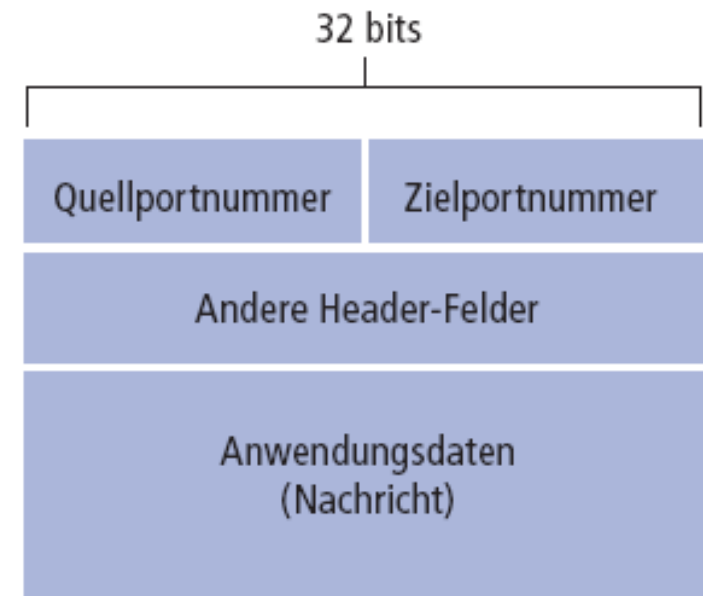
Demultiplexing beim Empfänger:

Empfangene Pakete am
richtigen Socket abliefern



Wie funktioniert Demultiplexing?

- **Host empfängt IP-Datagramme**
 - **Jedes Datagramm hat**
 - **eine Absender-IP-Adresse und**
 - **eine Empfänger-IP-Adresse**
 - **Jedes Datagramm beinhaltet ein Transportschichtpaket**
 - **Jedes Paket hat**
 - **eine Absender-Portnummer und**
 - **eine Empfänger-Portnummer**
- **Hosts nutzen IP-Adressen und Portnummern, um Pakete an den richtigen Socket weiterzuleiten.**



TCP/UDP-Paketformat

Well-known Ports

- **Hosts nutzen IP-Adressen und Portnummern, um Pakete an den richtigen Socket weiterzuleiten.**
- **Die Ports 0-1023 sind fest definiert / reserviert (well-known).**
- **Beispiele:**

Port	Anwendungs-Protokoll	Transport-schicht
20	FTP, Datenverbindung	TCP
21	FTP, Steuerverbindung	TCP
23	Telnet	TCP
25	SMTP	TCP
80	HTTP	TCP
110	POP3	TCP
443	HTTPS	TCP

Registrierte Ports

Registrierte Ports (1024 -49151) werden Benutzerprozessen oder -anwendungen zugeordnet. Diese Prozesse sind meist Einzelanwendungen, die ein Benutzer installiert hat. Also nicht solche, die einen Well-Known-Port erhalten würden. Einen registrierten Port, der nicht für eine Serverressource verwendet wird, kann ein Client dynamisch als Absender-Port auswählen.

Port	Anwendungs-Protokoll	Transport-schicht
1812	RADIUS-Authentifizierung	UDP
1863	MSN Messenger	TCP
2000	Cisco SCCP (VoIP)	UDP
5004	RTP (Sprach- und Video)	UDP
5060	SIP (VoIP)	UDP
8008	Alternativer HTTP-Port	TCP
8080	Alternativer HTTP-Port	TCP

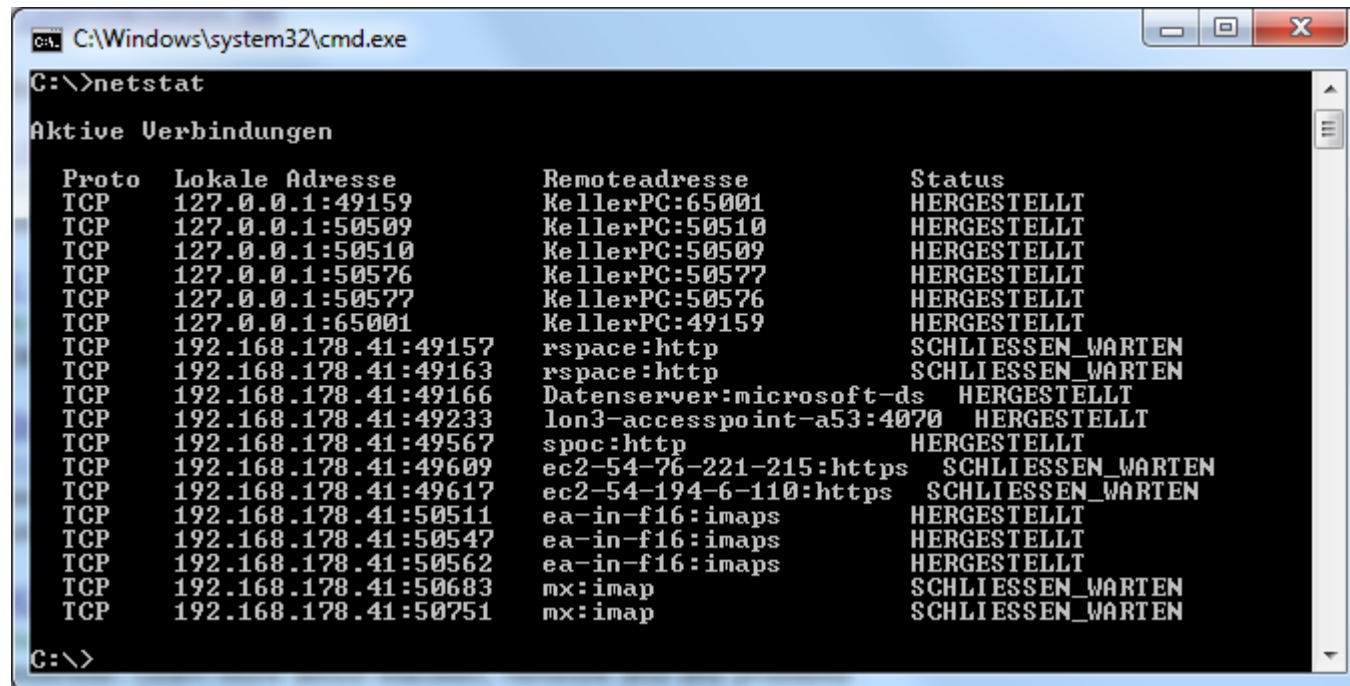
Weitere Ports

- **Dynamische oder private Ports (49152 -65535) werden meist in dynamischer Form Clientanwendungen zugewiesen, sobald eine Verbindung hergestellt wird. Verbindungsinitialisierung mit einem privaten Port ist eher ungewöhnlich (auch wenn einige Peer-to-Peer-Programme für das Filesharing dies tun).**
- **Einige Anwendungen können sowohl TCP als auch UDP verwenden (z.B. DNS). Sie tun dies über den gleichen Port.**

