Fakultät für Informatik

MNM TEAM
Munich Network Management Team

Institut für Informatik

LMU

Systempraktikum im Wintersemester 2009/2010 (LMU):
Vorlesung vom 07.01. – Foliensatz 7

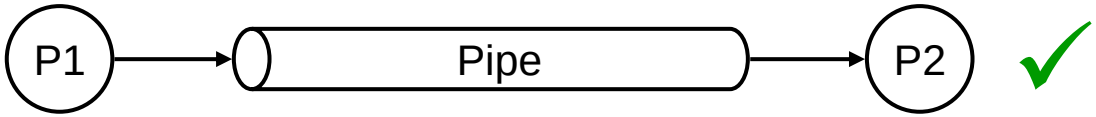
Rechnernetze & Verteilte Systeme (T)
Netzprogrammierung/Sockets (P)

Thomas Schaaf, Nils gentschen Felde

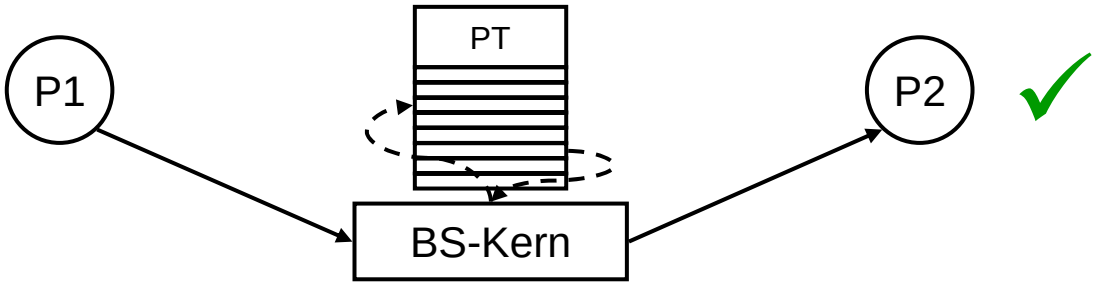
Interprozesskommunikation

MNM TEAM

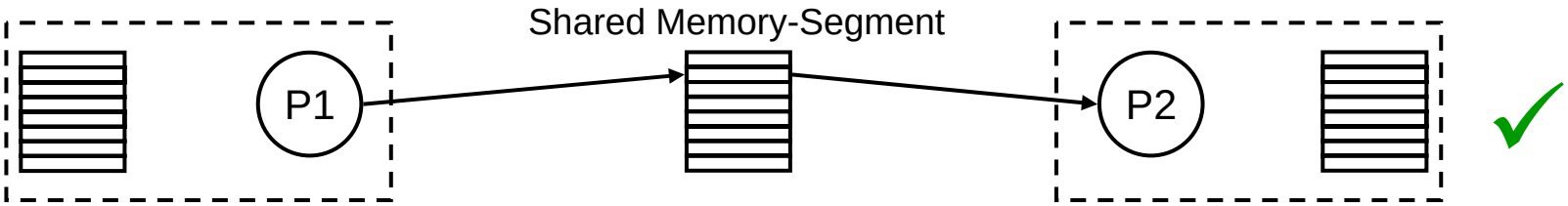
- Überblick: Interprozesskommunikation (stark abstrahiert)
  - Pipes und FIFOs:



  - Signale:



  - Speicherbasierte Kommunikation (Shared Memory-Konzept):



  - Netzbasierte Kommunikation (TCP/IP):

Heute

Protokolle

MNM TEAM

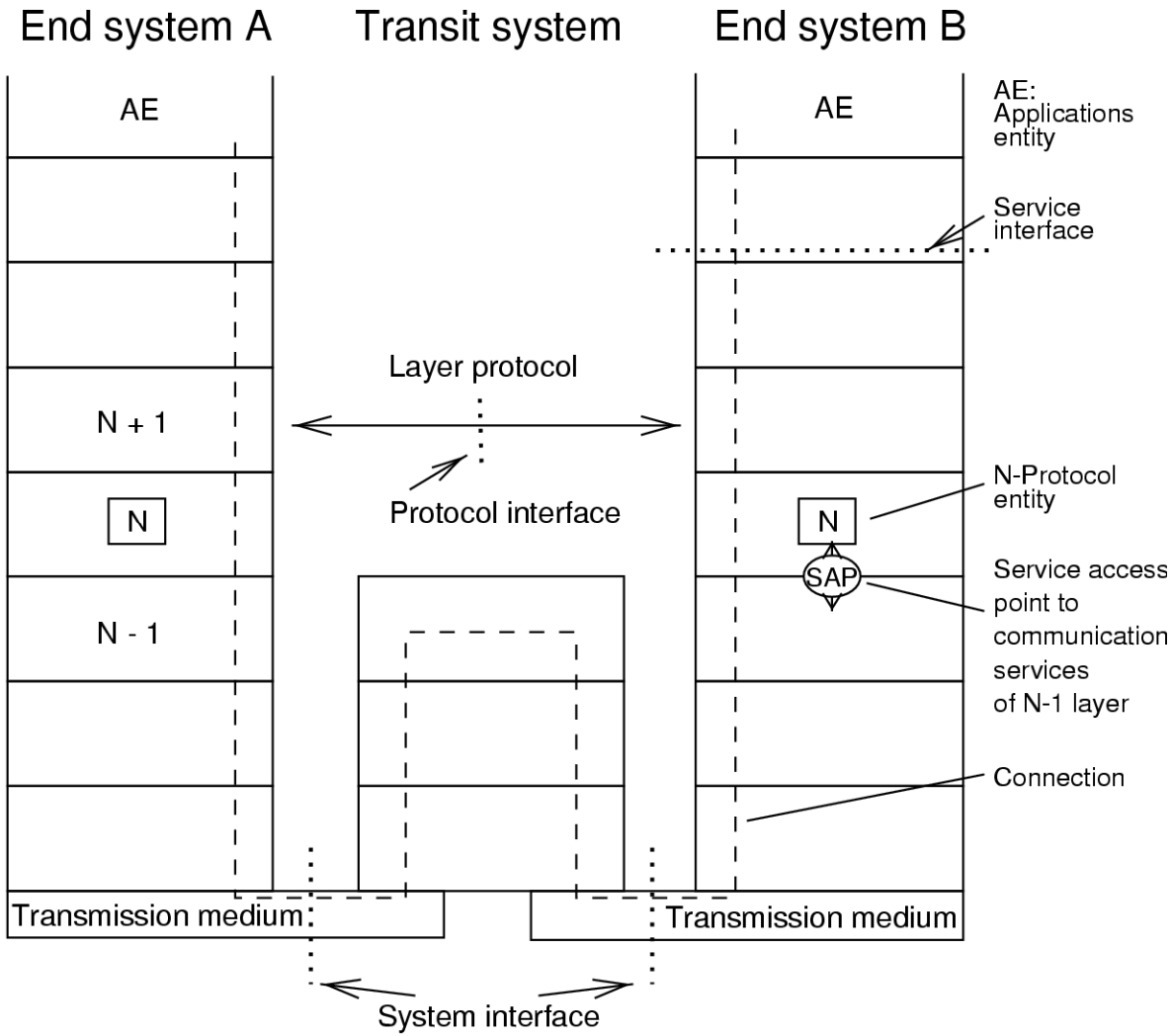
- Protokoll definiert "Regeln für den Informationsaustausch"
- Teile einer Protokolldefinition:
  - Verwendete Codes
  - Nachrichtenlänge
  - Nachrichtenformat
  - Form der Adressierung
  - Bestätigungen
  - u.a.

