10.Vorlesung Grundlagen der Informatik

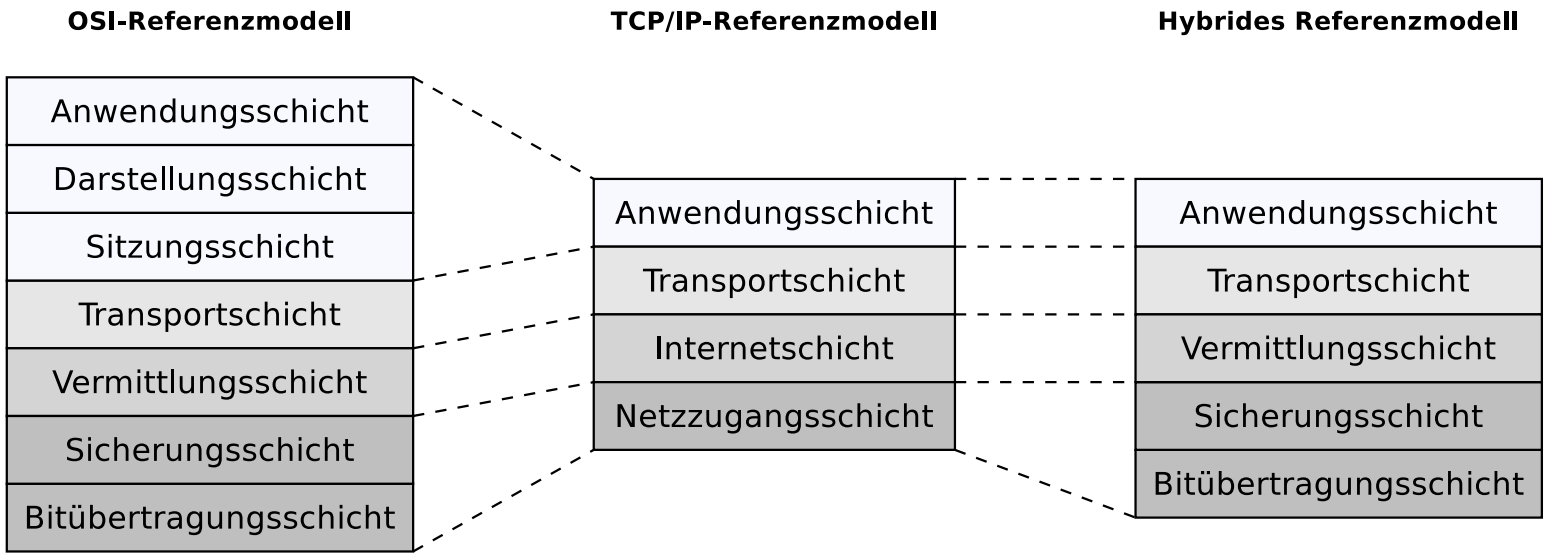
Dr. Christian Baun

Hochschule Darmstadt
Fachbereich Informatik
christian.baun@h-da.de

15.12.2011

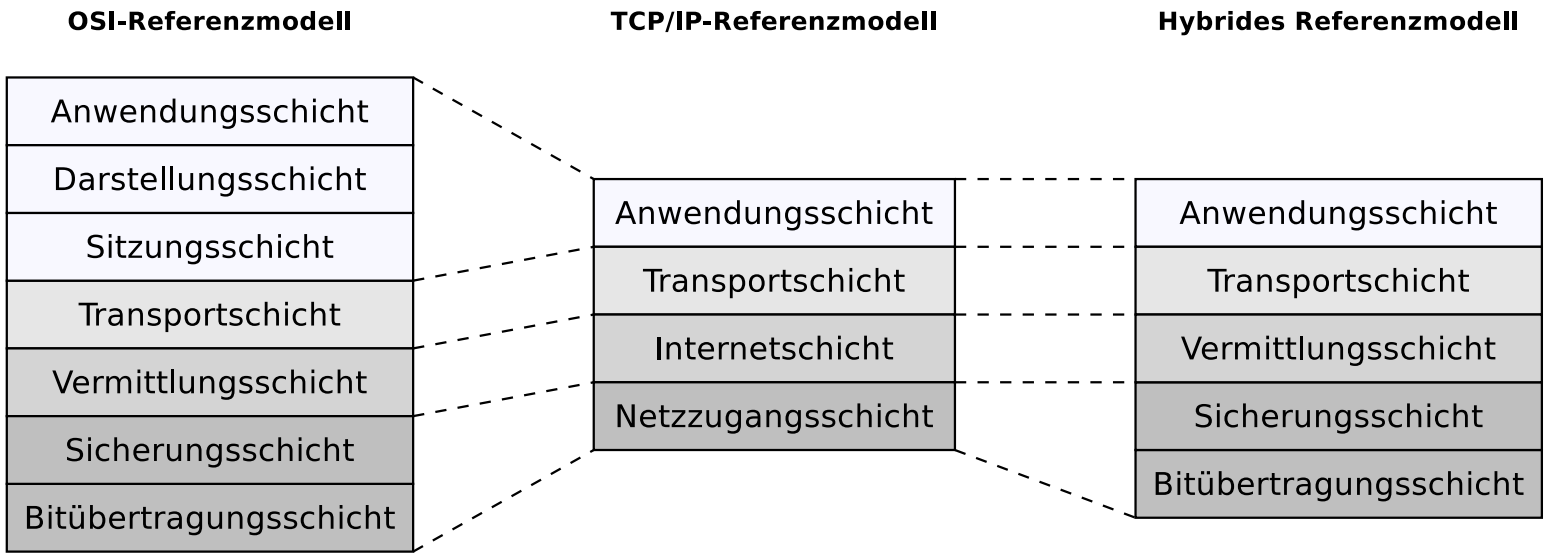
Wiederholung vom letzten Mal

- Bitübertragungsschicht
  - Kodierung von Daten
- Sicherungsschicht
  - Adressierung (MAC-Adressen)
  - Fehlererkennung (Zweidimensionale Parität, Zyklische Redundanzprüfung)
- Vermittlungsschicht
  - Link-State-Routing-Protokoll (Dijkstra-Algorithmus)



Heute

- Vermittlungsschicht
  - Adressierung (IP-Adressen)
- Transportschicht
  - Ports und Portnummern, Sockets
  - Transportprotokolle (UDP, TCP)
- Sitzungsschicht
- Darstellungsschicht
- Anwendungsschicht

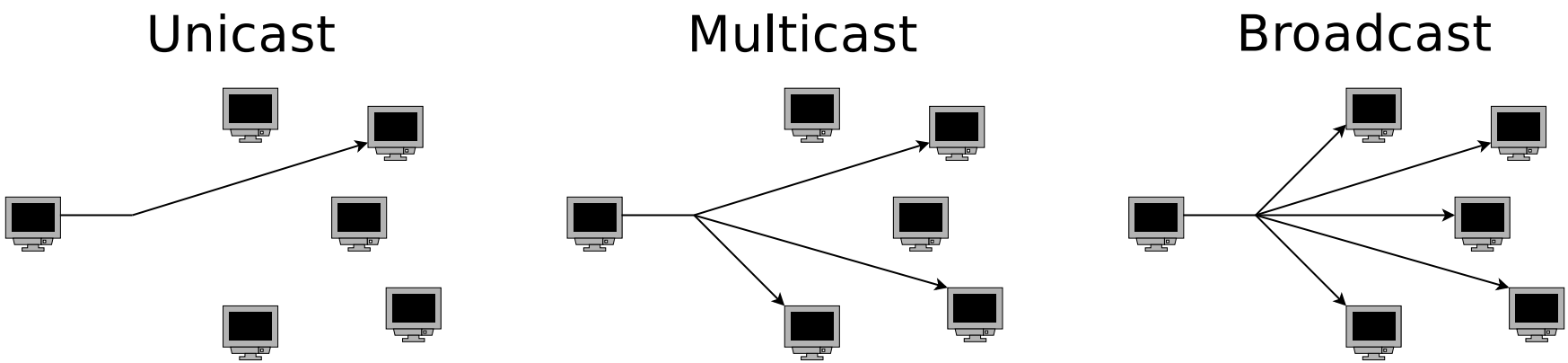


Adressierung und Adressarten

- **Adressraum** = Menge aller gültigen Netzadressen
- Verwendung der Adressen:
  - Bei verbindungsloser Kommunikation
    - Jede Nachricht enthält die Adressen von Sender und Empfänger
  - Bei verbindungsorientierter Kommunikation
    - Sender- und Empfängeradressen sind nur für den Verbindungsaufbau nötig
- **Physische Adressierung** (MAC-Adressen)
  - Direkte Angabe der Teilnehmernummer und Anschlussleitung
  - In großen Netzen wenig sinnvoll, da schlecht zu warten
  - Eingabefehler werden nicht als solche erkannt
- **Logische Adressierung** (IP-Adressen)
  - Adressen sind besser für Menschen lesbar

