

HTW SAARBRÜCKEN

# Rechnernetze I

---

*Skript zur Vorlesung von Dipl.-Ing. Wolfgang  
Pauly*

Sarah Theobald

1.1 Entwicklung des Internets .....	8
Phase 1: Experimentierphase .....	8
Phase 2: Skalierung .....	8
Phase 3: Universelle globale Anwendung .....	8
Phase 4: Allgegenwertiges Internet .....	9
1.2 Internetorganisationen und Gremien .....	9
1.3 RFC (request for comments) ( <a href="http://www.rfc-editor.org">www.rfc-editor.org</a> ) .....	9
1.3.1 Was ist ein RFC? .....	9
1.4 Das OSI – 7 – Schichten Referenzmodell der Computerkommunikation.....	10
1.4.1 Was ist ein Protokoll? .....	10
1.4.2 Was ist ein Netzwerk – Protokoll?.....	11
1.4.3 Die Eigenschaften / Funktionsweisen einer Protokoll – SW - Schicht in 5 Aussagen.....	11
1.4.3.1 Beispiele für Steuerverwaltungsinformationen .....	11
1.4.4 Aufbau der Schichten .....	11
Schicht 1: Bitübertragungsschicht = physical Layer .....	11
Schicht 2: Sicherungsschicht = Data Link Layer .....	12
Schicht 3: Vermittlungsschicht = Network Layer .....	12
Schicht 4: Transportschicht = Transport Layer .....	12
Schicht 5: Sitzungsschicht = Session Layer .....	12
Schicht 6: Darstellungsschicht = Presentation Layer .....	12
Schicht 7: Anwendungsschicht = Application Layer.....	13
2.1 Das Paket – Konzept .....	13
2.2 Netztopologien (Netzaufbauten) .....	14
2.2.1 Punkt – zu – Punkt – Netzwerk = Maschennetzwerk .....	14
2.2.2 LAN – Topologien .....	15
2.2.2.1 Stern – Topologie .....	15
2.2.2.2 Ring – Topologie .....	16
2.2.2.3 Bus – Topologie .....	16
Standardisierungsgremien .....	17
2.3 Busnetz: Ethernet 2 .....	18
2.3.1 Entwicklung .....	18
2.3.2 Ethernet.....	18
2.3.2.1 Zuweisung einer MAC – Adresse .....	21
a) statische Zuweisung .....	21

b) statisch konfigurierbare Zuweisung .....	21
c) dynamische Zuweisung .....	22
2.3.2.2 Broadcast – Adresse .....	22
2.3.2.3 Aufbau eines Ethernet – Frames .....	23
2.3.2.4 Aspekte des Ethernet .....	24
2.3.2.4.1 Ethernet Familien .....	24
2.3.2.4.2 Ethernet – Frames .....	24
2.4 kleine Hardwarekunde .....	24
2.5 Link Layer – Sicherungsschicht .....	25
2.6 Protokolle / Technologien .....	25
2.7 PPP – Point to Point – Protocol .....	25
2.7.1 Eigenschaften .....	25
2.7.2 PPP – Rahmen .....	25
2.7.3 Herstellung einer PPP - Verbindung .....	26
2.7.4 PPPoE – PPP over Ethernet .....	26
2.7.5 PPPoE – Paketaufbau .....	26
2.8 Repeater, HUB, Bridge, Switch .....	27
2.8.1 Repeater .....	27
2.8.2 Bridge .....	28
2.8.2.1 Spanning Tree Algorithmus .....	28
2.8.2.2 Spanning – Tree – Protocol – Familie .....	30
2.8.3 HUB .....	30
2.8.4 Switch .....	31
2.9 VLAN = Virtuelle LANs .....	32
2.9.1 Was sind VLANs? .....	32
2.9.2 Vorteile von VLANs an zwei Beispielen .....	33
2.9.3 Wie entstehen VLANs? .....	33
2.9.4 VLAN – Typen .....	34
2.9.4.1 Port Grouping VLAN (Switch – Port basierendes VLAN) .....	34
2.9.4.2 MAC – Layer Grouping VLAN (MAC-Adressen basiertes VLAN) .....	35
2.9.4.3 Network – Layer – Grouping VLAN (protokollbasiertes VLAN) .....	35
2.10 WAN – Technologien und Routing .....	36
2.10.1 Eigenschaften des WAN .....	36
3.1 Protokolle – Protokollstapel / Stackfamilie .....	38

3.2	Internetworks & IP - Adressen .....	38
3.2.1	Das IP - Dienstmodell.....	40
3.2.2	Die IP - Adresse.....	40
3.2.3	IP – Adresshierarchie.....	40
3.2.3.1	IP – Adressen Entwicklungsstufe 1 .....	41
3.2.3.2	Probleme – unvorhergesehene Beschränkungen der Classful – Adressierung.....	42
3.2.3.3	Subnetting .....	42
3.2.3.4	IP - Adressen Entwicklungsstufe 2.....	45
3.2.3.4.1	VSLM (variable length Subnet Mesh).....	45
3.2.3.4.2	Voraussetzungen für VLSM .....	45
3.2.3.5	IP – Entwicklungsstufe 3.....	46
3.2.3.6	Neue Lösung für das IP – Adressen – Problem.....	47
3.2.4	Spezielle IP – Adressen .....	48
3.2.4.1	Netzwerk – Adressen.....	48
3.2.4.2	gerichtete Broadcast – Adresse.....	48
3.2.4.3	begrenzte Broadcast – Adresse.....	48
3.2.4.4	Schleifenadresse – loopback .....	48
3.3	DHCP – Dynamic Host Configuration Protocol.....	49
3.3.1	DHCP – Kommunikationsablauf.....	50
3.4	Multicasting.....	50
3.4.1	IP – Multicast Ethernet – Multicast .....	51
3.4.2	Aufgaben eines Netzwerk - Interfaces .....	51
3.5	Die Verbindung einer IP – UNICAST – Adresse zu einer Ethernet – UNICAST – Adresse .....	52
3.5.1	ARP - Address Resolution Protocol.....	53
3.5.2	Gratuitous – ARP .....	55
Einschub:	Angriffe aus dem lokalen Netz .....	56
3.6	IP – Datagramme, Routing, Frequentierung .....	57
3.6.1	Der Aufbau des IP – Datagramms .....	57
3.6.2	Das Paketformat.....	58
3.6.3	Entstehung eines IP – Datagramms.....	59
3.6.4	Fragmentierungsgrundlagen .....	61
3.6.5	Zusammenbau eines IP – Datagramms (defragmentieren) .....	62
3.7	Das ICMP – Internet Control Message Protocol.....	62
3.7.1	Das ICMP – Datenpaket.....	63

3.7.3	Die wichtigsten ICMP – Nachrichten .....	63
3.7.4	ICMP - Anwendungen.....	64
3.7.5	Das Kommando traceroute (Unix / Linux).....	64
3.7.6	Feststellen der Path – MTU .....	65
3.8	Routing – Begriffe, Verfahren, Protokolle .....	65
3.9	Die Transportschicht und ihre Ende – zu – Ende Protokolle .....	66
3.9.1	UDP - Der Verbindungslose Transportdienst .....	66
EXKURS:	die Begriffe Port und Socket.....	67
	Protokollnummern – Aufteilung .....	67
	Socket .....	68
3.9.1.1	Aufbau eines UDP – Datagramms .....	69
3.9.2	TCP – der zuverlässige Bytestrom – Dienst .....	70
3.9.3	Der TCP – Header .....	71
3.9.3.1	die 6 TCP – Flags je 1 Bit .....	72
3.9.3.2	Grundsätzliches zum Verbindungs- Auf / Abbau .....	73
3.9.3.3	Experiment zu TCP – Verbindungsauf - / abbau.....	74
3.9.3.4	Die TCP - Datenflusskontrolle.....	74
4.1	einfache Standard – Dienste / - Server .....	75
4.2	Das Client – Server – Paradigma .....	76
4.3	Protokolle, Ports & Sockets .....	77
4.4	Dämonen und ihre Meister .....	79
Exkurs:	Sicherheit - tcpwrapper .....	79
4.5	Die Socket – Programmierschnittstelle / Socket - API.....	80
4.5.1	Das Unix Ein - / Ausgabe – Konzept.....	80
4.5.2	Sockets und das Unix Ein - / Ausgabe – Konzept.....	80
4.5.3	Serversoftware – Aufbau.....	80
4.6	Client – Server – Beispiel mit Sockets in C.....	81
4.6.1	Einsatz des Clients mit anderen Servern .....	82
4.6.2	Server über andere Client – Software abfragen.....	82
4.7	Die socket – basierten UNIX – r – utilities .....	82
4.8	RPC – Remote Procedure Call.....	83
4.8.1	Werkzeuge zur RPC - Programmierung .....	83
4.8.2	Die RPC – Paket - / Nachrichtenstruktur .....	84
4.8.3	Informationen zu RPC – Servern abfragen .....	84

4.9	Ein RPC – Client – Server Beispiel .....	84
5.1	Das DNS – Domain Name System / Service .....	85
5.2	Die DNS – Namenshierarchie .....	86
5.3	Das DNS Client – Server – Modell .....	86
5.4	DNS – Server Typen .....	87
5.4.1	primary Server .....	87
5.4.2	secondary Server .....	87
5.4.3	caching – only – Server .....	87
5.5	Elemente des DNS unter UNIX / LINUX .....	87
5.5.1	Client – Konfigurationsdateien .....	87
5.5.2	Server – Konfigurationsdateien .....	88
5.6	Name – Resolution - Prozess .....	88
5.7	DNS – technisch .....	89
5.8	Das DNS – Nachfrageformat .....	90
5.8.1	den DNS „von Hand“ abfragen .....	91
5.8.2	Whois .....	91
5.9	Elektronische Post – E – Mail .....	91
5.9.1	Aufgaben des SMTP – Servers .....	92
5.9.2	Die Mailbox .....	92
5.9.3	Aufbau einer E – Mail - Adresse .....	92
5.9.4	Aufbau einer E – Mail .....	92
5.9.5	Aufbau einer E – Mail – Header – Zeile .....	92
5.9.6	MIME – Header – Zeilen .....	93
5.9.7	POPv3 over SSL .....	94
5.9.8	E – Mail Weiterleitung .....	94
5.10	Spam <-> Ham .....	95
5.11	Greylisting .....	96
5.11.1	Das Vorgehen .....	96
5.11.2	Probleme .....	97
5.12	Fallstricke für Spam – Filter – Betreiber .....	97
5.13	E – Mail – Header Analyse .....	97
5.14	Return –Path .....	98
6.1	Geschichte .....	99
6.2	Adressierung im World Wide Web .....	99