Geschichtete Protokolle

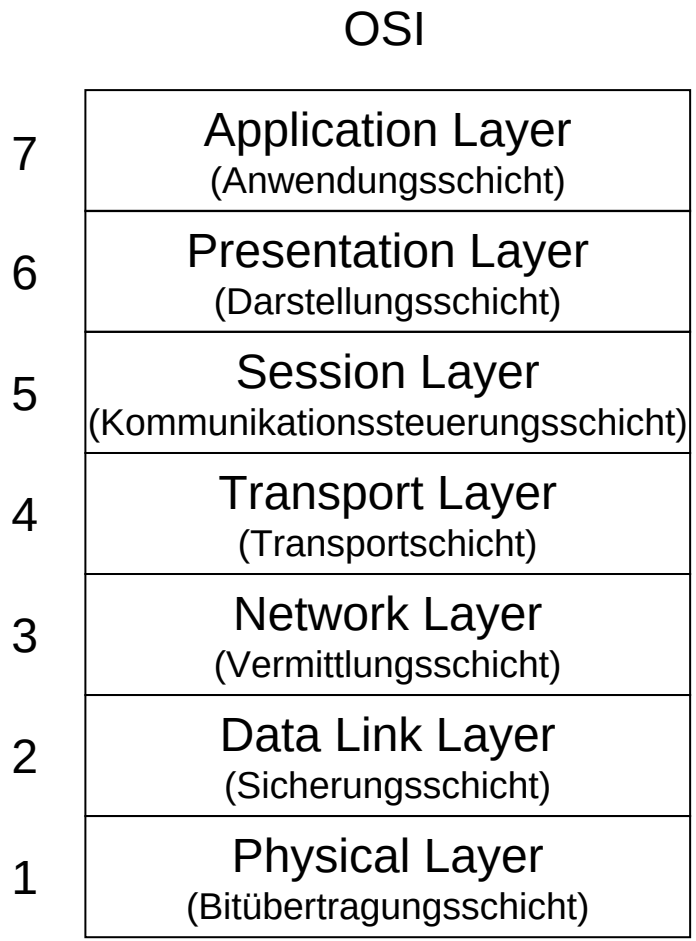
MNM TEAM

- Geschichtete Protokolle
  - Grundidee: Verschiedene Protokollebenen (statt ein allumfassendes Protokoll)
  - Teilprotokolle:
    - Kümmern sich jeweils um einen bestimmten Teil der Kommunikation
    - Bauen aufeinander auf
    - Setzen bestimmte gegenseitige Schnittstellen voraus
  - Geschichtete Teilprotokolle bilden einen Protokollstapel (Stack)
- Interoperabilität
  - Konsortien/Gremien definieren herstellerübergreifende Standards/Normen
  - Wichtige Gruppe: OSI (Open Systems Interconnection)

OSI-Architekturmodell

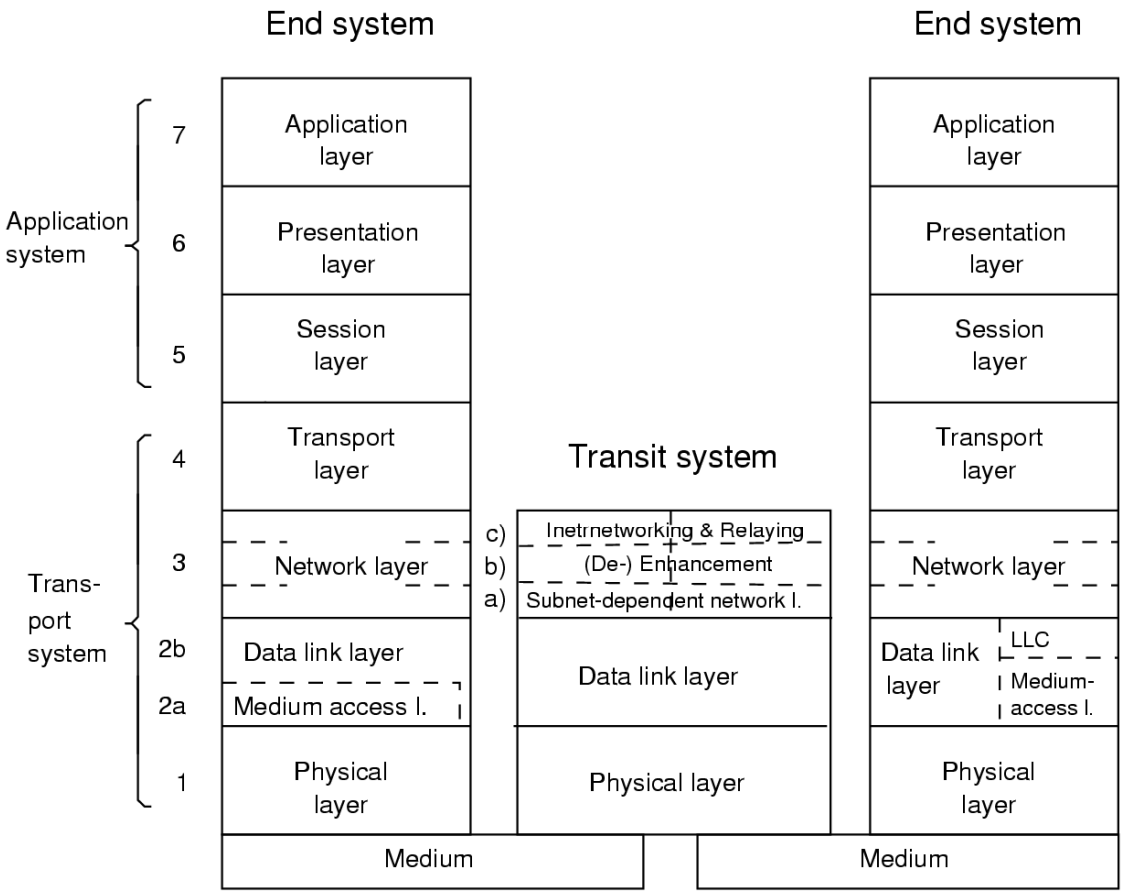


Das OSI-Schichtenmodell (1)



