
Rechnernetze und Telekommunikation

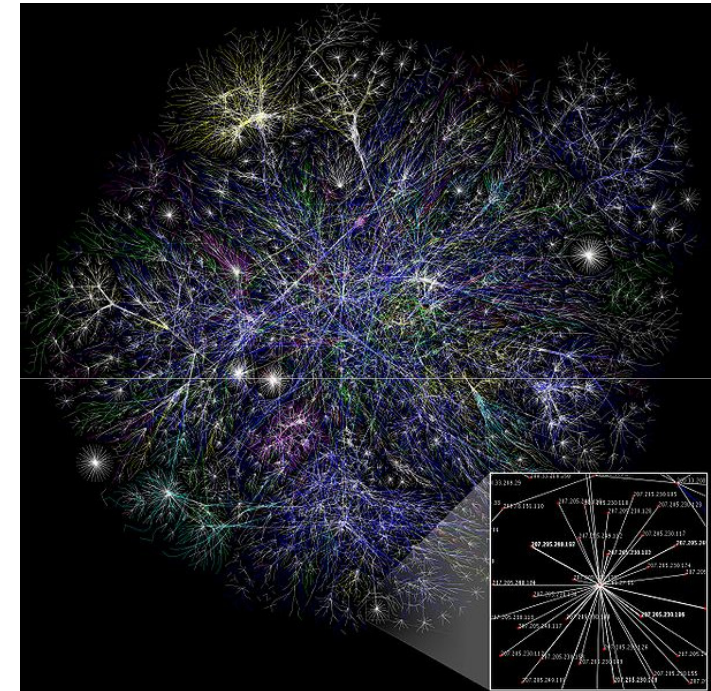
Internettechnologien

Übersicht

- ◆ **Definition und Organisation des Internets**
- ◆ **Protokolle, Schichten und Referenzmodell**
- ◆ **IP-Protokoll**
- ◆ **IP-Adressen**
- ◆ **TCP und UDP**

Definition: Internet

- ◆ Engl. Abk. für „Interconnected Network“
- ◆ Allgemein:
 - Die technische Vernetzung einzelner Computernetzwerke
- ◆ Speziell:
 - Das internationale Netz, das aus dem ehemaligen ARPAnet hervorgegangen ist
 - Auf dem Internetdienste wie WWW, Email, VoIP, etc. aufbauen.
- ◆ Oft Synonym für:
 - Für das WWW (World Wide Web), was aber nur ein Dienst auf dem Internet ist.



Teile einer "Karte" des Internets
opte.org am 15.01.2005

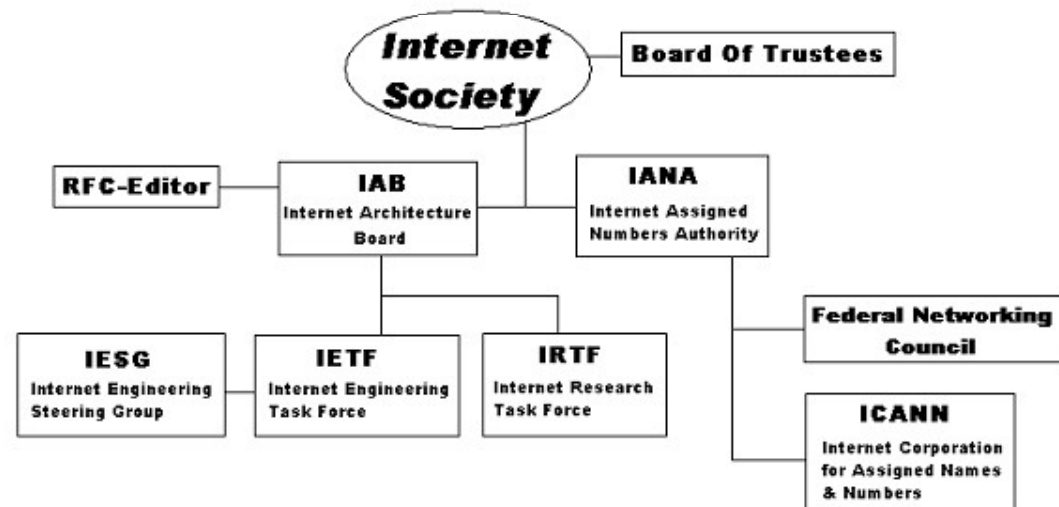
Geschichte des Internets

- 1969: Erstes „Internet“: Datenübertragung zwischen vier Rechnern der University of California at Los Angeles (UCLA), des Stanford Research Institute (SRI), der University of California at Santa Barbara (UCSB) und der University of Utah
- 1971: Betriebsaufnahme ARPAnet (erstes Internet-Backbone); Experiment zum Einloggen auf entfernten Rechnern; erstmalig Nutzung von E-Mail
- 1972: Erste öffentliche Demonstration des Netzwerkes
- 1973/74: Entwurf der TCP/IP-Protokolle zur Kopplung unterschiedlicher Netzwerke
- 1980: Integration der TCP/IP-Protokolle in UNIX (BSD)
- 1988: Das Internet umfasst auch Netze in Europa, Australien und Kanada. IP-Verbindung zum Internet aus Deutschland über EUNET Dortmund und Xlink Karlsruhe
- 1991: Erste öffentliche Demonstration des WWW am europäischen Kernforschungszentrum CERN

Wer hat etwas zu sagen im Internet? (1)

♦ Internet Society

- NGO seit 1992
- zur Pflege und Weiterentwicklung der Internetinfrastruktur des Internets
- Unterorganisationen
 - IANA: vergibt Adressen
 - ICANN: vergibt Top-Level-Domains
 - IETF: entwickelt neue Technologien
- Veröffentlichungen als RFC (Request for Comments)
 - z.B. RFC 793 (TCP);
 - RFC 959 (FTP)



Wer hat etwas zu sagen im Internet? (2)

- ◆ **World Wide Web Consortium (kurz: W3C)**
 - NGO seit 1994
 - zur Standardisierung der das World Wide Web betreffenden Techniken
- ◆ **Veröffentlicht *Recommendations* z.B. zu**
 - *Hypertext Markup Language (HTML)*
 - *Extensible Hypertext Markup Language (XHTML)*
 - *Extensible Markup Language (XML)*
 - *Extensible Stylesheet Language (XSL)*
 - *Cascading Style Sheets (CSS)*
 - *SOAP (SOAP)*
 - *Web Services Description Language (WSDL)*
 - ...

Wer hat etwas zu sagen im Internet? (3)

Außerdem

- ◆ **ITU: International Telecommunication Union**
 - 1865 gegründet als Internationalen Telegraphenverein
 - heute Sonderorganisation der Vereinten Nationen
 - für technischen Aspekten der Telekommunikation
 - Standards für Telefon(Netze), Mobilfunk, Frequenzen
 - ◆ **ISO: International Organization for Standardization**
 - seit 1946 für sonstige technische Standards
 - ◆ **IEEE: Institute of Electrical and Electronics Engineers**
 - Seit 1963 weltweiter Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informatik
 - Standardisierung von Techniken, Hardware und Software
 - z.B. Standards für Bussysteme und Protokolle, Ethernet, WLAN, ...
-

Ethische Aspekte des Internets

♦ Netzneutralität

- Darf das Netz (ein Netzwerkprovider) Inhalte und Anbieter bevorzugt behandeln?
- Wer zahlt die Kosten der Datenübertragung?

♦ Digitale Lücke

- Unterschiedliche Verfügbarkeit und Bandbreiten
- Stadt und Land, in den verschiedenen Ländern und Kontinenten

♦ Datenschutz

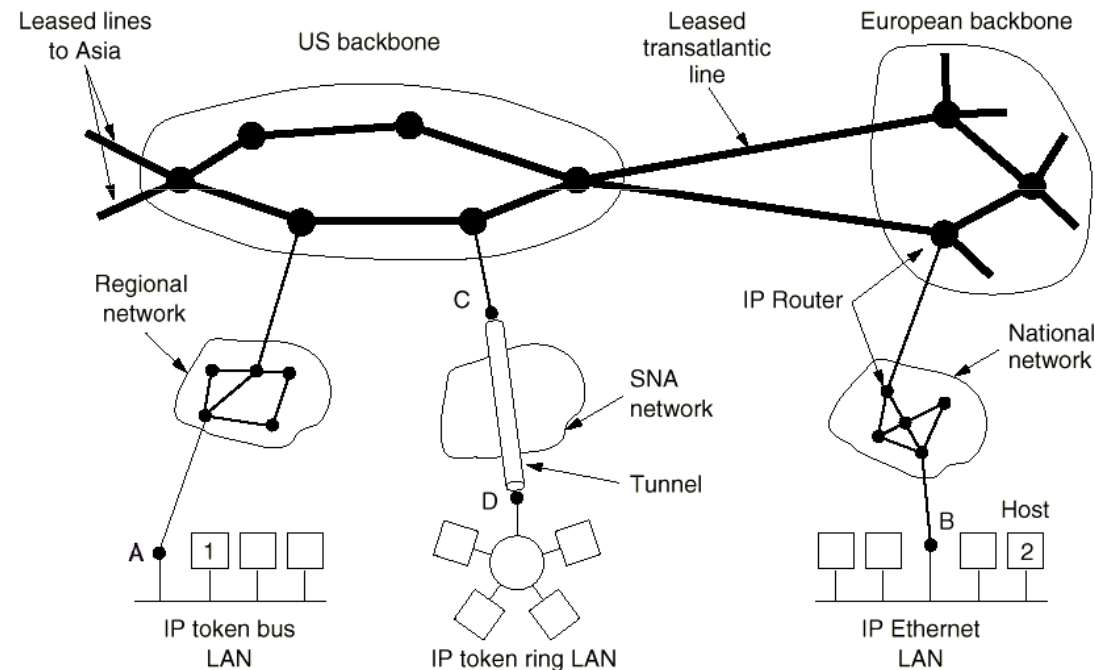
- Anonymität und informationelle Selbstbestimmung vs.
- Kommerzielle und staatliche Interessen