6.3	HTTP .....	100
6.4	HTTP – Kommunikation .....	100

# 1 Was ist INTERNET?

- Ein „Netzwerk von Computernetzen“
- Eine Anhäufung von elektronischen Diensten, die **eine** gemeinsame Kommunikationsinfrastruktur benutzen (TCP/IP)

## 1.1 Entwicklung des Internets

### Phase 1: Experimentierphase

- **1969:** ARPANET
- **1971:** erste E-Mail wird versendet  
7 Bit – ASCII – MIME
- **1973:**
  - „Internetting Project“ DARPA
  - Ethernet wird erfunden
- **1974/75:** Kahn/Cerf beginnen mit der Entwicklung von TCP/IP
- **1980:** ARPANET wird aufgeteilt in:
  - a) MILINET (Militär)
  - b) ARPANET (Universitäten)Parallel gibt es:
  - i) BITNET
  - ii) USENET
  - iii) CSNET
- **01.01.1983 0:00 Uhr:** TCP/IP wird die Protokoll – Familie für das ARPANET

### Phase 2: Skalierung

- **Wichtigster Motor:** UNIX mit TCP/IP
- **1986:** - NSFNET wird in den USA aufgebaut (National Science Networkstation)
  - Regionale Netze bis zum „normalen Menschen“
  - Einführung des DNS (Domain Name System /Service)
  - Gopher/ Wais / Hyper – G...

### Phase 3: Universelle globale Anwendung

- **1991:** Tim Bernnes – Lee **CERN**
- **April 1993:** Freigabe des CERN – http – Service [LYNX]
  - Erster Web – Browser: Mosaic → Netscape
- **Dez. 1993:** HTW bekam Internetanschluss im STL  
= erster experimenteller Web – Server der HTW

Killer – Application  
Web – Browser

- ISOC wird 1993 gegründet
- Wandel vom wissenschaftlich orientierten zum kommerziellen Netz



**Phase 4: Allgegenwertiges Internet**

- IP – Telefonie
- Online Banking
- Diverse Möglichkeiten des Netzzugriffs
  - ➔ neuartige Anwendungen
  - „Das Netz ist der Computer“

**1.2 Internetorganisationen und Gremien**

- a) **Internet Society (ISOC):** ([www.isoc.org](http://www.isoc.org))  
Gesellschaft für die Förderung der Entwicklung und des Wachstums des Internet als globale Kommunikationsinfrastruktur.
- b) Internet Architecture Board ([www.iab.org](http://www.iab.org))
- c) Internet Engineering Task Force ([www.ietf.org](http://www.ietf.org))
- d) Internet Research Task Force ([www.irtf.org](http://www.irtf.org))
- e) Internet Assigned Numbers Authority ([www.iana.org](http://www.iana.org))

**1.3 RFC (request for comments) ([www.rfc-editor.org](http://www.rfc-editor.org))****1.3.1 Was ist ein RFC?**

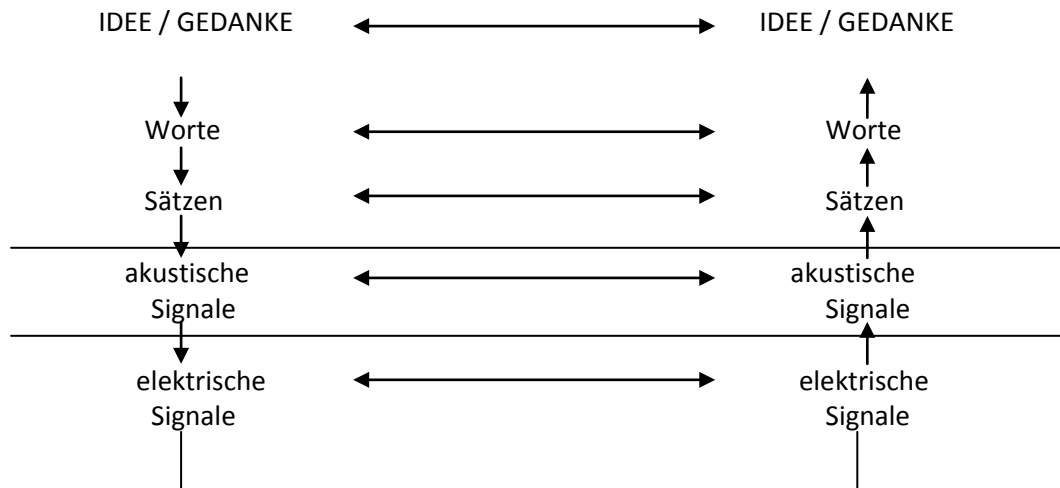
In RFCs werden alle Standards und Entwicklungsphasen der Internetprotokolle festgehalten.

RFC 3300: Internet Official Protocol Standards

<u>RFC</u>	<u>Zustand</u>	<u>Anforderungsstufe</u>
	Standard	Required
	Draft Standard	Recommended
	Proposed Standard	Elective
	Experimental	Limited use
	Informational	Not recommended
	historic	

RFC (1122/1123) – Host Requirements

<u>Protokoll</u>	<u>Zustand</u>	<u>Status</u>
IP	STANDARD	REQUIRED
ICMP	STANDARD	REQUIRED
TELENET	STANDARD	RECOMMENDED
ARP	STANDARD	ELECTIVE

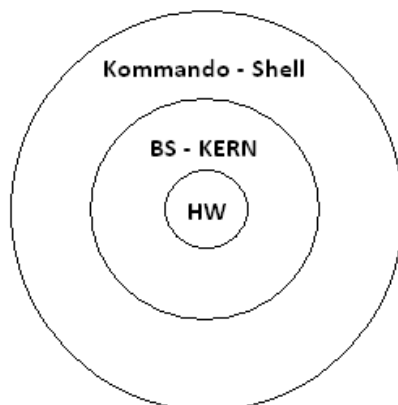


## 1.4 Das OSI – 7 – Schichten Referenzmodell der Computerkommunikation

OSI = Open System Interconnect Reference Model 1977

### Ziel:

Beliebige Anwendungsprogramme (AW – Prg) [Editoren, Compiler, Browser] die auf die beliebigen Rechnerplattform (MS Windows, Linux, UNIX, VMS...) laufen, sollen auf beliebige NW – Ressourcen (Festplatten, Drucker, Informations- Server) genauso zugreifen können wie auf lokale Ressourcen.



Benutzer
Kdo – Shell
BS – Kern
HW

→ POSIX

### 1.4.1 Was ist ein Protokoll?

Ein System von Regeln zum Austauschen von Informationen. Es besteht aus einer Sprache und aus Verhaltensregeln.

**Beispiel:** Pauly – Vorlesung

### 1.4.2 Was ist ein Netzwerk – Protokoll?

Ein NW – Protokoll legt das Format der auszutauschenden Nachrichten, deren Bedeutung (=Sprache) und alle Aktionen, die zur Übermittlung dieser Nachricht nötig sind, fest.

Die Protokoll – Software ist nicht ein Blocksoftware, sondern eine Schicht – Software und man spricht deshalb von Protokoll – Familien oder Protokoll – Suits / Stacks.

### 1.4.3 Die Eigenschaften / Funktionsweisen einer Protokoll – SW - Schicht in 5 Aussagen

- a) jede Schicht löst eine Teilaufgabe oder Teilproblem der Netzwerkkommunikation
- b) jede Schicht fügt beim Suchen der Daten ihre Steuer - / Verwaltungsinformation hinzu und übergibt das resultierende Datenpaket der nächst tieferen Schicht
- c) jede Schicht entfernt beim Empfangen von Daten ihre Steuer- / Verwaltungsinformation und gibt die „nackten“ Daten an die nächst höhere Schicht weiter
- d) jede Schicht benutzt der Dienst der nächst tieferen Schicht und stellt der nächst höheren Schicht ihren Dienst bereit
- e) jede Schicht verarbeitet die gleiche Nachricht wie ihre entfernte / remote Partnerschicht

#### 1.4.3.1 Beispiele für Steuerverwaltungsinformationen

- |    |           |   |            |       |   |
|----|-----------|---|------------|-------|---|
| 1) | Schicht 2 | → | Ethernet 2 | →     | Sender-, Quell - MAC<br>Empfänger-, Ziel – MAC<br>Typfeld |
| 2) | Schicht 3 | → | IP         | →     | IP – Sende / Empfängeradresse<br>Protokollfeld            |
| 3) | Schicht 4 | → | TCP        | _____ | Sende   |
|    |           | → | UDP        | _____ | & Empfänger Port  |

### 1.4.4 Aufbau der Schichten

#### ***Schicht 1:* Bitübertragungsschicht = physical Layer**

- verantwortlich für die Übertragung von einzelnen Bits über ein Medium
- diese Schicht definiert die elektromechanischen und technischen Eigenschaften der Netzkomponenten

Netzkomponenten sind:

- Stecker, Kabel, Bauformen, Belegungen
- Spannungen, Frequenzen, Übertragungsverfahren
- Verkabelungsart (**Topologien**)

**Beispiel:** RS 232

**Schicht 2:    Sicherungsschicht = Data Link Layer**

- das Format des Datenpaketes (**Rahmen** oder **Frames**)
- definiert das Adressierungsschema für Sender und Empfänger
- definiert ein Mediumzugriffsverfahren (CSMA / CD)
- Prüfungssummenberechnung

**Beispiel:** Ethernet 2 (802.2), PPP, PPPOE, SLIP

**Schicht 3:    Vermittlungsschicht = Network Layer**

- erlaubt die Kommunikation zwischen Rechnern, die **nicht** direkt über ein gemeinsames Medium verbunden sind
- erschafft das Internet = virtuelles Computernetzwerk

**Beispiel:** IP, IPX

**Schicht 4:    Transportschicht = Transport Layer**

- entkoppelt die höheren Schichten von den Kommunikationsdetails
- **Datenstrom / Bytestrom (TCP):**  
ist eine Folge von Datenpaketen, die fehlerfrei, vollständig und in der richtigen Reihenfolge beim Empfänger ankommt
- Nachrichten (UDP): **ein Datenpaket**
- Prog – to – Prog – Kommunikation
- Ende – zu - Ende – Verbindungen

