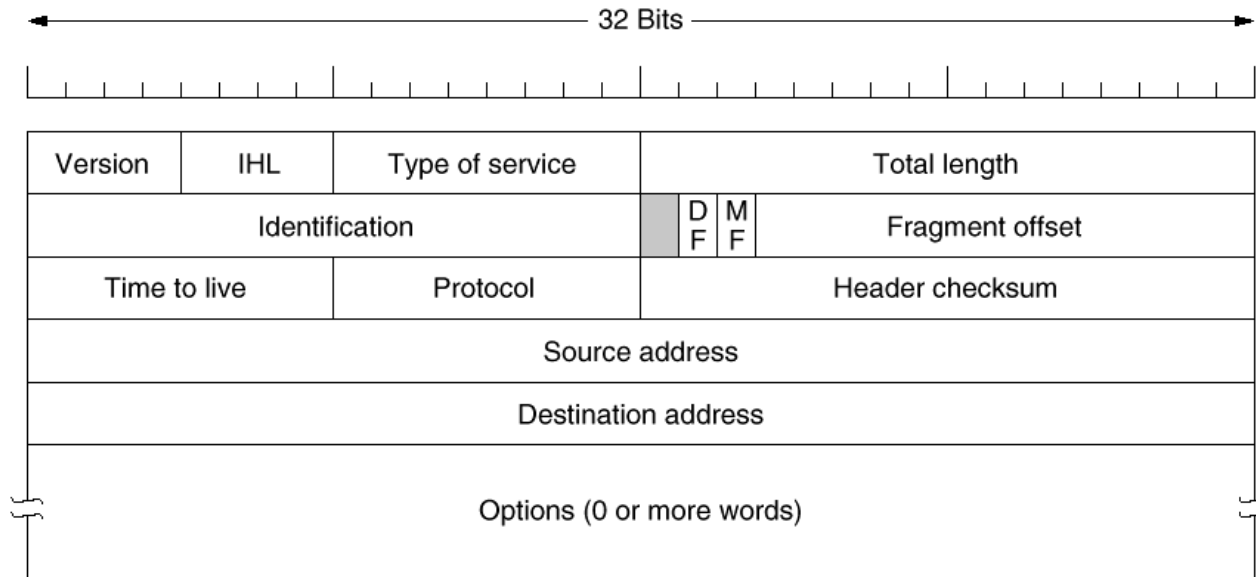

Rechnernetze und Telekommunikation

TCP/IP - Grundlagen

Funktionalität von IP

- ◆ **Best-Effort Dienst zum Transport von Datagrammen von der Quelle zum Ziel**
 - ◆ Best-Effort: kann klappen, muss aber nicht
 - ◆ Datagramme: einzelne Pakete, keine ganzen Datenströme
 - ◆ Quelle zum Ziel: von Rechner zu Rechner, nicht von Programm zu Programm
- ◆ **Unabhängig davon, ob diese Rechner im gleichen Netz liegen oder nicht**
- ◆ **Fragmentiert diese Datagramme und baut sie falls erforderlich wieder zusammen (reassembly)**
 - ◆ Um mit unterschiedlichen Maximal-Paketgrößen in verschiedenen Netzwerken umgehen zu können
 - ◆ Heute nur noch sehr selten verwendet!

IPv4 Header (1)



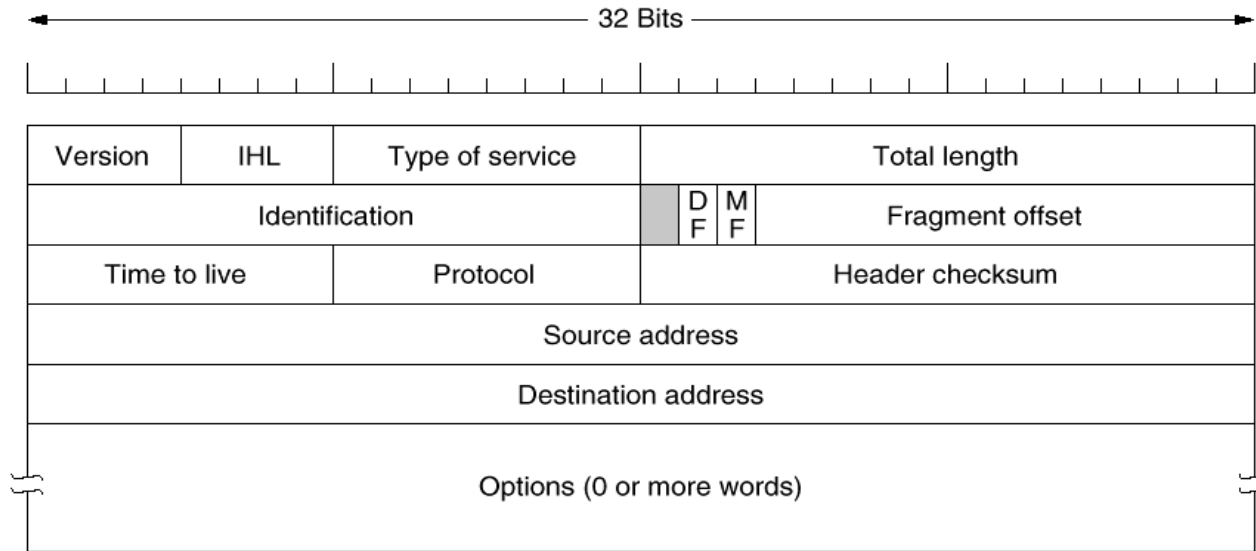
◆ Version

- Zz. v4, ermöglicht gemischten Betrieb mit neueren Versionen (IPv6!)

◆ IHL

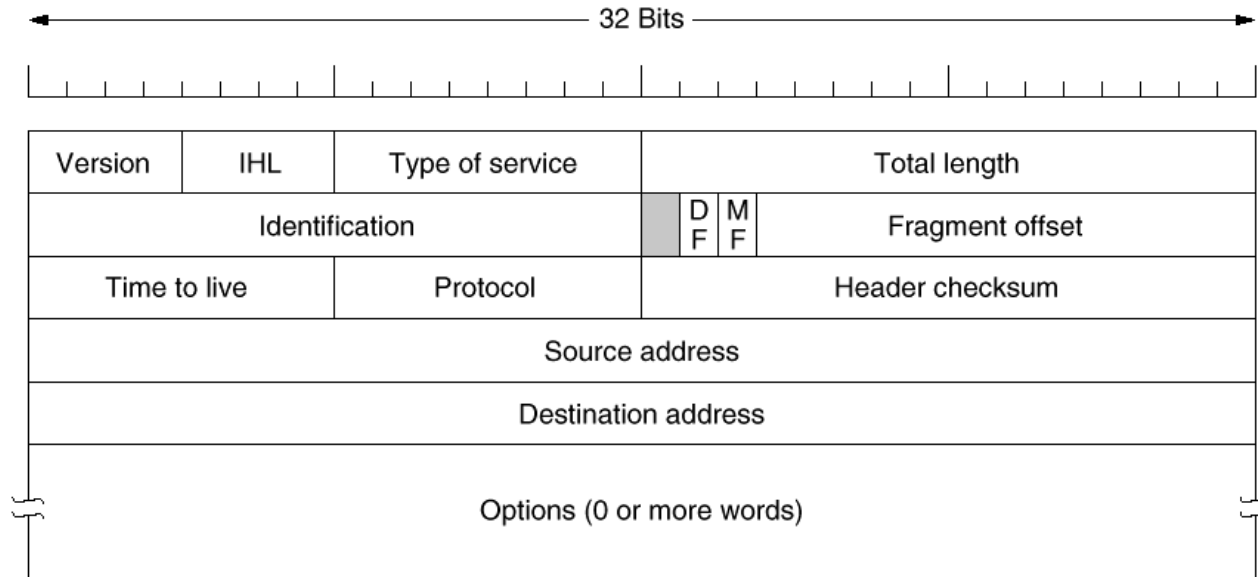
- Header Length (Einheiten von 32 Bits, min 5, max 15)

IPv4 Header (2)