Port	Anwendungs-Protokoll	Transport-schicht
53	DNS	UDP & TCP
161	SNMP	UDP & TCP
1433	MS SQL	UDP & TCP
2948	WAP	UDP & TCP

Netstat

- Der Befehl netstat liefert einen Überblick über die geöffneten Verbindungen.
- Netstat führt das verwendete Protokoll, die lokale Adresse und Port - Nummer, die Zieladresse und Ziel - Port - Nummer sowie den Status der Verbindung auf. Nicht nachvollziehbare TCP - Verbindungen können ein Hinweis darauf sein, dass irgend jemand oder irgend etwas mit dem lokalen Host verbunden ist.



```
C:\Windows\system32\cmd.exe
C:\>netstat

Aktive Verbindungen

Proto Lokale Adresse Remoteadresse Status
TCP 127.0.0.1:49159 KellerPC:65001 HERGESTELLT
TCP 127.0.0.1:50509 KellerPC:50510 HERGESTELLT
TCP 127.0.0.1:50510 KellerPC:50509 HERGESTELLT
TCP 127.0.0.1:50576 KellerPC:50577 HERGESTELLT
TCP 127.0.0.1:50577 KellerPC:50576 HERGESTELLT
TCP 127.0.0.1:65001 KellerPC:49159 HERGESTELLT
TCP 192.168.178.41:49157 rspace:http SCHLIESSEN_WARTEN
TCP 192.168.178.41:49163 rspace:http SCHLIESSEN_WARTEN
TCP 192.168.178.41:49166 Datenserver:microsoft-ds HERGESTELLT
TCP 192.168.178.41:49233 lon3-accesspoint-a53:4070 HERGESTELLT
TCP 192.168.178.41:49567 spoc:http HERGESTELLT
TCP 192.168.178.41:49609 ec2-54-76-221-215:https SCHLIESSEN_WARTEN
TCP 192.168.178.41:49617 ec2-54-194-6-110:https SCHLIESSEN_WARTEN
TCP 192.168.178.41:50511 ea-in-f16:imaps HERGESTELLT
TCP 192.168.178.41:50547 ea-in-f16:imaps HERGESTELLT
TCP 192.168.178.41:50562 ea-in-f16:imaps HERGESTELLT
TCP 192.168.178.41:50683 mx:imap SCHLIESSEN_WARTEN
TCP 192.168.178.41:50751 mx:imap SCHLIESSEN_WARTEN

C:\>
```

Netstat -o

Für den Befehl netstat gibt es viele nützliche Optionen. So kann beispielsweise mit dem Parameter -o Rückschluss auf die dazugehörige PID (Programm-ID) im Task-Manager gezogen werden.

```
C:\Windows\system32\cmd.exe

Aktive Verbindungen

Proto Lokale Adresse Remoteadresse Status PID
TCP 127.0.0.1:49159 KellerPC:65001 HERGESTELLT 2184
TCP 127.0.0.1:50509 KellerPC:50510 HERGESTELLT 5268
TCP 127.0.0.1:50510 KellerPC:50509 HERGESTELLT 5268
TCP 127.0.0.1:50576 KellerPC:50577 HERGESTELLT 2072
TCP 127.0.0.1:50577 KellerPC:50576 HERGESTELLT 2072
TCP 127.0.0.1:65001 KellerPC:49159 HERGESTELLT 2184
TCP 192.168.178.41:49157 rspace:http SCHLIESSEN_WARTEN 1328
TCP 192.168.178.41:49163 rspace:http SCHLIESSEN_WARTEN 1328
TCP 192.168.178.41:49609 ec2-54-76-221-215:https SCHLIESSEN_WARTEN 1628

C:\>
```

Name	PID	Beschreibung	Status	Gruppe
NetTcpActi...	1920	Net.Tcp Listener Adapter	Wird ...	
NetPipeActi...	1920	Net.Pipe Listener Adap...	Wird ...	
GfExperien...	1872	NVIDIA GeForce Experi...	Wird ...	Nicht zutre...
wcnscvc	1780	Windows-Sofortverbin...	Wird ...	LocalServic...
SSDPSRV	1780	SSDP-Suche	Wird ...	LocalServic...
FDResPub	1780	Funktionssuche-Resso...	Wird ...	LocalServic...
ClickToRunSvc	1708	Microsoft Office-Klick-u...	Wird ...	Nicht zutre...
AppHostSvc	1672	Anwendungshost-Hilfs...	Wird ...	apphost
NvStreamSvc	1636	NVIDIA Streamer Service	Wird ...	Nicht zutre...
AntiVirService	1628	Avira Echtzeit-Scanner	Wird ...	Nicht zutre...
MpsSvc	1484	Windows-Firewall	Wird ...	LocalServic...
DPS	1484	Diagnoserichtliniendienst	Wird ...	LocalServic...
BFE	1484	Basisfiltermodul	Wird ...	LocalServic...
AntiVirSche...	1392	Avira Planer	Wird ...	Nicht zutre...
NSM	1328	Norton Family	Wird ...	Nicht zutre...