**Beispiel:** TCP, UDP

**Schicht 5:    Sitzungsschicht = Session Layer**

- Kommunikations – Steuerungs – Schicht

**Beispiel:** ftp, http

**Schicht 6:    Darstellungsschicht = Presentation Layer**

- ASCII    →    EBCDIC
- Zahlen:    Big Endian                      <->                      Little Endian  
                 125                                      <->                      521  
                  $1 \cdot 10^2 + 2 \cdot 10^1 + 5 \cdot 10^0$                        $5 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2$

**Schicht 7: Anwendungsschicht = Application Layer**

- die Anwendungsprogramme SERVER / CLIENT
  - Web: http, https
  - FTP: ftp
  - Telnet: telnet
  - Mail: SMTP, POP, PoPo, SSL

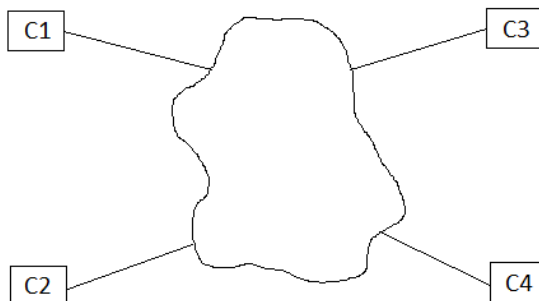
## 2 Grundlagen der LAN / WAN – Technologien

### 2.1 Das Paket – Konzept

Die meisten Rechnernetze sind paketvermittelnde Netze, das heißt sie versenden die von einer Anwendung erzeugten Daten in kleinen Datenblöcken (= **Datenpakete**).

**Frage :Warum Datenpakete?**

- bei gemeinsam benutzter Netzhardware ist „quasi“ parallele Kommunikation möglich
- man hat prompten Zugang zum Netz
- schnelle Fehlerkorrektur möglich
- schnellere Koordination von Sender und Empfänger möglich

**Beispiel 1: Übertragung großer Dateien**

- C1 sendet in einem Block 500 MByte an C4  
bei 5 MBit / s → 800 s □ 13 Minuten
- C2 will 1 kByte an C3 senden  
→ warte 13 min, sende ca. 1,6 ms

**Neue Regel:**

Es dürfen nur 1 kByte große Datenpakete gesendet werden. Nach jeder Sendung wird „neu verhandelt“, wer senden darf.

**aus Regel folgt:**

C1	sendet	1. DP	→ 1,6 ms
C1	sendet	2.DP	
.			
.			
.			
C2	sendet	sein Datenpaket	
C1	sendet	3.DP	

C1 hat eine Gesamtsendezeit von  $\square$  13 Minuten

C2 wartet kurze Zeit, sendet dann 1,6 ms

**Beispiel 2: Übertragungsfehler**

durchschnittliche Fehlerrate von 1:1 Mio. Bit

Es sollen 10 Mio. Bits in einem Stück übertragen werden

→  $\emptyset$  10 Fehlerbits

→ eine Übertragungszeit ist nicht vorhersehbar

1 Mio. Bits →  $\emptyset$  1 Fehler

100.000 Bits →  $\emptyset$  0,1 Fehler = 9 fehlerfreie und 1 fehlerhaftes DP

10.000 Bits →  $\emptyset$  0,01 Fehler

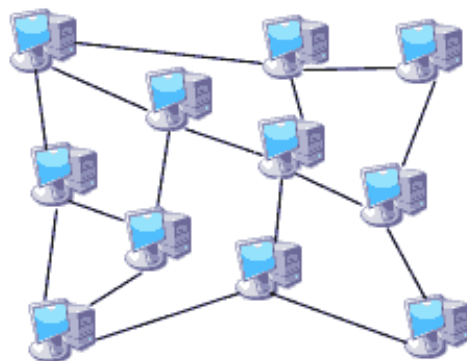
1.000 Bits →  $\emptyset$  0,001 Fehler = 999 fehlerfreie und 1 fehlerhaftes DP

**Reale Fehlerraten:**

- Kupfer: 1 :  $10^6 - 10^7$  Bits
- Licht: 1 :  $10^{12} - 10^{14}$  Bits

**2.2 Netztopologien (Netzaufbauten)****2.2.1 Punkt – zu – Punkt – Netzwerk = Maschennetzwerk**

in den 1960ern wurden Punkt – zu – Punkt –NW aufgebaut (Maschennetzwerke) aufgebaut



**Vorteile:**

- immer gleiche Bandbreite zum Kommunikationspartner
- exklusive Verbindung
- Ausfallsicherheit
- echte parallele Kommunikation

**Nachteile:**

- Vernetzungsaufwand
- für einen zusätzlichen Computer benötigt man  $\frac{N^2 - N}{2}$  Verbindungen
- WAN = Verbindung von Routern

**2.2.2 LAN – Topologien****2.2.2.1 Stern – Topologie**

HUB (Narbe)

- alle Cs sind an einem zentralen Punkt angeschlossen = **HUB**
- der HUB verteilt ein ankommendes Signal an alle Cs

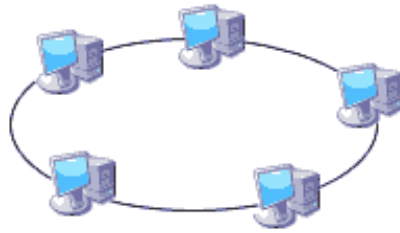
**Vorteile:**

- kappen einer Leitung verursacht keinen kompletten Ausfall des Netzes

**Nachteile:**

- Verkabelungsaufwand

### 2.2.2.2 Ring – Topologie



- jeder C ist mit seinem direkten Nachbarn verbunden

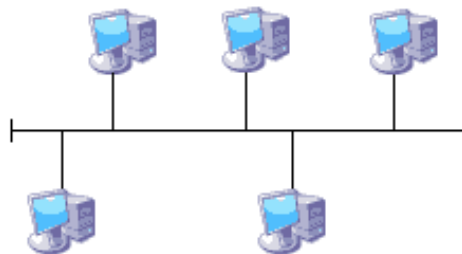
#### Vorteil:

- einfache “Sendekoordination“

#### Nachteil:

- wenn eine Leitung ausfällt oder bricht, kommt es zu einem Komplettausfall

### 2.2.2.3 Bus – Topologie



- alle Cs sind an einem Kabel angeschlossen, immer nur ein C kann Daten senden
- ein Sender belegt das ganze Medium

#### Vorteil:

- Einfache Verkabelung

#### Nachteil:

- Kabelbruch führt zum Komplettausfall

**Beispiel:** Ethernet



## Aspekte von LANs / MANs

(nicht Klausur relevant)

## Standardisierungsgremien

- IEEE: Institute of Electrical and Electronics Engineers
- ISO: International Standard Organization
- ECMA: European Computer Manufacturer Association

## a) Topologien

- Stern → wird benutzt
- Bus
- Ring

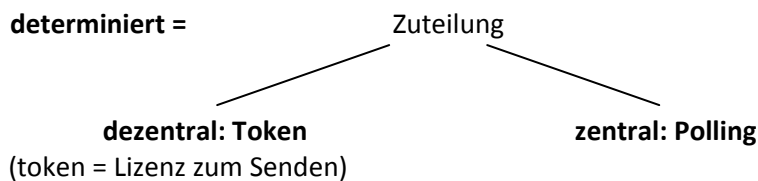
## b) Übertragungsmedien

- Koaxialkabel → Basisband ——— Thickwire  
Carrierband Thinwire  
Broadcast
- Twisted Pair → Shielded  
Unshielded  
Cat 2 ... 7
- Faseroptik → Monomode  
Multimode
- Wireless

## c) Medienzugriff

**stochastisch** = Wettbewerb = konkurrierender Zugriff

- CSMA / CD = Carrier sense with multiple access / Collision Detect
- CSMA / CA = Carrier sense with multiple access / Collision Avoidance

**determiniert =****determiniert = Reservierung**

- TDMA | GSM
- FDMA |
- CDMA | UMTS

## 2.3 Busnetz: Ethernet 2

### 2.3.1 Entwicklung

- entwickelt in den 1970er Jahren von XEROX, DIGITAL EQUIPMENT und INTEL  
→ DIX – Ethernet oder Ethernet 2

#### Thick – Ethernet

- max. 500 m
- $\Delta C$  min 3 m
- 10 MBit / s
- 10 Base 5 = 10 MBit Basisband 500 m

#### Thin – Ethernet (Cheepernet)

- max. 200 m
- 10 MBit / s
- 10 Base 2 = 10 MBit Basisband 200 m

#### weitere:

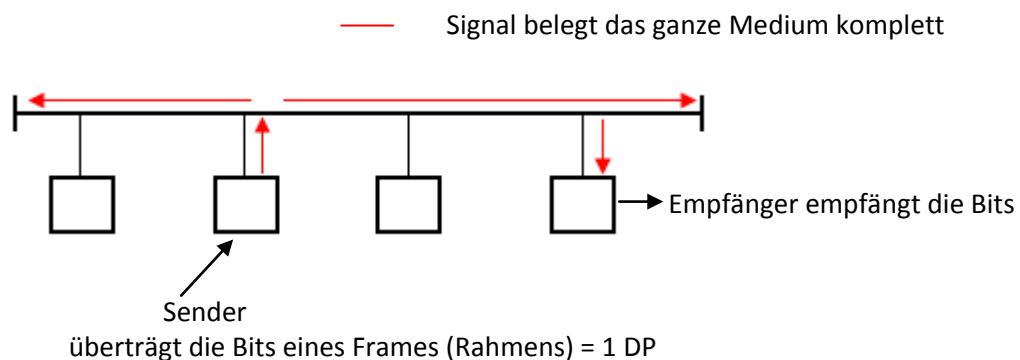
##### Kupfer

- 10 Base T
- 100 Base TX
- 1000 Base T
- 10 G Base T
- 100 G Base T

##### Glasfaser

- 100 Base FX
- 100 Base SX
- 100 Base LX

### 2.3.2 Ethernet



Während der Übertragung eines Frames nutzt ein Computer das gesamte Netz.  
Die anderen Computer müssen warten!