♦ Zensur

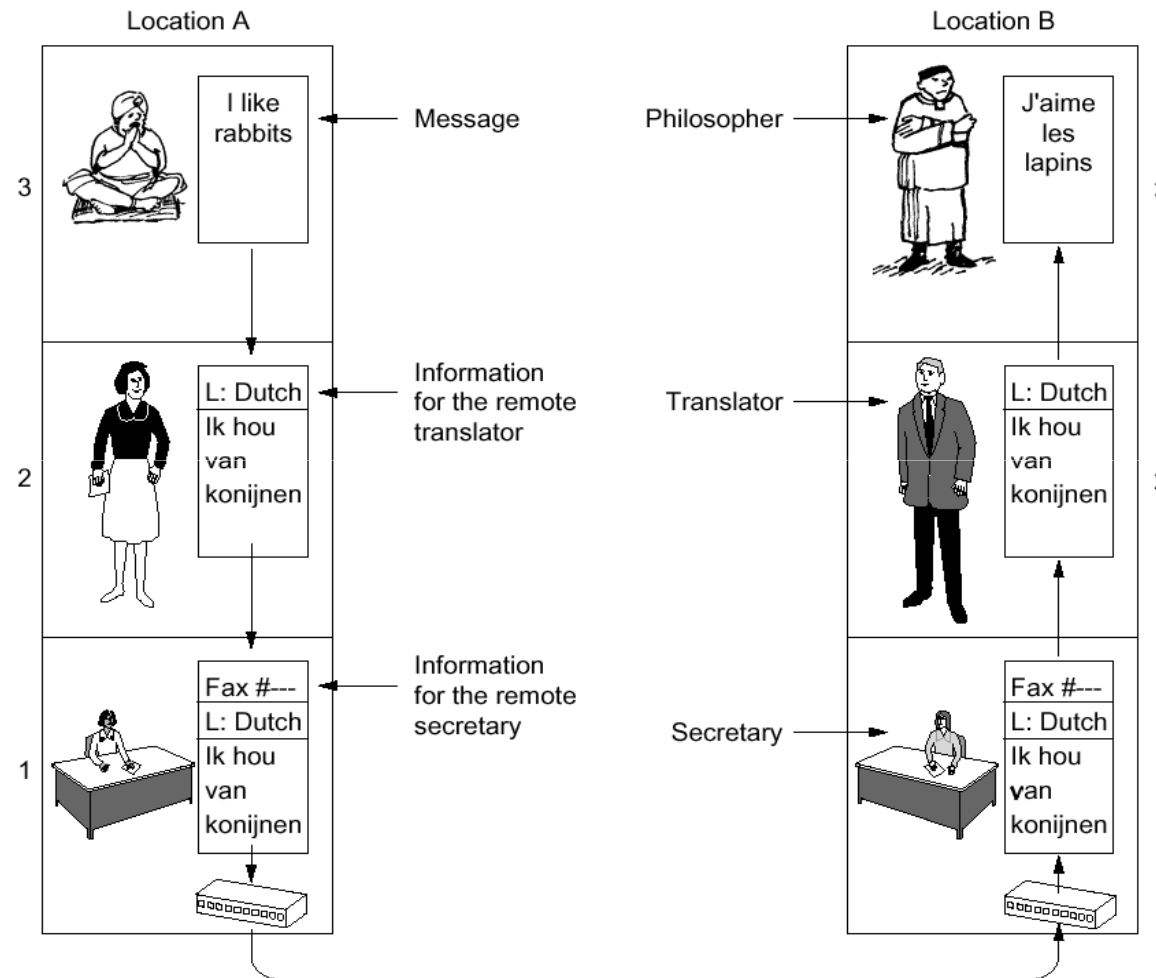
- Freie Meinungsäußerung vs.
- Internetkriminalität (von Copyright bis Kindesmissbrauch)

“Struktur” des Internet

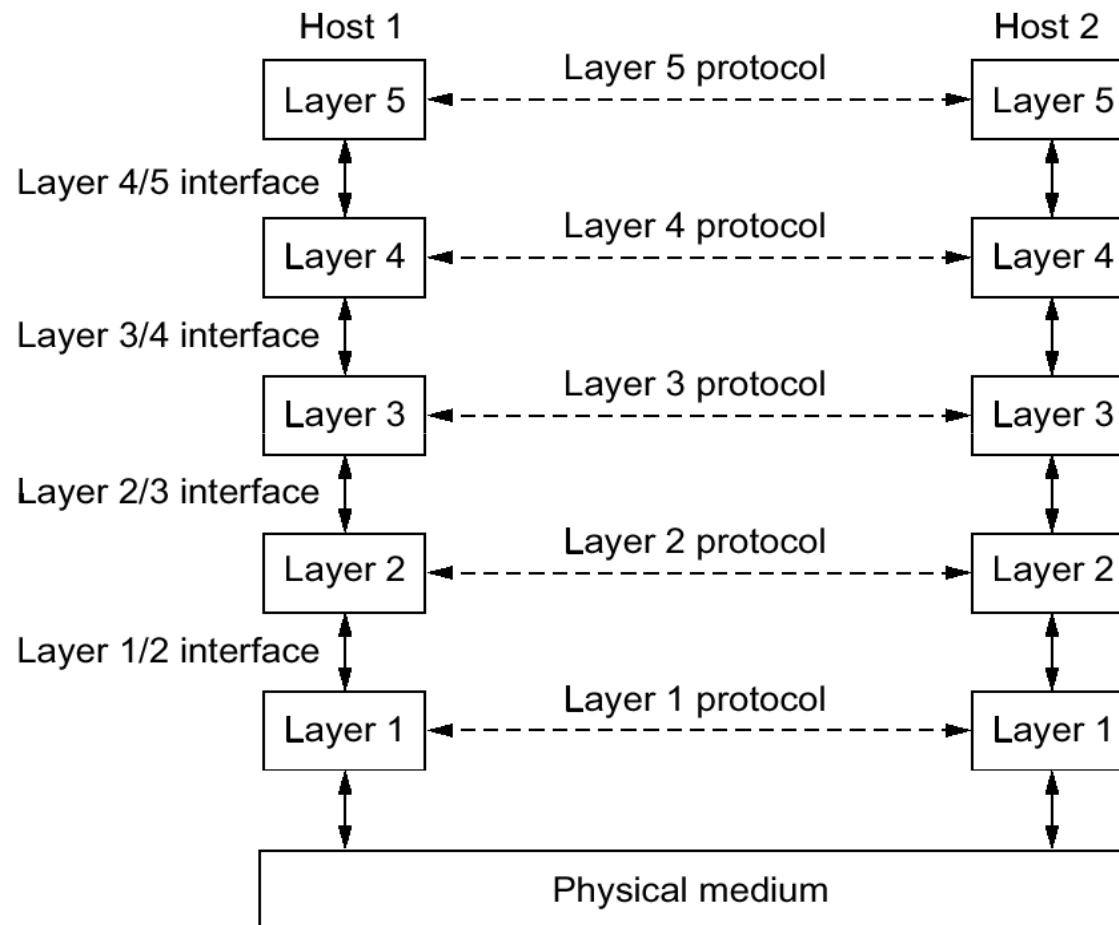
- ◆ Besteht aus zusammengeführten Netzen unterschiedlicher Organisationen
 - ◆ Sog. “Autonomen Systeme” (AS)
- ◆ IP, das “Internet Protocol” hält alles zusammen



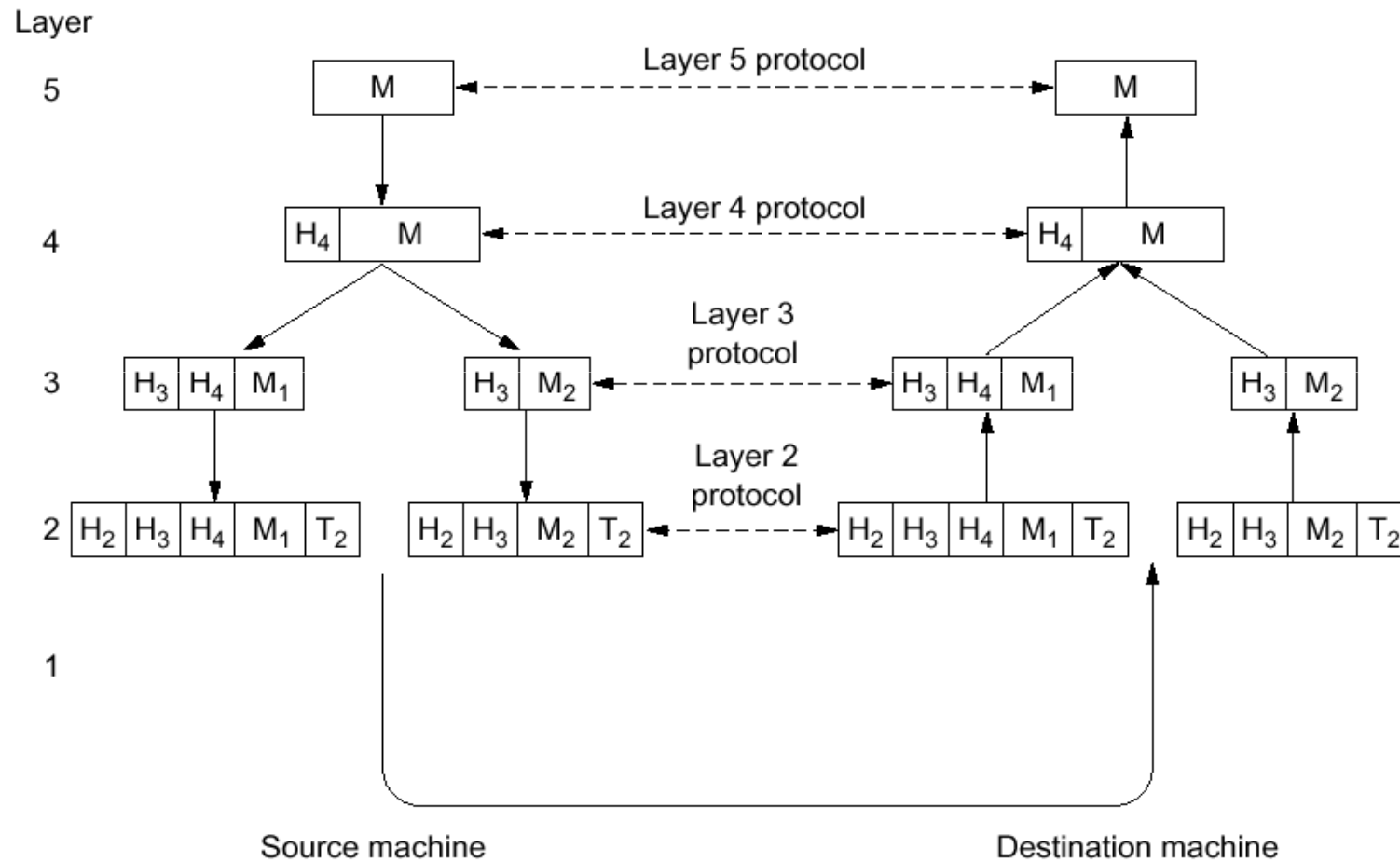
Beispiel: Philosoph, Übersetzer, Sekretär



