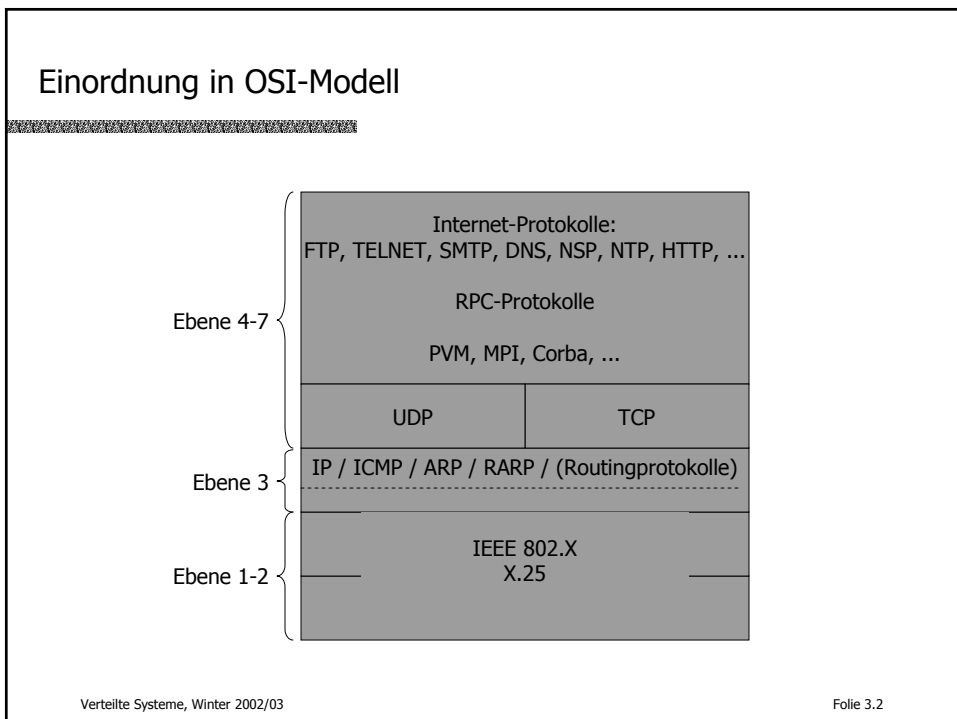


Verteilte Systeme

3. Protokollfamilie TCP/IP



IP-Adressen (IPv4)

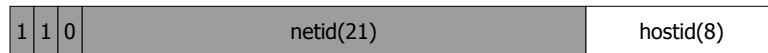
Class A: 2^7-2 Netze mit jeweils bis zu $2^{24}-2$ Rechnern



Class B: $2^{14}-2$ Netze mit jeweils bis zu $2^{16}-2$ Rechnern



Class C: $2^{21}-2$ Netze mit jeweils bis zu 2^8-2 Rechnern



Bit 1

8

16

24

Bit 32

Verteilte Systeme, Winter 2002/03

Folie 3.3

IP-Adressen (cont.)

Schreibweise: „Dotted Decimal“



= 136.199.54.210

Ist ein Class X-Netz?

Netzanteil?

Hostanteil?

Symbolische Namen

- 131.199.54.210 = balvenie.uni-trier.de
- Über verteilte „Datenbank“ (DNS, NSP, LADP, ...)

IP-Netadresse

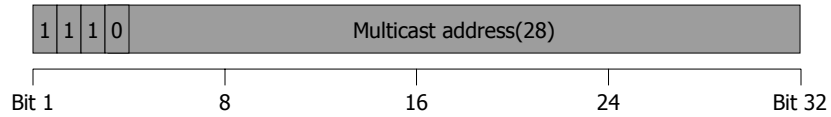
- Hostanteil ist 0

Verteilte Systeme, Winter 2002/03

Folie 3.4

Multicast- und Broadcast-Adressen

Multicast



- Direkte Abbildung bei Multicast-fähigen Netzen
- Flooding-Algorithmen sonst
- Beispiel: Mbone
 - Tunneln bei Unicast-Strecken

Verwaltung der Multicast-Gruppen

- IGMP = Internet Group Management Protocol

Broadcast

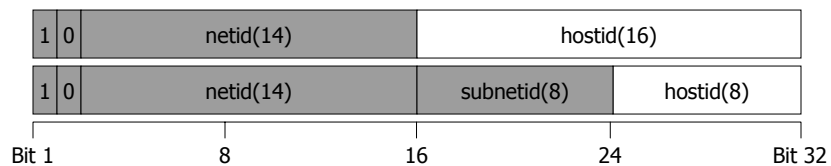
- Nur innerhalb eines Netzes
- Hostanteil alles 1