Wie wird die Datenübertragung koordiniert?Ziel:

gleichberechtigter prompter Zugang zum Netz, zum Übertragungsmedium

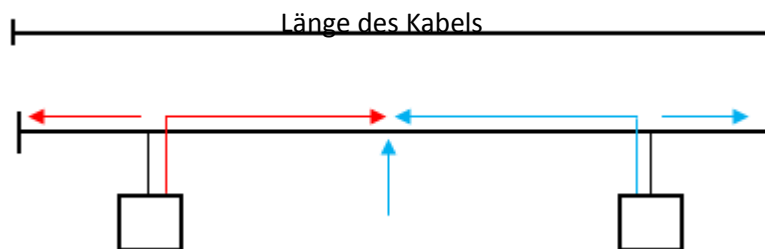
**! Keine zentrale Steuerung !**

das verteilte Koordinierungsschema: **CSMA** (Carrier sense with multiple access (Trägerübertragung))

- ist ein Signal auf dem Medium, dann warte
- ist **kein** Signal auf dem Medium, dann sende

Problem:

Kollision, 2 Computer beginnen gleichzeitig mit dem Datensenden

„gleichzeitig senden“:

die elektrische Signallaufzeit:  $\approx 2 \cdot 10^8 \frac{m}{s}$

Lösung:**CD – Collision Detect**

- jede Netzwerkkarte besitzt einen Sende – und Empfangskanal
- jede Karte überprüft beim Senden, ob Signal auf Sendekanal = Signal auf Empfangskanal

- a) Sind die Signale gleich → **alleiniger Sender**  
 b) Sind die Signale ungleich → **Kollision wurde entdeckt**  
     → **breche eigenes Senden ab, sende Störsignal**  
     → **warte zufällige Zeitspanne vor dem nächsten**  
     **Sendeversuch**

Grundbedingung:

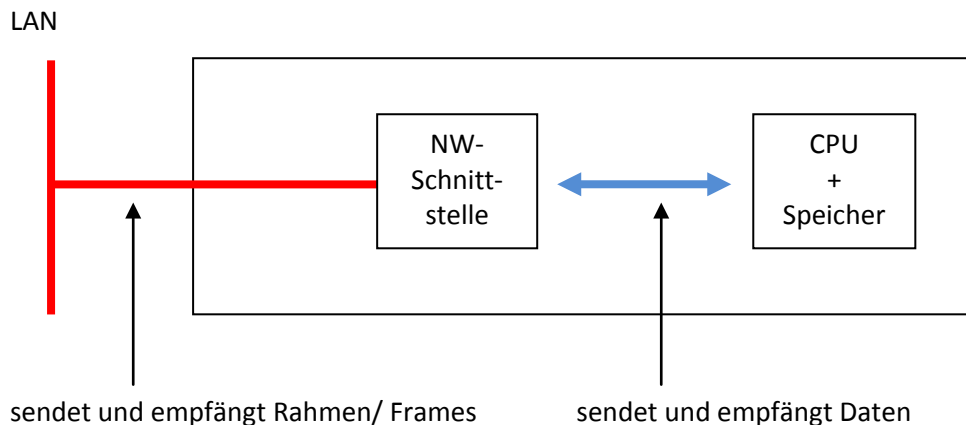
Das zu sendende Datenpaket muss das ganze Medium belegen, nur dann kann CD funktionieren!

10 MBit/s	64 Bytes	512 Bit	51,2 $\mu$ s
100 MBit/s	64 Bytes	512 Bit	5,12 $\mu$ s
1 GBit/s	512 Bytes	4016 Bit	4,016 $\mu$ s

**Erklärung:**

die gesendeten Rahmen / Frames / Datenpakete erreichen **alle** ans Netz angeschlossenen Geräte, normalerweise kommunizieren aber nur 2 Geräte miteinander.

→ die Geräte benötigen eine **eindeutige Adresse**.

**Aufbau eines Geräts:**

jeder Computer besitzt eine eindeutige

- HARDWARE – Address
- LINK – Address
- physical Address
- MAC – Address
- Ethernet – Address

diese Adresse ist ein Bestandteil der Netzwerkschnittstelle und ist **weltweit eindeutig**.

↓  
? sie ist doch nur im LAN relevant!

Herstellermix im LAN möglich!

**Erklärung:**

- die Netzwerkschnittstelle filtert selbstständig alle für sie bestimmten Rahmen aus dem Rahmenstrom des LAN, das heißt **alle Rahmen die ihre MAC – Adresse als Zieladresse führen**
- die CPU wird nur in der Zeit belastet, in der Rahmen ankommen

### 2.3.2.1 Zuweisung einer MAC - Adresse

### a) statische Zuweisung

der Hardware Hersteller vergibt die MAC – Adresse für seine Netzwerkkarten

**Vorteile:**

- einfache Installation (Herstellermix)
- dauerhaft (DHCP → IP)

Wir wissen:

- die Ethernet – Adresse ist 6 Bytes (6x8 Bit) lang

VV : VV : VV : XX : XX : XX

Hersteller – Code

Vendor – Code

- die IEEE vergibt Ethernet – Adressen Blocks an die Hersteller
- ein Block von  $2^{24}$  Adressen kostet ca. \$ 1250

xxxx xxXY : xxxx xxxx : xxxx xxxx

G/L – Bit (global / local)

G/I – Bit (group / individual)

G/I – Bit = 0 → die Adresse bezeichnet eine Station / System  
 = 1 → die Adresse bezeichnet eine logische Gruppe von System (VENDOR – Multicast)  
 setzt der Hersteller

G/L – Bit      = 0      → - das liefert IEEE aus  
                      - die Adresse ist **global** benutzbar  
                      = 1      → kann vom Hersteller als lokale, interne Adresse benutzt werden

### Ethernet - Adresse abfragen:

SOLARIS / LINUX → if config -a

WIN XP → ipconfig /all

jede Netzwerkkarte hat eine statische MAC – Adresse!

**ABER** alle modernen Netzwerkkarten sind im Hinblick auf die MAC – Adresse **konfigurierbar**.

### b) statisch konfigurierbare Zuweisung

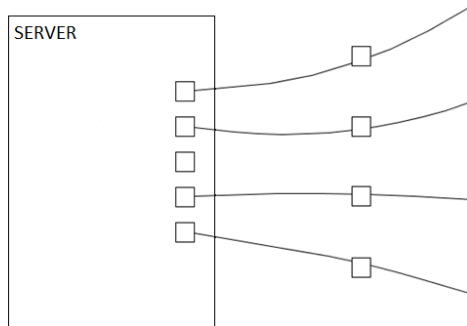
**Vorteile:**

- die Hersteller „müssten“ sich nicht koordinieren
- die Zuordnung zu der IP – Adresse ist einfacher
- HACKER-HILFE

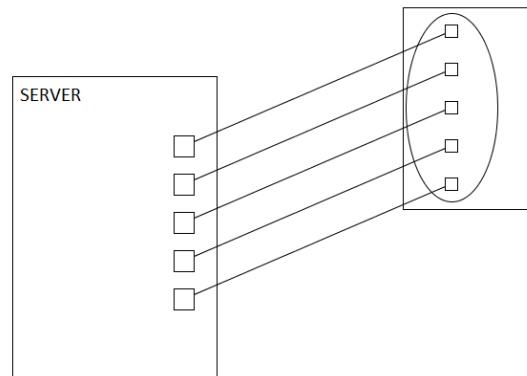
**Nachteile:**

- es kann Adressenkollisionen im LAN geben
- HACKER-HILFE (MAC – Spoofing)

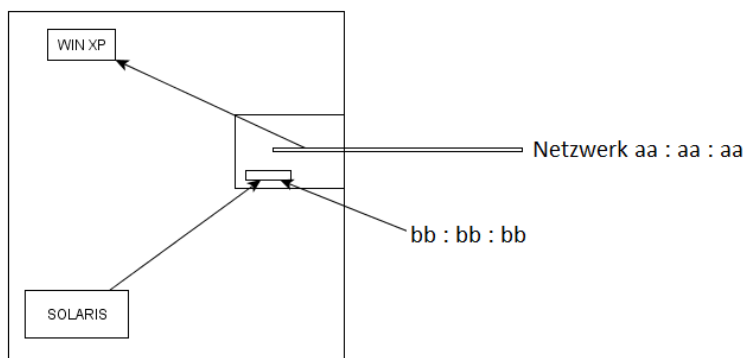
Servermaschinen besitzen oft mehrere Netzwerkkarten, die meist eine MAC – Adresse „auf konfiguriert“ haben



1 Server <-> verschiedene Netze



1 Server <-> 1 Netz mit 4 Pots



verschiedene Betriebssysteme <-> eine Netzwerkkarte  
(Kommando unter XP: get MAC)

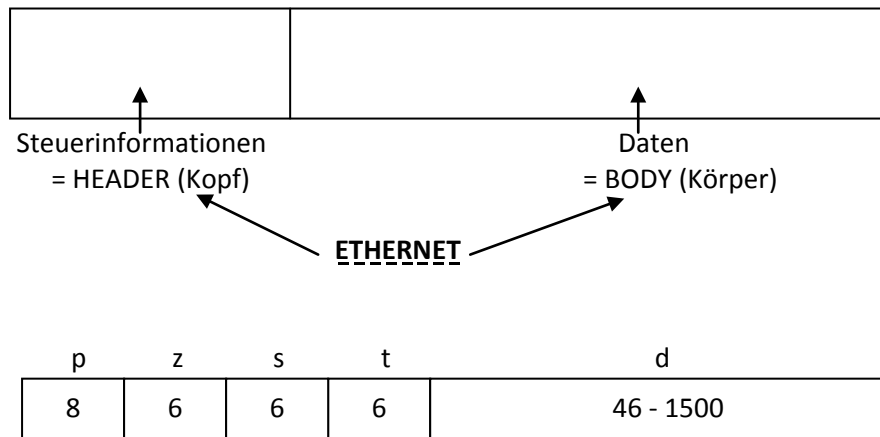
**c) dynamische Zuweisung**

beim Starten fragt der Rechner nach **seiner** MAC - Adresse

**2.3.2.2 Broadcast – Adresse**

- broadcasting → Rundsender (Radio, Fernseher)
- Ethernet Broadcast - Adresse : ff : ff : ff : ff : ff : ff, wobei f = 1111
- ein Rahmen mit der Zieladresse = Broadcast – Adresse wird von allen im LAN angeschlossenen Geräten empfangen und angemeldet
- die Daten werden an die CPU gesendet
- DHCP: Dynamic Host Configuration Protocol
- ARP: Address Resolution Protocol