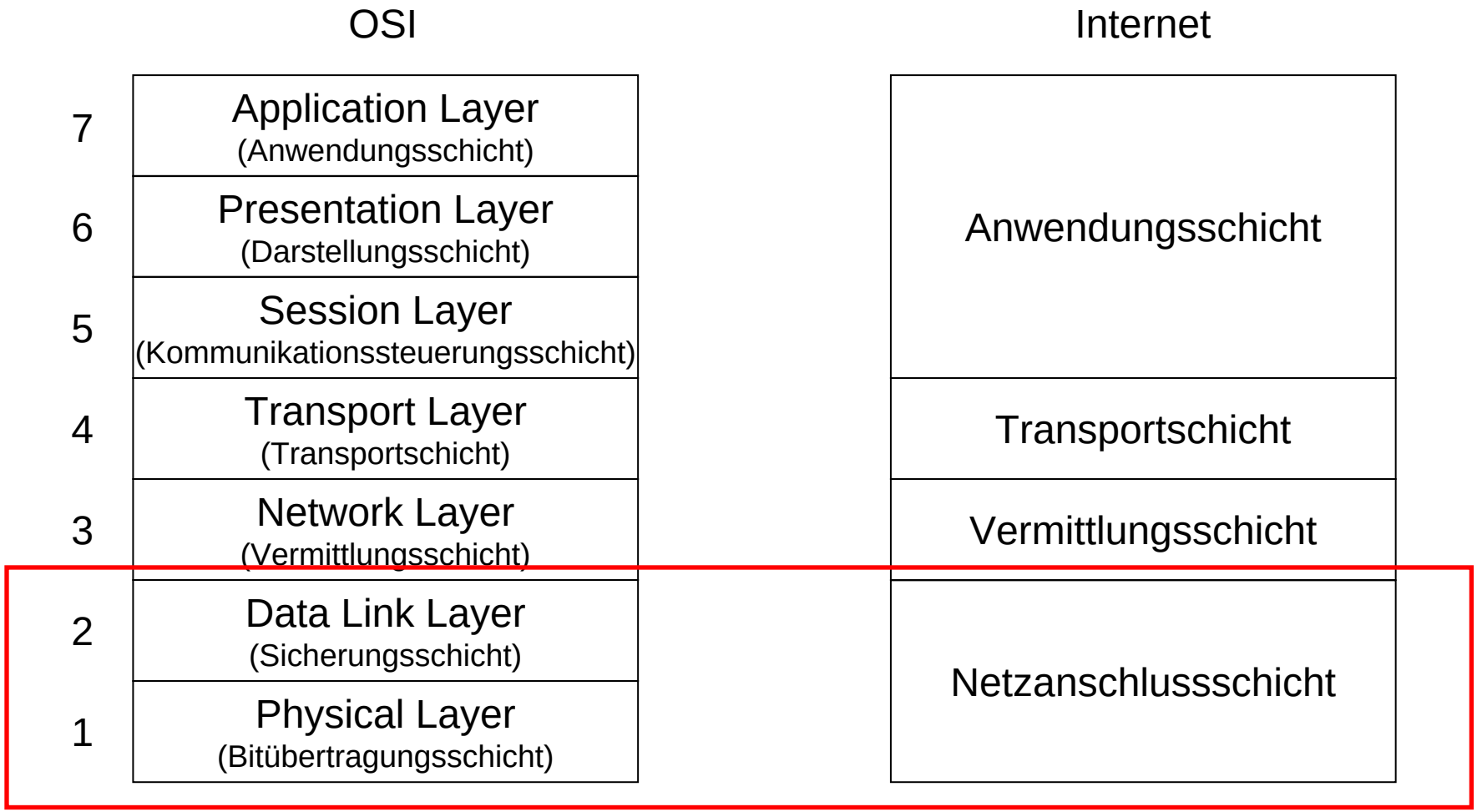
- Open System Interconnection (OSI) Basisreferenzmodell der International Standards Organisation (ISO)
- 7 Schichten (layers)
- Jede Schicht repräsentiert eine Funktion, kein Protokoll
- Kommunikation über Protokolle

Das OSI-Schichtenmodell (2)



c) Global network layer
b) Network adaption layer
a) Subnetwork network layer

OSI- und Internet-Schichtenmodell

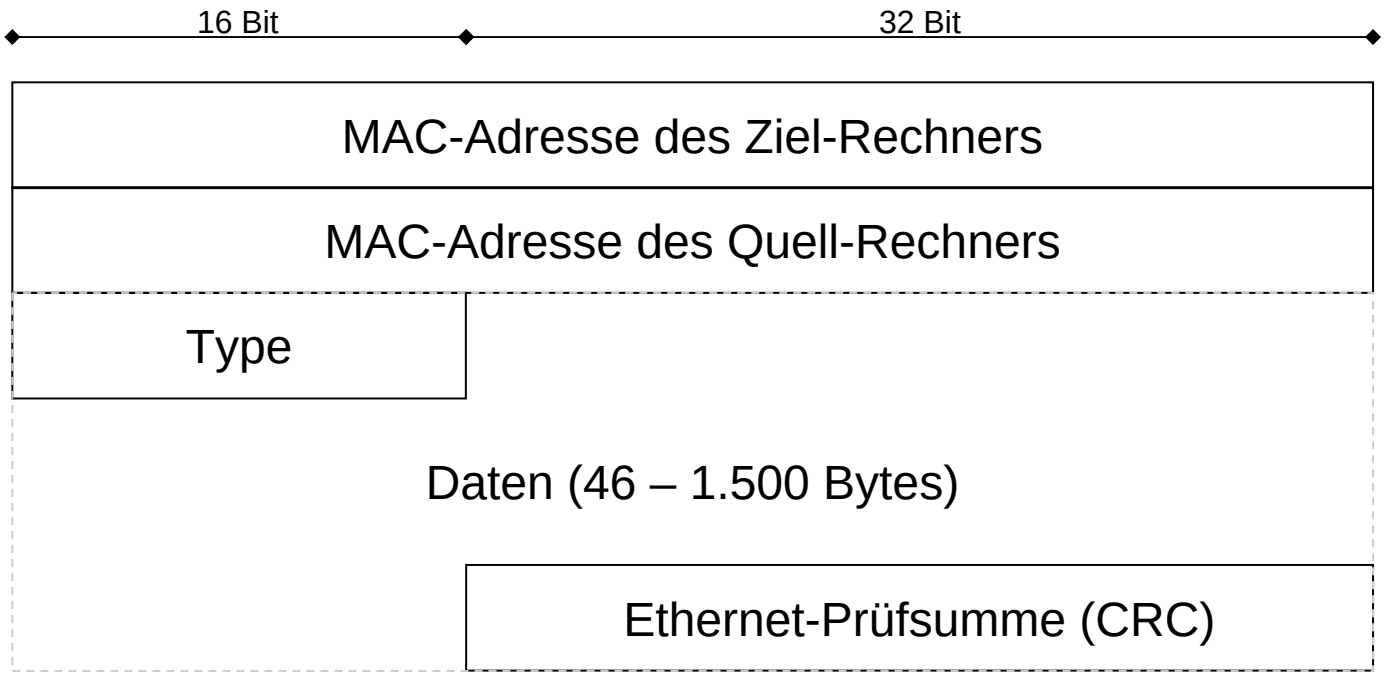


Ethernet

MNM TEAM

- Protokoll des Physical- und Data Link Layers (Schichten 1 und 2)
- Spezifiziert in RFC 894
- Identifikation über eindeutige MAC-Adresse
  - 6 Bytes lang
  - Ersten 3 Bytes beinhalten Herstellerkennung
  - Folgenden 3 Bytes sind fortlaufende Nummer
  - Beispiel: `00:0A:E4:` (Intel) `12:C4:62` (Nummer)
- Spezielle Adressen sind reserviert
  Bsp.: `ff:ff:ff:ff:ff:ff` für Broadcast
- Medienzugang nicht streng geregelt
  - Jeder darf senden, falls er das Medium als frei betrachtet
  - Medienzugangsverfahren CSMA/CD (Carrier Sense, Multiple Access, Collision Detection)