Adressen des Internet Protocol

- Zu jedem IP-Paket gehört eine Empfängeradresse ($\Longrightarrow$ IP-Adresse), die angibt, wohin das Paket geschickt werden soll
  - IP-Adressen sind Adressen in Computernetzen, die auf dem Internet Protocol (IP) basieren
- Eine IP-Adresse kann einen einzelnen Empfänger (**Unicast**) oder eine Gruppe von Empfängern bezeichnen (**Multicast** oder **Broadcast**)

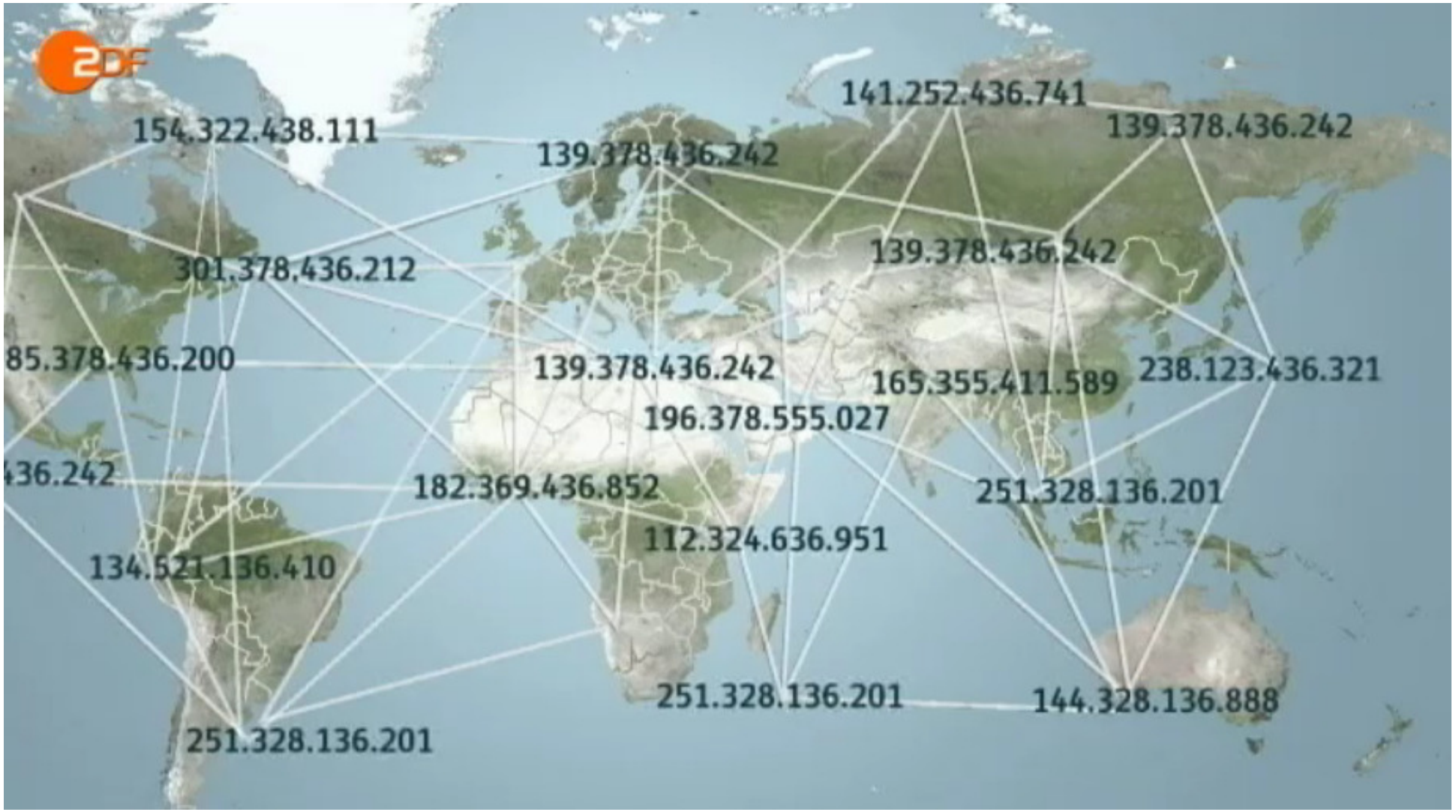


- Einem Computer können auch mehrere IP-Adressen zugeordnet sein

Aufbau von IPv4-Adressen

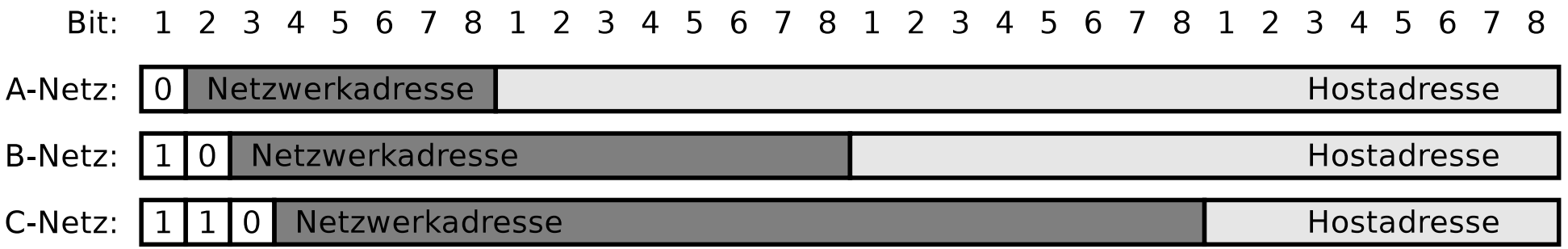
- IPv4-Adressen bestehen aus 32 Bit (4 Byte)
  - Daher können $2^{32} = 4.294.967.296$ Adressen dargestellt werden
- Üblich ist die Darstellung in der sogenannten *Dotted decimal notation*
  - Die 4 Oktetts werden als vier durch Punkte voneinander getrennte ganze Zahlen in Dezimaldarstellung im Bereich von 0 bis 255 geschrieben
    Beispiel: 141.52.166.25

ZDF heute am 3.2.2011 – Was stimmt hier nicht?



Netzklassen, Netzwerkteil und Geräteteil

- Ursprünglich wurden IPv4-Adressen in Klassen von A bis C eingeteilt
  - Es existierten auch die Klassen D und E für spezielle Aufgaben
- 1993: Einführung von klassenlosem Routing CIDR
  - CIDR = Classless Interdomain Routing
  - Seit CIDR spielt die Netzklasse einer IPv4-Adresse keine Rolle mehr
- Die 32 Bit einer IPv4-Adresse bestehen aus 2 Feldern:
  - **Netzwerkadresse** (Network Identifier) bzw. Netzwerk-ID
  - **Hostadresse** (Host Identifier) bzw. Host-ID
  - Klasse A: 7 Bit für Netzwerkadresse und 24 Bit für Hostadresse
  - Klasse B: 14 Bit für Netzwerkadresse und 16 Bit für Hostadresse
  - Klasse C: 21 Bit für Netzwerkadresse und 8 Bit für Hostadresse



Netzklassen (1)

- Adressräume der Netzklassen

Klasse	Adressbereich	Präfix	Netzlänge (ohne Präfix)	Hostlänge
A	0.0.0.0 - 127.255.255.255	0	7 Bit	24 Bit
B	128.0.0.0 - 191.255.255.255	10	14 Bit	16 Bit
C	192.0.0.0 - 223.255.255.255	110	21 Bit	8 Bit
D	224.0.0.0 - 239.255.255.255	1110		
E	240.0.0.0 - 255.255.255.255	1111		