Schichten, Protokolle und Interfaces



Kommunikation zwischen den verschiedenen Schichten



Grundideen der ISO OSI-Schichtenaufteilung

ISO OSI := ISO Open System Interconnection

Eine Schicht

- stellt eine neue Abstraktionsebene dar.
- sollte genau definierte Funktionen erfüllen
- sollte auch im Hinblick auf bestehende intern. Protokolle festgelegt werden
- sollte so definiert sein, dass an ihren Interfaces nur minimalen Informationsfluss nötig ist

Das gesamte Modell sollte

- so viele Schichten haben wie nötig, um unterschiedliche Funktionen auch in verschiedene Schichten zu separieren
- so wenige Schichten haben wie möglich um Unübersichtlichkeit zu vermeiden

Aufgaben der 7 Schichten des ISO OSI-Modells

1-3

Bitübertragungsschicht	
Schicht 1 Physical	Maßnahmen und Verfahren zur Übertragung von Bits
	Die Bitübertragungsschicht definiert die elektrische, mechanische und funktionale Schnittstelle zum Übertragungsmedium. Die Protokolle dieser Schicht unterscheiden sich nur nach dem eingesetzten Übertragungsmedium und -verfahren. Das Übertragungsmedium ist jedoch kein Bestandteil der Schicht 1.
Sicherungsschicht	
Schicht 2 Data Link	Logische Verbindungen mit Datenpaketen und elementare Fehlererkennungsmechanismen
	Die Sicherungsschicht sorgt für eine zuverlässige und funktionierende Verbindung zwischen Endgerät und Übertragungsmedium. Zur Vermeidung von Übertragungsfehlern und Datenverlust enthält diese Schicht Funktionen zur Fehlererkennung, Fehlerbehebung und Datenflusskontrolle. Auf dieser Schicht findet auch die physikalische Adressierung von Datenpaketen statt.
Vermittlungsschicht	
Schicht 3 Network	Routing und Datenflusskontrolle
	Die Vermittlungsschicht steuert die zeitliche und logische getrennte Kommunikation zwischen den Endgeräten, unabhängig vom Übertragungsmedium und -topologie. Auf dieser Schicht erfolgt erstmals die logische Adressierung der Endgeräte. Die Adressierung ist eng mit dem Routing (Wegfindung vom Sender zum Empfänger) verbunden.

Quelle: <http://www.elektronik-kompodium.de>

Aufgaben der 7 Schichten des ISO OSI-Modells

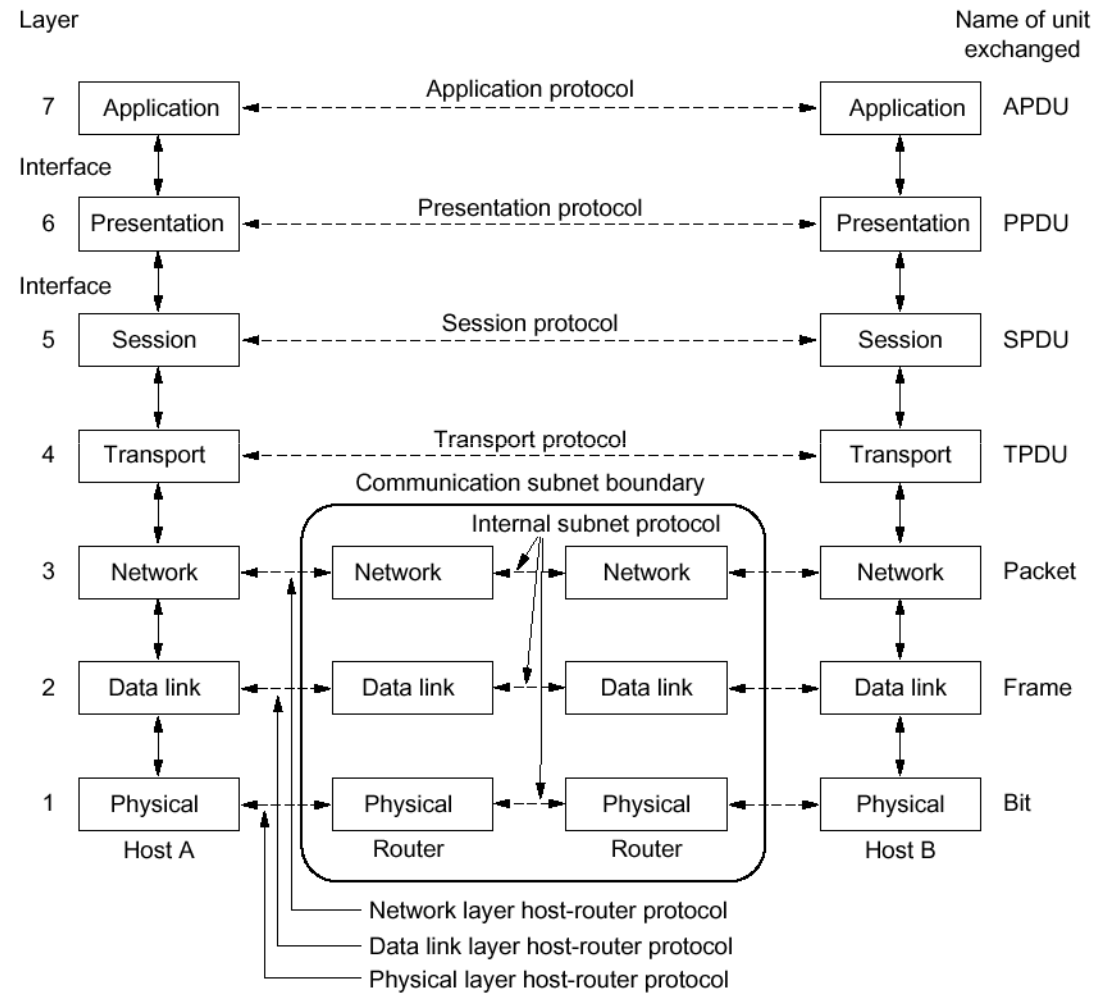
4-7

Transportschicht	
Schicht 4 Transport	Logische Ende-zu-Ende-Verbindungen
	Die Transportschicht ist das Bindeglied zwischen den transportorientierten und anwendungsorientierten Schichten. Hier werden die Datenpakete einer Anwendung zugeordnet.
Kommunikationsschicht	
Schicht 5 Session	Prozeß-zu-Prozeß-Verbindungen
	Die Kommunikationsschicht organisiert die Verbindungen zwischen den Endsystemen. Dazu sind Steuerungs- und Kontrollmechanismen für die Verbindung und dem Datenaustausch implementiert.
Darstellungsschicht	
Schicht 6 Presentation	Ausgabe von Daten in Standardformate
	Die Darstellungsschicht wandelt die Daten in verschiedene Codecs und Formate. Hier werden die Daten zu oder von der Anwendungsschicht in ein geeignetes Format umgewandelt.
Anwendungsschicht	
Schicht 7 Application	Dienste, Anwendungen und Netzmanagement
	Die Anwendungsschicht stellt Funktionen für die Anwendungen zur Verfügung. Diese Schicht stellt die Verbindung zu den unteren Schichten her. Auf dieser Ebene findet die Dateneingabe und -ausgabe statt.

Quelle: <http://www.elektronik-kompodium.de>

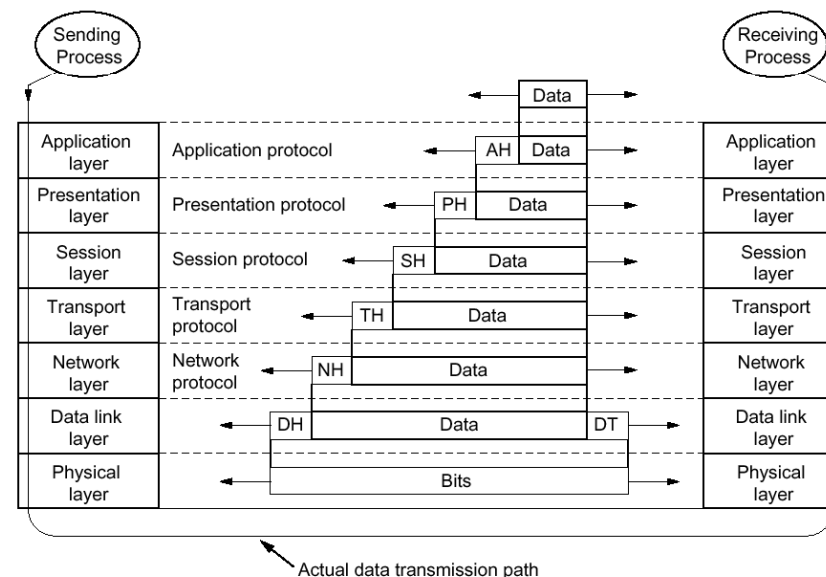
Das ISO OSI-Referenzmodell

Übersicht und Komponenten

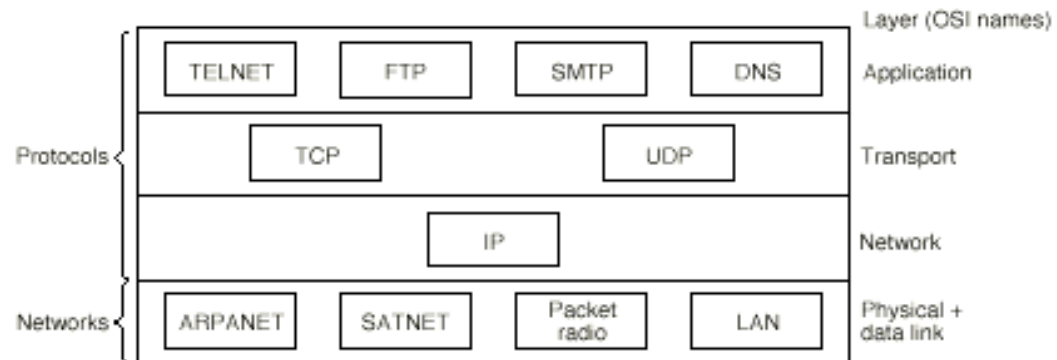
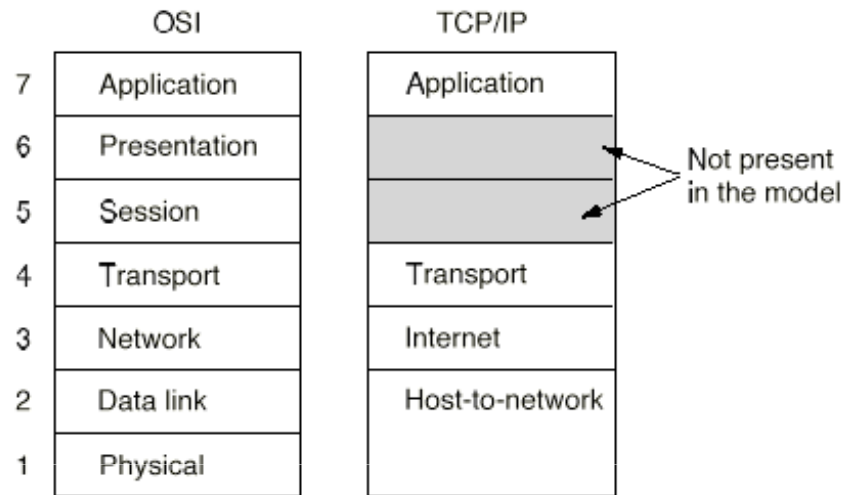