Der Ethernet-Header



- Beispiele für den „Type“
  - 0800: Daten
  - 0806: ARP
  - 0835: RARP
- Prüfsumme:
  CRC-32 als Generatorpolynom

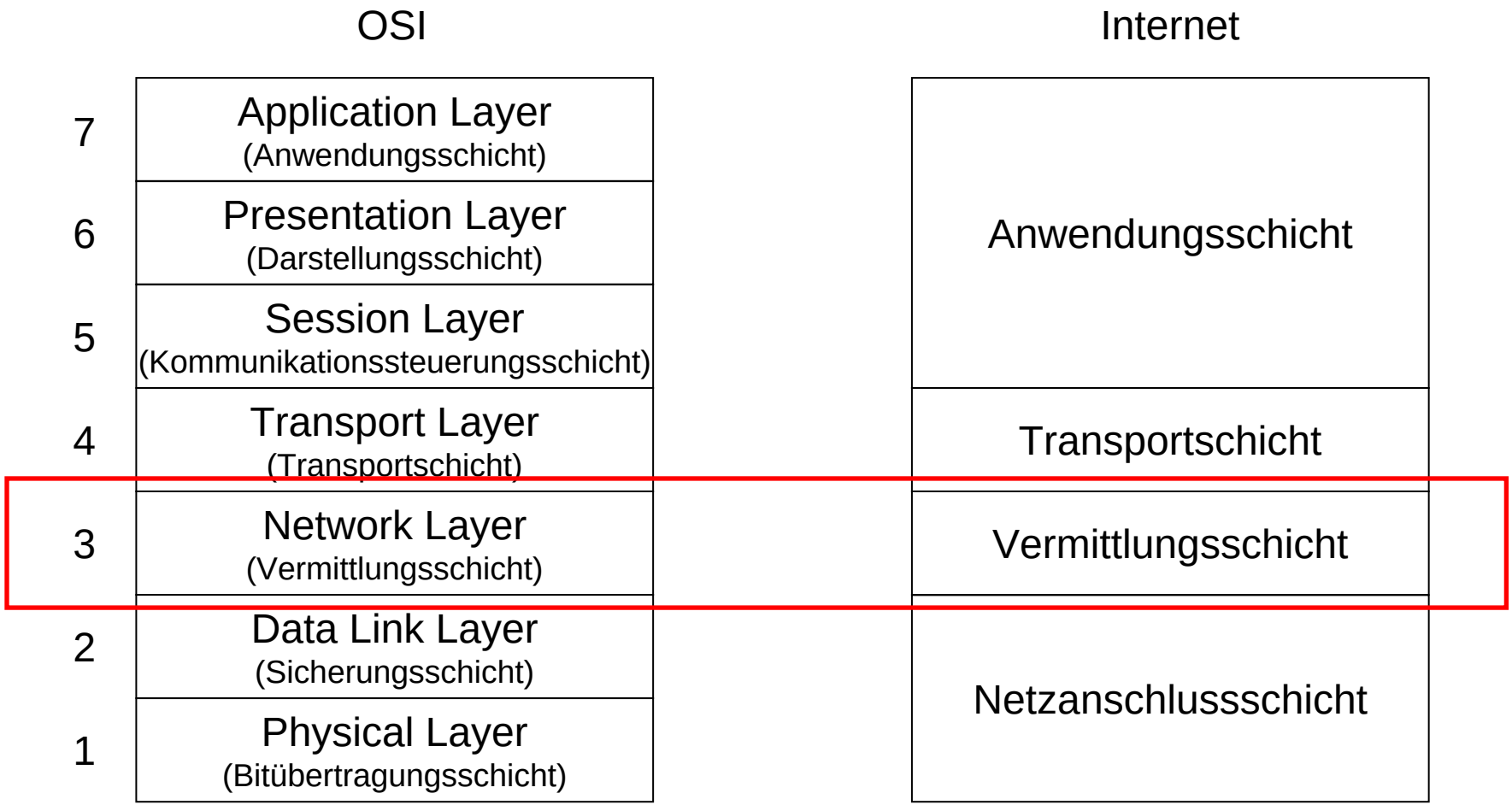
Repeater und Switches

MNM TEAM

- Repeater
  - Arbeitet auf Netzanschlussebene
  - Nimmt Signale auf, gibt sie an alle angeschlossenen Stationen weiter
  - Verstärkung/Rekonstruktion der Datenimpulse

- Switch
  - Ähnlich dem Repeater
  - Filterung des Datenverkehrs nach MAC-Adresse
  - Überprüft Pakete auf Korrektheit (CRC-Check)
  - Vorteil: erhöhte Performanz
  - Nachteile z.B. bei Fehlersuche

OSI- und Internet-Schichtenmodell



Das Internet Protocol (IPv4)

MNM TEAM

- 32 Bit Adressen (4 Byte)
- Gängige Schreibweise: „Dotted-Decimal“
  Bsp.: 192.168.218.43
- IP-Adresse besteht aus 2 Teilen:
  - Netzadresse
  - Hostadresse
- Längen von Netz- und Hostadresse variieren
- Längen von Netz- und Hostadresse spezifiziert durch Netzmasken
  - CIDR (Classless Inter-Domain Routing, RFC 1519)
  - Ebenfalls 32 Bit
  - Durch AND-Verknüpfung von IP-Adresse mit Netzmaske ergibt sich die Netzadresse
  - Durch XOR-Verknüpfung der Netzadresse mit der IP-Adresse ergibt sich die Hostadresse

Beispiel

MNM TEAM

- IP-Adresse: 192.168.218.43
  (11000000.10101000.11011010.00101011)
- Netzmaske: 255.255.255.0
  (11111111.11111111.11111111.00000000)

```
     11000000.10101000.11011010.00101011
AND  11111111.11111111.11111111.00000000
-----------------------------------------
     11000000.10101000.11011010.00000000
     (192.168.218.0) -> Netzadresse
```