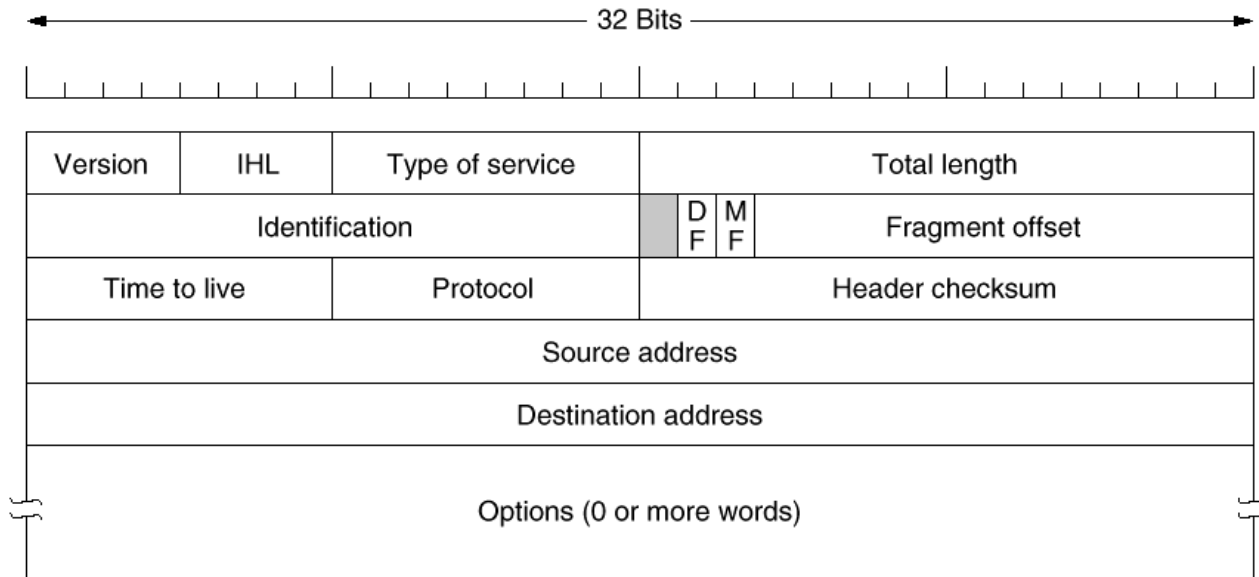
- ◆ **Type of service (usually ignored)**
 - 3 bits *precedence (priority)*, *normal to network control*
 - 3 flags (Delay, Throughput, Reliability)
- ◆ **Total length**
 - Length of header and data in bytes (max. 65535)

IPv4 Header (3)



- ◆ **Identification** - identifies parts of a fragment
- ◆ **DF** - “Don’t Fragment”
- ◆ **MF** - “More Fragments”
- ◆ **Fragment Offset** (in 8 Byte Einheiten)

IPv4 Header (4)



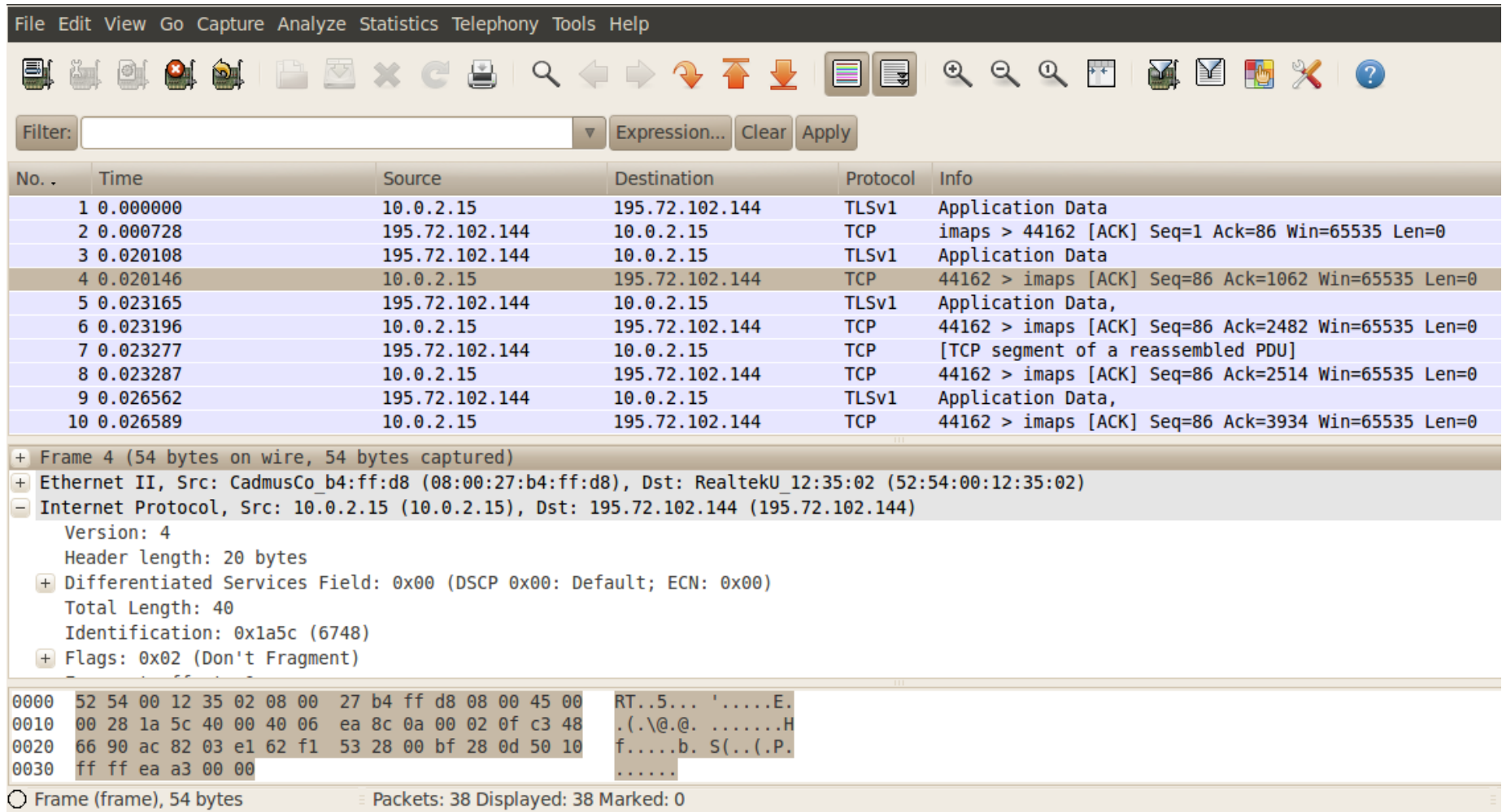
◆ Time to live

- Verbleibende Zeit in sec (max. 255), normalerweise “Hops”

◆ Protocol

- Transportprotokoll zu dem das Datagramm gehört (TCP,UDP)

Ein Paket im Netzwerk-Sniffer (Wireshark)



The image shows the Wireshark network sniffer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A filter bar is present with a text input field and buttons for Expression..., Clear, and Apply.

The main packet list displays the following data:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.0.2.15	195.72.102.144	TLSv1	Application Data
2	0.000728	195.72.102.144	10.0.2.15	TCP	imaps > 44162 [ACK] Seq=1 Ack=86 Win=65535 Len=0
3	0.020108	195.72.102.144	10.0.2.15	TLSv1	Application Data
4	0.020146	10.0.2.15	195.72.102.144	TCP	44162 > imaps [ACK] Seq=86 Ack=1062 Win=65535 Len=0
5	0.023165	195.72.102.144	10.0.2.15	TLSv1	Application Data,
6	0.023196	10.0.2.15	195.72.102.144	TCP	44162 > imaps [ACK] Seq=86 Ack=2482 Win=65535 Len=0
7	0.023277	195.72.102.144	10.0.2.15	TCP	[TCP segment of a reassembled PDU]
8	0.023287	10.0.2.15	195.72.102.144	TCP	44162 > imaps [ACK] Seq=86 Ack=2514 Win=65535 Len=0
9	0.026562	195.72.102.144	10.0.2.15	TLSv1	Application Data,
10	0.026589	10.0.2.15	195.72.102.144	TCP	44162 > imaps [ACK] Seq=86 Ack=3934 Win=65535 Len=0

The detailed view of Frame 4 (54 bytes on wire, 54 bytes captured) shows the following structure:

- Ethernet II, Src: CadmusCo_b4:ff:d8 (08:00:27:b4:ff:d8), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
- Internet Protocol, Src: 10.0.2.15 (10.0.2.15), Dst: 195.72.102.144 (195.72.102.144)
 - Version: 4
 - Header length: 20 bytes
 - Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
 - Total Length: 40
 - Identification: 0x1a5c (6748)
 - Flags: 0x02 (Don't Fragment)