### 2.3.2.3 Aufbau eines Ethernet – Frames



#### Präambel (p):

- ist 8 Bytes lang:  
7 x 10 10 10 10  
1 x 10 10 10 11
- die empfangende Hardware nutzt das Bitmuster um sich mit dem Signal zu synchronisieren

#### Zieladresse (z):

- die Adresse des Empfängers

#### Sende – oder Quelladresse (s):

- die Adresse des Senders oder der Quelle

#### Typ - Feld (t):

- Rahmen – Typ – Feld
- Protokoll – Typ – Feld
- **beschreibt die ART der im Body transportierten Daten**  
(definiert an welche Software auf der höheren Schicht die Daten weitergegeben werden)

#### Datenarten:

- 0 x 0800 = IP v 4
- 0 x 8138 } = IPX
- 0 x 8137 }
- 0 x 86D0 = IP v 6

#### Datenbereich (d):

- mindestens 46 Bytes → maximal 1500 in Ethernet 2

#### CRC (c):

- Prüfsumme

### 2.3.2.4 Aspekte des Ethernet

#### 2.3.2.4.1 Ethernet Familien

- Ethernet: 1 – 10 MBit/s
- Fast – Ethernet: 100 MBit/s
- Gigabit – Ethernet: 1 GBit/s
- 10 Gigabit – Ethernet: 10 GBit/s

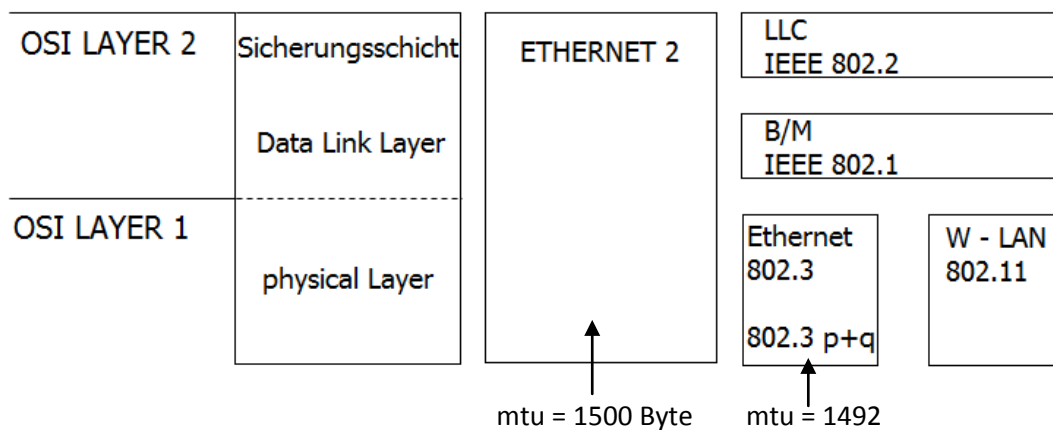
#### 2.3.2.4.2 Ethernet – Frames

##### a) Standard – Frames:

- Ethernet 2 / DIX Ethernet
  - Ethernet IEEE 802.3
- } Host – to – Host

##### b) Tagged – Frames:

- Ethernet IEEE 802.3 p + q (Switch, Router)



- mtu = maximum transport unit
- Ethernet 2: Rechnertypfeld > 1500
- Ethernet IEEE 802.3: Längenfeld < 1500
- path – mtu: f (Netzweg zwischen den Kommunikationspartnern)

##### Beispiel:

RFC 1111

## 2.4 kleine Hardwarekunde

nur in der Vorlesung besprochen!



## 2.5 Link Layer – Sicherungsschicht

- physische Verbindung von Netzwerkgeräten zum LAN
- Daten von Schicht 3 transportieren
- Internet:  $\left( \begin{array}{c} \text{IP - Daten} \\ \text{ARP - Daten} \end{array} \right)$  werden transportiert

## 2.6 Protokolle / Technologien

### a) LAN:

- Ethernet 2 / 802.3
- Token – Ring
- ...

### b) WAN:

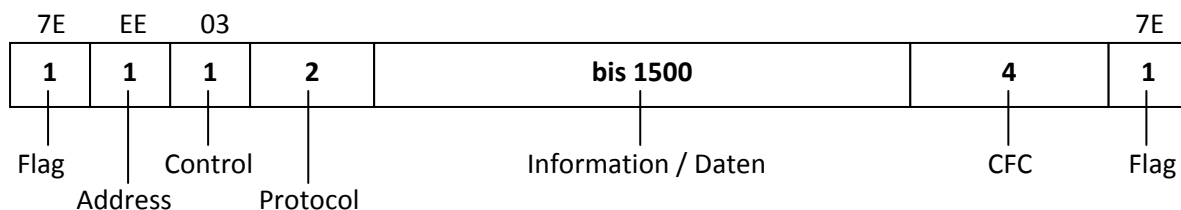
- SLIP
- PPP ← ISDN
- PPOE ← Router / DSL

## 2.7 PPP – Point to Point – Protocol

### 2.7.1 Eigenschaften

- beschrieben in RFC 1661
- ist heute das Standard – Protokoll von HOME – PC zum Internet – Provider bei ANALOG / ISDN Modems
- umfasst 3 Komponenten:
  - a) Die Möglichkeit IP – Pakete zu kapseln (beliebige Schicht 3 Protokolle)
  - b) ein LINK CONTROL PROTOCOL [LCP]
    - beschrieben in RFC 1548
    - es konfiguriert + testet die Verbindung zum Provider
  - c) eine NETWORK CONTROL PROTOCOL [NCP] Familie
    - um die verschiedenen Schicht 3 Protokolle zu transportieren (IP v 4, IPX, Appletalk...)
    - beschrieben in IP – NCP – RFC 1332

### 2.7.2 PPP – Rahmen



- 0 x 0021 → Information = IP v 4 – Dg (Datagram)
- 0 x C021 → Information = LCP – Dg
- 0 x 8021 → Information = IP – NCP – Dg

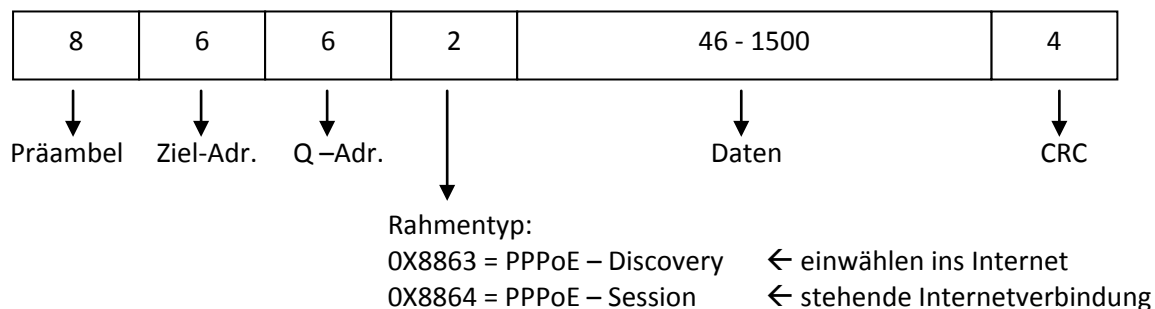
### 2.7.3 Herstellung einer PPP - Verbindung

1. Verbindungsaufbau + Aushandlung der Konfiguration  
→ versenden von LCP – Daten
2. (Optionale) Phase = Bestimmung der Verbindungsqualität  
→ Optimierung der Geschwindigkeit
3. Aushandeln der Konfiguration des Vermittlungsschichtprotokolls ( z. B: IP)  
→ verschickt werden NCP – Daten
4. versenden von z.B. IP – Daten
5. Beenden der Verbindung – NCP – LCP

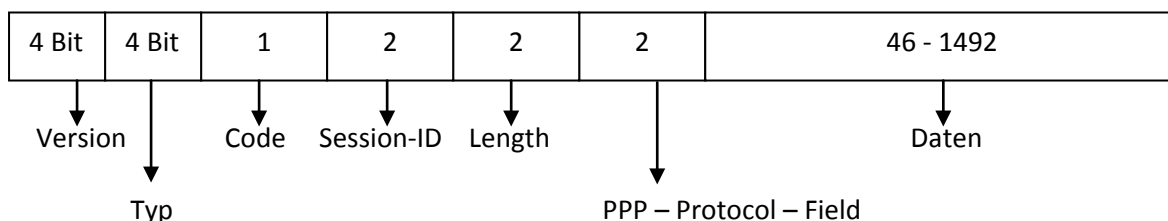
### 2.7.4 PPPoE – PPP over Ethernet

- beschrieben in RFC 2516
- wird in BRD (T – DSL/A – DSL) zwischen HOME – PC und Internet – Provider benutzt

### 2.7.5 PPPoE – Paketaufbau



#### PPPoE - Paket:



#### a) Version:

4 Bit → 0001 = PPPoE v 1

#### b) Typ:

4 Bit → 0001 = PPPoE Typ 1

#### c) Length:

2 Bytes = die Anzahl der Nutzdaten - Bytes

#### d) PPP – Protocol – Field:

2 Bytes = wie bei PPP

-----

**e) Code:**

- 1 Byte
- **0x09 = PADI → PPPoE Active Discovery Initiation**  
wird gesendet, wenn sich ein Internetnutzer via DSL „einwählen“ will
- **Zweck:**  
suchen eines Point – of – Presence (POPs) seines Internetproviders (POP = Einwahlknoten des Internetproviders)
- Ethernet – Ziel – Adresse: Broadcast Address
- Ethernet – Quell – Adresse: Ethernet - Adresse des DSL – Routers
- **0x07 = PADO → PPPoE Active Discovery Offer**  
wird von denen POP gesendet, die ein PADI erhalten haben
- das Antwortpaket enthält:
  - die POP – Ethernet – Address
  - den POP – Namen
  - die POP – Dienstbezeichnung
- Kommen mehrere PADO's am DSL – Router an, so wählt sich der DSL – Router sich einen POP aus
- **0x19 = PADR → PPPoE Active Discovery Request**  
der DSL – Router sendet diesen Code an den von ihm ausgewählten POP
- **0x65 = PADS → PPPoE Active Discovery Session Confirmation**  
der POP bestätigt dem DSL – Router seine Auswahl, vergibt eine Session – ID, die ab sofort beim Datenaustausch angegeben wird  
(vorher war die Session – ID = 0x00)
- 0x00 = Session Data
- **0xd7 = PADT → PPPoE Active Discovery Termination**
  - beendet die “Internetverbindung“
  - kann vom POP oder DSL – Router gesendet werden

**2.8 Repeater, HUB, Bridge, Switch**

Die Größe von traditionellen LANs (LAN – Segmenten) ist beschränkt (10 Base 5 / 2); sie wird außerdem durch die maximale Verzögerungszeit bestimmt (CSMA / CD).