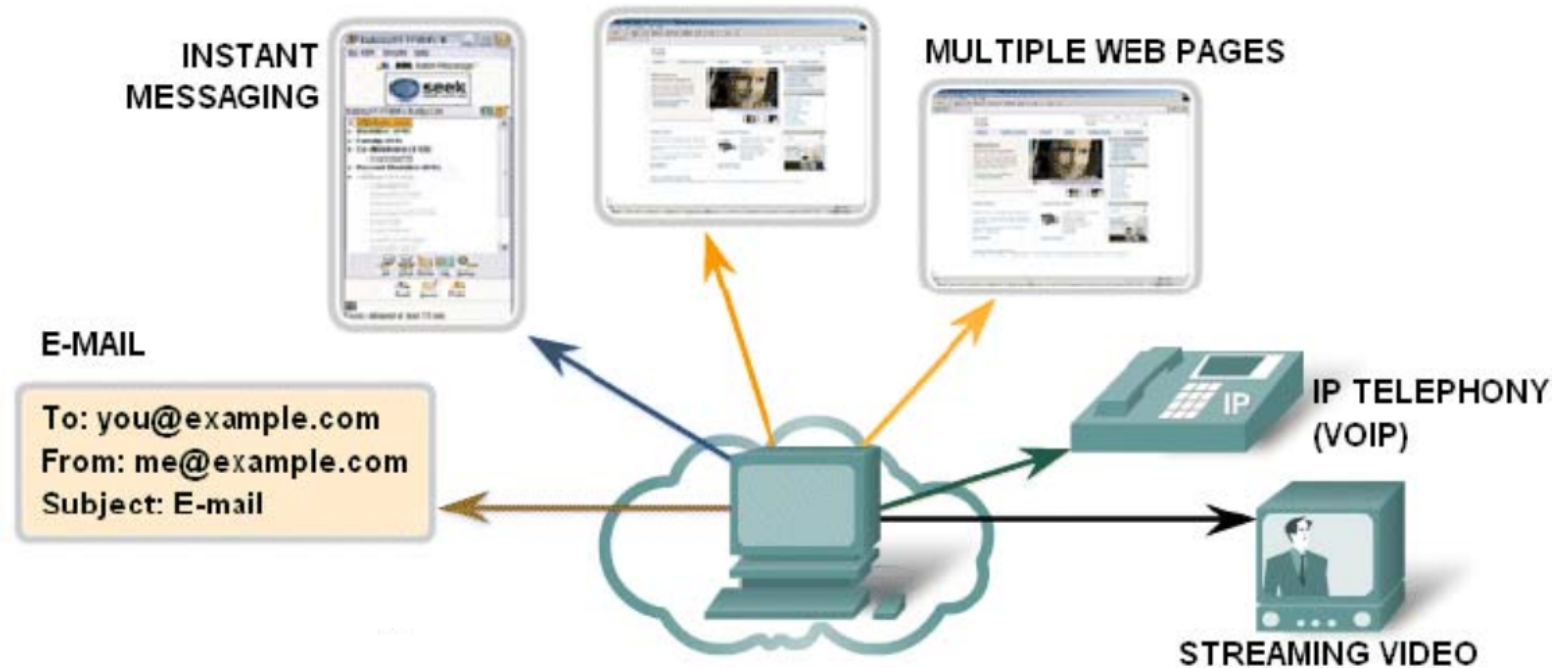
Flusskontrolle

- **Ressourcen von Netzwerkhosts (Speicher oder Bandbreite), sind stets begrenzt.**
- **Bei Überbeanspruchung sind einige Protokolle in der Lage, die sendende Anwendung zur Verringerung ihrer Datenrate aufzufordern.**
- **Regulierung der Datenmenge, die der Absender überträgt**
- **Verhindert den Verlust von Segmenten und beseitigt so die Notwendigkeit einer Neuübertragung.**
- **Detaillierte Behandlung später...**

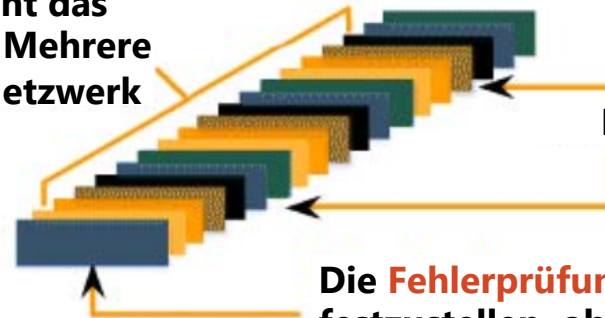
Fehlerwiederherstellung

- **Aus verschiedenen Gründen ist es möglich, dass Dateneinheiten bei der Übertragung durch das Netzwerk beschädigt werden oder verloren gehen.**
- **Die Transportschicht kann sicherstellen, dass alle Teile ihr Ziel erreichen.**
- **Hierzu muss das Absendergerät alle verloren gegangenen Daten neu senden.**

Zusammenfassung: Dienste der Transportschicht



Die Segmentierung ermöglicht das **Multiplexing** von Sitzungen: Mehrere Anwendungen können das Netzwerk zur selben Zeit verwenden



Die Segmentierung ermöglicht die **Übertragung von Daten** durch die unteren Schichten des Netzwerks.

Die **Fehlerprüfung** kann an Daten im Segment vorgenommen werden, um festzustellen, ob das Segment während der Übertragung geändert wurde.

Transportdienste und –protokolle

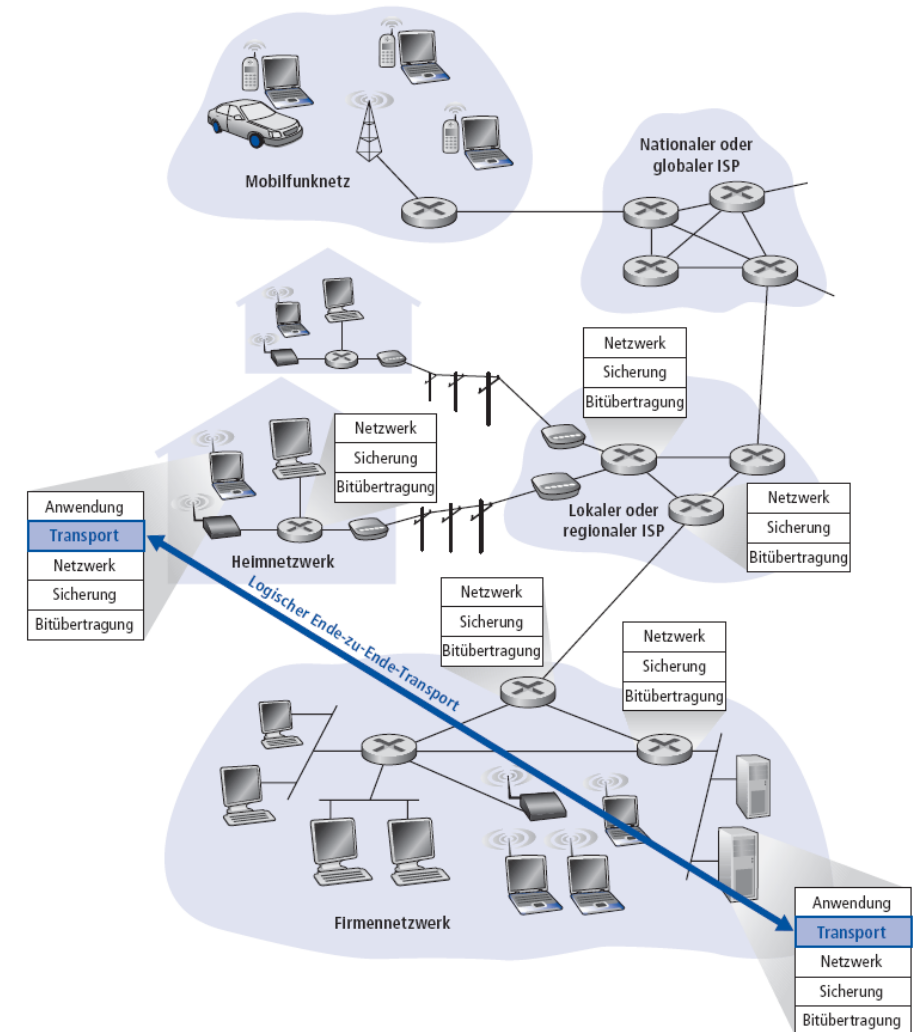
Sie stellen logische Kommunikation zwischen Anwendungsprozessen auf verschiedenen Hosts zur Verfügung