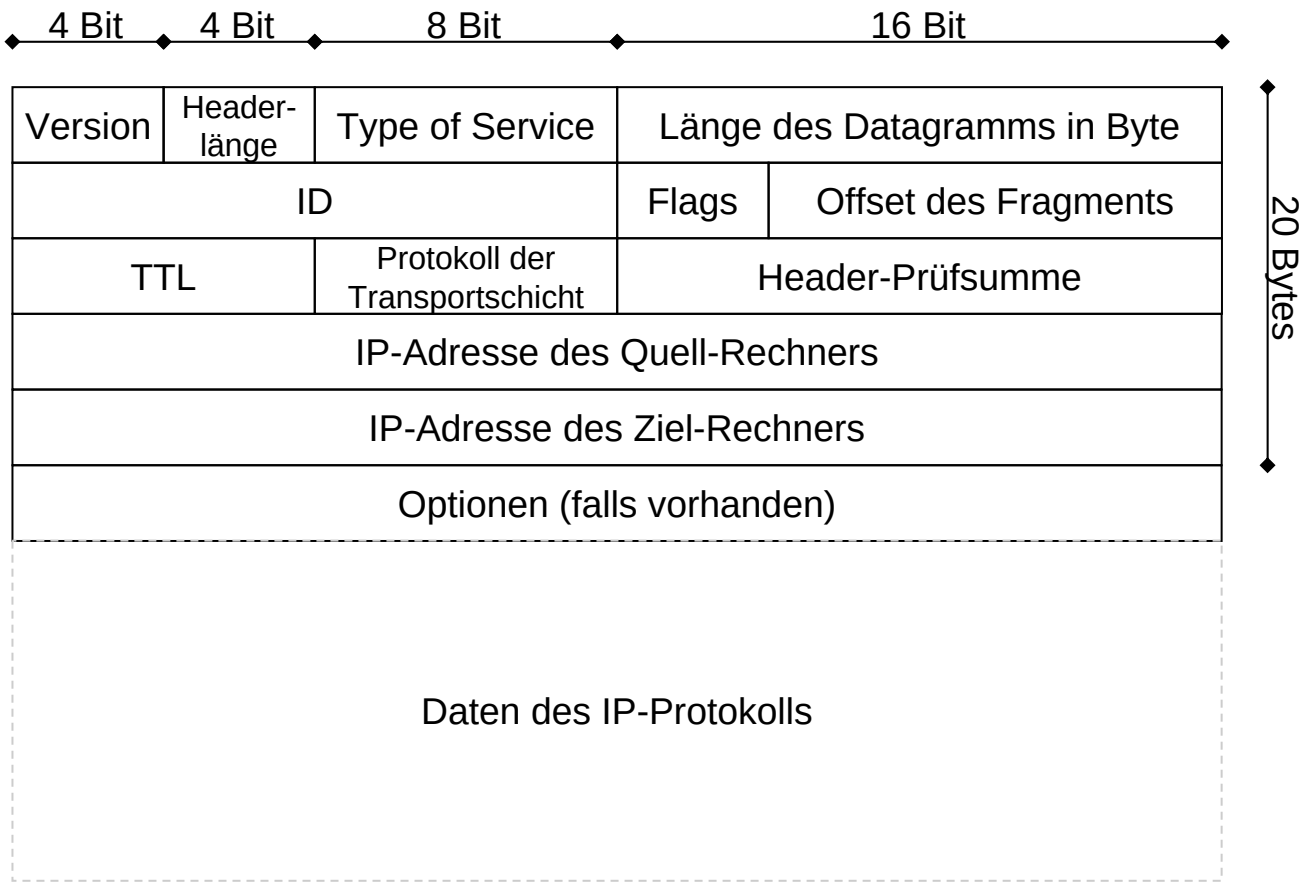
```
     11000000.10101000.11011010.00000000
XOR  11000000.10101000.11011010.00101011
-----------------------------------------
     00000000.00000000.00000000.00101011
     (0.0.0.43) -> Hostadresse
```