- `Klasse A`: $2^7 = 128$ Netze mit max. $2^{24} = 16.777.216$ Adressen
- `Klasse B`: $2^{14} = 16.384$ Netze mit max. $2^{16} = 65.536$ Adressen
- `Klasse C`: $2^{21} = 2.097.152$ Netze mit max. $2^8 = 256$ Adressen
- `Klasse D`: Multicast-Adressen (z.B. für IPTV)
- `Klasse E`: Reservierte Adressen (für zukünftige Zwecke)

Netzklassen (2)

- Praktisch relevant sind nur die Klassen A, B und C
- Ursprünglich war beabsichtigt, durch die Netzwerkadresse ein physisches Netzwerk eindeutig zu identifizieren
  - Dieses Vorgehen bringt eine Reihe von Nachteilen mit sich
  - An einer Hochschule mit mehreren internen Netzwerken müsste man für jedes Netzwerk mindestens ein Klasse C Netz (255 Adressen) reservieren
  - Unternehmen mit > 255 Netzwerkgeräten bräuchten Klasse B Netze

Subnetze

- **Nachteile der Netzklassen:**
  - Sie können nicht **dynamisch an Veränderungen angepasst** werden können und **verschwenden viele Adressen**
  - Ein Klasse C Netz mit 2 Geräten verschwendet 253 Adressen
  - Ein Klasse B Netz mit 256 Geräten verschwendet über 64.000 Adressen
  - Es gibt nur wenige Klasse A Netze
  - Bei Klasse C Netzen kann der Adressraum rasch knapp werden
    - Eine spätere Migration ist schwierig/lästig
- Einfache Möglichkeit IP-Adressen effizienter zu verwenden: **Teilnetze**, die meist **Subnetze** genannt werden
  - Man teilt verfügbare Knoten-Adressen auf mehrere Subnetze auf
  - Subnetze sollten räumlich nahe beieinander liegen, da sie von einer **Netzwerknummer** repräsentiert werden
    - Typisches Szenario: Unterschiedliche Abteilungen eines Unternehmens

Netzmaske

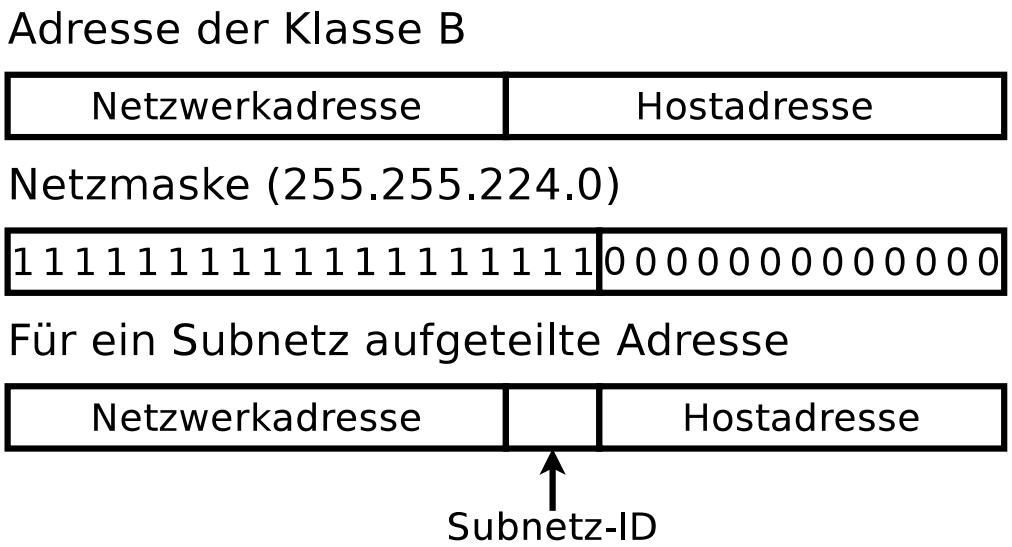
- Um Subnetzen zu bilden braucht man eine **(Sub-)Netzmaske**
  - Alle Knoten in einem Netzwerk bekommen eine Netzmaske zugewiesen
  - Die Netzmaske unterliegt der Kontrolle des Netzwerkverwalters
  - Die Netzmaske ist wie eine IPv4-Adresse eine 32 Bit-Ziffer, mit der die Zahl der Subnetze und Knoten festgelegt wird

Klasse	Standard-Netzmaske (Dezimale Punktschreibweise)	Netzmaske-Maske (Hexadezimale Punktschreibweise)
A	255.0.0.0	FF.0.0.0
B	255.255.0.0	FF.FF.0.0
C	255.255.255.0	FF.FF.FF.0

- Aufbau der Netzmaske:
  - Einsen kennzeichnen den Subnetz-Nummernteil eines Adressraumes
  - Nullen kennzeichnen den Teil des Adressraumes, der für die Knoten-Adressen zur Verfügung steht

Netzmaske

- Die Netzmaske unterteilt die Hostadresse in **Subnetznummer** (häufig **Subnetz-ID** genannt) und Hostadresse



- Die Netzmaske fügt eine weitere Hierarchieebene in die IP-Adresse ein
- Seit der Einführung des **Classless Interdomain Routing** 1993 werden Adressbereiche in der Notation `Anfangsadresse/Netzbits` vergeben
  - Die Netzmaske wird als Zahl hinter einem Schrägstrich angegeben
  - Die Zahl ist die Anzahl der Einsen im Netzwerkteil der Netzmaske

Alternative Schreibweise – CIDR

- Beispiel: 130.94.122.195/27

	Dezimal	Binär
IP-Adresse	130.094.122.195	10000010 01011110 01111010 11000011
Netzmaske	255.255.255.224	11111111 11111111 11111111 11100000

Schrägstrichformat	/25	/26	/27	/28	/29	/30	/31	/32
Netzmaske	128	192	224	240	248	252	254	255
Bit	1	2	3	4	5	6	7	8
Subnetzadressen (gesamt)		4	8	16	32	64		
Subnetze (maximal)		2	6	14	30	62		
Hostadressen (gesamt)	128	64	32	16	8	4	2	1
Hosts (maximal)	126	62	30	14	6	2	0	einzelner Host

Warum können 2 Hostadressen nicht an Knoten vergeben werden?

Jedes (Sub-)-Netzwerk hat eine Adresse (Netzdeskriptor) für das Netz selbst (alle Bits im Hostteil auf Null) und eine Broadcast-Adresse (alle Bits im Hostteil auf Eins)

Warum können 2 Subnetzadressen nicht verwendet werden?

Die Subnetzadressen, die ausschließlich aus Nullen und ausschließlich aus Einsen bestehen, dürfen nicht verwendet werden

Bit-Bestimmung bei Subnetzen

- Aufgabe des Netzwerkverwalters ist es bei der Einrichtung eines Netzwerks abzuschätzen, wie viele Subnetze und Hosts zu erwarten sind
- Beispiel: 5 Subnetze mit jeweils maximal 25 Hosts sind einzurichten
  - Für 25 Hosts sind 5 Hostbits nötig
  - Für 5 Subnetze sind 3 Subnetzbits nötig
  - Somit ist /27 geeignet

Schrägstrichformat	/25	/26	/27	/28	/29	/30	/31	/32
Netzmaske	128	192	224	240	248	252	254	255
Bit	1	2	3	4	5	6	7	8
Subnetzadressen (gesamt)		4	8	16	32	64		
Subnetze (maximal)		2	6	14	30	62		
Hostadressen (gesamt)	128	64	32	16	8	4	2	1
Hosts (maximal)	126	62	30	14	6	2	0	einzelner Host

Rechenbeispiel (von Wikipedia)

- Beispiel: IPv4-Adresse 130.94.122.195/27
- IP-Adresse AND Netzmaske = Subnetznummer bzw. Subnetz-ID

IP-Adresse	130.094.122.195	10000010 01011110 01111010 11000011
Netzmaske	255.255.255.224	11111111 11111111 11111111 11100000
Subnetz-ID	130.094.122.192	10000010 01011110 01111010 11000000

- IP-Adresse AND (NOT Netzmaske) = Hostnummer bzw. Hostadresse

IP-Adresse	130.094.122.195	10000010 01011110 01111010 11000011
Netzmaske	255.255.255.224	11111111 11111111 11111111 11100000
NOT		00000000 00000000 00000000 00011111
Hostadresse	3	00000000 00000000 00000000 00000011