**2.8.1 Repeater**

- Hardware – Gerät (OSI – Layer 1)
- überträgt / verstärkt Signale von einem Segment in alle anderen angeschlossenen Segmente
- ist für die kommunizierenden Computer **transparent**
- maximal 4 Repeater dürfen zwischen 2 Computern liegen, damit CSMA / CD noch funktioniert

**Vorteile / Zweck:**

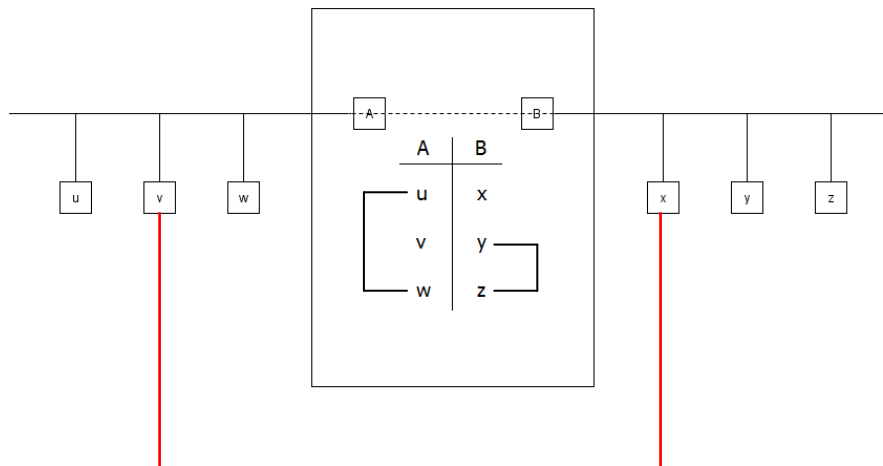
- Vergrößern des LAN

**Nachteile:**

- Repeater bauen **ein** LAN auf
- es kann immer nur **ein** Computer senden; dieser belegt das komplette Netz / LAN
- Repeater übertragen **Signale** : Bits von Ethernet – Frames, Störsignale

## 2.8.2 Bridge

- ist ein einfacher Computer mit CPU und Hauptspeicher
- überträgt nur **vollständige fehlerfreie** Rahmen / Frames  
→ sie ist ein Gerät des OSI Layer 2  
→ Störsignale werden **nicht** weitergeleitet
- Bridges leiten nur die nötigen Rahmen weiter → **Rahmenfilterung**
- eine Bridge lernt welche Rechner an welchem ihrer Ports angeschlossen sind und leitet



- Broadcasts / Multicasts leiten Bridges **immer** weiter
- Qualitätssprung:  
echte parallele Kommunikation in den Teilsegmenten

### 2.8.2.1 Spanning Tree Algorithmus

#### Problem:

zirkulierende Frames durch LAN – Schleifen

#### Entstehung:

ausgelöst durch Broadcast / Multicast – Rahmen, die von den Bridges immer weitergeleitet und „überflüssigen“ Bridges, um die Ausfallsicherheit des Netzes (LAN) zu erhöhen

#### Lösung:

- Radia Perlman → DEC
- Spanning Tree Algorithmus
- die Bridges bauen durch Nachrichtenaustausch ein schleifenfreies Netz auf

#### Wie geht's?

Die Bridges tauschen via BPDU (Bridge Protocol Data Unit) ihre Ausschlussdaten aus und legen jeweils ihre aktiven und passiven Ports fest.

Das ganze kontinuierlich, sodass nach Ausfall einer Bridge nach kurzer Zeit ein neues zyklfreies LAN entsteht.

#### Aktiv:

- ankommende Frames werden weitergeleitet

**Passiv:**

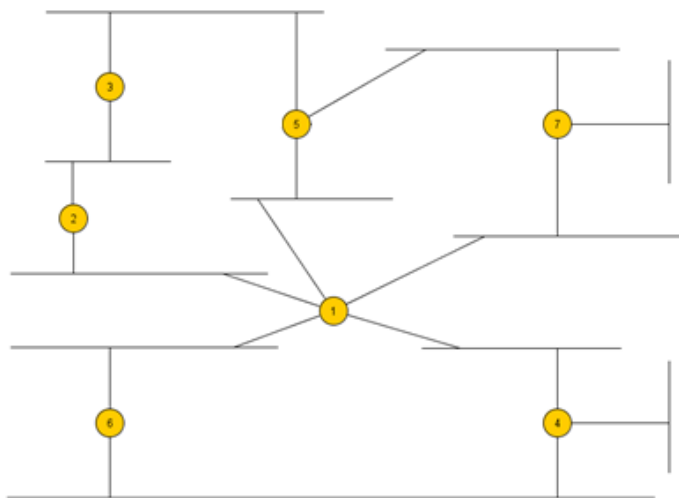
- ankommende Frames werden **nicht** weitergeleitet

**Anschlussdaten:**

- Bridge – Kennung: MAC – Adresse
- Port hat „Qualitätsmerkmal“

**Kurzbeschreibung des ST – (Spanning – Tree) Algorithmus**

- die BPDU – Nachrichten enthalten unter anderem folgende Informationen:
  - die Kennung der Bridge, welche die BPDU – Nachricht sendet (real die MAC – Adresse des Sender – Ports)
  - die Kennung der Bridge, welche die sendende Bridge für die **Wurzel / Root** – Bridge hält
  - die Entfernung der sendenden Bridge von der Root – Bridge
- anfangs hält sich jede Bridge für die Root – Bridge und sendet (RB = ich, E = 0, SB = ich)
- anfangs leiten die Bridges keine Frames weiter, die von Computern kommen, sondern nur BPDU – Pakete
- Alle Bridges erhalten von ihren Nachbarn BPDU – Pakete, die sie auswerten und dann die BPDU – modifizieren:
  1. wenn das ankommende BPDU – Paket eine Wurzel mit einer kleineren Kennung benennt, so wird diese Wurzel als neue Wurzel akzeptiert

**Beispiel:**

B3 → hat bisher gesendet (RB = B3, E = 0, SB = B3)  
 B2 → (RB = B2, E = 0, SB = B2)  
       ↓  
       (RB = B2, E = 1, SB = B2)  
       ↑  
 B1 → (RB = B1, E = 0, SB = B1)  
 B2 → (RB = B1, E = 1, SB = B2)  
 B3 → (RB = B1, E = 2, SB = B3)

2. Sind die Wurzel – oder Rootkennungen gleich, die Entfernungen unterschiedlich, wähle die kürzere Entfernung

**Beispiel:**

B2	→	B3	→	(RB = B1, E = 1, SB = B2)
B5	→	B3	→	(RB = B1, E = 0, SB = B1)
B5			→	(RB = B1, E = 1, SB = B5)

3. Wenn die Root – Kennung und die Entfernung gleich sind, wähle die kleinere Sendebridge aus.

**Beispiel:**

B2	→	B3	→	(RB = B1, E = 1, SB = B2)
B5	→	B3	→	(RB = B1, E = 1, SB = B5)
B3			→	(RB = B1, E = 1, SB = B3)

- die Dauer der Bridgekommunikation bis zur Stabilisierung liegt im Sekundenbereich
- nach der Stabilisierung haben alle Bridges **aktive** und **passive** Ports geschaltet und die Root – Bridge sendet zyklisch ihre Informationen
- fällt eine Bridge aus  
→ Algorithmus beginnt von Neuem

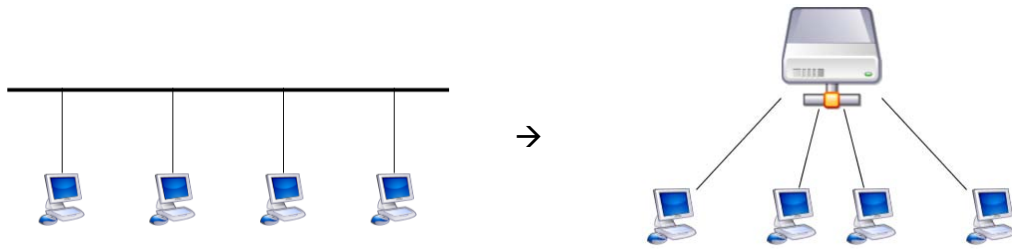
**2.8.2.2 Spanning – Tree – Protocol – Familie**

- beschrieben in IEEE 802.1 d
- Rapid – STP IEEE 802.1w
- Multiple – STP IEEE 802.1s

**2.8.3 HUB**

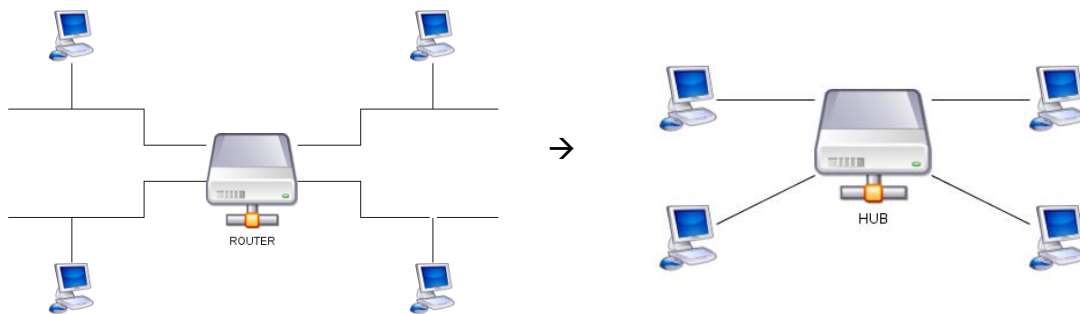
physisch ist HUB und Switch gleich!