Transportprotokolle laufen auf Endsystemen

- **Sender:** teilt Anwendungsnachrichten in Pakete auf, gibt diese an die Netzwerkschicht weiter
- **Empfänger:** fügt Pakete wieder zu Anwendungsnachrichten zusammen, gibt diese an die Anwendungsschicht weiter

Es existieren verschiedene Transportschichtprotokolle

- **Internet: TCP und UDP**



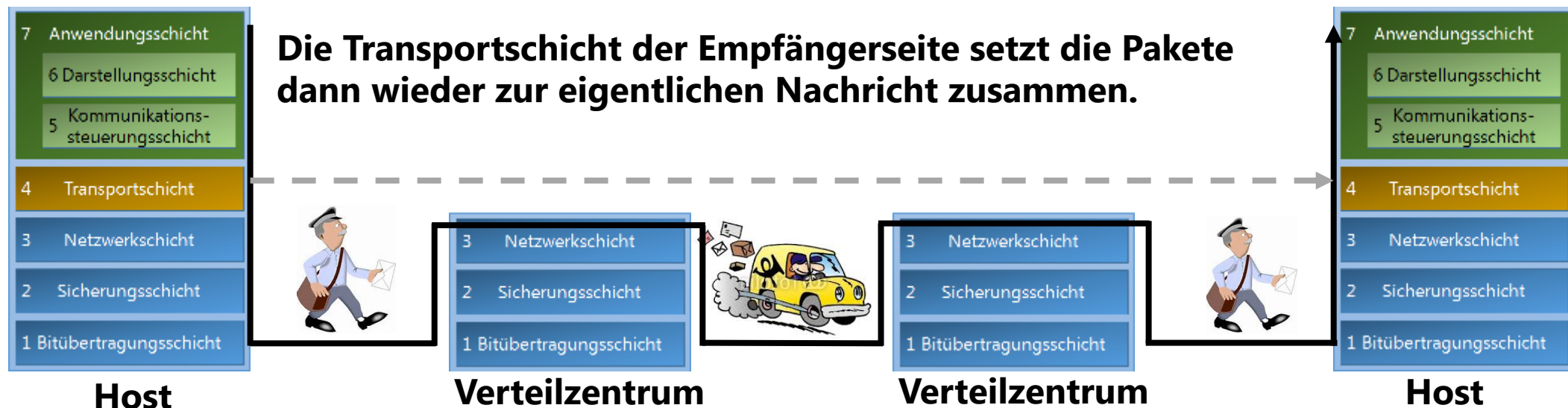
Nachrichtentransport durch Layer 1-4

Anwendung
Transport
Netzwerk
Sicherung
Bitübertragung

Die Transportschicht stellt eine logische Verbindung zwischen den **Prozessen auf den Hosts** her. Sie verwendet und erweitert die Dienste der Netzwerkschicht.

Der eigentliche Transport findet nur auf den Layern 1-3 statt. Layer 4 stellt eine logische Verbindung zwischen den **Hosts selber** her.

Um die „Belastbarkeit des Briefträgers“ nicht überzustrapazieren, werden große Sendungen in kleinere Stücke (**Pakete**) zerlegt.



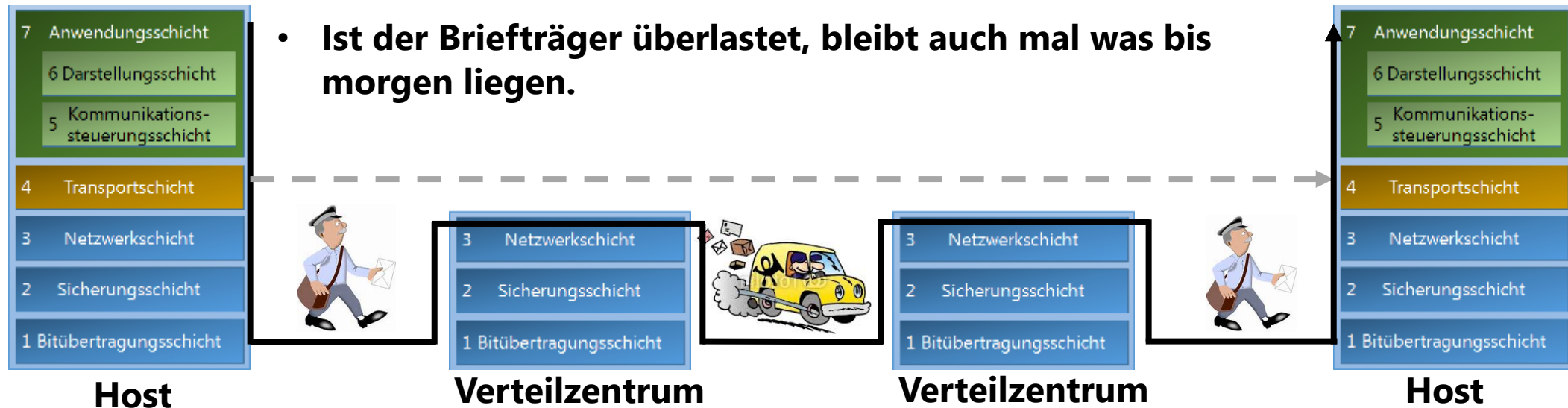
Beispiel: Nachrichtentransport durch TCP [RFC 793]

Anwendung
Transport
Netzwerk
Sicherung
Bitübertragung

Herr Theuerkauf und Frau Carlsen sind in ihren Firmen für den Posttransport im Haus zuständig. Sie machen das sehr gewissenhaft und rücksichtsvoll:



- Alles per Einschreiben mit Rückschein
- Posteingangs- und Ausgangsbücher
- Sie fragen schriftlich an, ob Datensendung OK wär.
- Ist der Briefträger überlastet, bleibt auch mal was bis morgen liegen.