Ergebnis

	Dezimal	Binär
IP-Adresse	130.094.122.195	10000010 01011110 01111010 11000011
Netzmaske	255.255.255.224	11111111 11111111 11111111 11100000
Subnetznummer	130.094.122.192	10000010 01011110 01111010 11000000
Hostadresse	3	00000000 00000000 00000000 00000011

- Bei einer Netzmaske mit 27 gesetzten Bits ergibt sich die Subnetznummer 130.94.122.192
- Es verbleiben 5 Bits und damit $2^5 = 32$ Adressen für den Hostadresse
  - Davon werden noch je eine Adresse (Netzdeskriptor) für das Netz selbst alle Bits im Hostteil auf Null) und für den Broadcast (alle Bits im Hostteil auf Eins) benötigt
  - Somit stehen 30 Adressen für Geräte zur Verfügung

Private Netze – Private IP-Adressen

- In privaten, lokalen Netzen (LAN) müssen IP-Adressen vergeben werden
- Diese sollten nicht mit real existierenden Internetangeboten kollidieren
- Es existieren private IP-Adressen, die im Internet nicht geroutet werden

Netzadressbereich:	10.0.0.0 bis 10.255.255.255
CIDR-Notation:	10.0.0.0/8
Anzahl Adressen:	$2^{24} = 16.777.216$
Netzklasse:	Klasse A. 1 privates Netz mit 16.777.216 Adressen

Netzadressbereich:	172.16.0.0 bis 172.31.255.255
CIDR-Notation:	172.16.0.0/12
Anzahl Adressen:	$2^{20} = 1.048.576$
Netzklasse:	Klasse B. 16 private Netze mit jeweils 65.536 Adressen

Netzadressbereich:	192.168.0.0 bis 192.168.255.255
CIDR-Notation:	192.168.0.0/16
Anzahl Adressen:	$2^{16} = 65.536$
Netzklasse:	Klasse C. 256 private Netze mit jeweils 256 Adressen

Status von IPv4

ZEIT ONLINE | INTERNET

INTERNET PROTOKOLL

Bye, bye IPv4

Die letzten Adressblöcke des alten Internet Protokolls Version vier sind vergeben. Die Umstellung auf IPv6, die seit Jahren nicht vorankommt, wird nun beginnen müssen.

VON: Monika Ermert | 2.2.2011 - 16:36 Uhr

Im Netz hat eine neue Zeitrechnung begonnen: In der Nacht zum Dienstag hat die Internet Assigned Numbers Authority (IANA) die letzten freien IPv4-Adressen verteilt. Wer künftig IP-Adressen an Nutzer vergeben möchte, sei es für Mobiltelefone, PCs oder internetfähige Autos, muss sich mit der nächsten Generation von "Rufnummern" befassen, mit der Internet-Protokoll Version 6 – IPv6.

Das Internet-Protokoll ist Teil der komplexen Struktur, die notwendig ist, damit Computer miteinander Daten austauschen können. Es sorgt darin für die korrekte Vermittlung der transportierten Informationen. IPv4 nutzt Adressen mit einer Länge von 32 Bit, was die Zahl der insgesamt verfügbaren IPs auf 4.294.967.296 oder 4,2 Milliarden Stück beschränkte.

Das klingt viel. Aber bei 6,5 Milliarden Menschen weltweit und angesichts des Trends, mehr und mehr Geräte internetfähig zu machen, ist seit Jahren klar, dass die IPv4-Adressen knapp werden. Netzanbieter nutzten daher dynamische Adressen, vergaben also keine festen für jedes einzelne Gerät. Doch auch diese Technik ist begrenzt, weswegen seit vielen Jahren an einem neuen Internet-Protokoll gearbeitet wurde.

IPv6 basiert auf längeren Nummern und bietet damit für die Zukunft die nicht mehr so richtig vorstellbare Zahl von 340 Sextillionen eindeutiger Internetadressen. Jedes Sandkorn könnte damit künftig eine IP-Adresse bekommen.

Bis heute allerdings kam die technische Umstellung nur langsam voran. Nun sind jedoch die letzten freien IPv4-Blöcke an den für Asien zuständigen regionalen IP-Adressverwalter vergeben worden. Bis diese an die einzelnen Netzbetreiber und deren Kunden verteilt sind, wird es noch eine Weile dauern. Außerdem bekommt jede der weltweit fünf Verwaltungen in den kommenden Tagen noch eine Reserve von 16 Millionen IPv4-Adressen, doch der Zeitraum ist absehbar.

Aufbau von IPv6-Adressen

- IPv6-Adressen bestehen aus 128 Bits (16 Bytes)
- Daher können 2^{128}, also $\approx 3{,}4 * 10^{38}$ Adressen dargestellt werden
- Einführung ist wegen des begrenzten Adressraums von IPv4 notwendig
- Dezimaldarstellung ist unübersichtlich
  - Aus diesem Grund stellt man IPv6-Adressen hexadezimal dar
  - Zur weiteren Vereinfachung der Darstellung werden jeweils zwei Oktetts zusammengefasst und in Gruppen durch Doppelpunkt getrennt dargestellt
    Beispiel: 2001:0db8:85a3:08d3:1319:8a2e:0370:7344

Eigenschaften von Transportprotokollen

- Einschränkungen/Nachteile von Netzwerken sind u.a.
  - Nachrichten gehen eventuell verloren oder werden verworfen
  - Nachrichten kommen in der falschen Reihenfolge an
  - Mehrere Kopien einer Nachricht erreichen das Ziel
  - Nachrichten dürfen eine bestimmte Größe nicht überschreiten
  - Die Verzögerung des Netzwerks kann schwanken
- Gewünschte Eigenschaften von Transportprotokollen sind u.a.
  - Garantierte Nachrichtenübertragung
  - Einhaltung der Reihenfolge von Nachrichten
  - Unterstützung beliebig großer Nachrichten
  - Der Sender soll in der Lage sein, den Datenfluss zu kontrollieren
    - Vergleichbar mit der Flusskontrolle in der Sicherungsschicht
- Gesucht: Transportportokolle, die die negativen Einschränkungen der Netzwerke in die (positiven) Eigenschaften umwandeln, die von Transportportokollen erwartet werden $\Longrightarrow$ UDP und TCP
- Zuvor klären: **Ports** und **Sockets**

Ports (1/2)

- Jede Anwendung, die die Transportportokolle TCP oder UDP nutzt, hat eine Portnummer
  - Diese gibt an, welcher Dienst auf dem Quell-Host angesprochen wird
  - Bei TCP und UDP ist die Portnummer 16 Bit groß (0 bis 65535)
- Portnummern können im Prinzip beliebig vergeben werden
  - Es gibt Konventionen, welche Standardanwendungen welche Ports nutzen

Portnummer	Dienst	Beschreibung
21	FTP	Dateitransfer
22	SSH	Verschlüsselte Terminalemulation (Secure Shell)
23	Telnet	Terminalemulation
25	SMTP	E-Mail-Versand
53	DNS	Auflösung von Domainnamen in IP-Adressen
67	DHCP	Zuweisung der Netzwerkkonfiguration an Clients
80	HTTP	Webserver
110	POP3	Client-Zugriff für E-Mail-Server
143	IMAP	Client-Zugriff für E-Mail-Server
443	HTTPS	Webserver (verschlüsselt)
993	IMAPS	Client-Zugriff für E-Mail-Server (verschlüsselt)
995	POP3S	Client-Zugriff für E-Mail-Server (verschlüsselt)

- Die Tabelle enthält nur eine kleine Auswahl bekannter Portnummern

Ports (2/2)

- Die Portnummern sind in 3 Gruppen unterteilt:
  - 0 bis 1023 (*Well Known Ports*)
    - Sind fest zugeordnet und allgemein bekannt
  - 1024 bis 49151 (*Registered Ports*)
    - Können sich Anwendungsentwickler für eigene Protokolle registrieren
  - 49152 bis 65535 (*Private Ports*)
    - Sind nicht registriert
- Unter Linux/UNIX existiert die Datei `/etc/services`
  - Hier ist die Zuordnung von Anwendungen zu Portnummern festgelegt
- Unter Windows: `\WINNT\SYSTEM32\ETC\SERVICES`