- ein HUB ist ein Gerät des OSI Layer 1, arbeitet wie ein Repeater, ist aber zusätzlich ein Kabelkonzentrator
- ein HUB simuliert das gemeinsame Medium, das Kabel, in einem Gehäuse mit vielen Anschlüssen



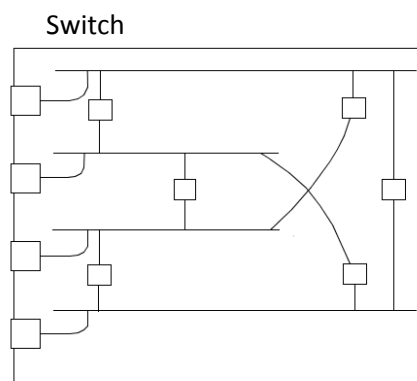
über eine HUB können immer nur 2 Computer gleichzeitig kommunizieren!

- ein HUB ist ein Multiport – Repeater mit einem Computer an einem Port



### 2.8.4 Switch

- ein Switch ist ein Gerät des OSI Layer 2
- ein Switch simuliert ein vollständig gebrides LAN



- ein Switch ist eine besondere Multiport – Bridge

! parallele Kommunikation zwischen den Computern mit voller Sendeleistung !

**Verdeutlichung:**

Repeater / Hub	Bridge / Switch
OSI Layer 1	OSI Layer 2
Collision Domain	Broadcast Domain
immer nur einer sendet	↑ IP – Netzwerk

**2.9 VLAN = Virtuelle LANs**

Die HTW hat ein vollständig „geswitchtes“, hausübergreifendes LAN

**Vorteile:**

- jede, an Netzwerk / LAN angeschlossene Komponente könnte mit jedem andren direkt auf Layer 2 kommunizieren  
Abstandsgrenzen gibt es keine mehr.

**Nachteile:**

- Sicherheit
- Broadcast

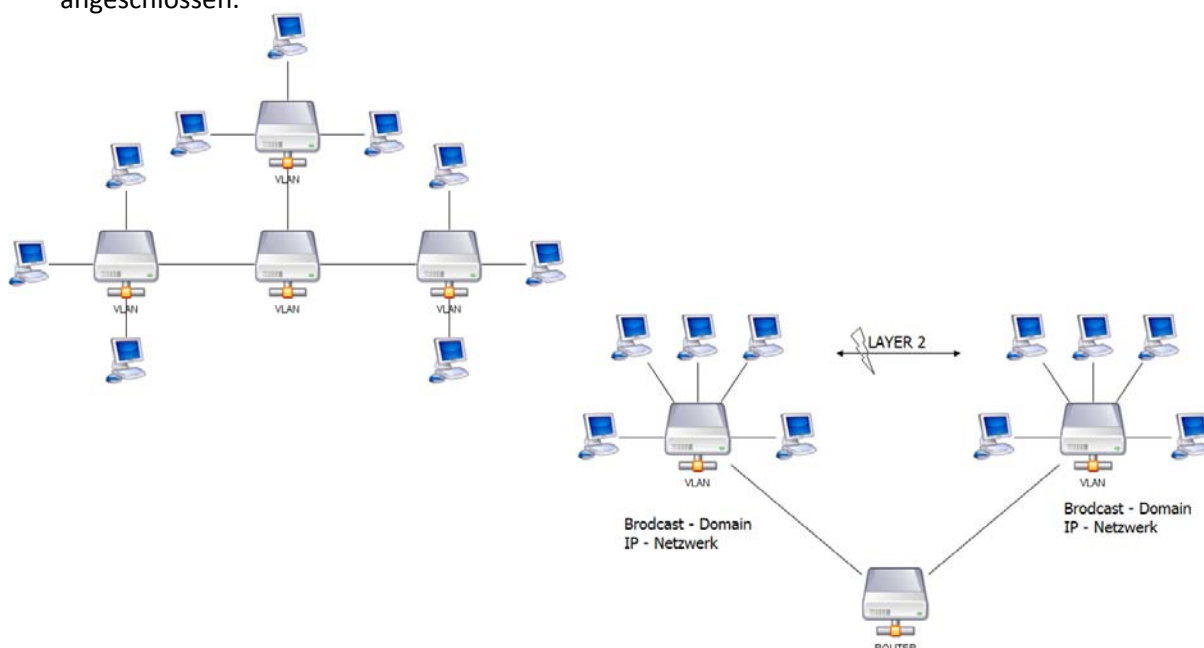
→ Aufteilung des einen physischen LANs logisch per Software

→ logischen LAN = virtuelles LAN

**2.9.1 Was sind VLANs?**

Netzwerkgeräte werden unabhängig von ihrem Standort zu „Geräte – Gruppen = VLANs“ zusammengefasst.

Die Gruppenmitglieder kommunizieren untereinander so als wären sie an ein physisches LAN angeschlossen.





## 2.9.2 Vorteile von VLANs an zwei Beispielen

### Beispiel 1:

VLANs vereinfachen das „Wandern“ von NW – Geräten

### Szenario:

Pauly wird Rektor, wandert von 8210 → 2200

### Problem:

wegen langer Wege liegt in 2200 kein Netzkabel von STL

in 8210 → STL – IP – Netzwerk

in 2200 → Rektor – IP – Netzwerk

### Traditionelle Lösung:

- Paulys PC wird um konfiguriert, das heißt neue IP – Adresse / DNS – Namen

- Firewall:

Anpassung von:

- allen STL – Firewalls
- STL – SAMBA / NFS / NIS+

Zeitaufwand: ~ 90 min

### VLAN Lösung:

Die NW – Anschlüsse in Rektor – Zimmer in 220 werden in das VLAN des STL aufgenommen

Port Grouping V-LAN → 2 min

MAC Grouping V-LAN → 0 sec

### Beispiel 2:

Abschottung von Rechnergruppen

### Szenario:

Verwaltungsbüros liegen in Bau 2 und Bau 8 an der HTW

### Problem:

lange Wege, das heißt in Bau 2 und Bau 8 liegen verschiedene LANs.

→ die Verwaltungsrechner sind in verschiedenen IP- Netzwerken zu Hause.

Auch andere, nicht –Verwaltungsrechner arbeiten in diesen Netzwerken.

- SICHERHEITSPROBLEM  
Layer 2 (MAC – Spoofing)

### VLAN Lösung:

- Alle Verwaltungsrechner in einem V-LAN, das ist durch eine Firewall gesichert
- alle Verwaltungsrechner in einem nicht zugänglichen IP – Netzwerk

## 2.9.3 Wie entstehen VLANs?

- die Standards IEEE 802.1p und 802.1q, veröffentlicht 1998, beschreiben ein herstellerunabhängiges Verfahren zum erzeugen von VLANs in geschwichten Netzwerken
- es wird **ein neues Ethernet-Rahmen-Format** definiert:  
→ IEEE 802.1q, das das Standard Ethernet – Frame um 4 Bytes erweitert
- 1518 Bytes → 1522 Bytes, die 4 Bytes werden vor das **Ether – Typ** bzw. **Ether – Längenfeld** eingefügt

**Die Felder im Einzelnen:**

TPI → Tag Protocol Identifier Ethernet mit dem Wert 0x8100  
das Paket enthält VLAN Informationen

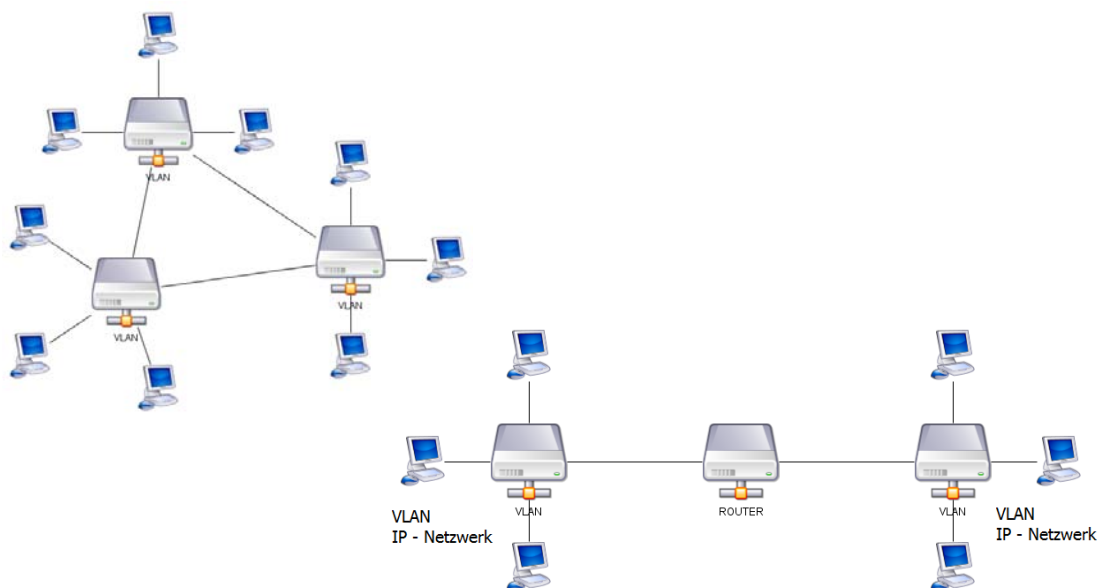
TAG → eigentlich TCI = Tag Control Information  
wird in **3 Felder** aufgeteilt:

- a) VLAN - ID → 12 Bits =  $2^{12} = 4096$  V-LANs (STL – VLA 180)
- b) TR → 1 Bit = Token – Ring spezifisch
- c) Priority → 3 Bit
  - kodiert die Priorität des Datenpakets
  - Es gibt 8 mögliche Prioritätsstufen ( $2^3$ ) definiert als Class of Service in IEEE 802.1p

- jedes Datenpaket, das zu einem bestimmten VLAN gehört, trägt dessen eindeutigen VLAN - ID
- die VLAN – ID wird vom Netzwerkadministrator festgelegt und im SWITCH – Verbund konfiguriert
- am SWITCH ankommende Ethernet – Frames (2 / 802.2) werden vom Switch nach Administratorvorgabe in Ethernet – Frames 802.1q umgepackt
- bevor die Ethernet – Frames den Switch – Verbund verlassen, werden sie wieder in „normale“ Ethernet – Frames umgepackt