Subnetze

Maximale Rechnernanzahl in Class-A- und Class-B-Netzen für ein Netz (z.B. Ethernet) zu groß

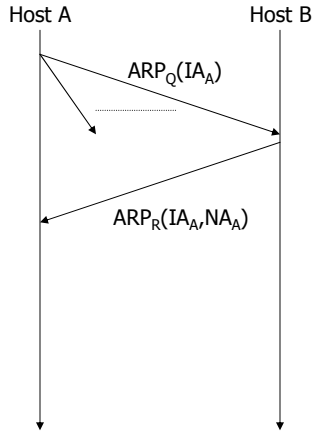
- Class A: 16.777.214 Rechner
- Class B: 65.534 Rechner

Unterteilung in Unternetze



Netzmaske definiert Grenze zwischen Subnetz und Hostanteil

ARP und RARP



ARP = Address Resolution Protocol

- Ermittlung einer Netzwerkadresse im gleichem Netz
 - Gegeben IA
 - Gesucht NA
- Voraussetzung Broadcast-fähiges Netz
 - Bei dedizierten Leitungen Konfigurationssache

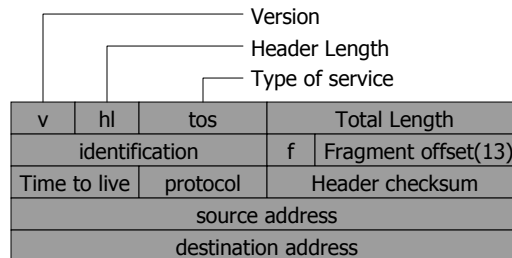
RARP = Reverse ARP

- Gegeben NA, Gesucht IA
- Diskless Clients

Verteilte Systeme, Winter 2002/03

Folie 3.7

IP-Protokoll (IPv4)



20 Byte Header

Maximale Größe IP-Datagramm: 64 Kbyte inkl. Header

Type of Service (tos)

- Hohe Zuverlässigkeit, Hoher Durchsatz, Geringe Latenz, Prioritäten

Fragmentierung

- Identifikation zusammengehörender Fragmente
- Offset des Fragments im IP-Datagramm

Time-to-live

- Hop-Count; wird von Routern dekrementiert

Verteilte Systeme, Winter 2002/03

Folie 3.8

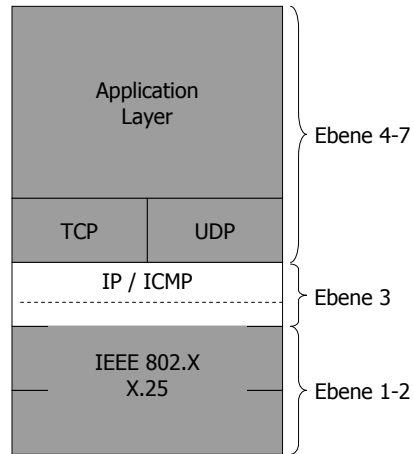
Aufgaben der IP-Netzwerkebene

Übertragung von
IP-Datagrammen

- Routing
- Fragmentierung und Assembly

Netzverwaltung und Melden
von Fehlern

- ICMP



Verteilte Systeme, Winter 2002/03

Folie 3.9

IP: Routing

„Normale“ Rechner

- Empfänger im selben Netz
ARP falls notwendig
Direkt Senden
- Empfänger in anderem Netz
Nachricht an
„Default Router“

Router

- Routingtabelle durchsuchen
Host-Match
Network-Match
Default Router
- An ermittelte NA weiterleiten
- „Routing Mechanism“

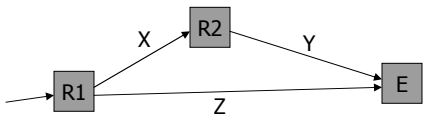
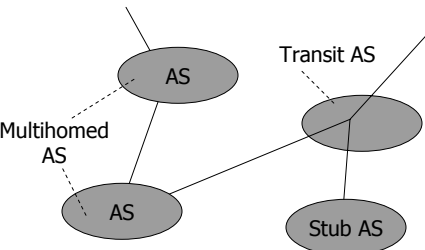


Routing Policy meist durch zusätzliche
Protokolle (Applikationsebene)

Verteilte Systeme, Winter 2002/03

Folie 3.10

Routing-Protokolle

Autonome Systeme (AS)