IPv4-Netze und reservierte Bereiche

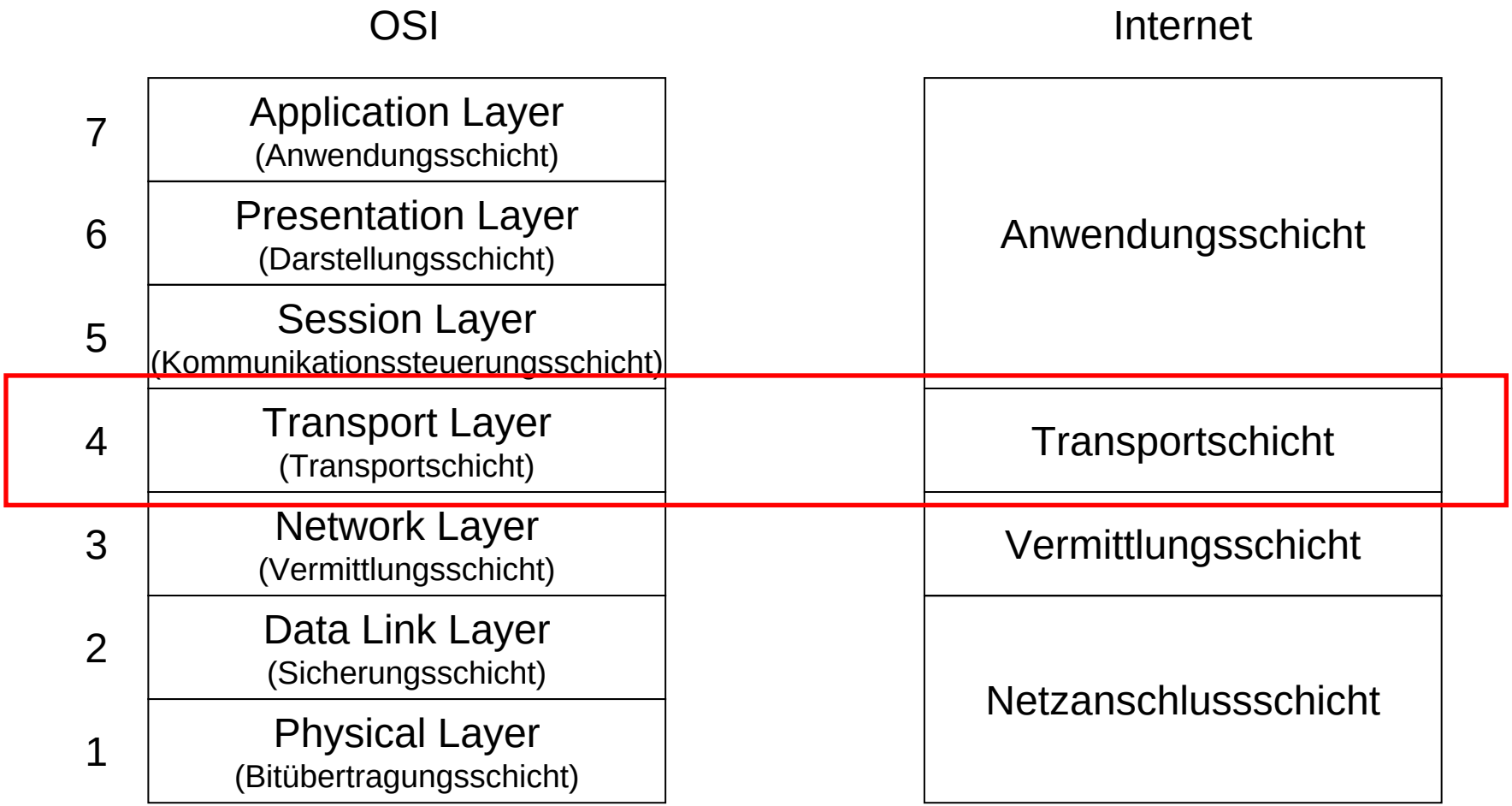
MNM TEAM

Klasse	Netzadresse	Standard Netzmaske	Reservierte Netze	
A	{0…127}.0.0.0	255.0.0.0 bzw. „/8“	0.0.0.0	Default-Route
			10.0.0.0	privater Adressbereich
			127.0.0.1	Loopback-Interface
B	{128…191}.xxx.0.0	255.255.0.0 bzw. „/16“	172.{16…31}.0.0	privater Adressbereich
C	{192…223}.xxx.xxx.0	255.255.255.0 bzw. „/24“	192.168.xxx.0	privater Adressbereich
D, E	{224…255}.xxx.xxx.xxx	spezielle Multicast-Adressen bzw. reserviert für zukünftige Zwecke		

Aufbau des IPv4-Headers



OSI- und Internet-Schichtenmodell



Transmission Control Protocol (TCP)

MNM TEAM

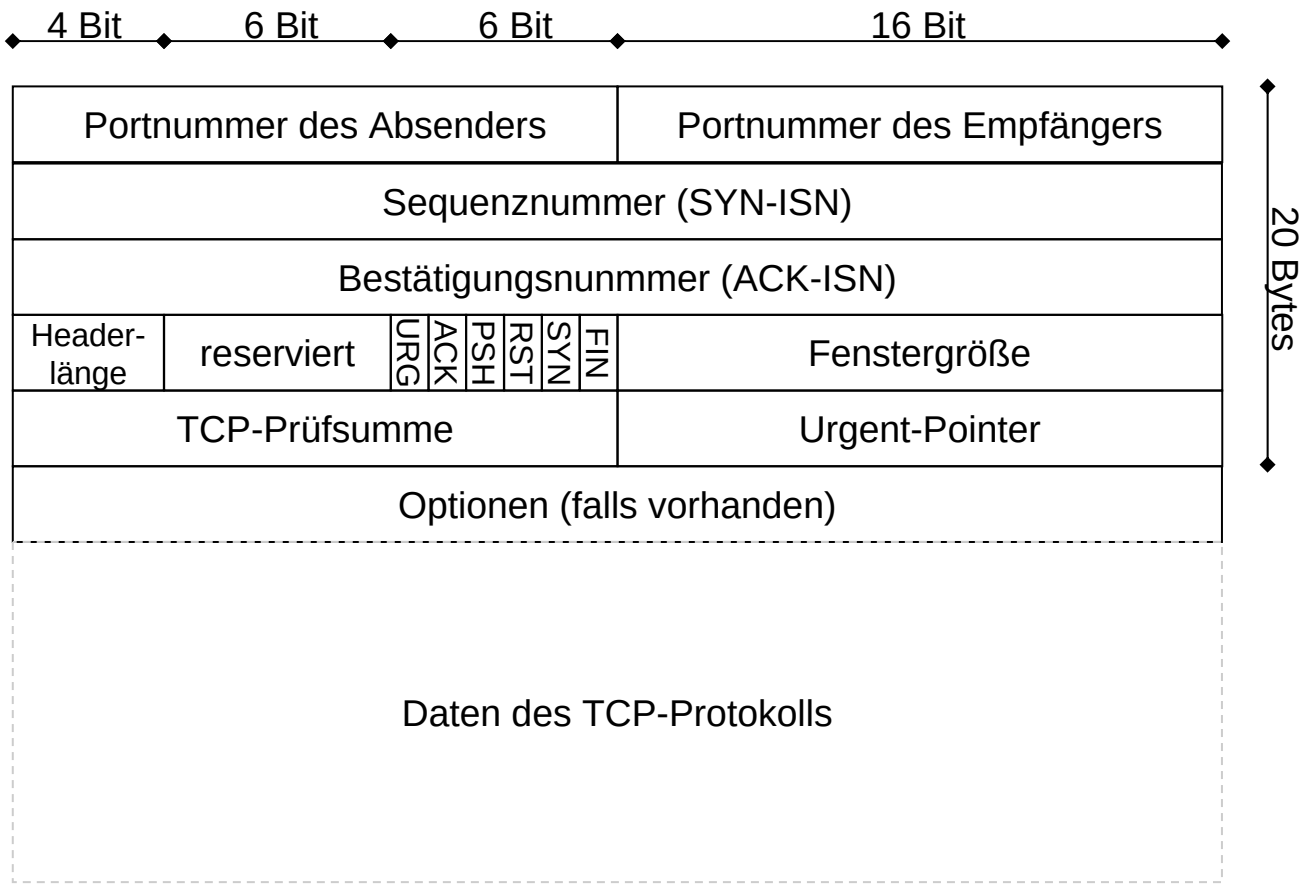
- Protokoll des Transport Layers (Schicht 4)
- Spezifikation
  - Ursprünglich in RFC 793
  - Bugfixes in RFC 1122
  - Erweiterungen in RFC 1323
- IP Protokoll-Nummer: 6
- Zuverlässige Datenübertragung
  - „Positive Acknowledgement with Retransmission“ (PAR)
  - Prüfsumme zur Fehlererkennung
- Reihenfolgesicherung durch „Initial Sequence Numbers“ (ISN)
- Verbindungsorientiert
  - 3-Way Handshake
  - Beidseitiger Verbindungsabbau
- Fair gegenüber anderen Datenströmen
  - Sendefenster-Mechanismus
  - slow start / congestion control
  - Keine weitere Betrachtung
- Es existieren diverse Implementierungen mit unterschiedlichen Charakteristika