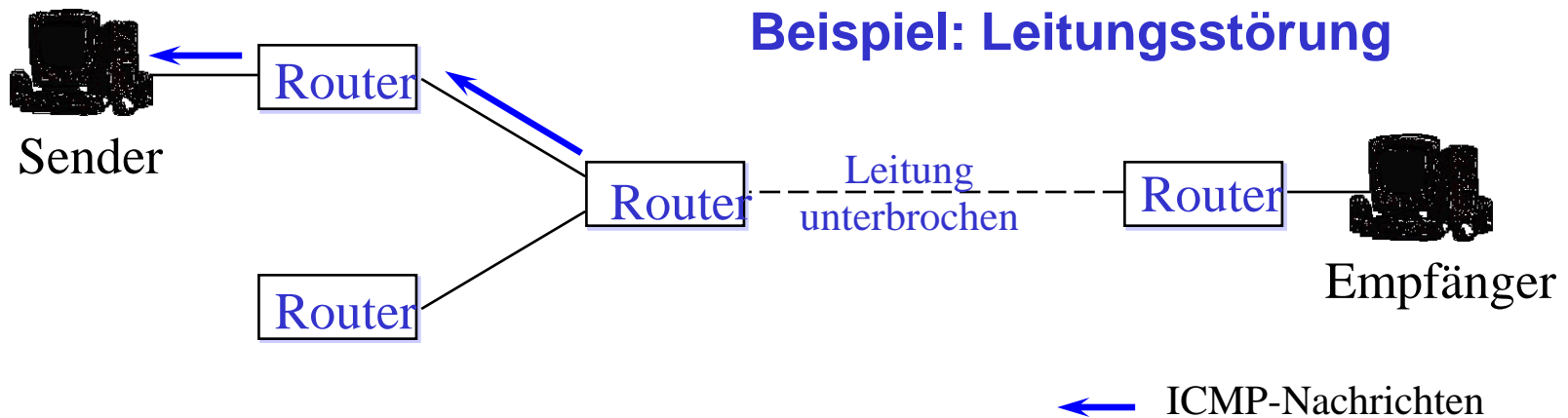
The packet bytes are displayed in hexadecimal and ASCII format:

```
0000 52 54 00 12 35 02 08 00 27 b4 ff d8 08 00 45 00 RT..5... '....E.
0010 00 28 1a 5c 40 00 40 06 ea 8c 0a 00 02 0f c3 48 .(. \@.@. ....H
0020 66 90 ac 82 03 e1 62 f1 53 28 00 bf 28 0d 50 10 f....b. S(..(P.
0030 ff ff ea a3 00 00 .....
```

The status bar at the bottom indicates: Frame (frame), 54 bytes | Packets: 38 Displayed: 38 Marked: 0

ICMP – Internet Control Message Protocol (1)

- ◆ Einzelne Paketverluste werden im Normalfall von IP nicht gemeldet (unzuverlässiger Datagrammdienst).
- ◆ Schwerwiegende Probleme werden zur Vermeidung von Folgefehlern mittels ICMP den Kommunikationspartnern mitgeteilt.



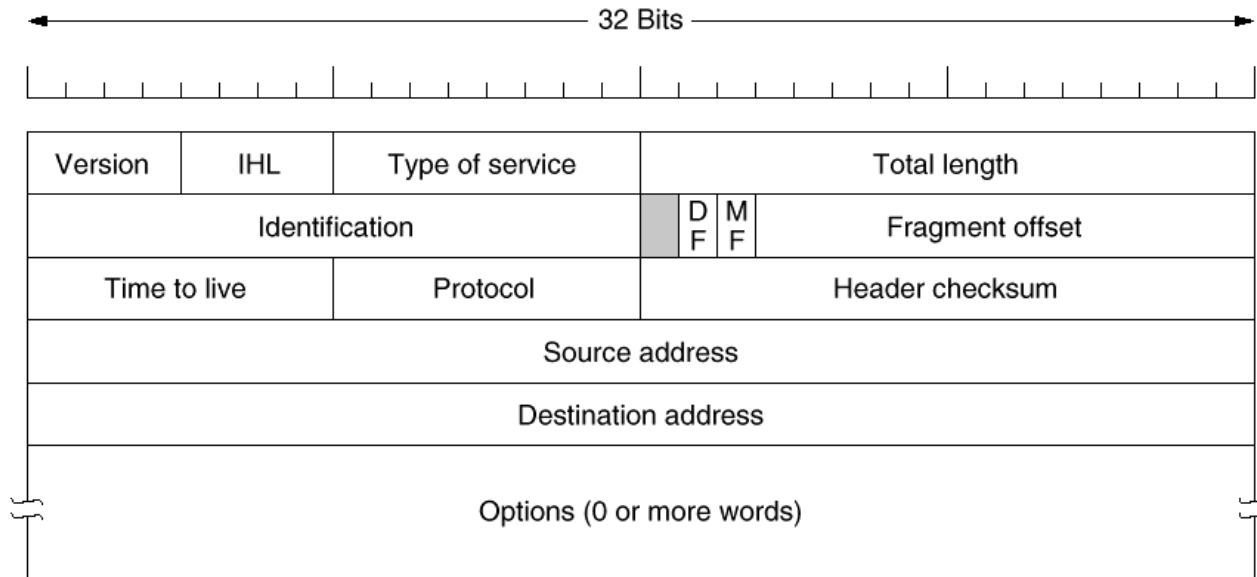
- ◆ ICMP unterstützt den Austausch von Fehlermeldungen, Statusanfragen und Zustandsinformation.

ICMP - Internet Control Message Protocol (2)

◆ ICMP-Nachrichtentypen

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

The IPv4 Header Format (5)