- Verwaltungseinheiten

Interior Gateway Protocols

- Distanzvektor-basiert
 - RIP (Routing Information Protocol)
 - RIP-2
- Link-state-basiert
 - OSPF (Open Shortest Path First)

Exterior Gateway Protocols

- EGP
- BGP (Border Gateway Protocol)

Größe der Routingtabellen

- CIDR (Classless Interdomain Routing)

Verteilte Systeme, Winter 2002/03
Folie 3.11

IP: Fragmentierung und Assembly

Intranet-Fragmentierung

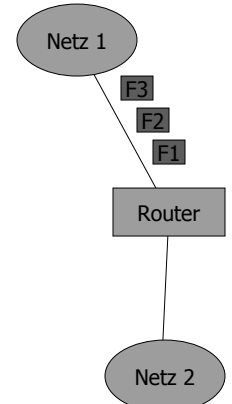
- An jedem Hop Assembly und anschließende Fragmentierung für ausgehende MTU
- Vorteil: Nutzung volle MTU
- Nachteil: Zusätzliche Kosten
- Problem: Fragmentverluste

Internet-Fragmentierung

- Nur Fragmentierung an Hops mit kleiner werdender MTU
- Vor- und Nachteile?

MTU-Path-Discovery

- Kleinste MTU auf dem Weg zum Empfänger bestimmen und entsprechend fragmentieren
- Wann sinnvoll?



Verteilte Systeme, Winter 2002/03
Folie 3.12

ICMP = Internet Control Message Protocol

Echo (Ping)

- request
- reply

Destination Unreachable

- network unreachable
- host unreachable
- protocol unreachable
- port unreachable
- fragmentation needed but don't fragment bit set
- source route failed
- destination network unknown
- destination host unknown
- destination administratively prohibited
- network/host unreachable for TOS
- ...

Source Quench

Redirect

- redirect for network
- redirect for host
- redirect for TOS and network
- redirect for TOS and host

Router

- advertisement
- solicitation

Time exceeded

Anfragen

- timestamp request
- information request
- address mask request

Verteilte Systeme, Winter 2002/03

Folie 3.13

UDP

Source port	Destination port
UDP length	UDP checksum

UDP = User Datagram Protocol

- End-to-End-Version von IP (Portnummer)

Best-Effort

- Nachrichtenverluste
- Keine Ordnung
- Keine Flußkontrolle

8 Byte Header

Maximale Datagramm-Länge: 64 Kbyte inkl. Header

UDP-Multicast möglich

- Angabe einer Multicast-IP-Adresse als Empfänger
- Meist von Sysadmins ausgeschaltet

Verteilte Systeme, Winter 2002/03

Folie 3.14

TCP

source port		destination port	
Sequence number			
hl		flags	Window size
checksum		Urgent pointer	

TCP = Transmission Control Protocol

Reliable Stream

- Verbindungsorientiert
- Flußkontrolle (Sliding-Window)
- Duplex

Segmente

- Maximale Datenmenge die in einem Schritt übertragen wird
- Maximale Segmentgröße (MSS) „handeln“ Sender und Empfänger aus

Socketschnittstelle in UNIX

Socket = Kommunikationsendpunkt (Ebene 4)

UDP- oder TCP-Adresse:

```
struct sockaddr_in {
    u_short sin_family;
    u_short sin_port;
    u_long sin_addr;
}

sin_family = Protokollfamilie (TCP/IP = PF_INET)
sin_addr = IP-Adresse
sin_port = Portnummer
```

Erzeugen und Schließen von Sockets vergleichbar
Dateideskriptor

Zugriffsfunktionen

- UDP: `sendto` und `recvfrom`
- TCP: Einbettung in Dateischnittstelle (`read` und `write`)

Sockets erzeugen und schließen

Erzeugung:

```
int sock;
sock = socket (PF_INET, service, 0);
    UDP: service = SOCK_DGRAM
    TCP: service = SOCK_STREAM
– Es wird noch keine Portnummer vergeben
```

Schließen:

```
close (sock);
```



Verteilte Systeme, Winter 2002/03

Folie 3.17

Portnummer zuweisen

Funktion bind:

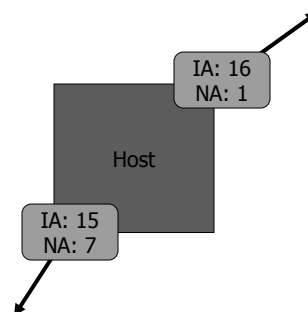
```
int bind (
    int socket,
    sockaddr_in *addr,
    int addr_len
)
```

Aufrufvarianten

- Portnummer:
 - Transient: port = 0
 - Fest (z.B. ftp-Server TCP-Port 21): port = <Portnummer>
- Netz-Interface:
 - Bestimmtes Interface: sin_addr = <IP-Adresse>
 - An allen angeschlossenen Interfaces: sin_addr = INADDR_ANY

Zahlen müssen im Netzwerkstandard angegeben werden

- Konvertierungsroutinen: htonl, htons, ...