TCP-Ports

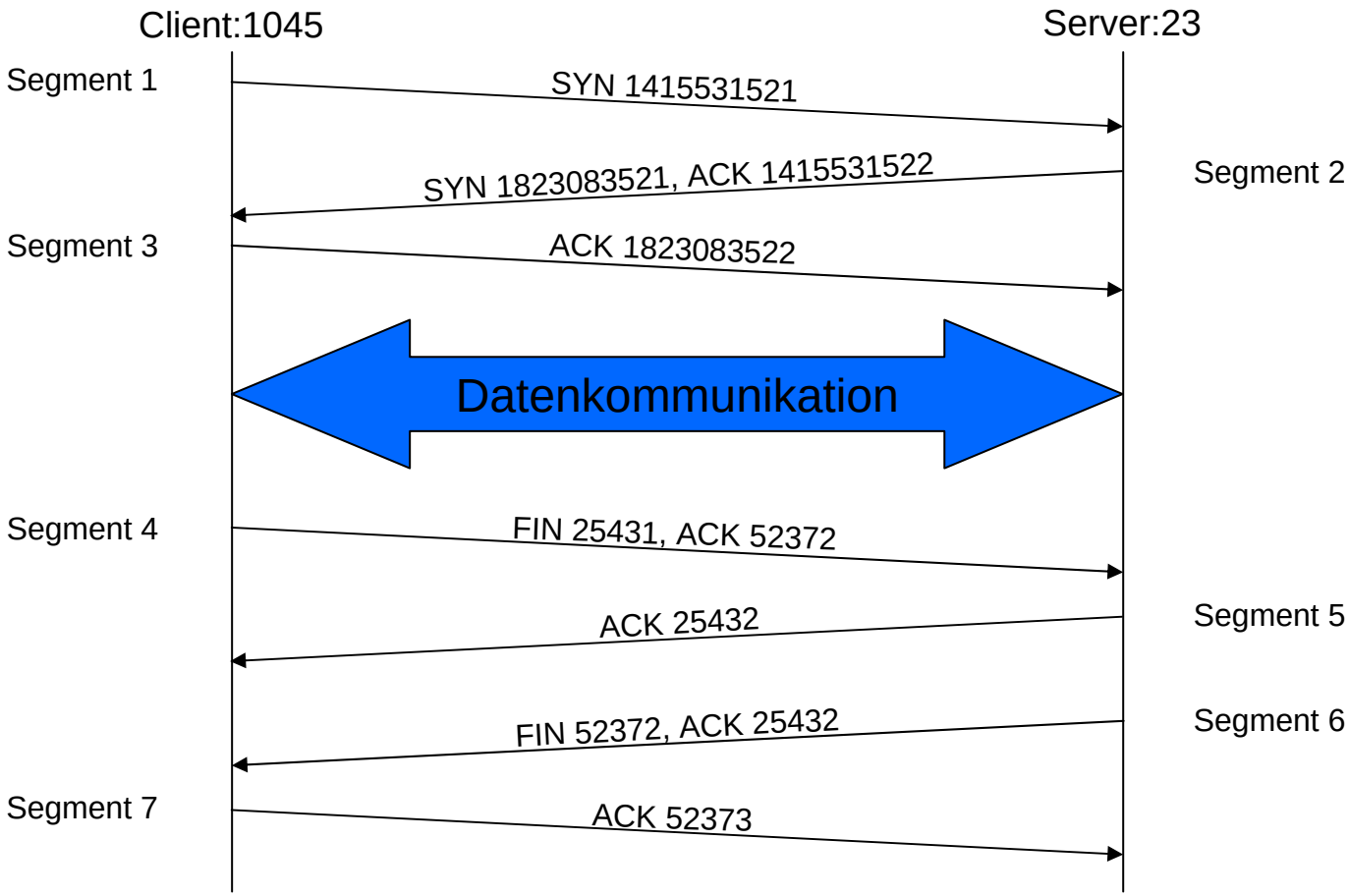
MNM TEAM

- Bietet Multiplexing-/Demultiplexing-Schnittstelle zur Netzebene
- Ports charakteristisch für einige Dienste
- Ursprünglich waren nur Ports <1024 reserviert
- Einige „well known ports“:
  - 20: FTP (data)
  - 21: FTP (control)
  - 22: SSH
  - 23: telnet
  - 25: SMTP
  - 53: DNS
  - 80: HTTP
  - 110: POP-3
  - 137-139: diverse NETBIOS ports
  - 445: Microsoft DS
- Offizielle Liste unter: http://www.iana.org/assignments/port-numbers
- … oder unter Unix: `cat /etc/services`

Aufbau des TCP-Headers



TCP-Verbindungsauf- und -abbau

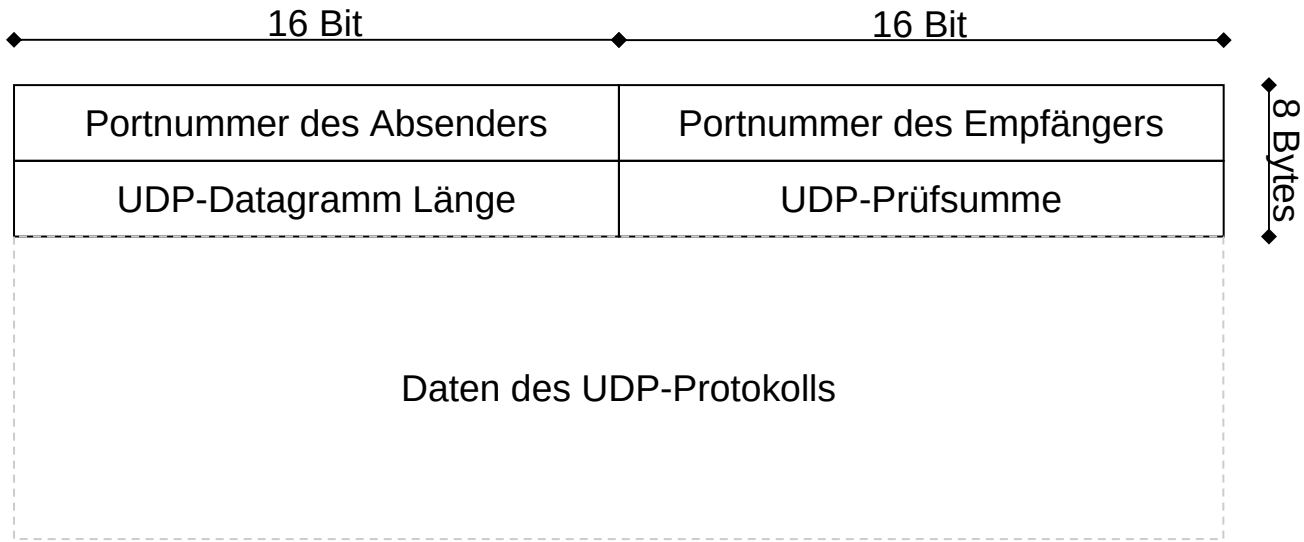


User Datagram Protocol (UDP)

MNM TEAM

- Protokoll des Transport Layers (Schicht 4)
- Spezifiziert in RFC 768
- IP Protokoll-Nummer: 17
- Verbindungslos
  - Kein Verbindungsaufbau
  - Kein Verbindungsabbau
- Nachteile:
  - Nicht reihenfolgegesichert
  - Keine zwingende Fehlererkennung
  - Keine Flusskontrolle bzw. Retransmissions
- Vorteile:
  - Wenig Overhead
  - Einfache Schnittstelle zur Netzwerkebene
  - Multiplexing/Demultiplexing durch Ports

Aufbau des UDP-Headers



- Länge: Angabe in Bytes inkl. 8-Byte Header
- Prüfsumme ist optional!