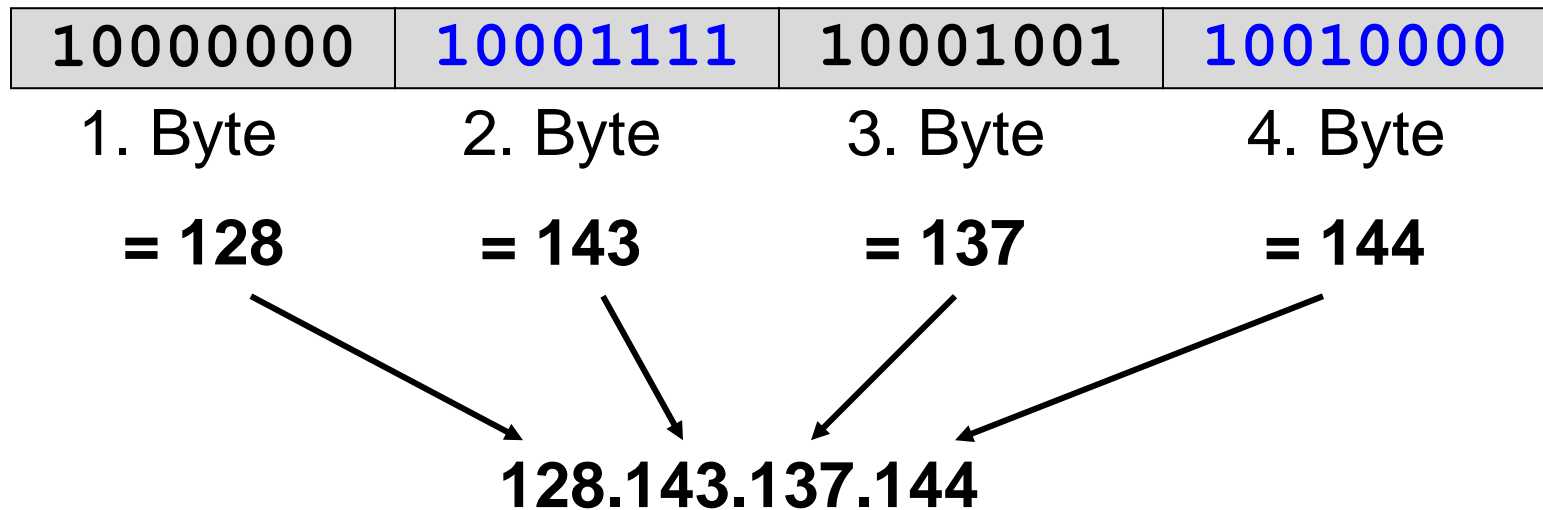


◆ Options

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

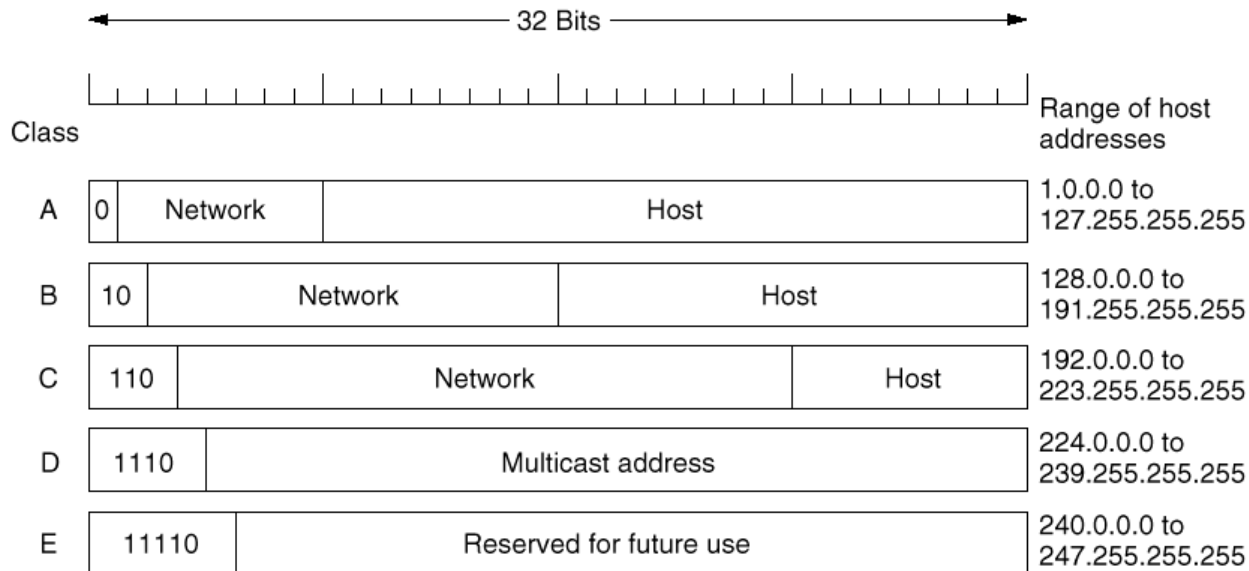
IPv4-Adressen

- ♦ **32 Bit-Werte**
 - d.h. es gibt max. 4.294.967.296 verschiedene Adressen
- ♦ **Dargestellt meist als 4 Bytes in Dezimaldarstellung durch Punkte getrennt**
 - Historisch bedingt, extrem unpraktisch, aber Standard
- ♦ **Beispiel:**



IPv4 klassische Adressformate

- ◆ Unterteilung in Netzwerk (geroutet) und Hostteil (lokal)
- ◆ Class A: 126 Netzwerke mit 16 Millionen Hosts
- ◆ Class B: 16382 Netzwerke mit 64k Hosts
- ◆ Class C: ca. 2 Millionen Netzwerke mit 254 Hosts



Spezielle IPv4 Adressen

- ◆ **Broadcast-Adresse eines Netzes**
 - ◆ Broadcast = Nachricht an alle Hosts des Netzes
 - ◆ Letzte Adresse des Netzes reserviert für Broadcast
 - ◆ d.h. Hostteil alles Einsen
 - ◆ Beispiel: Broadcast-Adresse des Class C Netzes 192.168.0.0 ist 192.168.0.255

- ◆ **Universelle Broadcast-Adresse 255.255.255.255**
 - ◆ Nachricht an alle im eigenen Netz (egal, wie das Netz heißt)

- ◆ **Loopback Adresse 127.x.x.x**
 - ◆ Alle Adressen in diesem Class A-Netz gehen an den eigenen Rechner, z.B. meist 127.0.0.1

IPv4 Adressen in privaten Netzen

- ◆ **Geregelt im RFC 1918 (Address Allocation for Private Internets)**
 - Jeder kann aus diesen Bereichen den Adressbereich für sein eigenes privates Netz auswählen

Die folgenden Adressbereiche sind für private Netze reserviert:

- ◆ **Klasse A:** 10.0.0.0
 - Privates Klasse A-Netz: 10.0.0.0 bis 10.255.255.254
- ◆ **Klasse B:** 172.16.0.0 bis 172.31.0.0
 - Es sind 16 Klasse B-Netze reserviert
Jedes dieser Netze kann aus bis zu 65.534 Hosts bestehen
(z.B. ein Netz mit den Adressen von 172.17.0.1 bis 172.17.255.254).
- ◆ **Klasse C:** 192.168.0.0 bis 192.168.255.0
 - 256 Klasse C-Netze stehen zur privaten Nutzung zur Verfügung.
Jedes dieser Netze kann jeweils 254 Hosts enthalten
 - Häufig genutzt bei DSL-Routern

CIDR - Classless InterDomain Routing

◆ Problemen

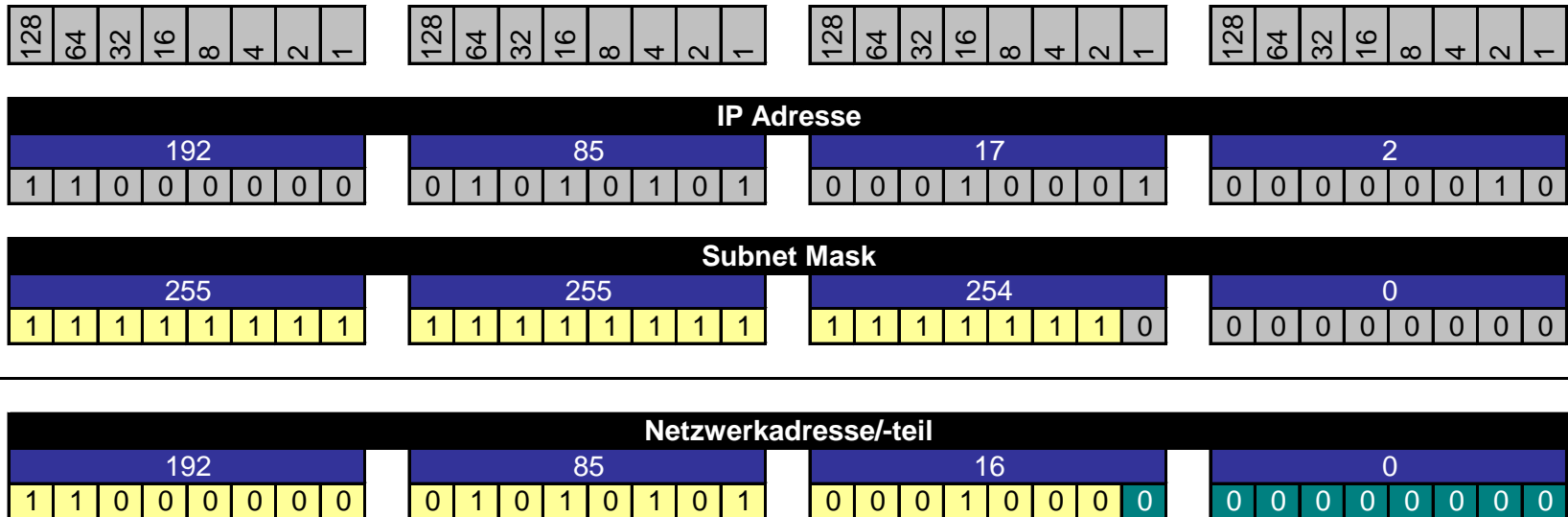
- IP Adressen wurden knapp
- Class A und B Netzwerke sind zu groß, Class C zu klein
- Explosion der Routing Tabellen sollte vermieden werden

◆ Lösung

- 1993 eingeführt (RFC 1518, RFC 1519)
- Länge von Netzwerk- und Hostteil kann beliebig gewählt werden
- Vergabe der Netzwerke in Größen von 2^n
- Generell wird bei Netzen immer die Länge der Adresse mit angegeben
 - Als Netzmaske (markiert Bits im Netzteil mit 1):
 - z.B. 255.255.254.0
 - Oder äquivalent als Anzahl der Bits im Netzteil mit “/”
 - z.B. 192.85.16.0/23

CIDR - Beispiel

◆ IP-Adresse 192.85.17.2 und Subnetmask 255.255.254.0 (/23)



◆ Durch log. UND ergibt sich aus der gesamten Adresse die Netzadresse:

- In diesem Beispiel: 192.85.16.0

Noch offene Punkte zu IP (werden in weiteren Vorlesungen besprochen)

- ◆ **Details zur Struktur von IP-Subnetzen und IP-Adressen**
 - **Beispiele zur Adressrechnung**
- ◆ **Routing in IP-Netzen**
- ◆ **IPv6**

TCP (Transmission Control Protocol)

◆ Ziel:

- Zuverlässiger, verbindungsorientierter Byte-Strom über ein unzuverlässiges Netz (Internet)

◆ Anforderungen:

- Der Byte-Strom des Benutzers wird in Pakete von max. 64 KByte Größe unterteilt

◆ Erbrachter Dienst:

- Wiederherstellung des ursprünglichen Byte-Stroms durch Ordnung der Pakete in der richtigen Reihenfolge
- Timeout und Wiederholung um die Zuverlässigkeit der Übertragung zu gewährleisten