Sockets

- Sockets sind die plattformunabhängige, standardisierte **Schnittstelle** zwischen der Implementierung der Netzwerkprotokolle im Betriebssystem und den Anwendungen
- **Ein Socket besteht aus einem Port mit einer IP-Adresse**
- Man unterscheidet zwischen Stream Sockets und Datagram Sockets
  - Stream Sockets $\Longrightarrow$ TCP
  - Datagram Sockets $\Longrightarrow$ UDP
- Übersicht der Sockets unter Linux/UNIX: `netstat -n`

Alternativen zu Sockets in der Interprozesskommunikation

Pipes, Message Queues und gemeinsamer Speicher (Shared Memory)

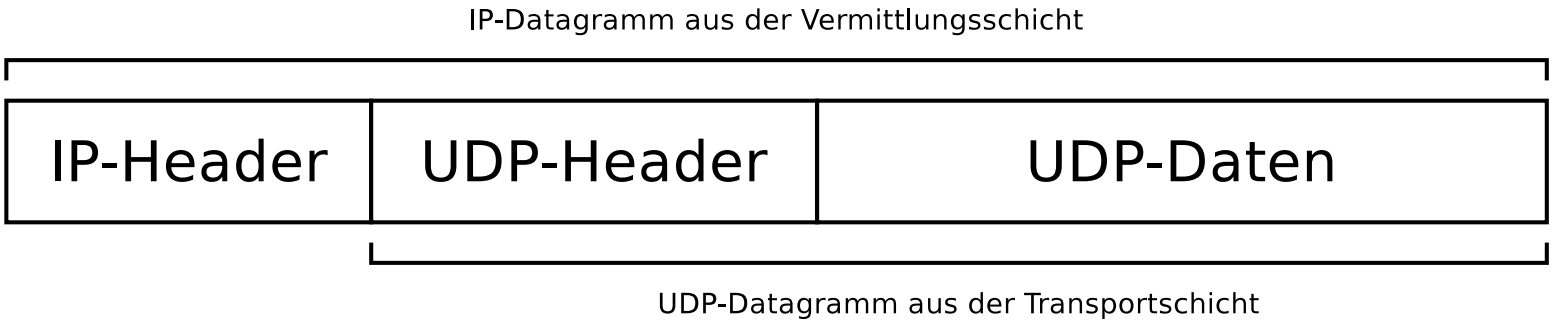
User Datagram Protocol (UDP)

- **Verbindungsloses Transportprotokoll**
  - Datenübertragungen finden ohne vorherigen Verbindungsaufbau statt
- Einfacheres Protokoll als das verbindungsorientierte TCP
  - Nur für die Adressierung zuständig
  - Keine Sicherung der Datenübertragung
  - Übertragungen werden nicht vom Empfänger beim Sender bestätigt
    - Daten können/dürfen verloren gehen
- UDP ist u.a. für Videoübertragungen optimal
  - Geht bei TCP ein Paket (Bild) verloren, wird es neu angefordert
  - Es käme zu Aussetzern bzw. ein Wiedergabepuffer wäre notwendig
  - Bei UDP wäre nur ein Bild verloren oder die Qualität kurzzeitig schlechter
- Datagramm hat im OSI-Modell unterschiedliche Bezeichnungen

OSI-Schicht	**Datagrammbezeichnung**
Schicht 4 (Transportschicht)	Datensegment
Schicht 3 (Vermittlungsschicht)	Datenpaket (IP-Paket)
Schicht 2 (Sicherungsschicht)	Datenrahmen (Frame)

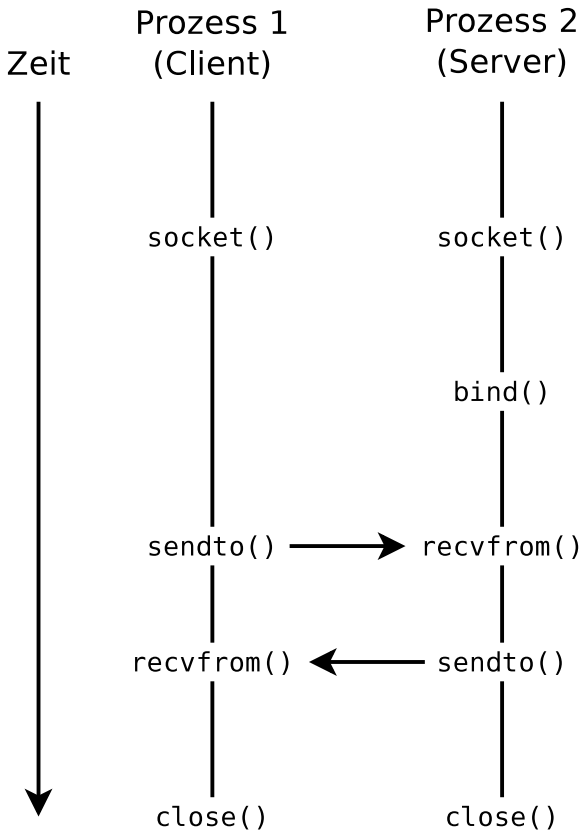
Aufbau von UDP-Datagrammen

- UDP-Pakete werden via Internet Protokoll (IP) übertragen
  - IP setzt vor jedes UDP-Paket einen weiteren Header (den IP-Header)



- Im UDP-Header steht:
  - Port-Nummer des sendenden Prozesses
  - Port-Nummer des Prozesses, der das Paket empfangen soll
  - Länge des kompletten Datagramms (Header und Nutzdaten)
  - Prüfsumme über (Header und Nutzdaten)

Verbindungslose Kommunikation mit Sockets – UDP

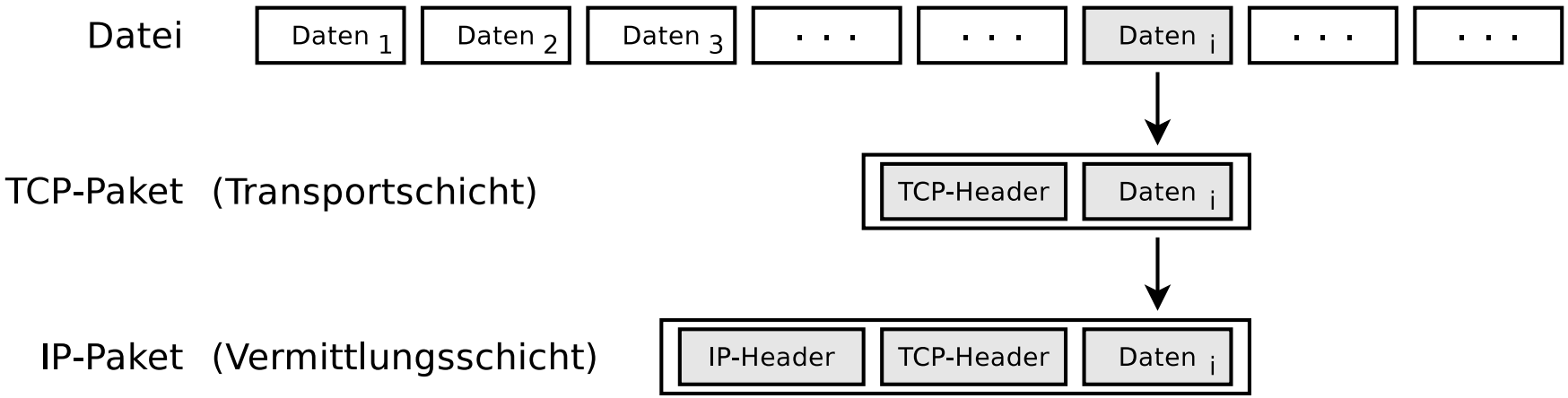


- **Client**
  - Socket erstellen (`socket`)
  - Daten senden (`sendto`) und empfangen (`recvfrom`)
  - Socket schließen (`close`)
- **Server**
  - Socket erstellen (`socket`)
  - Socket an einen Port binden (`bind`)
  - Daten senden (`sendto`) und empfangen (`recvfrom`)
  - Socket schließen (`close`)

Transmission Control Protocol (TCP)