Datenübertragung im OSI-Referenzmodell

- ♦ Die Daten fließen in den Schichten 2-7 vertikal
 - von oben nach unten, bzw. wieder von unten nach oben
- ♦ Nur in Schicht 1 (physical) fließen Daten von einem Rechner zum anderen
- ♦ Jede Schicht betrachtet die Daten der darüber liegenden als zu transportierende Daten
- ♦ Jede Schicht kann eigene Daten hinzufügen
 - Vor den Daten (Header) oder
 - danach (Trailer)



Im Vergleich: das (einfachere) TCP/IP Referenzmodell



Diskussion des OSI-Modells

- ◆ Die ausdrückliche Trennung in die verschiedenen Konzepte ist zentral im OSI-Modell
 - Services
 - Interfaces
 - Protokolle

- ◆ Warum hatte es trotzdem nicht den Erfolg?
 - Schlechtes Timing
 - Schlechte Technik
 - Schlechte Implementierung
 - Schlechte Politik

Diskussion des TCP/IP-Modells

♦ Vorteile:

- Protokolle passen perfekt zum Model ;-)
- Gute, weniger komplizierte und freie Implementierung in BSD UNIX

♦ Nachteile:

- Keine klare Trennung zwischen den Konzepten von Services, Interfaces und Protokollen
- Nicht allgemein
- Host-to-Network Schicht ist eigentlich nur ein Interface
- Physikalische und Sicherungsschicht nicht abgedeckt
- Einige Protokolle der Anwendungsschicht sind eher ein Hack (Telnet)

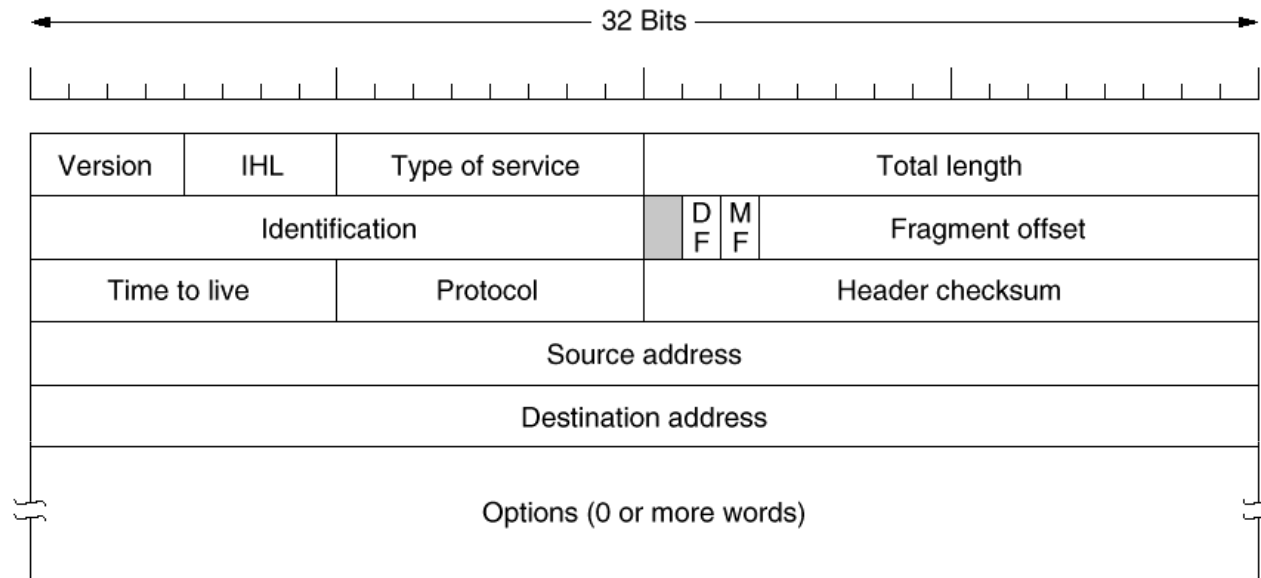
♦ Zusammenfassend:

- OSI: Gutes Modell (außer Session und Presentation Layer), aber die Protokolle wurden kaum genutzt
- TCP/IP: Praktisch kein eigenständiges Modell, aber Protokolle, die weithin genutzt werden

Funktionalität von IP

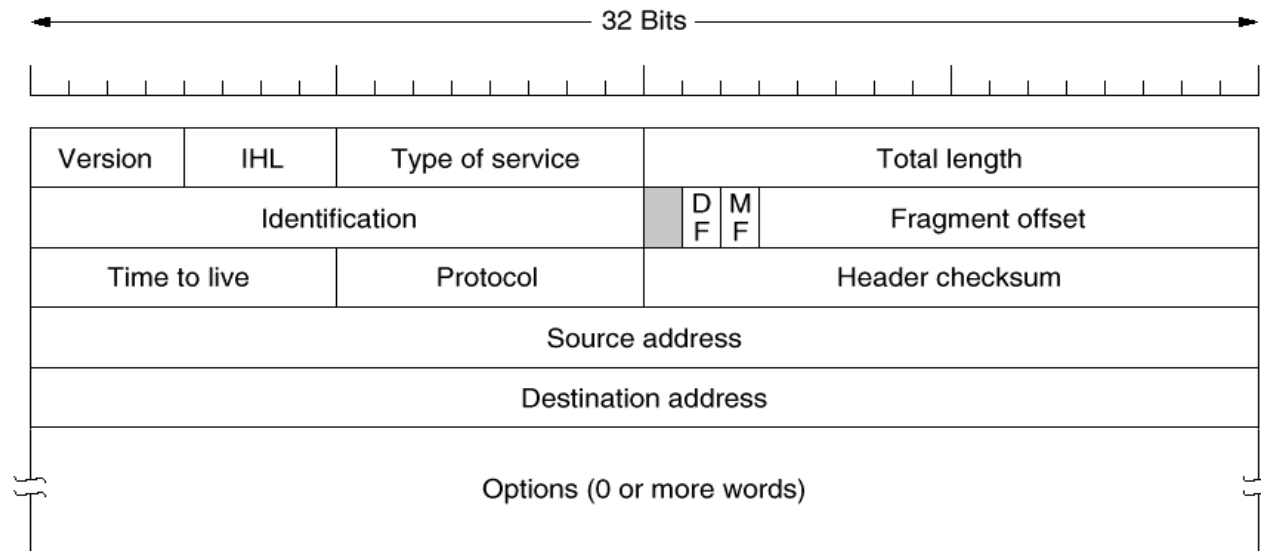
- ◆ **Best-Effort Dienst zum Transport von Datagrammen von der Quelle zum Ziel**
 - ◆ **Best-Effort: kann klappen, muss aber nicht**
 - ◆ **Datagramme: einzelne Pakete, keine ganzen Datenströme**
 - ◆ **Quelle zum Ziel: von Rechner zu Rechner, nicht von Programm zu Programm**
- ◆ **Unabhängig davon,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

IPv4 Header (1)



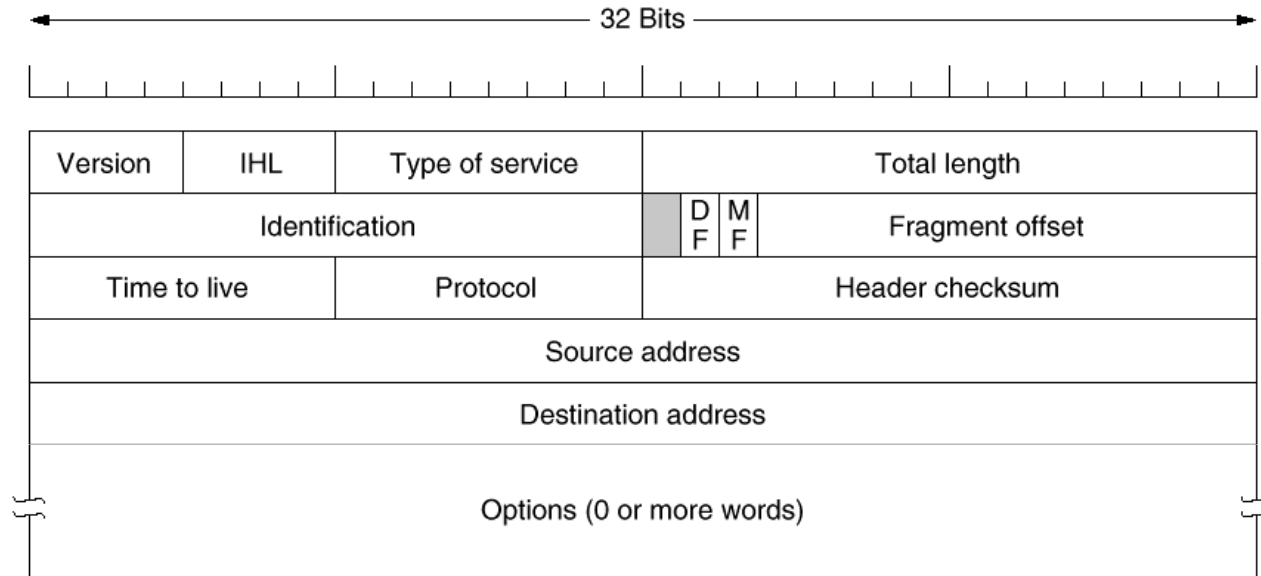
- ◆ **Version**
 - Zz. v4, ermöglicht gemischten Betrieb mit neueren Versionen (IPv6!)
- ◆ **IHL**
 - Header Length (Einheiten von 32 Bits, min 5, max 15)