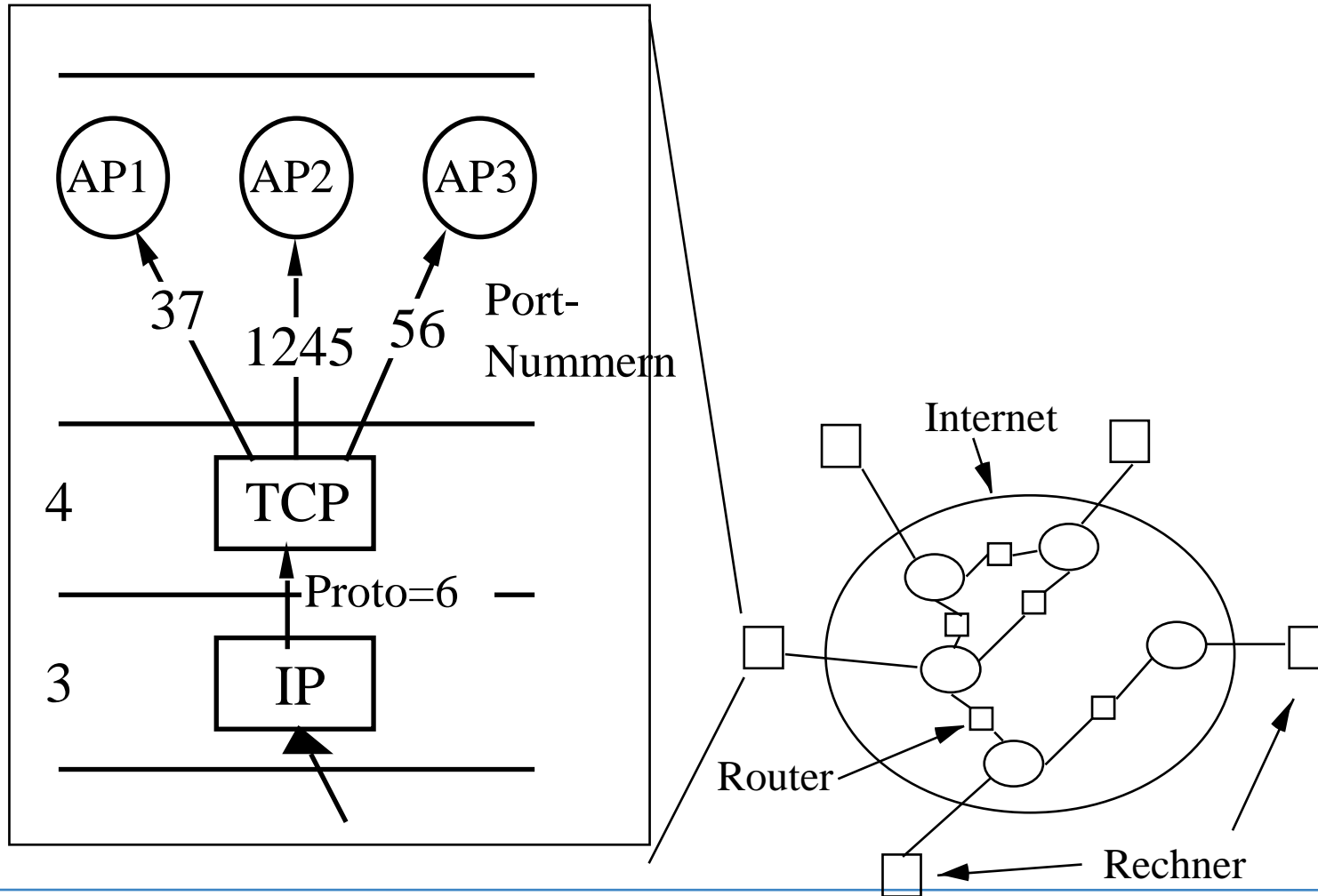
◆ Das TCP Servicemodell

- Sender und Empfänger erzeugen als Endpunkte sog. Sockets
- Jeder Socket hat als ID (Adresse) eine lokale Nummer (sog. Port)
- Um den TCP Dienst wird auf einer Verbindung zwischen den Sockets von Sender und Empfänger erbracht
- Ein Socket kann mehrere Verbindungen zu einem Zeitpunkt haben
- Verbindungen werden durch die Socket-IDs beider Enden bezeichnet: (Socket1, Socket2)

TCP - Portnummern

- ◆ Adressierung der Applikationen
- ◆ Portnummern sind 16 Bit groß (65.535 TCP-Verbindungen)
- ◆ Portnummern sind nicht einzigartig zwischen den Transportprotokollen, die Transportprotokolle haben jeweils eigene Adressräume.
- ◆ Eine IP-Adresse zusammen mit der Portnummer spezifiziert einen *Socket*.
- ◆ auf UNIX-Systemen sind Portnummern in der Datei „/etc/services“ definiert.
- ◆ Portnummern sind in drei Bereiche aufgeteilt:
 - 0 – 1023 well-known ports (root-Rechte!)
 - 1024 – 49151 registered ports
 - 49152 – 65535 dynamic and/or private ports

Adressierung von Anwendungsprozessen: Beispiel TCP/IP - Portnummern



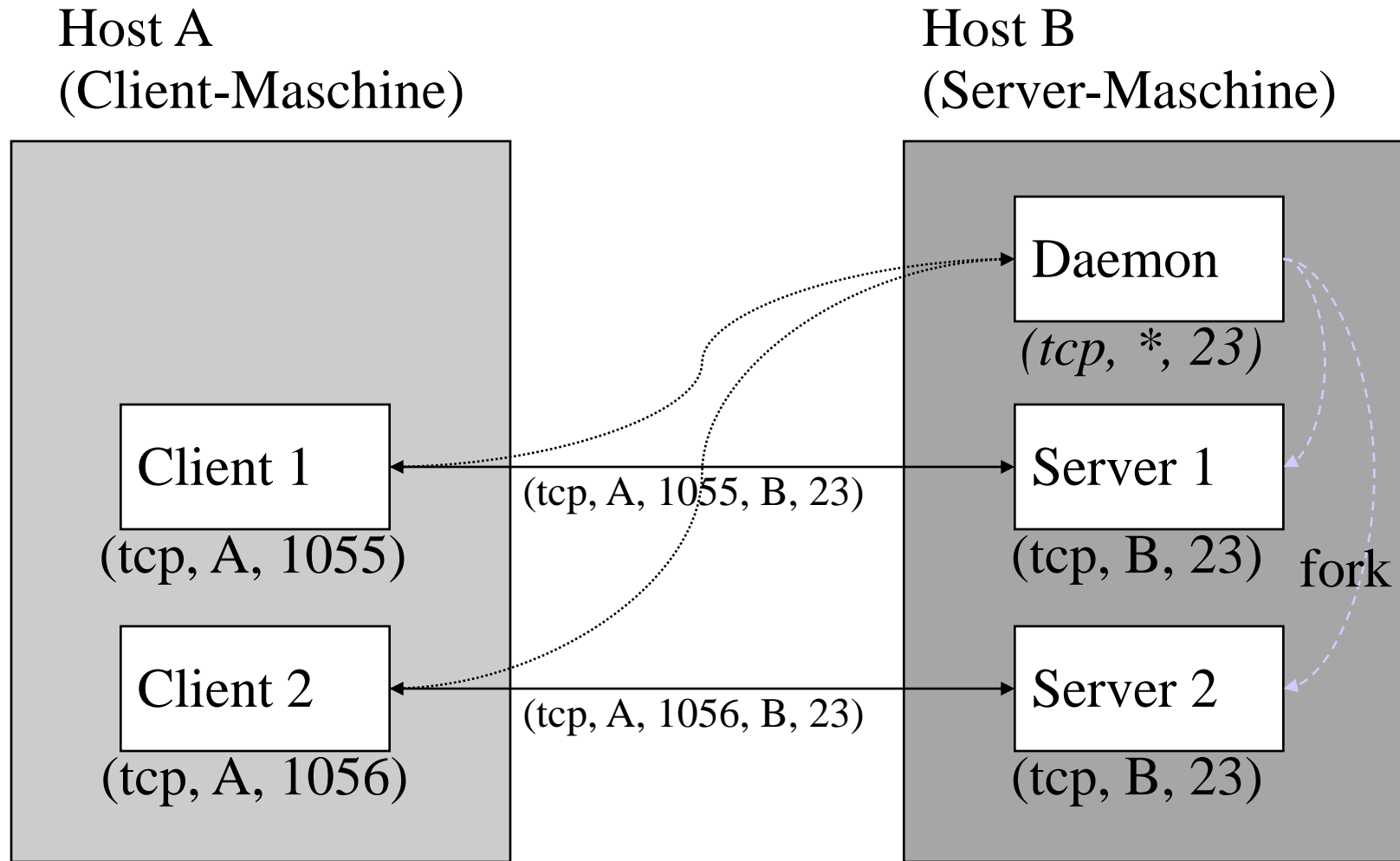
Well Known Ports (Auswahl)