- **Verbindungsorientiertes Transportprotokoll**
  - Das bedeutet, dass eine TCP-Verbindung wie eine Datei geöffnet und geschlossen wird und das man die Position im Datenstrom bestimmen kann
  - Genau wie man bei einer Datei wird die Position der Lese- oder Schreibposition angeben
- Garantiert, dass Pakete vollständig und in der richtigen Reihenfolge ihr Ziel erreichen
- Erweitert das IP-Protokoll um die Zuverlässigkeit, die für viele Anwendungen gewünscht bzw. notwendig ist
- TCP ist das meist genutzte Transportprotokoll in IP-Netzen
  - Wegen der hohen Verbreitung wird anstatt vom IP-Protokoll häufig einfach von TCP/IP gesprochen
- Neben den Nutzdaten existiert auch bei TCP-Paketen ein Header
  - Im Gegensatz zu UDP-Datagrammen ist der Header von TCP-Paketen deutlich komplexer

Arbeitsweise von TCP (1/2)



- TCP-Pakete haben eindeutige Folgenummern (**Sequenznummern**)
  - Damit kann die Reihenfolge der Segmente korrigiert und doppelt angekommene Segmente können aussortiert werden
- Die Länge des Segments ist aus dem IP-Header bekannt
  - So können Lücken im Datenstrom entdeckt werden, und der Empfänger kann verlorengegangene Segmente neu anfordern
- Beim Öffnen einer TCP-Verbindung (**Dreiwege-Handshake**) tauschen beide Kommunikationspartner Kontrollinformationen aus
  - So ist sichergestellt, dass der jeweilige Partner existiert und Daten annehmen kann

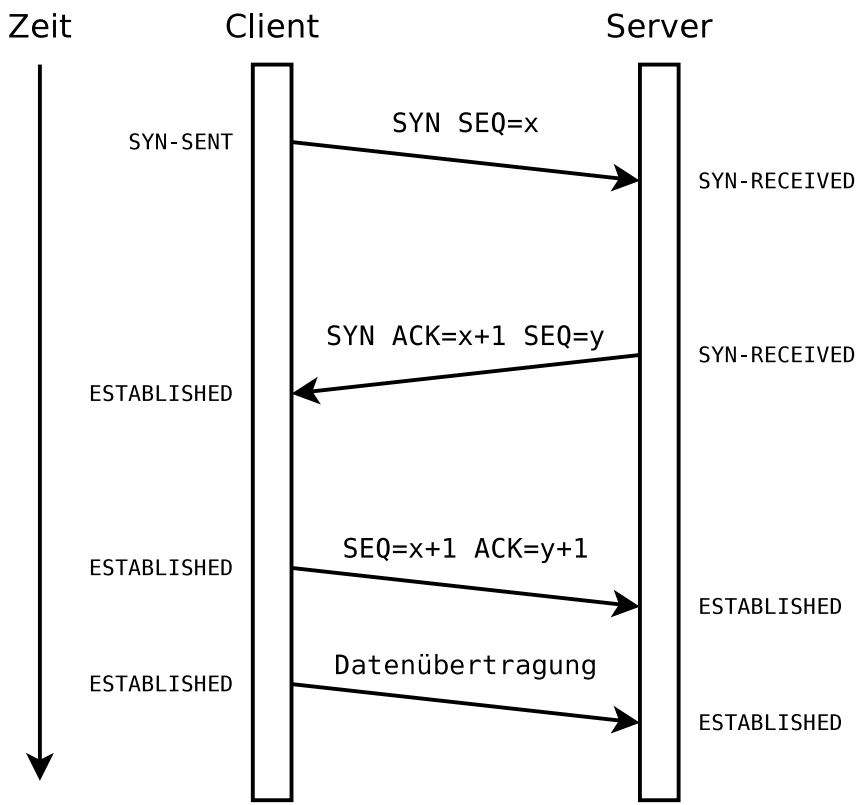
Arbeitsweise von TCP (2/2)

- Dazu schickt Station A ein Paket mit der Aufforderung, die Folgenummern zu synchronisieren
- Das einleitende Paket mit gesetztem SYN-Bit (*Synchronize*) gibt die Anfangs-Sequenznummer des Clients bekannt
  - Die Anfangs-Sequenznummer wird zufällig bestimmt
- Bei allen nachfolgenden Paketen ist das ACK-Bit (*Acknowledge*) gesetzt
  - Das ACK-Bit ist mit einer Quittung gleichzusetzen
- Der Server antwortet mit ACK, SYN und gibt auch seine Anfangs-Sequenznummer für Übertragungen in die Gegenrichtung an
- Der Client bestätigt mit ACK und die Verbindung ist aufgebaut
  - Nun können Daten über die Verbindung ausgetauscht werden
- Diese Art des Austausches von Kontrollinformationen, bei der jede Seite die Aktionen der Gegenseite bestätigen muss, ehe sie wirksam werden, nennt man **Dreiwege-Handshake**

TCP-Verbindungsaufbau (Dreiwege-Handshake)

- Der Server wartet passiv auf eine ankommende Verbindung
  - Funktionen `listen` und `accept`

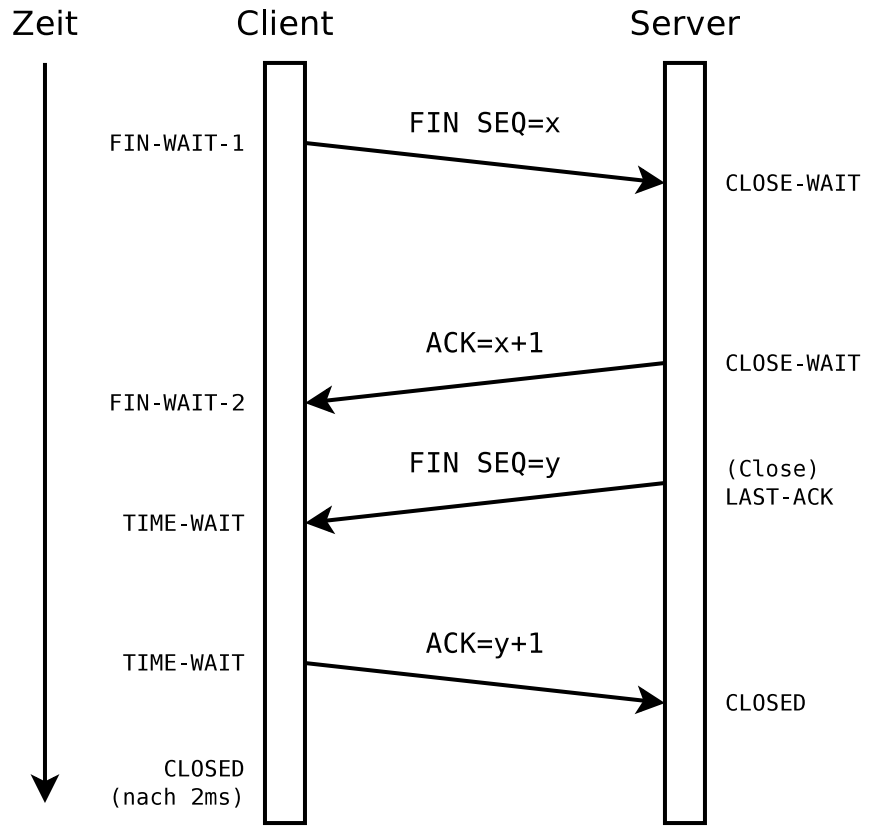
1. Der Client führt `connect` aus, sendet ein TCP-Paket mit SYN=1 und ACK=0 und wartet auf die Antwort
   $\Longrightarrow$ *Synchronize*
2. Läuft am Port ein Serverprozess, der die Verbindung annimmt, sendet er als Bestätigung ein TCP-Paket mit SYN=1 und ACK=1
   $\Longrightarrow$ *Synchronize Acknowledge*
3. Der Client bestätigt mit einem TCP-Paket mit SYN=0 und ACK=1 und die Verbindung ist hergestellt
   $\Longrightarrow$ *Acknowledge*