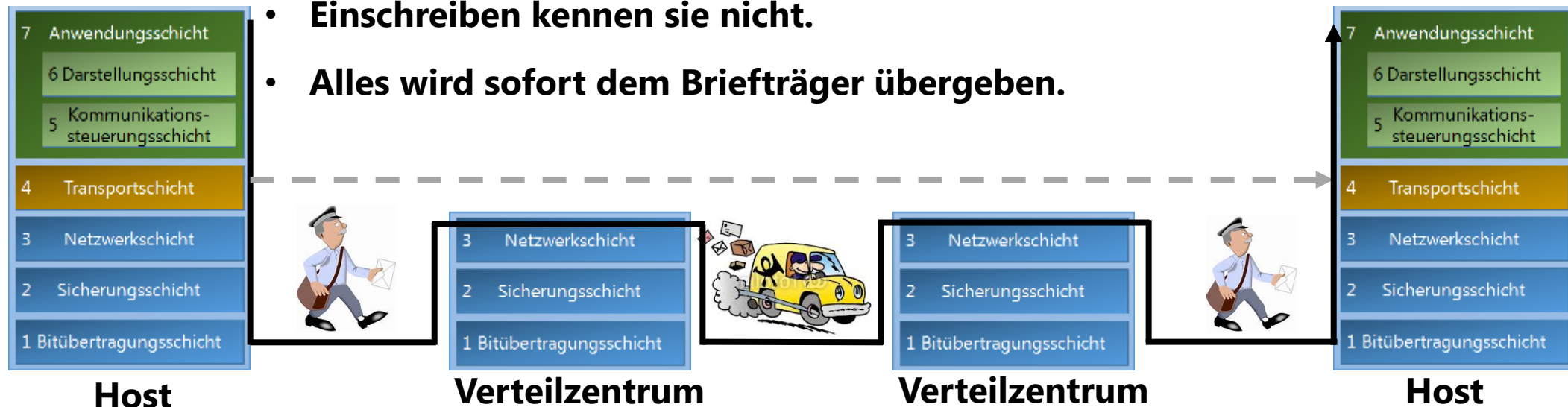
Beispiel: Nachrichtentransport durch UDP [RFC 768]

Anwendung
Transport
Netzwerk
Sicherung
Bitübertragung

Uli und Doreen sind in ihren Firmen für den Posttransport im Haus zuständig. Sie sehen vieles eher locker:



- Wenn mal was runterfällt oder vom Hund gefressen wird, ist es halt weg.
- Ihnen ist egal, ob und wieviel Post rein oder raus geht. Protokolliert wird nichts, das ist nur unnütze Arbeit.
- Einschreiben kennen sie nicht.
- Alles wird sofort dem Briefträger übergeben.



Wiederholung: Dienste der Transportprotokolle

Anwendung
Transport
Netzwerk
Sicherung
Binübertragung

TCP-Dienste:

- **Verbindungsorientierung:**
Herstellen einer Verbindung zwischen Client und Server
- **Zuverlässiger Transport** zwischen sendendem und empfangendem Prozess
- **Flusskontrolle:** Sender überlastet den Empfänger nicht
- **Überlastkontrolle:** Bremsen des Senders, wenn das Netzwerk überlastet ist
- **Nicht: Zeit- und Bandbreitengarantien**

UDP-Dienste:

- **Unzuverlässiger Transport** von Daten zwischen Sender und Empfänger
- **Nicht: Verbindungsorientierung, Zuverlässigkeit, Flusskontrolle, Überlastkontrolle, Zeit- oder Bandbreitengarantien**

	Datenverlust	Bandbreite	Echtzeit
UDP	Unzuverlässiger Transport von Daten zwischen Sender und Empfänger	Keine Bandbreitengarantien	Kein Zeitgarantien, aber schneller als TCP
TCP	Zuverlässiger Transport zwischen sendendem und empfangendem Prozess		Kein Zeitgarantien