Vordefinierte Dienste

[ftp](#) 21/tcp File Transfer [Control]
[telnet](#) 23/tcp Telnet
[smtp](#) 25/tcp Simple Mail Transfer
[smtp](#) 24/tcp any private mail system
[time](#) 37/tcp Time
[time](#) 37/udp Time
[rap](#) 38/tcp Route Access Protocol
[rap](#) 38/udp Route Access Protocol
[nicname](#) 43/tcp Who Is
[login](#) 49/tcp Login Host Protocol
[xns-time](#) 52/tcp XNS Time Protocol
[dns](#) 53/tcp Domain Name Server
[sql*net](#) 66/tcp Oracle SQL*NET

[bootpc](#) 68/udp Bootstrap Protocol Client
[tftp](#) 69/udp Trivial File Transfer
[http](#) 80/tcp World Wide Web HTTP
hosts2-ns
[pop](#) 110/tcp Mail abholen
[nntp](#) 119/tcp Network News
Transfer Protocol
[imap2](#) 43/tcp Interactive Mail Access
Protocol v2
[https](#) 443/tcp https
[irc](#) 6665-6669/tcp chatten

Identifikation von Verbindungen



Der TCP Segment-Header

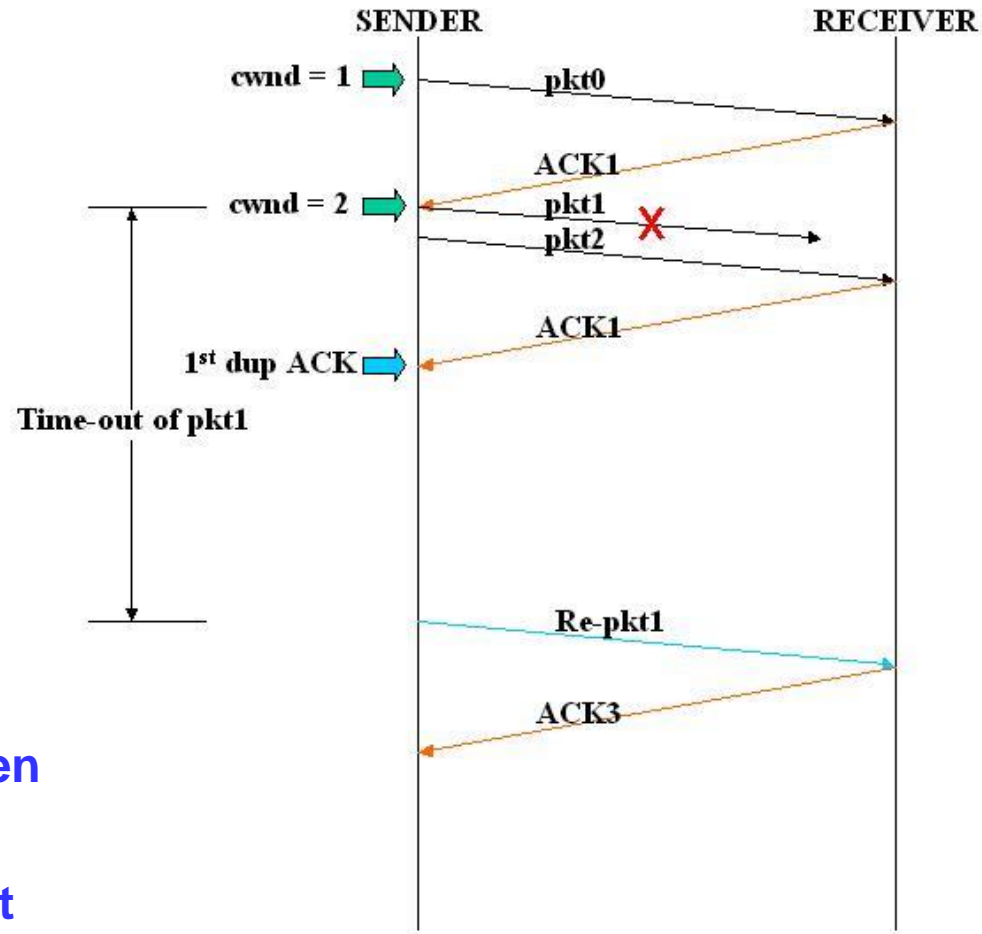
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Quell-Portnummer															Ziel-Portnummer																
Sequenznummer																															
Quittungsnummer																															
Header				Reserviert				Code Bits				Fenstergröße																			
Prüfsumme															Urgent-Zeiger																
Optionen																								Füllzeichen							

- ◆ **Quell-Port, Ziel-Port** (jew. 2 Bytes): identifiziert Anfangs- und Endpunkt einer Verbindung
- ◆ **Sequenznummer, Quittungsnummer** (4 Bytes): Folgenummern – werden beim Verbindungsaufbau generiert und fortlaufend erhöht
- ◆ **Header** (1 Byte): Anzahl der 32-Bit-Wörter in TCP-Header (variables Optionenfeld)
- ◆ **Code Bits** (6 Bits): URG, SYN, ACK, FIN, RST und PSH
- ◆ **Fenstergröße** (2 Bytes): variabler Schiebefenstermechanismus zur Flusssteuerung
- ◆ **Urgent-Zeiger** (2 Bytes): Zeigt auf Daten innerhalb des Payloads, die von besonderer Wichtigkeit sind

Fehlerbehandlung bei TCP

Garantie eines zuverlässigen Datenstroms

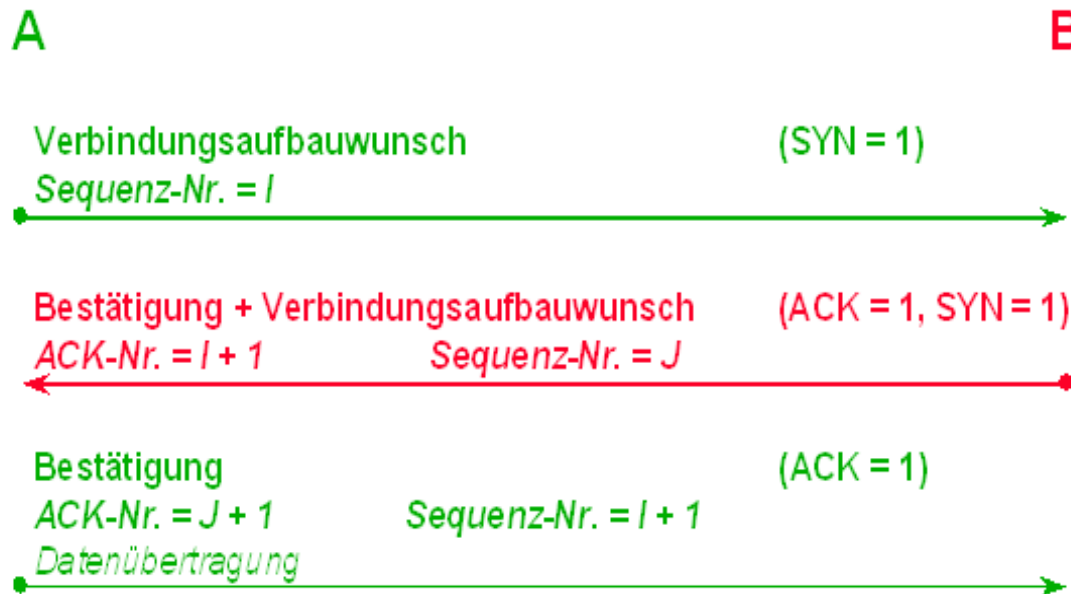
- ◆ Datenbytes haben eine Sequenznummer
 - **SEQ:** Nummer der gesendeten Daten
- ◆ Empfangene Bytes werden quittiert
 - **ACK:** Nummer des ersten noch nicht empfangenen Bytes
- ◆ Empfängt der Sender nicht innerhalb eines Timeouts ein ACK, wiederholt er die Daten
 - Funktioniert sowohl, wenn Daten als auch ACKs verloren gehen
 - Empfänger erkennt und verwirft doppelte Daten