IPv4 Header (2)



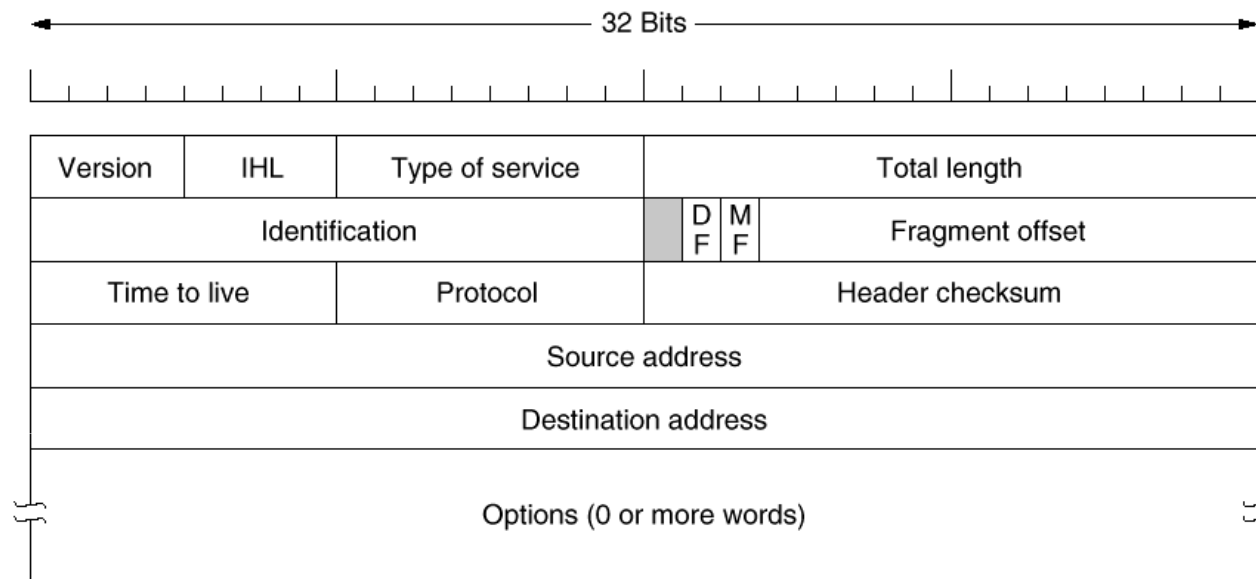
- ◆ **Type of service (usually ignored)**
 - 3 bits *precedence (priority)*, *normal to network control*
 - 3 flags (Delay, Throughput, Reliability)
- ◆ **Total length**
 - Length of header and data in bytes (max. 65535)

IPv4 Header (3)



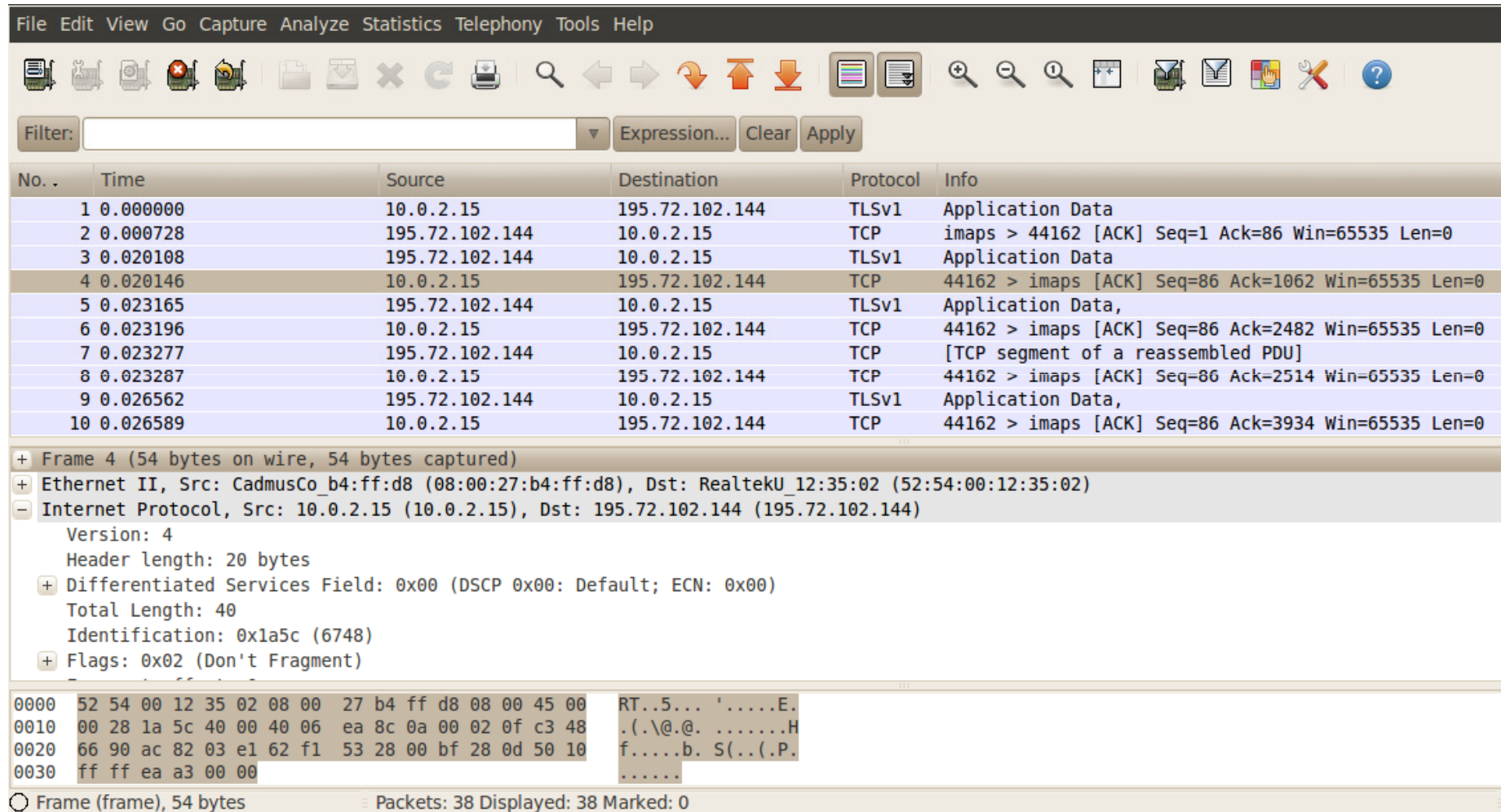
- ◆ Identification - identifies parts of a fragment
- ◆ DF - “Don’t Fragment”
- ◆ MF - “More Fragments”
- ◆ Fragment Offset (in 8 Byte Einheiten)

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:

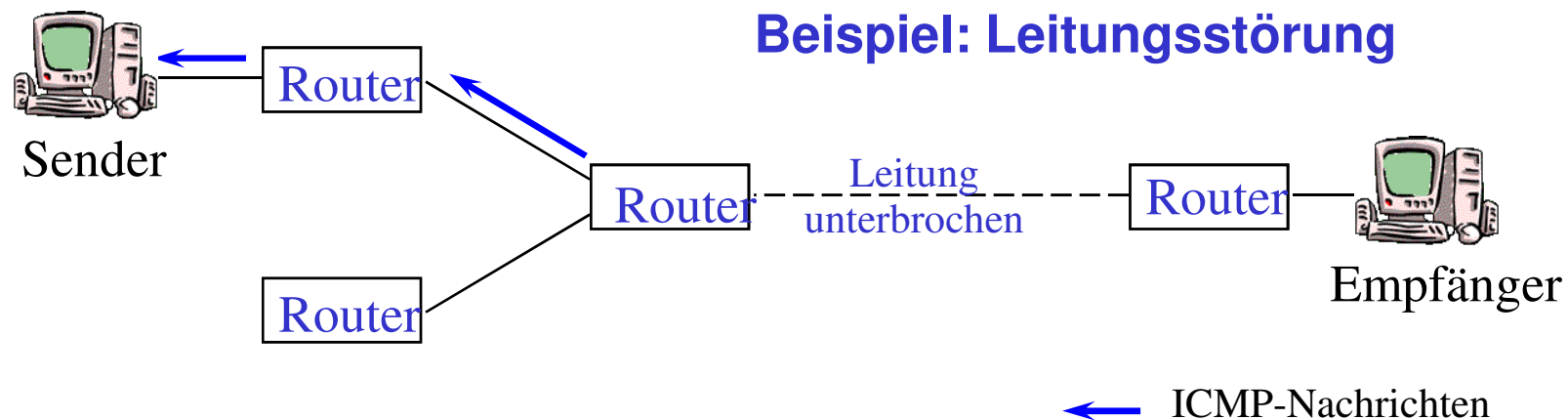
```
0000 52 54 00 12 35 02 08 00 27 b4 ff d8 08 00 45 00 RT..5... '....E.
0010 00 28 1a 5c 40 00 40 06 ea 8c 0a 00 02 0f c3 48 .(. \@.@. ....H
0020 66 90 ac 82 03 e1 62 f1 53 28 00 bf 28 0d 50 10 f....b. S(..(P.
0030 ff ff ea a3 00 00 .....

```

At the bottom, the status bar 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.