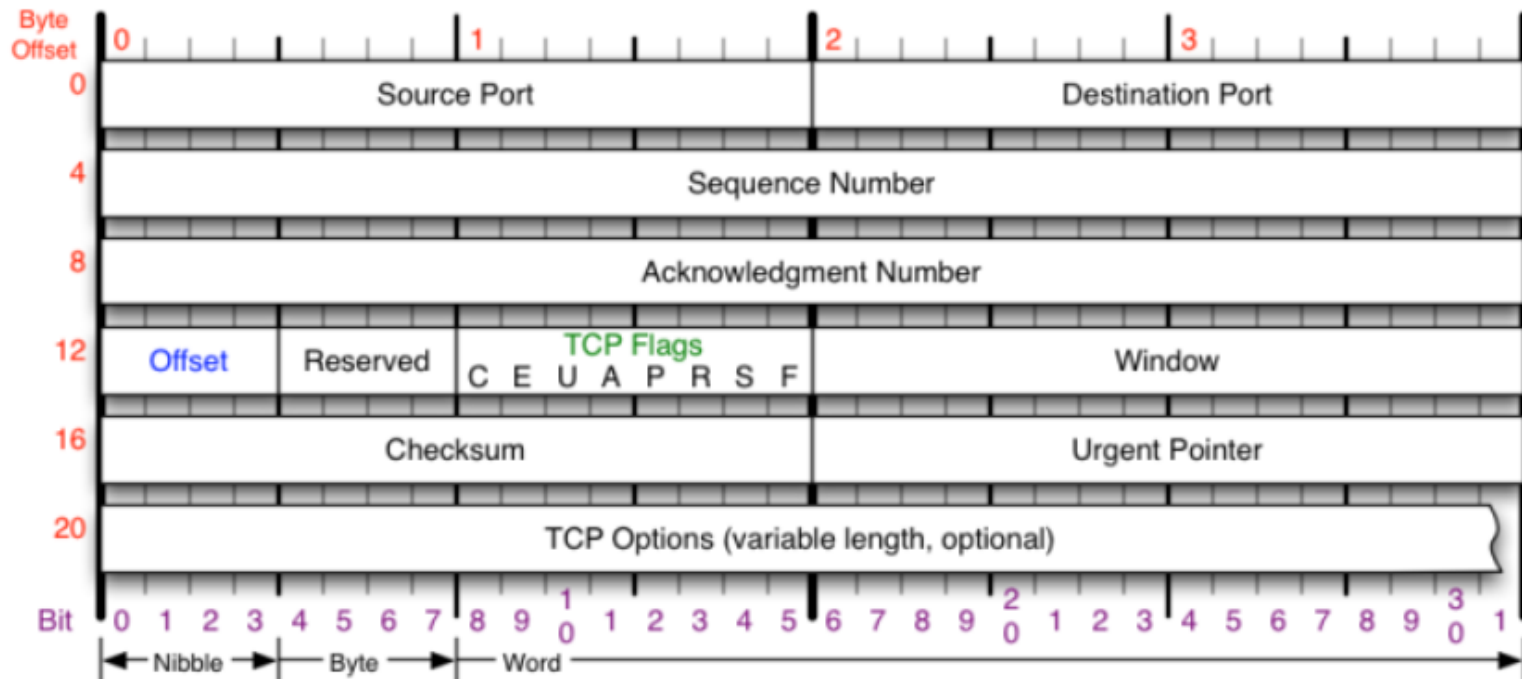
Übersicht über die bisherigen Protokolle und Schichten

Internet-Protokolle							
OSI-Schicht		Internet Protokoll Suite					DOD Schicht
7	Anwendung	File Transfer	Electronic Mail	Terminal Emulation	World Wide Web	Domain Name Service	Art der Kommuni- kation
6	Darstellung	File Transfer Protocol (FTP) RFC 959	Simple Mail Transfer Protocol (SMTP) RFC 821	Telnet Protocol (Telnet) RFC 854	Hypertext Transfer Protocol (HTTP) RFC 2616	Domain Name Service (DNS) RFC 1034	Applikation
5	Sitzung						
4	Transport	Transmission Control Protocol (TCP) RFC 793					User Datagram Protocol (UDP) RFC 768 Host to Host Kommuni- kation

DNS sendet UDP, solange die Daten in ein Segment passen. Bei Datenteilung wird auf TCP umgestellt.

TCP-Paketformat

TCP header



Quelle: <http://nmap.org/book/tcpip-ref.html>

TCP Flags

C E U A P R S F

Congestion Window

C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

20
Bytes

Offset

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

Offset

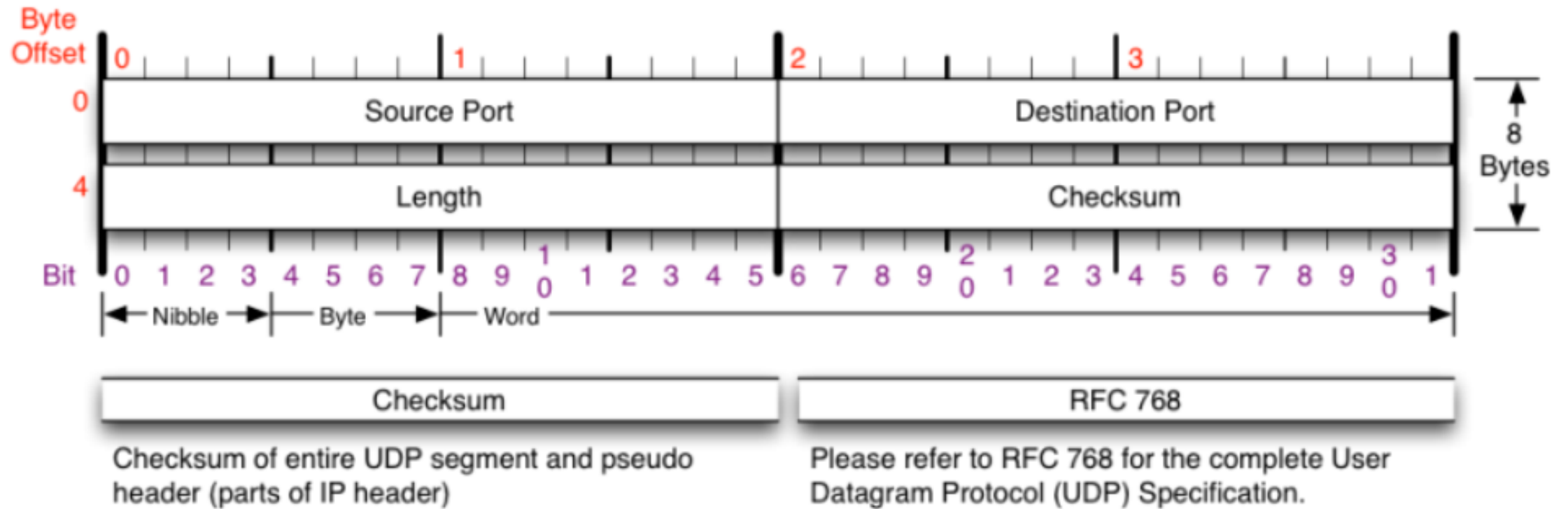
Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

UDP-Paketformat

UDP header



Quelle: <http://nmap.org/book/tcpip-ref.html>

Vergleich Demultiplexing UDP/TCP

UDP-Socket wird durch ein 2-Tupel identifiziert:

- **Empfänger-IP-Adresse**
- **Empfänger-Portnummer**

Sockets mit Portnummer anlegen:

```
DatagramSocket mySocket1 = new  
DatagramSocket(12534);
```

```
DatagramSocket mySocket2 = new  
DatagramSocket(12535);
```

TCP-Socket wird durch ein 4-Tupel identifiziert:

- **Absender-IP-Adresse**
- **Absender-Portnummer**
- **Empfänger-IP-Adresse**
- **Empfänger-Portnummer**