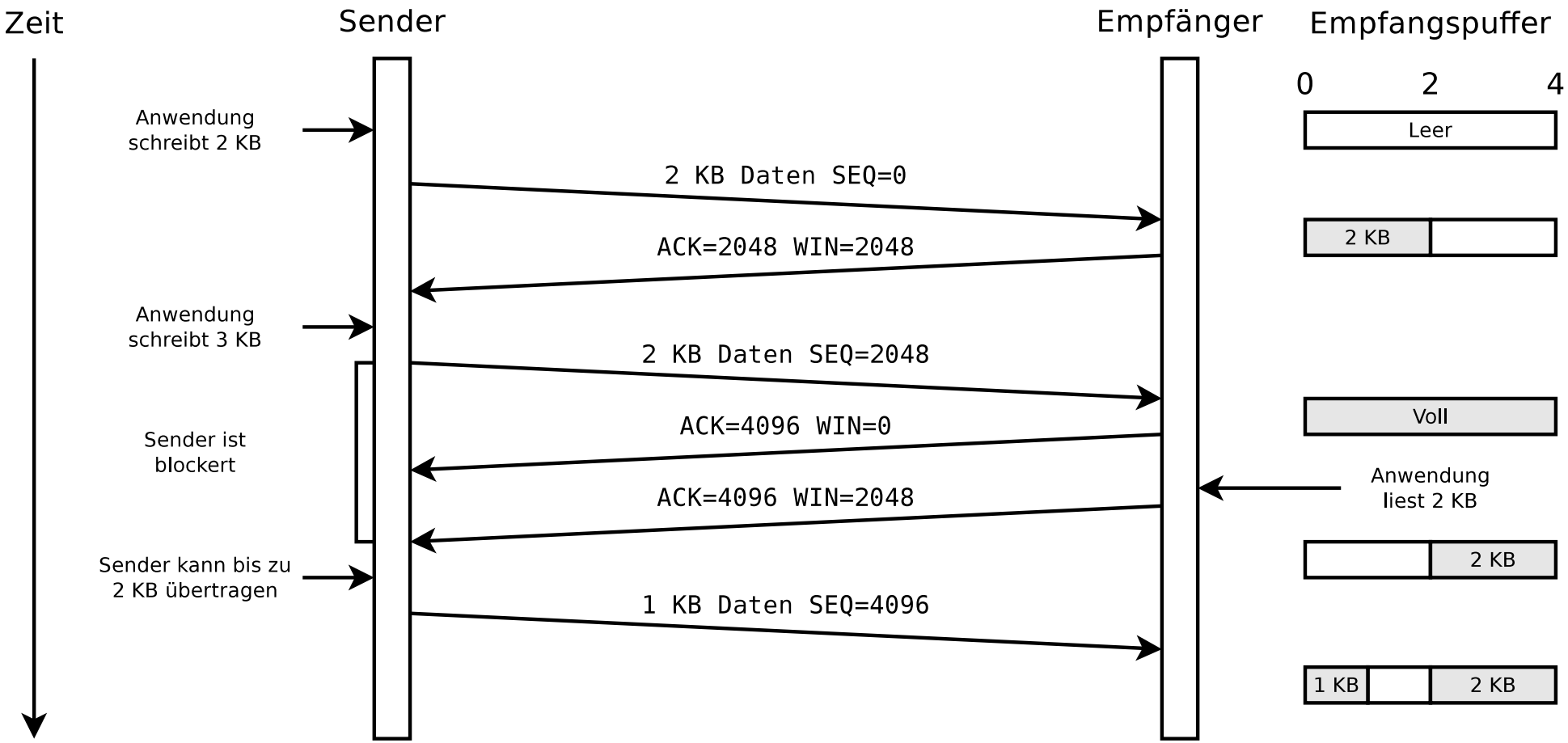
TCP-Verbindungsabbau

- Der geregelte Verbindungsabbau ist dem Verbindungsaufbau ähnlich
- Statt des SYN-Bit kommt das FIN-Bit zum Einsatz, welches anzeigt, dass keine Daten mehr vom Sender kommen werden
  - Funktion `close`

1. Client sendet den Verbindungsabbauwunsch mit FIN=1 und ACK=1 und wartet auf die Antwort
2. Server sendet eine Bestätigung mit ACK=1
3. Server sendet den Verbindungsabbauwunsch mit FIN=1 und ACK=1 und wartet auf die Antwort
4. Client sendet eine Bestätigung mit ACK=1