Sockets (1)

MNM TEAM

- Idee: Informationen zwischen (entfernten) Rechnern/Prozessen austauschen
- Definition: Ein Socket ist ein Kommunikationsendpunkt auf Softwareebene, der eine spezifische Schnittstelle zwischen einem Anwendungsprogramm und dem Transportmedium darstellt.

- Socket-Arten
  - Stream-Socket (`SOCK_STREAM`): Zuverlässiger, verbindungsorientierter Bytestrom mit Sequencing und Fehlerkorrektur
  - Datagramm-Socket (`SOCK_DGRAM`): Unzuverlässige, verbindungslose Paketübertragung ohne Sequencing und ohne Fehlerkorrektur

Sockets (2)

MNM TEAM

- Socket-Adressierung
  - Socket-Domäne: vom Socket verwendete Protokollfamilie → Nur Sockets der gleichen Domäne können miteinander kommunizieren
  - Domänen:
    - `PF_UNIX`, `PF_LOCAL`: Unix-Adresse → Nachteil: keine entfernte Kommunikation
    - `PF_INET`: IP-Adresse (IPv4: 32 Bits) + Portnummer (16 Bits)
    - `PF_INET6`: IP-Adresse (IPv6: 128 Bits) + Portnummer (16 Bits)

- Wir betrachten: Sockets der Internet-Domäne (Internet-Sockets)

Netzprogrammierung (1)

MNM TEAM

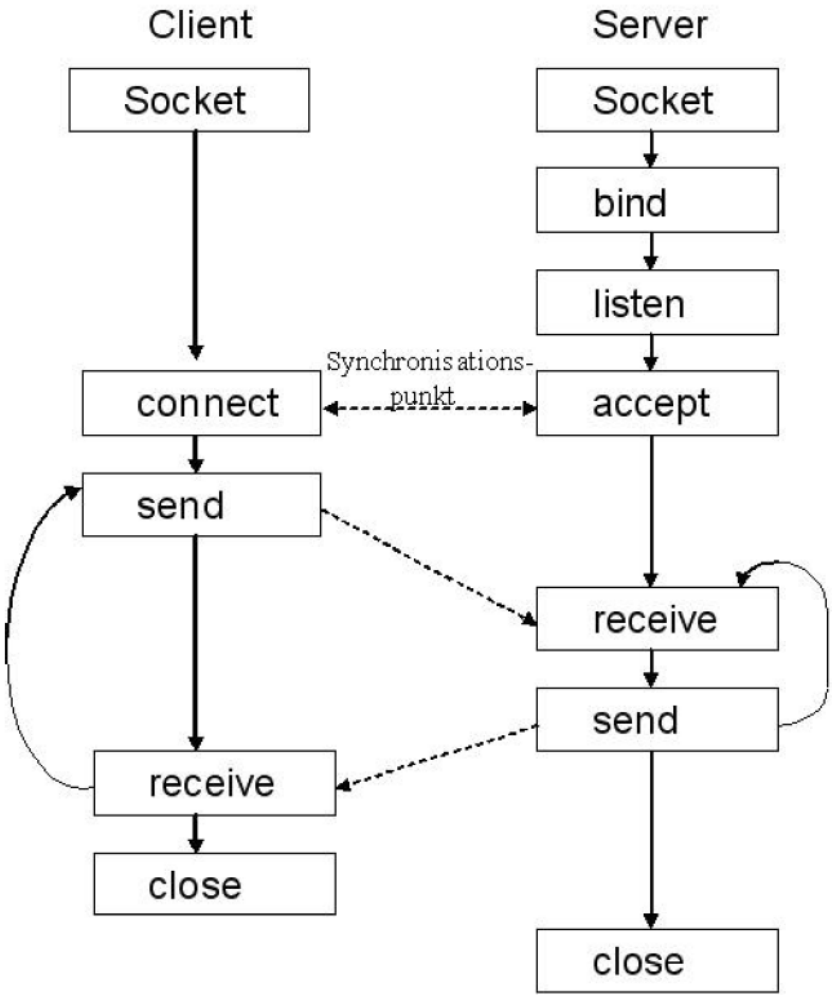
- Die Struktur `struct sockaddr_in`

```
struct sockaddr_in {
   short sin_family;
   unsigned short sin_port;  // Portnummer
   struct in_addr sin_addr;  // Struktur für IP-Adresse
   char sin_zero[8];         // zum Auffüllen der Struktur,
                             // damit alle Socket-Adressen
                             // mind. 16 Bytes groß sind
};

struct in_addr {
   unsigned long saddr;      // IP-Adresse
};
```

  - wird durch Einbinden von `netinet/in.h` verfügbar
- Client/Server-Prinzip
  - Client und Server sind Rollen
  - Server: bietet (mindestens) einen Dienst an, wartet passiv auf eine Anfrage
  - Client: sucht aktiv nach einem Server, dessen Dienst er benötigt

- Systemaufrufe für Sockets



Funktionssignaturen:
siehe `man`-Pages

MNM TEAM

- Beispiel: Iterativer Socket-Server (socksrv.c)

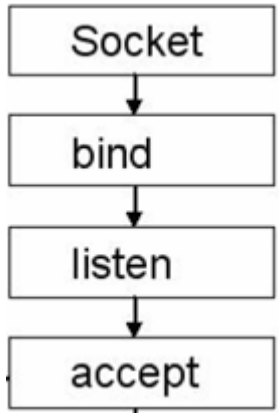
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main() {

   // 1. Socket anlegen
   int sock = socket(PF_INET, SOCK_STREAM, 0);
   // 2. Binden einer Adresse an den Socket
   struct sockaddr_in server;
   server.sin_family = PF_INET;
   server.sin_addr.s_addr = INADDR_ANY;
   server.sin_port = htons(4711);
   bind(sock, (struct sockaddr *) &server, sizeof(server));

   // 3. Verbindungen akzeptieren
   listen(sock, 5);
   // 4. Auf Anfragen warten
   struct sockaddr_in client;
   int fd, client_len;
   client_len = sizeof(client);
   fd = accept(sock, (struct sockaddr *) &client, &client_len);

   return 0;
}
```



Netzprogrammierung (4)

MNM TEAM

- Bemerkungen zum Beispiel
  - Zwei Filedeskriptoren:
    - `sock`: Rendezvous-Deskriptor
      → akzeptiert neue Verbindungen, nicht relevant für Ein-/Ausgabe
    - `fd`: Verbindungsdeskriptor
      → ermöglicht die Kommunikation mit dem Client
  - Systemfunktion `htons()`:
    transformiert die Portnummer in Network-Byte-Order

Zusammenfassung

MNM TEAM

- Das ISO-OSI-Referenzmodell
- Der Internet-Protokollstapel
- Das Verbindungsprotokoll IP
- Adressen und Ports
- Die Transportprotokolle TCP und UDP

- Socket-Arten und -Adressierung
- Systemaufrufe für Sockets