Empfänger nutzt alle vier Werte, um den richtigen TCP-Socket zu identifizieren

Vergleich Demultiplexing UDP/TCP

Wenn ein Host ein UDP-Segment empfängt:

- Lese Empfänger-Portnummer
- Das UDP-Segment wird an den UDP-Socket mit dieser Portnummer weitergeleitet

IP-Datagramme mit anderer Absender-IP-Adresse oder anderer Absender-Portnummer werden an denselben Socket ausgeliefert

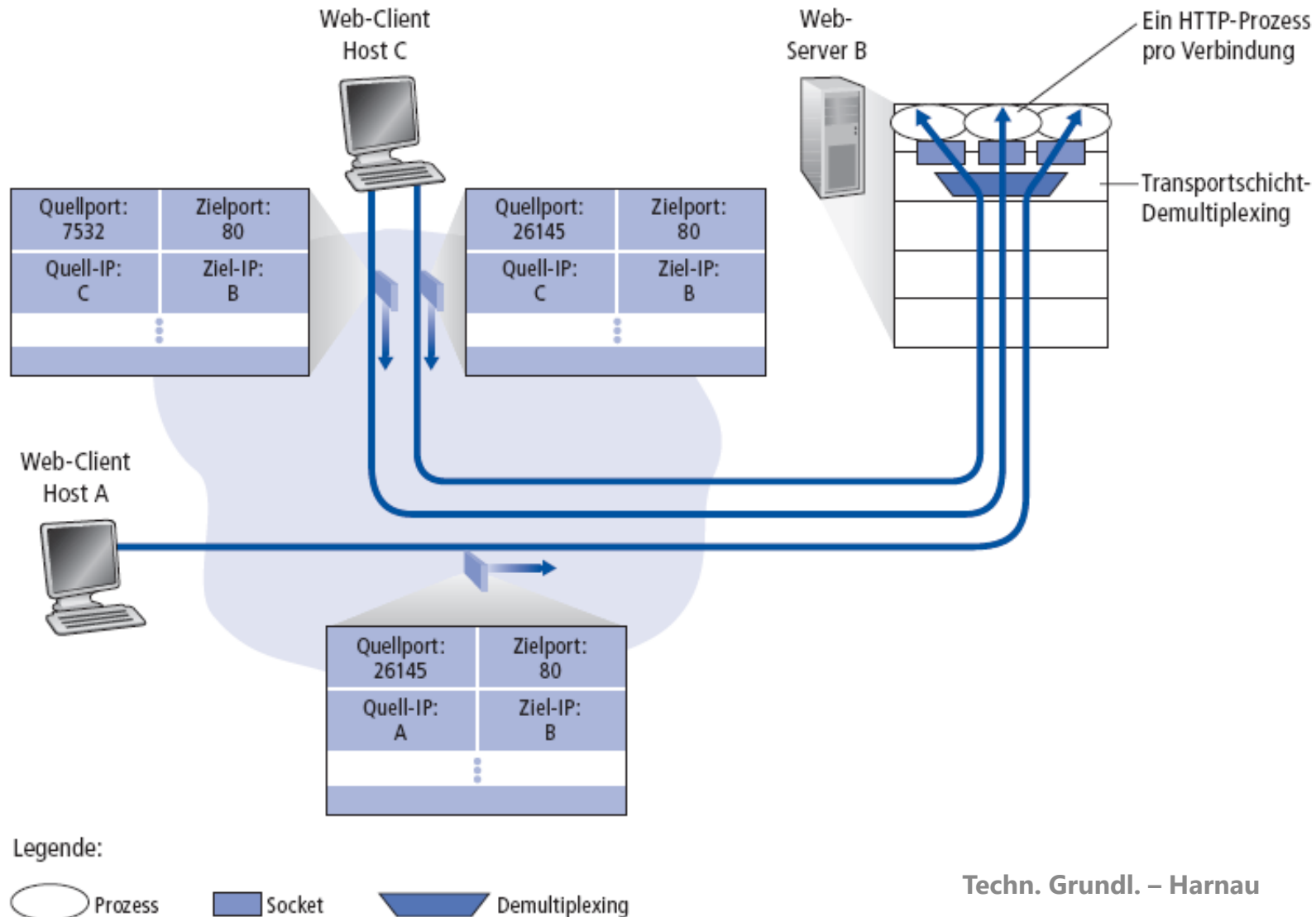
Server kann viele TCP-Sockets gleichzeitig offen haben:

- Jeder Socket wird durch sein eigenes 4-Tupel identifiziert

Webserver haben verschiedene Sockets für jeden einzelnen Client

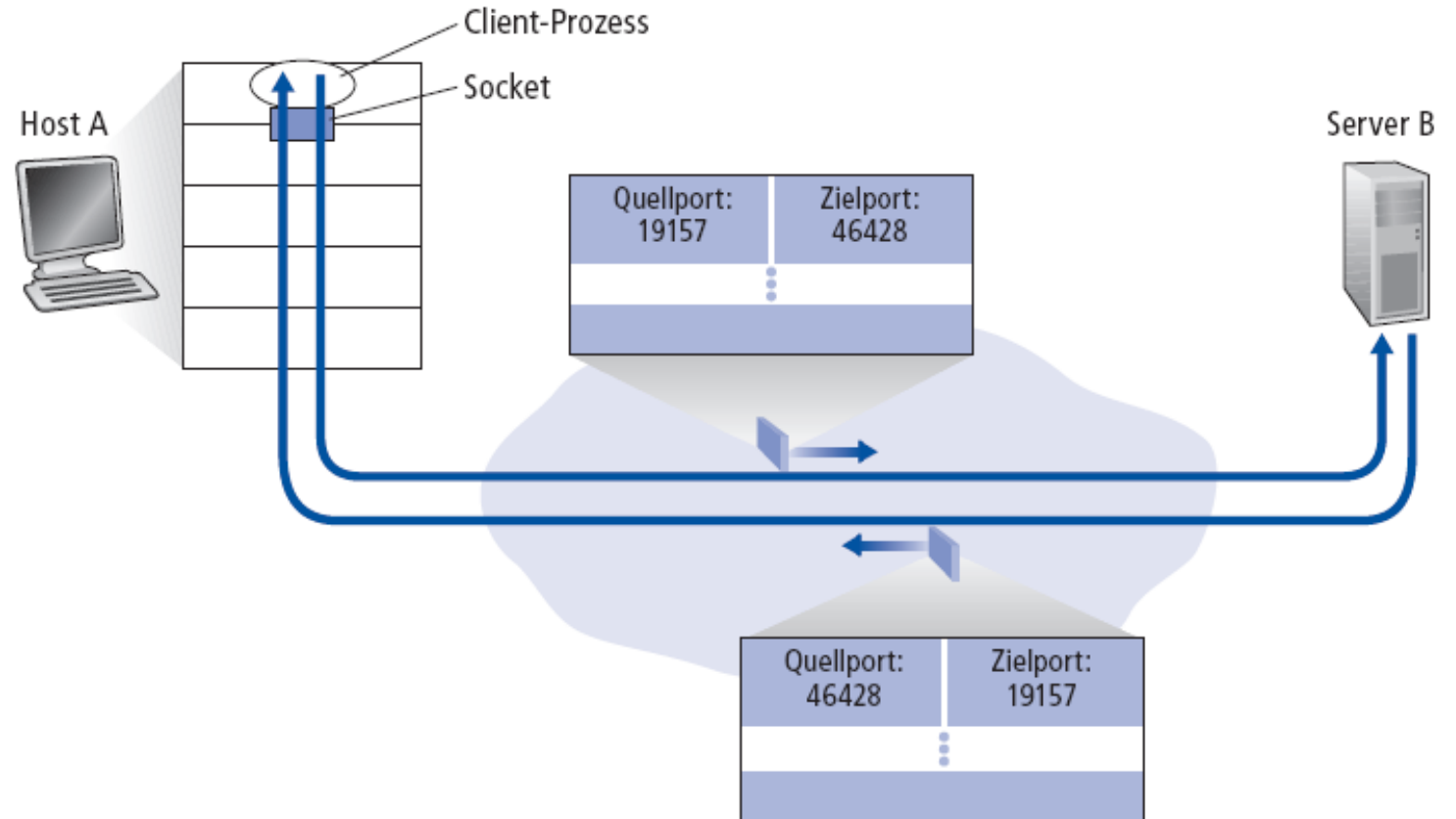
- Bei **nichtpersistentem HTTP** wird jede Anfrage über einen eigenen Socket beantwortet (dieser wird nach jeder Anfrage wieder geschlossen)

Verbindungsorientiertes Demultiplexing (TCP)



Verbindungsloses Demultiplexing (UDP)

```
DatagramSocket  
serverSocket = new  
DatagramSocket(46428);
```



Quellport ist der Port, an den geantwortet werden soll

UDP: User Datagram Protocol [RFC 768]

Minimales Internet-Transportprotokoll

“Best-Effort”-Dienst; UDP-Segmente können:

- verloren gehen.
- in der falschen Reihenfolge an die Anwendung ausgeliefert werden.

Verbindungslos (Nicht persistent):

- Kein Handshake zum Verbindungsaufbau
- Jedes UDP-Segment wird unabhängig von allen anderen behandelt

Warum gibt es UDP?

- Kein Verbindungsaufbau (der zu Verzögerungen führen kann)
- Einfach: kein Verbindungszustand im Sender oder Empfänger
- Kleiner Header
- Keine Überlastkontrolle: UDP kann so schnell, wie von der Anwendung gewünscht, senden

UDP: Fortsetzung

Häufig für Anwendungen im Bereich Multimedia-Streaming eingesetzt

- verlusttolerant
- Mindestrate

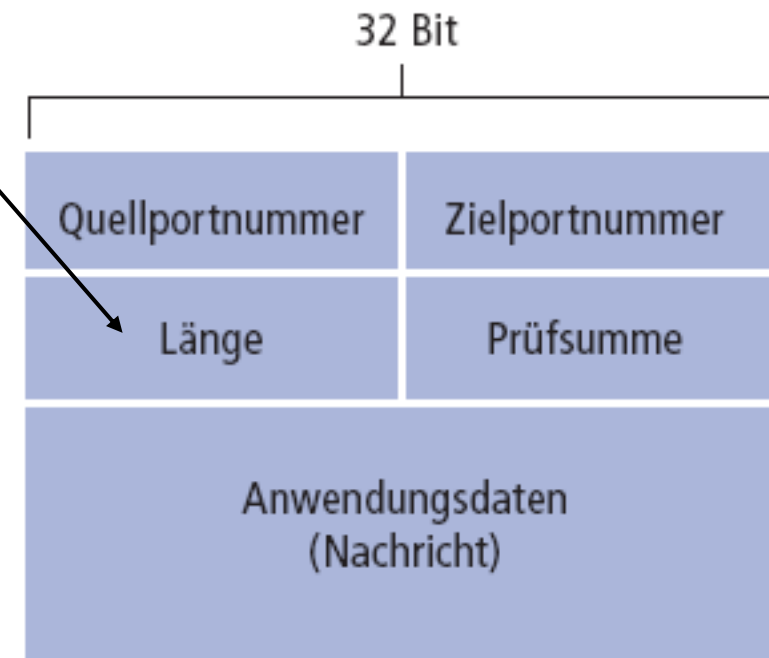
Andere Einsatzgebiete

- DNS
- SNMP

**Zuverlässiger Datentransfer über UDP:
Zuverlässigkeit auf der Anwendungsschicht
implementieren**

- Anwendungsspezifische Fehlerkorrektur!

**Länge (in
Byte) des
UDP-Segments,
inklusive
Header**



UDP-Segment-Format

Ziel: Fehler im übertragenen Segment erkennen (z.B. verfälschte Bits)

Sender

Betrachte Segment als Folge von 16-Bit-Integer-Werten

Prüfsumme: Addition (1er-Komplement-Summe) dieser Werte

Sender legt das invertierte Resultat im UDP-Prüfsummenfeld ab

Empfänger

Berechne die Prüfsumme des empfangenen Segments inkl. des Prüfsummenfeldes

Sind im Resultat alle Bits 1?

- **NEIN – Fehler erkannt**
- **JA – kein Fehler erkannt**

Prüfsummenbeispiel

Anmerkung

- Wenn Zahlen addiert werden, dann wird ein Übertrag aus der höchsten Stelle zum Resultat an der niedrigsten Stelle addiert

Beispiel: Addiere zwei 16-Bit-Integer-Werte

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		<hr/>															
Übertrag	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
		<hr/>															
Summe		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
Prüfsumme		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1