Transmission Control Protocol

Verbindungsaufbau – Three-Way-Handshake

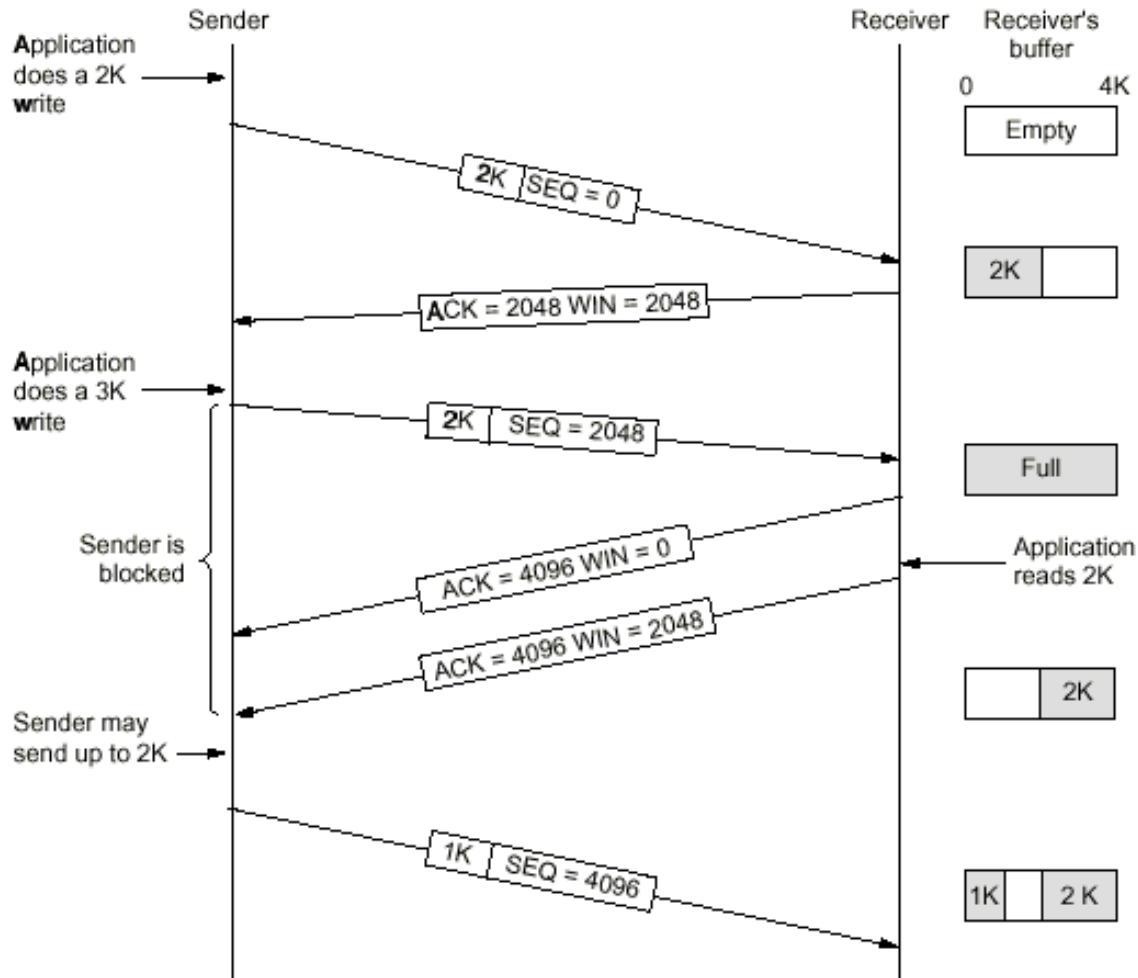
- ◆ Aktives Öffnen einer Verbindung (SYN-Nachricht)
- ◆ Passive Seite nimmt eine Verbindung auf einer bestimmten Port-Nummer entgegen
- ◆ Die initialen Sequenznummern werden auf jeder Seite zufällig gewählt und bestätigt.



Window management bei TCP

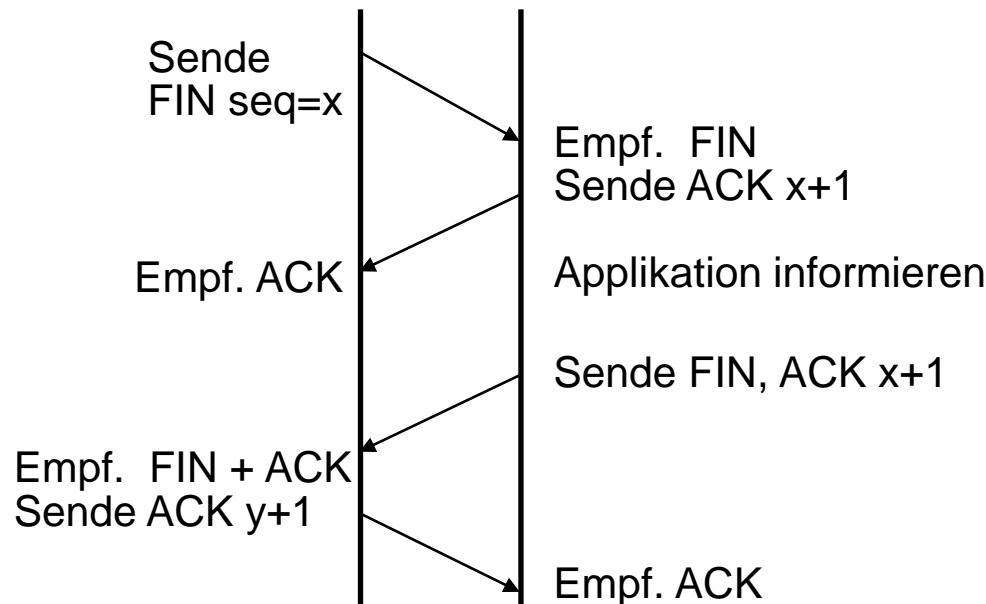
- ◆ **Fenster (Window) = Teil des Datenstroms, der jetzt maximal empfangen werden kann**
- ◆ **Die Fenstergrösse im TCP ist ein "Sliding Window" Protokoll, d.h. Fenstergrösse wird an den "Füllstand" des Netzes bzw. des Empfängers angepasst**
- ◆ **Zur Flusssteuerung**
 - **Damit kann der Empfänger verhindern, dass er vom Sender "überrannt" wird**
 - **Jedes Bestätigungspaket enthält einen "window advertisement" Wert, in dem der Empfänger angibt, für wieviele weitere Pakete er noch freie Kapazität hat (das Fenster kann also grösser oder kleiner werden)**

Window management bei TCP - Beispiel



Abbau einer TCP-Verbindung

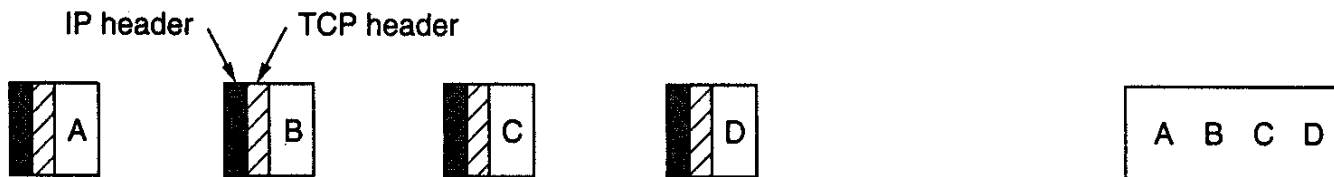
- ◆ 4-fach-Handshake; jede Seite wird separat beendet (TCP half close)
- ◆ Beispiel: Aktive Seite (links) schliesst Verbindung mit FIN-Flag
- ◆ Neue Daten werden nicht mehr übertragen, von rechts ankommende Daten werden jedoch noch bestätigt.



(Berkeley) Sockets Primitive für TCP

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

◆ Byte-Strom (NICHT Nachrichten-Strom)



TCP-Kommunikation mit Sockets

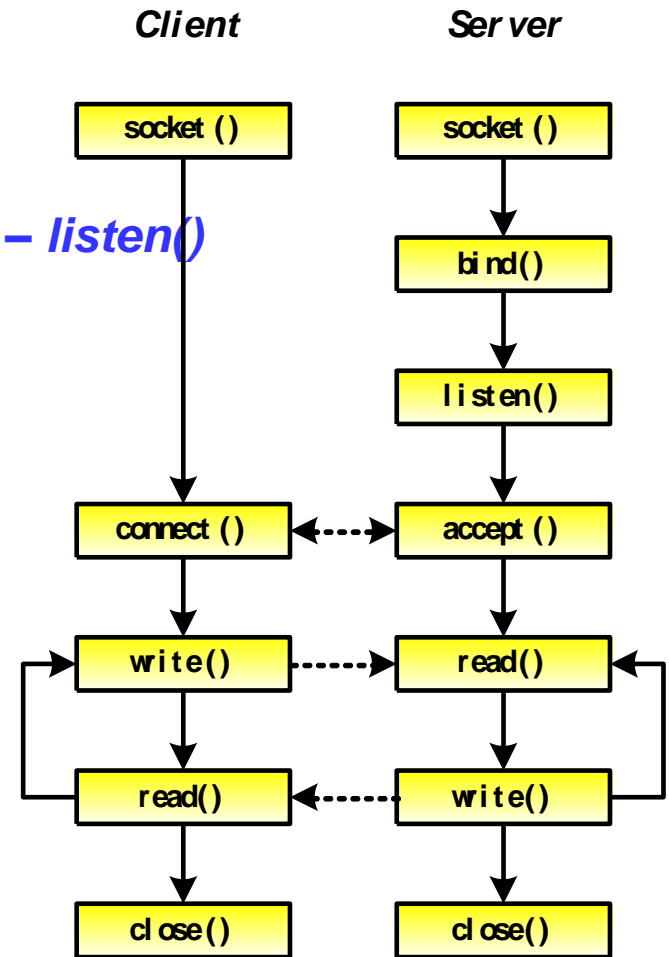
Client/Server

◆ Verbindungsaufbau asymmetrisch:

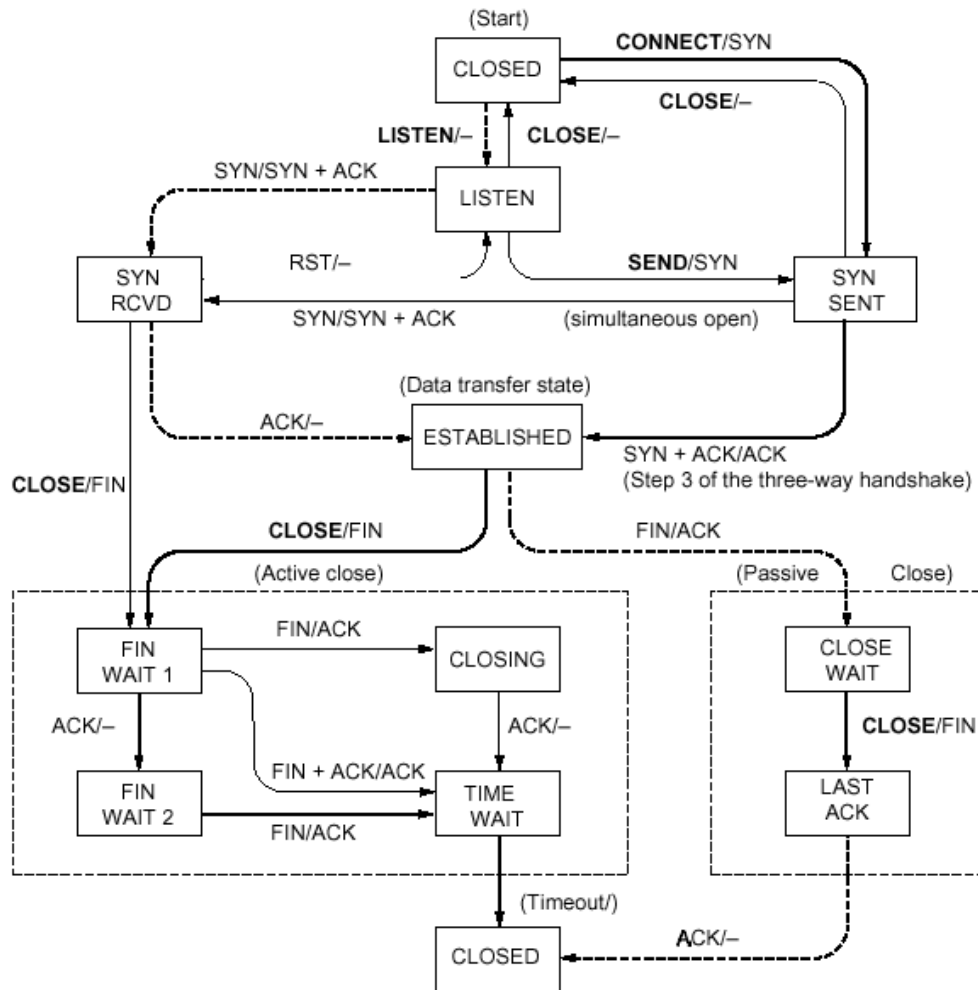
- Server nutzt eine bestimmte Adresse (IP, Port) – *bind()*
- Server bereitet sich auf Verbindungen vor – *listen()*
- Client öffnet Verbindung – *connect()*
- Server nimmt Verbindung an – *accept()*

◆ Kommunikation dann symmetrisch

- Client und Server können beide Byte-Ströme schreiben und lesen – *read()*, *write()* [auch *send()* und *receive()*]
- Client und Server können beide die Kommunikation beenden – *close()*



TCP Zustandsautomat



State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

TCP Zustandsübergänge

- > Client (normal)
- - - - -> Server (normal)
- > Client (selten)
- - - - -> Server (selten)

Der UDP Datagramm-Header

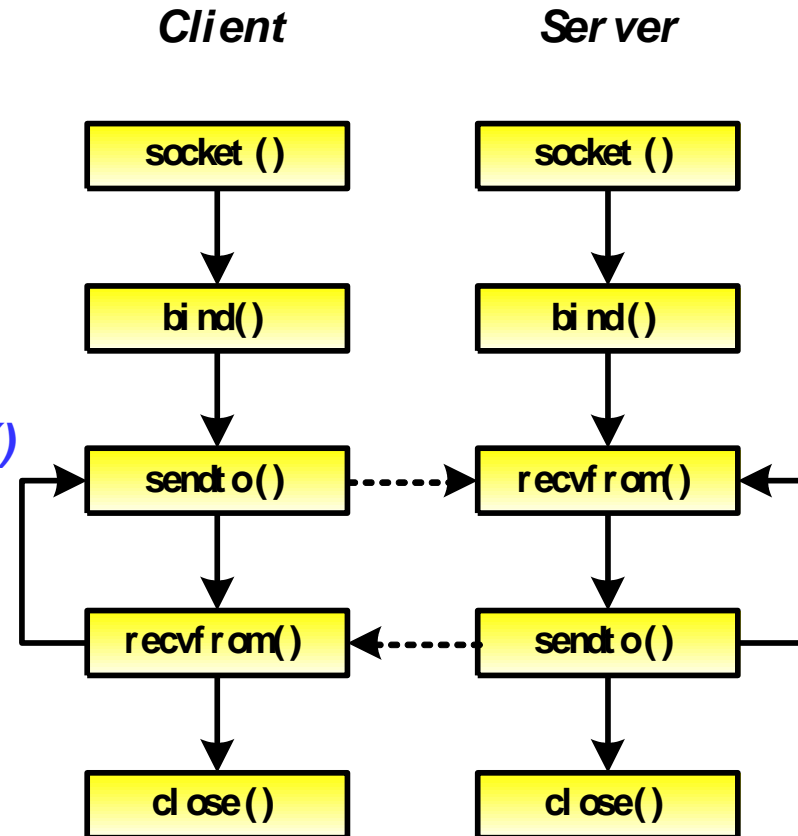
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Quell-Portnummer															Ziel-Portnummer																
Länge															Prüfsumme																
Daten																															

- ◆ UDP ist verbindungslos (im Gegensatz zu TCP)
- ◆ setzt auf dem unter ihm liegenden IP-Protokoll auf
- ◆ besitzt nur einen kleinen Overhead (keine Transportquittungen oder – bis auf Prüfsumme – keine Sicherheitsmaßnahmen)
 - Quell-Port, Ziel-Port (jew. 2 Bytes): identifiziert Anfangs- und Endpunkt einer Verbindung
 - Länge(2 Bytes): Länge des UDP-Headers
 - Prüfsumme (2 Bytes): alle Daten werden als 16-Bit-Wörter addiert und dann das Einerkomplement gebildet
 - Daten: zu übertragene Payload

UDP-Kommunikation mit Sockets

◆ Kommunikation symmetrisch

- Client und Server nutzt eine bestimmte Adresse (IP, Port) – *bind()*
 - Ohne *bind()* wird eine beliebige Portnummer gewählt
- Client und Server können beide Byte-Ströme schreiben und lesen – *sendto()* und *recvfrom()*
- Client und Server können beide die Kommunikation beenden – *close()*



Zusammenfassung

- ◆ **Das IP-Protokoll hält das Internet zusammen**
 - Bringt Pakete vom sendenden Rechner zum empfangenden
 - IP-Adressen sind strukturiert nach Netz- und Hostteil

- ◆ **TCP und UDP sind die Protokolle, die Prozesse/Programme nutzen**
 - TCP für den zuverlässigen Transport von Byteströme
 - UDP für den Best-Effort-Transport einzelner Datagramme

- ◆ **Alle erforderlichen Details zu TCP finden Sie im Dokument „Beschreibung TCP“ im StudIP – bitte lesen!**