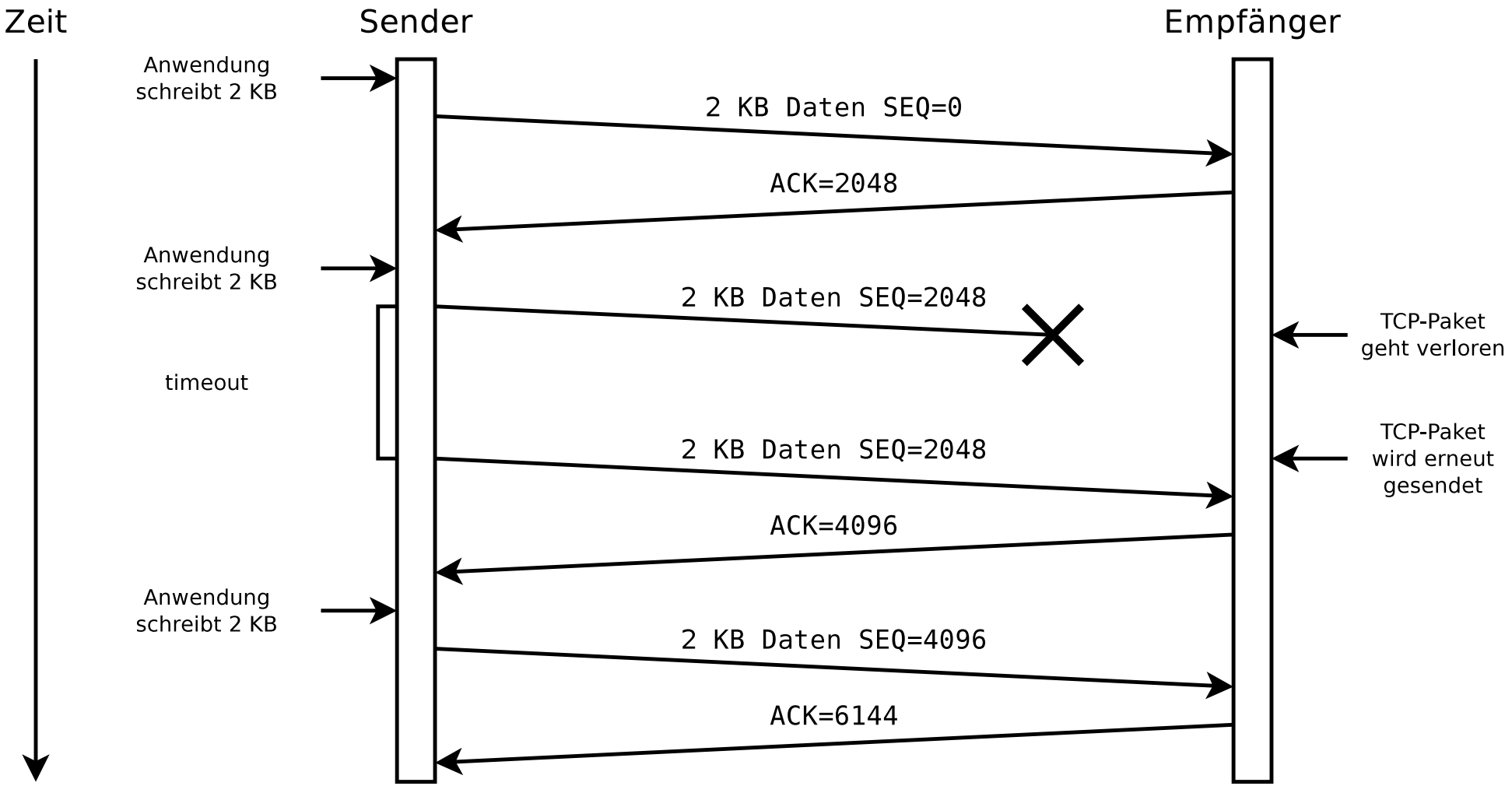


Flusskontrolle bei TCP

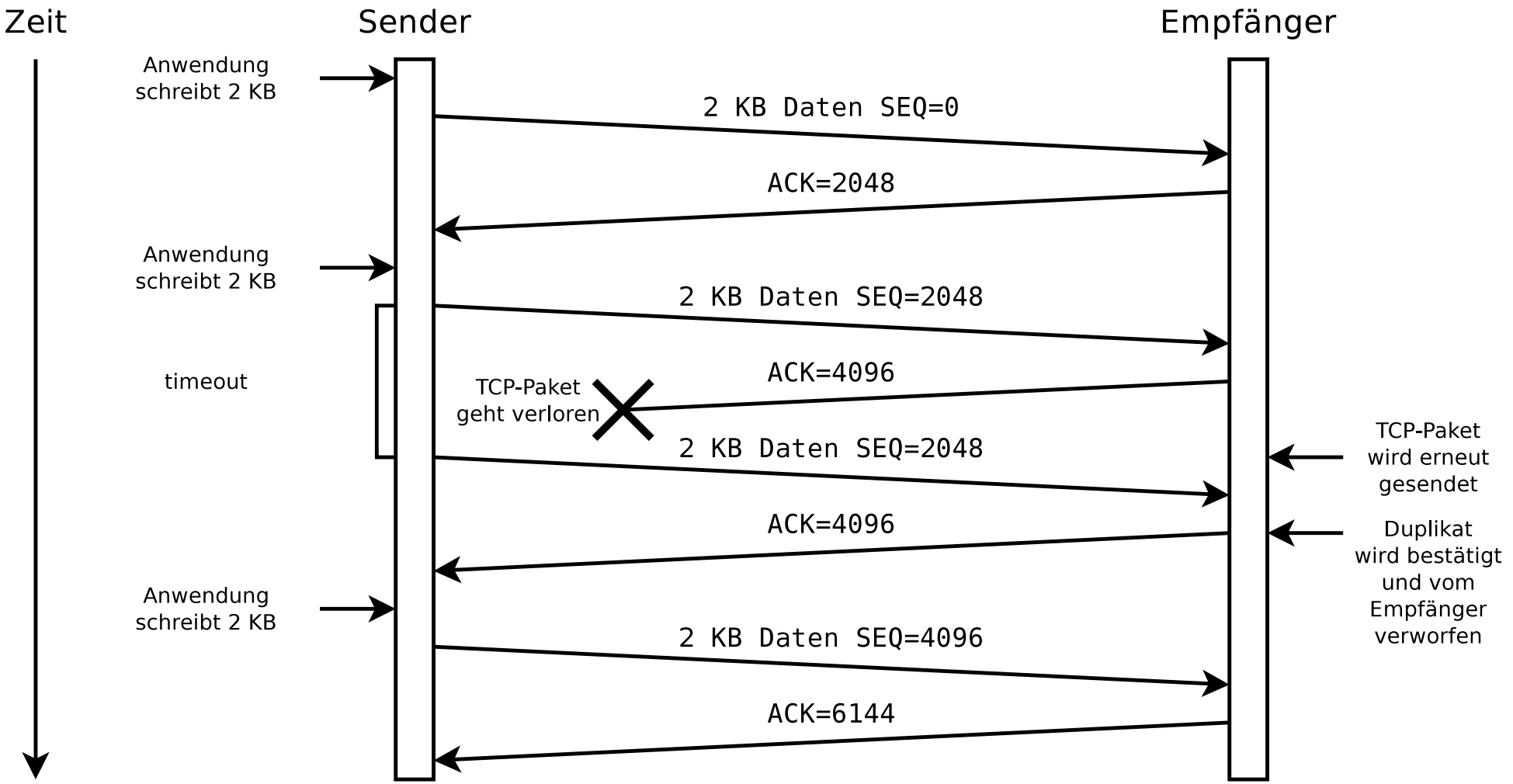


Quelle: Andrew S. Tanenbaum

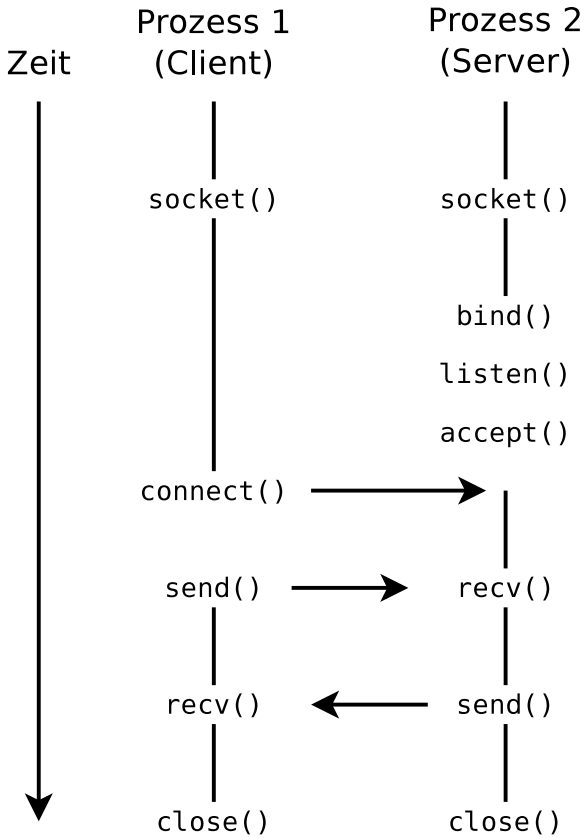
Verhalten von TCP beim Verlust von Paketen (Szenario 1)



Verhalten von TCP beim Verlust von Paketen (Szenario 2)



Verbindungsorientierte Kommunikation mit Sockets – TCP



- **Client**
  - Socket erstellen (`socket`)
  - Client mit Server-Socket verbinden (`connect`)
  - Daten senden (`send`) und empfangen (`recv`)
  - Socket schließen (`close`)
- **Server**
  - Socket erstellen (`socket`)
  - Socket an einen Port binden (`bind`)
  - Socket empfangsbereit machen (`listen`)
    - Richtete eine Warteschlange für Verbindungen mit Clients ein
  - Server akzeptiert Verbindungen (`accept`)
  - Daten senden (`send`) und empfangen (`recv`)
  - Socket schließen (`close`)

Sitzungsschicht

- Verantwortlich für **Aufbau, Überwachung und Beenden einer Sitzung**
  - Eine Sitzung ist die Grundlage für eine virtuelle Verbindung zwischen 2 Anwendungen auf physisch unabhängigen Rechnern
- Funktionen zur **Dialogkontrolle** (welcher Teilnehmer gerade spricht)
  - Sorgt für Verbindungsaufbau und Verbindungsabbau
  - Sorgt für die Darstellung der Daten in einer für die darüberliegende Schicht unabhängigen Form
- Funktionen zur **Synchronisierung**
  - Kontrollpunkte können in längeren Übertragungen eingebaut werden
  - Kommt es zum Verbindungsabbruch, kann zum nächsten Kontrollpunkt zurückgekehrt werden und die Übertragung muss nicht von vorne beginnen
- Wird in der Praxis kaum benutzt
- Beispiel für ein Protokoll in dieser Schicht: Remote Procedure Calls (RPC) als Technik zur Interprozesskommunikation

Darstellungsschicht

- Enthält Regeln zur **Formatierung (Präsentation) der Nachrichten**
  - Der Sender kann den Empfänger informieren, dass eine Nachricht in einem bestimmten **Format** (z.B. ASCII) vorliegt
  - Datensätze können hier mit Feldern (z.B. Name, Matrikelnummer...) definiert werden
  - **Art und Länge der Datentypen** können definiert werden
  - **Komprimierung und Verschlüsselung** können hier eine Rolle spielen
- Wird in der Praxis kaum benutzt
- Beispiele für Protokolle in dieser Schicht: T.73 und EHKP-6 für Bildschirmtext

Situation heute

Die Funktionalitäten, die für die Sitzungsschicht und Darstellungsschicht vorgesehen waren, sind heute fast immer Teil der Protokolle und Dienste in der Anwendungsschicht

Anwendungsschicht

- Enthält Anwendungsprotokolle und darauf aufbauende Dienste u.a. zur Datenübertragung, Synchronisierung und Fernsteuerung von Rechnern und Namensauflösung
- Einige Anwendungsprotokolle und deren Funktionalität:
  - Namensauflösung (DNS)
  - Automatische Vergabe von Adressen (DHCP)
  - Zeitsynchronisierung (NTP)
  - Fernsteuerung von Computern (Telnet, SSH)
  - Übertragung von Daten (HTTP)
  - Emails austauschen (SMTP)
  - Emails herunterladen (POP3)
  - Dateien hochladen und herunterladen (FTP)

Nächste Vorlesung

Nächste Vorlesung:

22.12.2011