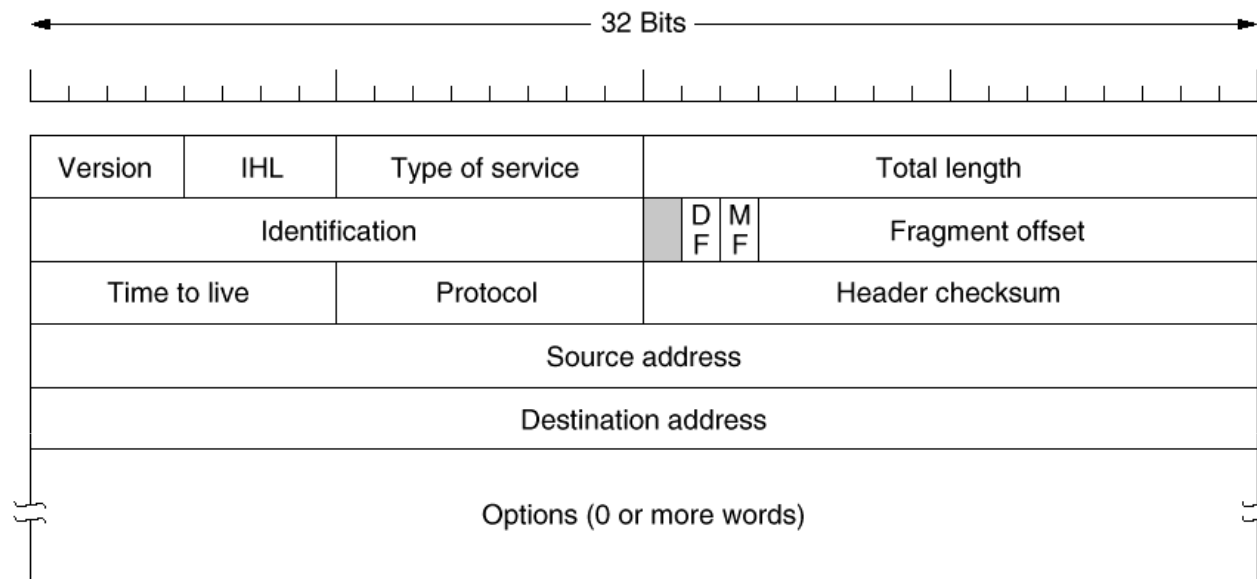
- ◆ ICMP unterstützt den Austausch von Fehlermeldungen, Statusanfragen und Zustandsinformation.
-

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

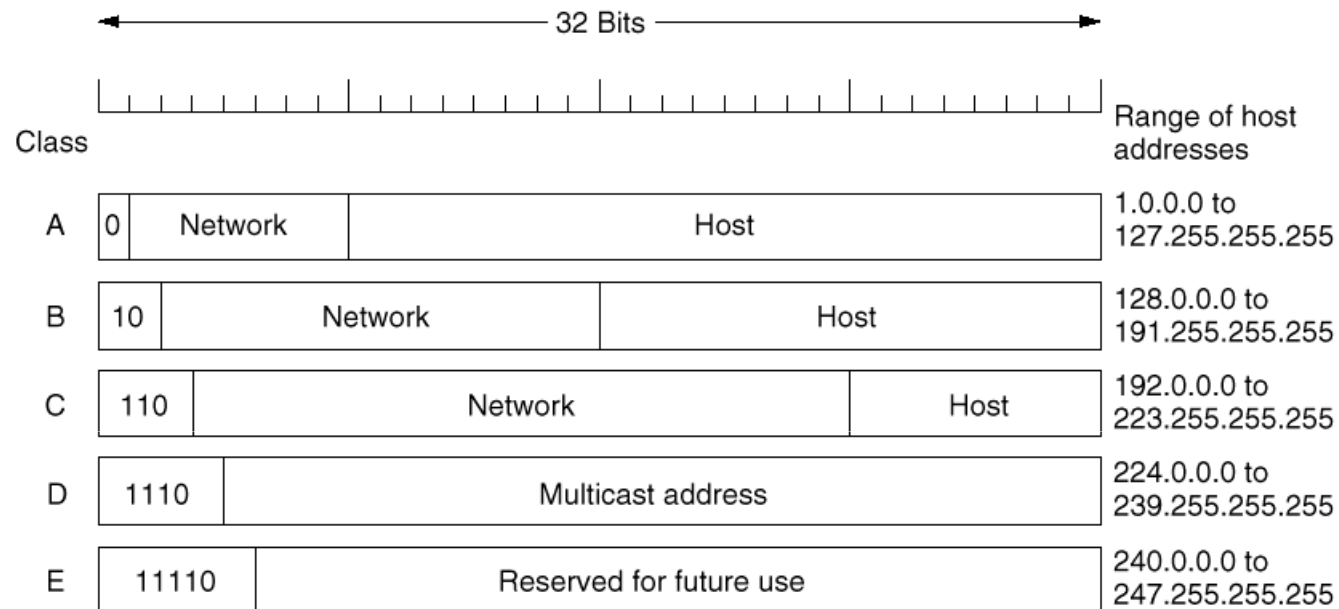
The IPv4 Header Format (5)



◆ 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 klassische Adressformate



- ◆ 126 Class A Netzwerke mit 16 Millionen Hosts
- ◆ 16382 Class B Netzwerke mit 64k Hosts
- ◆ 2 Millionen Class C Netzwerke mit 254 Hosts

Spezielle IPv4 Adressen

0 0																														This host	
0 0		...		0 0		Host																								A host on this network	
1 1																														Broadcast on the local network	
Network								1 1 1 1				...		1 1 1 1				Broadcast on a distant network													
127				(Anything)																										Loopback	

IPv4 Adressen in privaten Netzen

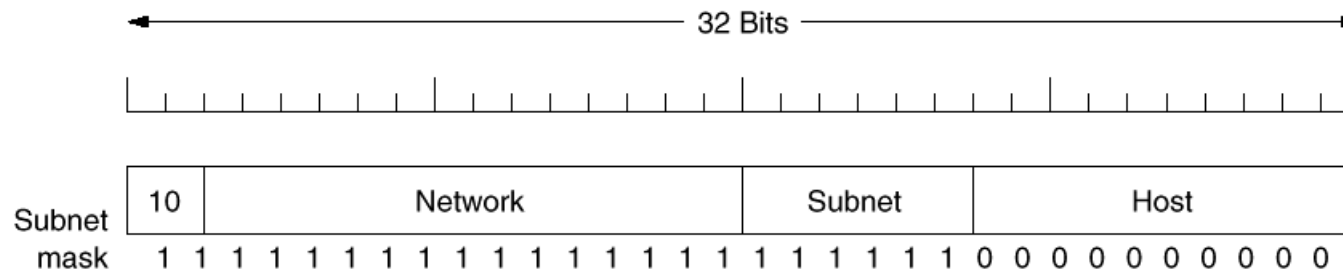
- ◆ **Geregelt im RFC 1918 (Address Allocation for Private Internets)**
 - Jeder kann aus diesen Bereichen den Adressbereich für sein eigenes privates Netz auswählen

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 65.000 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

Subnets

- ◆ Führt eine neue Hierarchie-Ebene unterhalb der IP Netzwerke (Class A, B, or C) ein
- ◆ Beispiel 1: Class B Netzwerk mit 6 Bit Subnet Nummer (64 Subnets mit je 1022 Hosts):



Address: 130.50.15.6

Net: 130.50, Subnet 3, Host 774

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)
- Vergabe der Netzwerke in Größen von 2^n
 - Nutzt Sub-Netzwerke und
 - *Super-Netzwerke* (Zusammenfassung von Class C-Netzwerken)
- Generell wird bei Netzen immer die Länge der Adresse mit angegeben
 - Als Netzmaske:
 - z.B. 255.255.254.0
 - Oder als Anzahl der Bits mit “/”
 - z.B. 192.85.16.0/23

Adressrechnung (1) – Netzwerkadresse berechnen

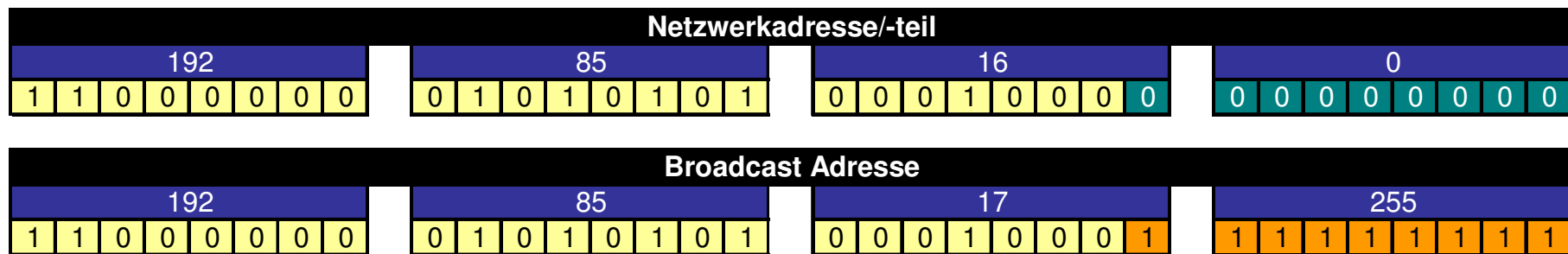
- ♦ IP-Adresse 192.85.17.2 und Subnetmask 255.255.254.0 (CIDR /23)

128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
IP Adresse																															
192								85								17								2							
1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
Subnet Mask																															
255								255								254								0							
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Netzwerkadresse/-teil																															
192								85								16								0							
1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

- ♦ Durch log. UND ergibt sich aus der gesamten Adresse die Netzadresse:
 - In diesem Beispiel: 192.85.16.0

Adressrechnung (2) – Broadcast-Adresse

- ♦ Füllt man den Hostteil des ermittelten Netzwerks mit Einsen, erhält man die Broadcast Adresse
 - im Beispiel: 192.85.17.255



- ♦ Die erste (Hostteil alles 0) und letzte (alles 1) Adresse eines Netzwerks können nicht als IP Adresse für einen Host verwendet werden
 - Netzwerkadresse (erste)
 - Broadcastadresse (letzte)

Adressrechnung (3) - Subnetting

Klasse-B : 255.255.0.0 oder / 16 (Schreibweise !!)

Klasse-B mit 64 Teilnetzen : 255.255.252.0 oder / 22

Teilnetz 0 : 1 0 0 0 █ 0 0 1 0 █ 0 0 1 1 █ 0 0 1 0 █ 0 0 0 0 █ 0 0 0 0 █ 0 0 0 0 █ 0 0 0 1 130.50.0.0

Teilnetz 1: 1 0 0 0 █ 0 0 1 0 █ 0 0 1 1 █ 0 0 1 0 █ 0 0 0 0 █ 0 1 0 0 █ 0 0 0 0 █ 0 0 0 1 130.50.4.0

Teilnetz 2 : 1 0 0 0 █ 0 0 1 0 █ 0 0 1 1 █ 0 0 1 0 █ 0 0 0 0 █ 1 0 0 0 █ 0 0 0 0 █ 0 0 0 1 130.50.8.0

... weitere 61 Teilnetze ...