Verteilte Systeme, Winter 2002/03

Folie 3.18

Beispiel: TCP-Port 21 erzeugen

```
int s;
struct sockaddr_in saddr;
int ret;

s = socket(AF_INET, SOCK_STREAM, 0);
bzero(&saddr, sizeof saddr);
saddr.sin_family = AF_INET;
saddr.sin_addr.s_addr = INADDR_ANY;
saddr.sin_port = htons(21);
ret = bind(s, &saddr, sizeof(saddr));
assert(ret != -1);

...
```

Verteilte Systeme, Winter 2002/03

Folie 3.19

UDP-basierte Kommunikation

Senden eines Datagramms:

```
retcode = sendto (
    int socket,
    char *msg, int msg_len,
    int flags,
    sockaddr_in *recv, int recv_len
)
```

Empfangen eines Datagramms:

```
rectcode = recvfrom (
    int socket,
    char *msgbuf, int msgbuf_len,
    int flags,
    sockaddr_in *sender, int *sender_len
)
```

Rückgabewert ist die tatsächliche Datagrammlänge

Verteilte Systeme, Winter 2002/03

Folie 3.20

TCP-basierte Kommunikation

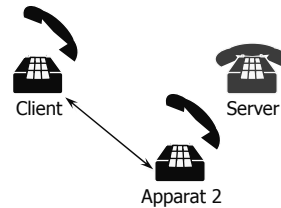
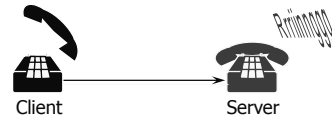
Asymmetrischer Verbindungsaufbau

Passive Seite (Server):

1. Socket erzeugen
2. Socket an Portnummer binden
3. Initialisierung der Verbindungsannahme (`listen`)
4. Auf Verbindungswunsch warten (`accept`)

Aktive Seite (Client):

1. Socket erzeugen
2. Server ermitteln (IP-Adresse, Portnummer)
3. Verbindung aufbauen (`connect`)



Verteilte Systeme, Winter 2002/03

Folie 3.21

Listen, Accept und Connect

Server-Seite

Initialisierung des passiven Wartens:

```
- retcode = listen (
    int socket,
    int queue_length
)
```

Socket wird passiv

Auf Verbindungswünsche warten:

```
- retcode = accept (
    int socket,
    sockaddr_in *client,
    int *client_len
)
```

Client-Seite

Verbindungswunsch herstellen:

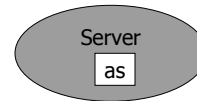
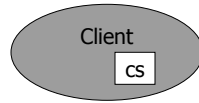
```
- retcode = connect (
    int socket,
    sockaddr_in *server,
    int server_len
)
```

Verteilte Systeme, Winter 2002/03

Folie 3.22

Ablauf eines Verbindungsaufbaus

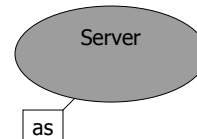
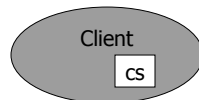
(1) Sockets erzeugen



```
cs = socket(..., SOCK_STREAM, 0);    as = socket(..., SOCK_STREAM, 0);
```

Ablauf eines Verbindungsaufbaus

(2) Server-Adresse (Portnummer) festlegen



```
cs = socket(..., SOCK_STREAM, 0);    as = socket(..., SOCK_STREAM, 0);  
bind(as, server_port);
```

Ablauf eines Verbindungsaufbaus

(3) Server-Socket wird passiv



```
cs = socket(...,SOCK_STREAM,0);    as = socket(..., SOCK_STREAM,0);
                                     bind(as,server_port);
                                     listen(as,4);
```

Ablauf eines Verbindungsaufbaus

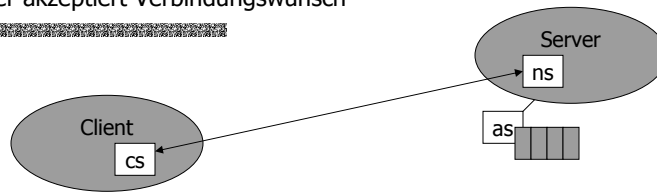
(4) Verbindungswunsch des Clients



```
cs = socket(...,SOCK_STREAM,0);    as = socket(..., SOCK_STREAM,0);
                                     bind(as,server_port);
                                     listen(as,4);
connect(cs,server(host,port));
```

Ablauf eines Verbindungsaufbaus

(5) Server akzeptiert Verbindungswunsch



```

cs = socket(...,SOCK_STREAM,0);
connect(cs,server(host,port));

as = socket(...,SOCK_STREAM,0);
bind(as,server_port);
listen(as,4);
ns = accept(ss,...);

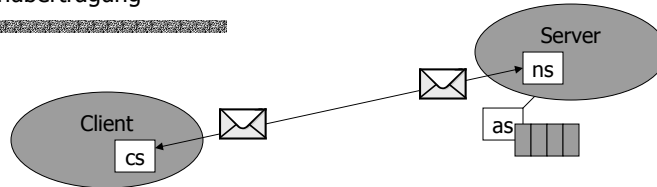
```

Verteilte Systeme, Winter 2002/03

Folie 3.27

Ablauf eines Verbindungsaufbaus

(6) Datenübertragung



```

cs = socket(... ,SOCK_STREAM,0);
connect(cs,server(host,port));
write(cs,...);
write(cs,...);

as = socket(... ,SOCK_STREAM,0);
bind(as,server_port);
listen(as,4);
ns = accept(ss,...);
read(ns,...);

```

Verteilte Systeme, Winter 2002/03

Folie 3.28

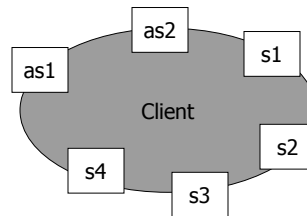
Nicht-selektives Warten

Auf verschiedene Ereignisse blockierend Warten

- Empfangs- und sendebereite Sockets
- Lese- und schreibbereite Dateien

Funktion **select**:

```
int sock1, sock2;
fd_set fds;
sock1 = socket(...);
sock2 = socket(...);
FD_ZERO(&fds);
FD_SET(sock1, &fds);
FD_SET(sock2, &fds);
ret = select(FD_SETSIZE, &fds, ..., timeout);
...
if (FD_ISSET(sock1, &fds) { ... });
```



Zugegeben, die Funktion ist etwas komplex :-)

Windows und TCP/IP

Aller Anfang ist schwer

Seit Windows 95 winsock

- Explizite Initialisierung des Protokoll-Stacks
- Alle gängigen Funktionen anschließend verfügbar

Ab .NET

- Klassen im .NET Framework
 - Direkter Zugriff auf Socket-Klassen
 - Cooked: TcpClient, TcpListener, ...
- Wegen CLR in allen „managed“ Sprachen verfügbar
- Winsock weiterhin vorhanden

Beispiel?

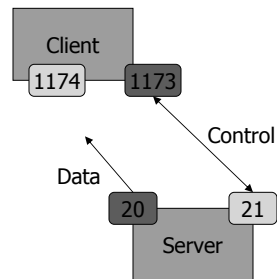
FTP

File Transfer Protocol

- TCP-basierte Kommunikation, Port 21
- Unterstützt beschränkte Menge an Dateitypen (ASCII, Binär, ...) und Dateistrukturen (Byte Stream, Records)

Zwei TCP-Verbindungen

- Kontrollverbindung
 - FTP-Server passiv
 - FTP-Client aktiv
- Datenverbindung
 - Client öffnet passiven Socket
 - IP-Adresse und Portnummer wird über die Kontrollverbindung an Server übertragen
 - Server stellt Verbindung her



Verteilte Systeme, Winter 2002/03

Folie 3.31

Literatur

D.E. Comer, D.L. Stevens
Internetworking with TCP/IP, Volume I-III
 Prentice-Hall, 1993

W. Stevens
TCP/IP Illustrated, Volume 1: The Protocols (1994)
TCP/IP Illustrated, Volume 3: T/TCP, HTTP, ... (1996)
 Addison-Wesley

G. Wright, W. Stevens
TCP/IP Illustrated, Volume 2: The Implementation
 Addison-Wesley, 1995

Verteilte Systeme, Winter 2002/03

Folie 3.32