1 0 0 0 █ 0 0 1 0 █ 0 0 1 1 █ 0 0 1 0 █ 0 0 0 0 █ 1 1 1 1 █ 0 0 0 0 █ 0 1 1 0 130.50.15.6

1 1 1 1 █ 1 1 1 1 █ 1 1 1 1 █ 1 1 1 1 █ 1 1 1 1 █ 1 1 0 0 █ 0 0 0 0 █ 0 0 0 0 255.255.252.0

Routing : logisches UND mit der Maske 255.255.252.0 ergibt das Teilnetz 130.50.12.0

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

Bekannte Ports (Auswahl)

[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

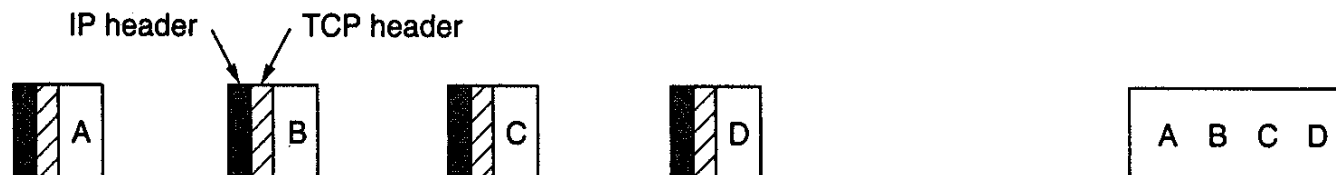
[https](#) 443/tcp https

[irc](#) 6665-6669/tcp chatten

(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

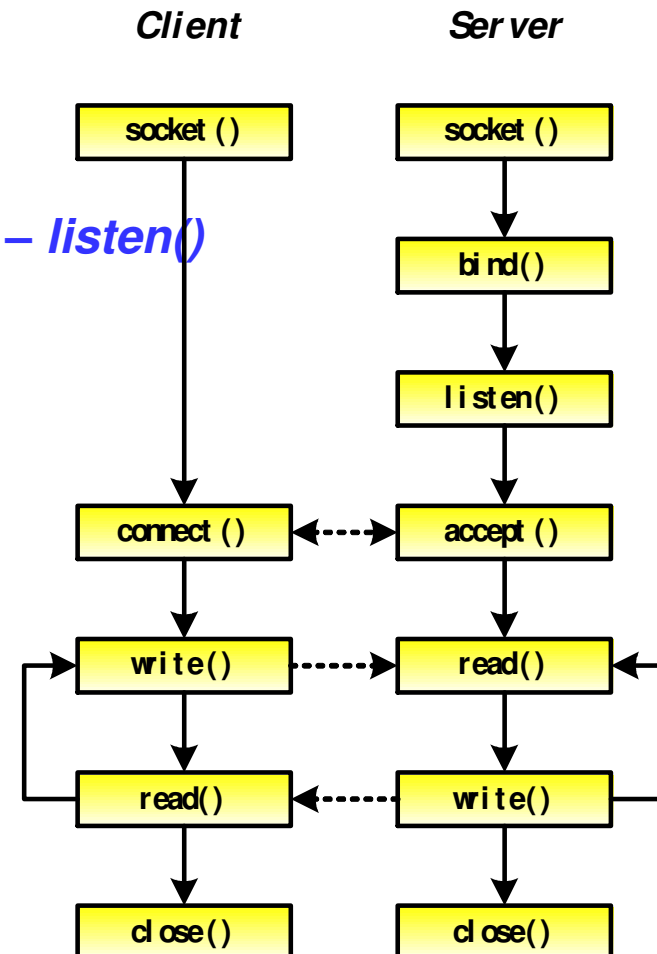
Client/Server

◆ 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()*



Beispiel: Einfacher Server

```
SOCKET listen_socket = socket(AF_INET, SOCK_STREAM, 0);

struct sockaddr_in local;
local.sin_family = AF_INET;
local.sin_addr.s_addr = INADDR_ANY;
local.sin_port = htons(502);

bind(listen_socket, (struct sockaddr*)&local, sizeof(local));

listen(listen_socket, 5);

while (1)
{
    struct sockaddr addr;
    int addrlen = sizeof(addr);
    SOCKET sock = accept(listen_socket, &addr, &addrlen);

    double d;
    int r = recv(sock, (char*)&d, sizeof(d), 0);
    while (1) {
        if (r != sizeof(d))
        {
            close(sock);
            break;
        }

        r = send(sock, (char*)&d, sizeof(d), 0);
    }
}
```

Beispiel: Client

```
SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);

struct sockaddr_in to;
memset(&to, 0, sizeof(to));
to.sin_family = AF_INET;
to.sin_addr.S_un.S_addr = inet_addr("192.168.1.11");
to.sin_port = htons(502);

int ret = connect(sock, (struct sockaddr*)&to, sizeof(to));

BOOL data = TRUE;
int size = sizeof(data);
ret = setsockopt
(
    sock,    // int socket,
    IPPROTO_TCP, // int level,
    TCP_NODELAY, // int optname,
    (char*)&data, // char FAR* optval,
    size      // int FAR* optlen
);

double dr = 0.0;

int r = send(sock, (char*)&d, sizeof(d), 0);

r = recv(sock, (char*)&dr, sizeof(dr), 0);

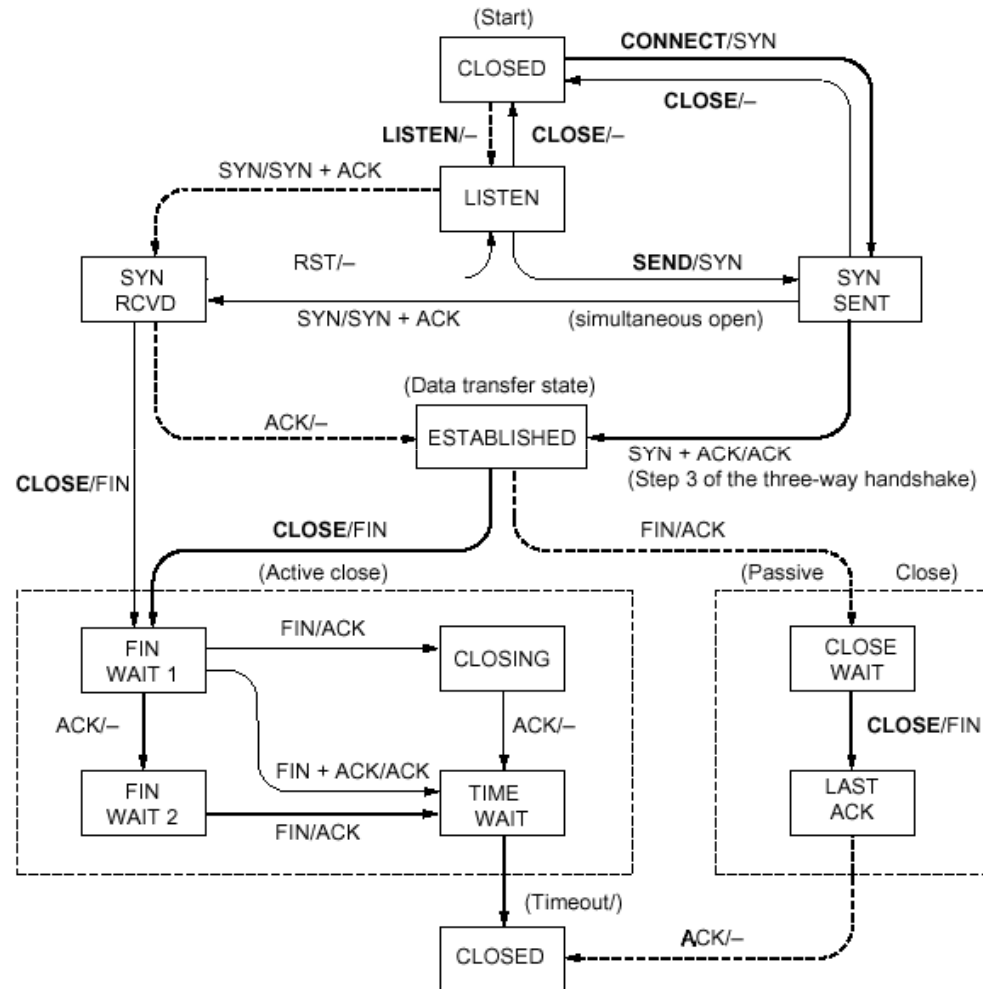
close(sock);
```

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

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

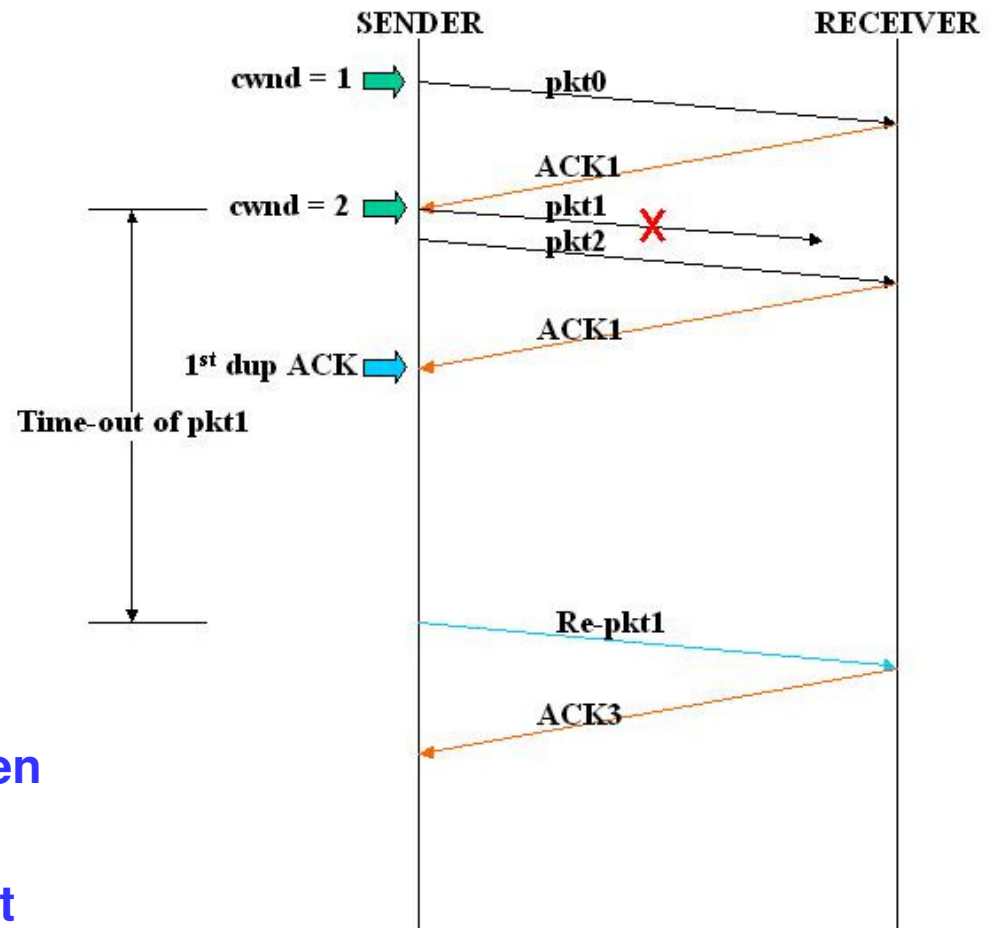
TCP Zustandsübergänge

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

Fehlerbehandlung bei TCP

Garantie eines zuverlässigen Datenstroms

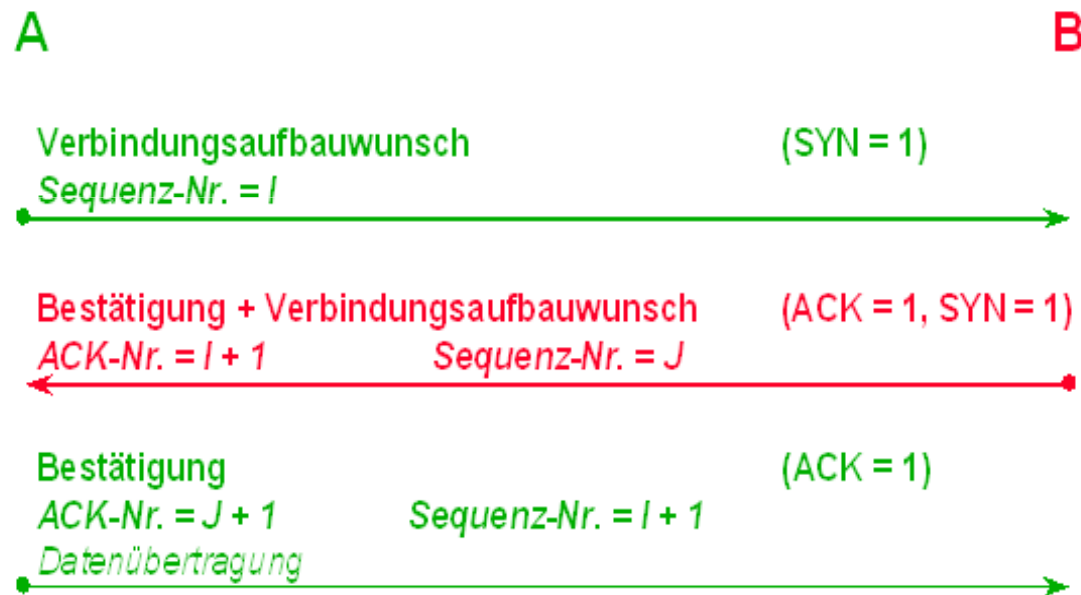
- ◆ Datenbytes haben eine Sequenznummer
 - **SEQ: Nummer der gesendeten Daten**
- ◆ Empfangene Bytes werden quitiert
 - **ACK: Nummer des ersten noch nicht empfangenen Bytes**
- ◆ Empfängt der Sender nicht innerhalb eines Timeouts ein ACK, wiederholt er die Daten
 - Funktioniert sowohl, wenn Daten als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



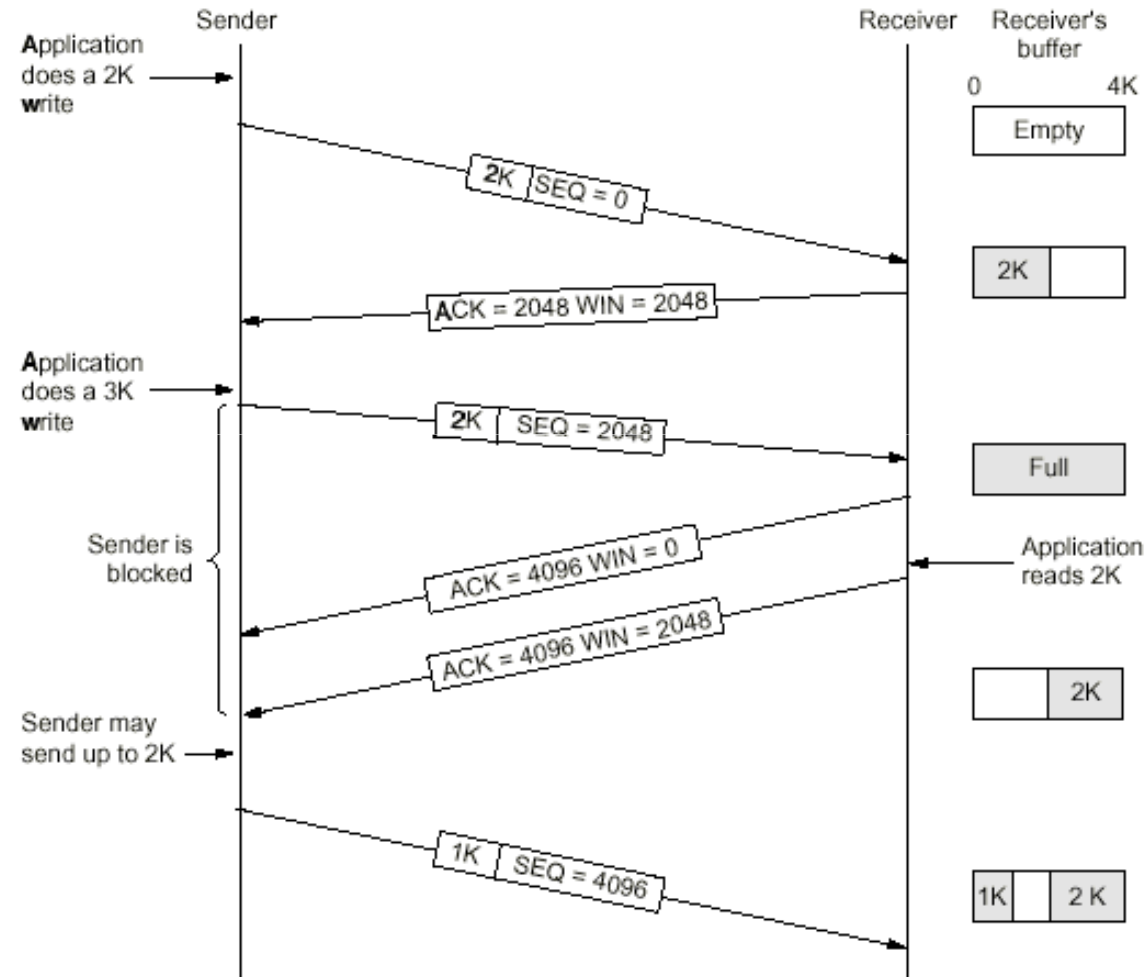
Transmission Control Protocol

Verbindungsaufbau

TCP - Verbindungsaufbau (Three-Way-Handshake)



Window management (Transmission Policy) bei TCP



Der UDP Datagramm-Header

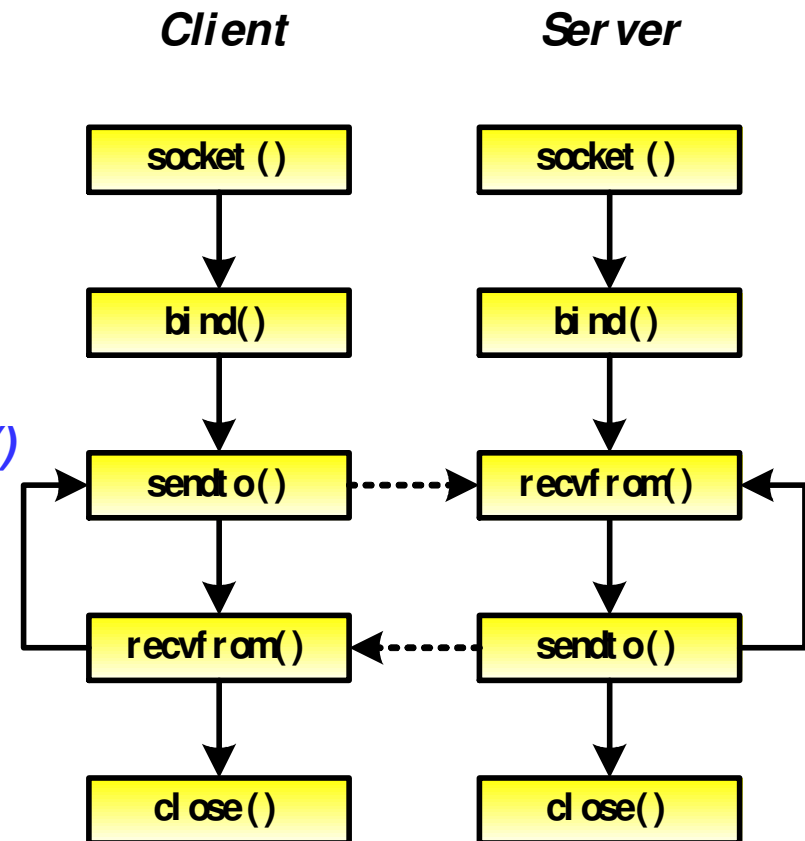
0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Quell-Portnummer															Ziel-Portnummer																
Länge															Prüfsumme																
Daten																															

- ♦ 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 Internet ist ein weltweiter Zusammenschluss verschiedener Netze**
 - WWW ist nur ein Dienst auf dem Internet
 - ◆ **Protokolle, Schichten und Interfaces sind in einem Referenzmodell definiert**
 - Das 7-Schichten-Modell ist DAS Modell der Rechnernetze
 - Auch wenn TCP/IP nur einen Teil davon implementiert
 - ◆ **Das IP-Protokoll hält das Internet zusammen**
 - Bringt Pakete vom sendenden Rechner zum empfangenden
 - IP-Adressen sind strukturiert nach Netz-, (Subnetz) und Hostteil
 - IPv6 wird irgendwann das bisherige IPv4 ablösen
 - ◆ **TCP und UDP sind die Protokolle, die Prozesse/Programme nutzen**
 - TCP für Datenströme, UDP für einzelne Datagramme
-