**2.9.4 VLAN – Typen****2.9.4.1 Port Grouping VLAN (Switch – Port basierendes VLAN)**

- gängigster Typ, wird auch an der HTW genutzt
- einzelne Ports von einem oder mehreren Switches werden einem VLAN durch den Administrator zugeordnet



**!** alle an einen Port angeschlossenen Geräte gehören zu dem VLAN des Ports

**Vorteile:**

- einfache Konfiguration
- einfache Fehlersuche
- keine neue Endgeräte – Software

**Nachteile:**

- jedes, an einen Port angeschlossene Gerät, gehört zu dem VLAN des Ports
- sollen mehrere VLANs in einem Raum liegen, so müssen mehrere Switch – Ports / Anschlussdosen im Raum verfügbar sein
- [ • standardmäßig an der HTW > 4 Netzwerk – Anschlüsse) ]

**2.9.4.2 MAC – Layer Grouping VLAN (MAC-Adressen basiertes VLAN)**

- die MAC - Adresse des Netzwerk – Gerätes legt das VLAN des Gerätes fest
- Der NW – Admin ordnet die MAC – Adresse eines Netzwerk - Gerätes einem VLAN zu
- Die Switch – Software extrahiert die Quell – Ethernet – Adresse aus dem ankommenden Ethernet – Paket und ordnet danach das Ethernet – Paket einem VLAN zu

**Vorteile:**

- keine neue Endgeräte – Software
- das Gerät gehört zu einem VLAN
- keine redundante Verkabelung der Räume

**Nachteile:**

- die AMC -Adresse gehört zu einem VLAN (MAC – Spoofing)
- Switch – Konfiguration mit MAC – Adresse
- Fehlersuche ist komplexer

**2.9.4.3 Network – Layer – Grouping VLAN (protokollbasiertes VLAN)**

- wird oft durch sogenannte Layer 3 – Switches realisiert, das heißt die Grouping – Merkmale stecken in den Layer 3 – Merkmalen

**a) Zuordnung über das Ethernet – Typfeld:**

- Port 6 -> IP – Datagram → VLAN 200
- Port 6 -> IPX – Datagram → VLAN 170

**b) Zuordnung über die IP – Quell – Adresse / Sende – Adresse**

- Port 7 -> IP – Datagram = IP 134.96.216.17 → VLAN 180
- Port 7 -> IP – Datagram = IP 134.96.220.99 → VLAN 70

die Geräte besitzen „feste“ IP – Adresse!

**Switch → Layer 2 Switch:**

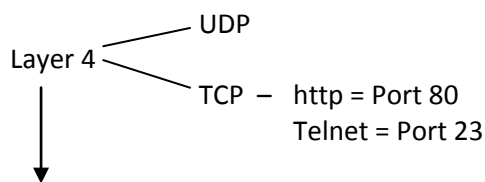
- zur Weiterleitung des Rahmen / Frames werden nur die Sender- und Empfänger – Adressen ausgewertet
- der Switch führt pro Port eine MAC – Tabelle  
(zum Beispiel verwaltet ein Gigabit - Switch bis zu 20.000 Adresseinträge)

**Layer 3 Switching:**

- auch Informationen in den Layer 3 – Protokoll – Feldern werden zum Weiterleiten der Rahmen / Frames verwendet
  - zum Beispiel:
    - die IP – Sender / Empfänger – Adresse
- die Switches übernehmen Routing – Aufgaben

**Layer 4 Switching:****Problem:**

Bandbreite:    - www – http  
                   - Videokonferenz – yyy  
                   - Voice over IP – VoIP



Priorisierung der Ethernet – Pakete anhand der UDP / TCP - Portnummern

**Layer 7 Switches:**

- Server Load Balancing

**2.10 WAN – Technologien und Routing****LAN:**

- vernetzt eine kleine Anzahl von Computern innerhalb von Gebäuden / Campus
- ist in Bezug auf:
  - Anzahl der Geräte
  - Lage der Verbindungen (CSMA/CD)
 begrenzt.

**WAN:**

- sollte beliebig **skalierbar** sein, das heißt bei Bedarf immer erweiterbar
- vernetzt theoretisch beliebig viele Computer mit beliebigen Standorten auf der Welt

**2.10.1 Eigenschaften des WAN**

- wie LAN ein paketvermittelndes Netzwerk
- **kein** gemeinsames Medium, das heißt **keine direkte** Kommunikation auf Ebene 2 möglich
- **kein** einheitliches **Medium** (Cu, LWL, Funk,...)
- zentrales Element ist der **Router**
  - WAN ist ein virtuelles Netzwerk mit Routern als Verbindungselemente zwischen den einzelnen Computern oder LANs
- **alle** an einen WAN angeschlossenen Computer können parallel kommunizieren, also Daten austauschen
  - das ermöglichen die Router mit ihrem Schema Store – and – Forward (Speichervermittlung)

**Das Prinzip (Store – and – Forward):**

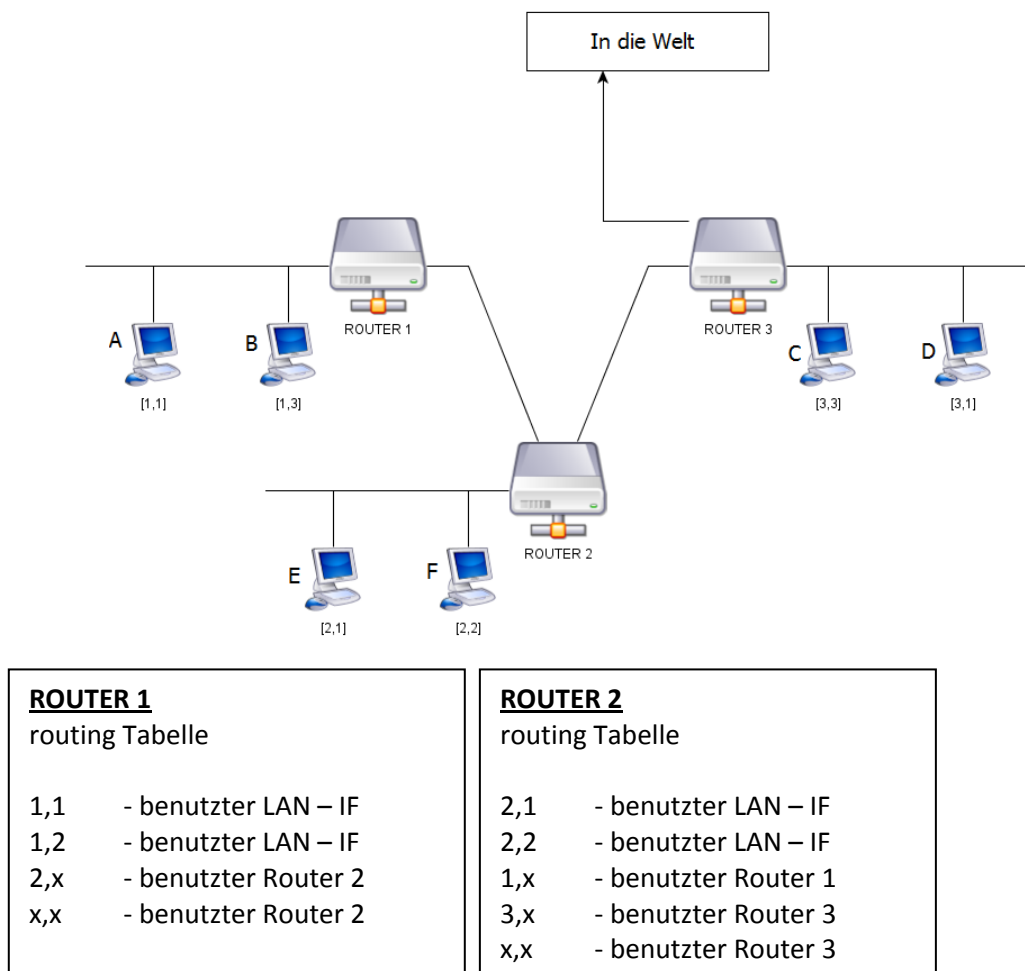
- Computer sendet sein Datenpaket an den Router
- Router empfängt das Datenpaket, speichert es ab und analysiert es
- Weg zum Ziel suchen, Datenpaket in Weg – Werte – Schlange schreiben
- Datenpaket an das Ziel oder den nächsten Router senden

**Problem (Weg finden!):**

→ hierarchische Adressen

Computer besitzt zwei Adressen:

WAN – Adresse → IP – Adresse → Layer 3  
 LAN – Adresse → Ethernet – Adresse → Layer 2

**Routing – Algorithmus → Teilstreckenverfahren**

x,x = DEFAULT – ROUTE

C<sub>A</sub> → C<sub>B</sub> → gleiches LAN / WAN – Teilnetzwerk → Kommunikation direkt via Layer 2

$C_A \longrightarrow C_E \longrightarrow$  ungleiches WAN – Teilnetzwerk  $\rightarrow$  verschiedene LANs  
 $\rightarrow C_A$  sendet Datenpaket an R1  
 $\rightarrow$  R1 sucht Weg, findet R2  
 $\rightarrow$  R2 kennt Zielrechner  $C_E$   
 $\rightarrow$  R2 sendet Datenpaket an  $C_E$

**! Jeder Rechner besitzt eine Routing – Tabelle**

UNIX / LINUX/ WIN: netstat - r

↓  
route print

**Hinweis:**

es gibt statisches und dynamisches Routing.

**a) statisch:**

Systemadministrator legt Routinginfos fest

**b) dynamisch:**

Routingprotokolle über die sich Router über das Netz informieren

## 3 TCP / IP in UNIX / WINDOWS - Umfeld

### 3.1 Protokolle – Protokollstapel / Stackfamilie

(kurze Wiederholung des Stoffes)

### 3.2 Internetworks & IP - Adressen

**Internetwork:**

- bezeichnet eine beliebige anzahl zusammengeschalteten (LAN-) Netzwerke die Pakete zwischen zwei Computern übertragen können
- ist ein logisches = virtuelles Netzwerk, wird durch die auf Computern und Routern laufende IP – Software realisiert

**Zu lösende Probleme:**

- man braucht eine einheitliche Adressierung für alle Netzwerkgeräte weltweit
- Routing = Weg finden

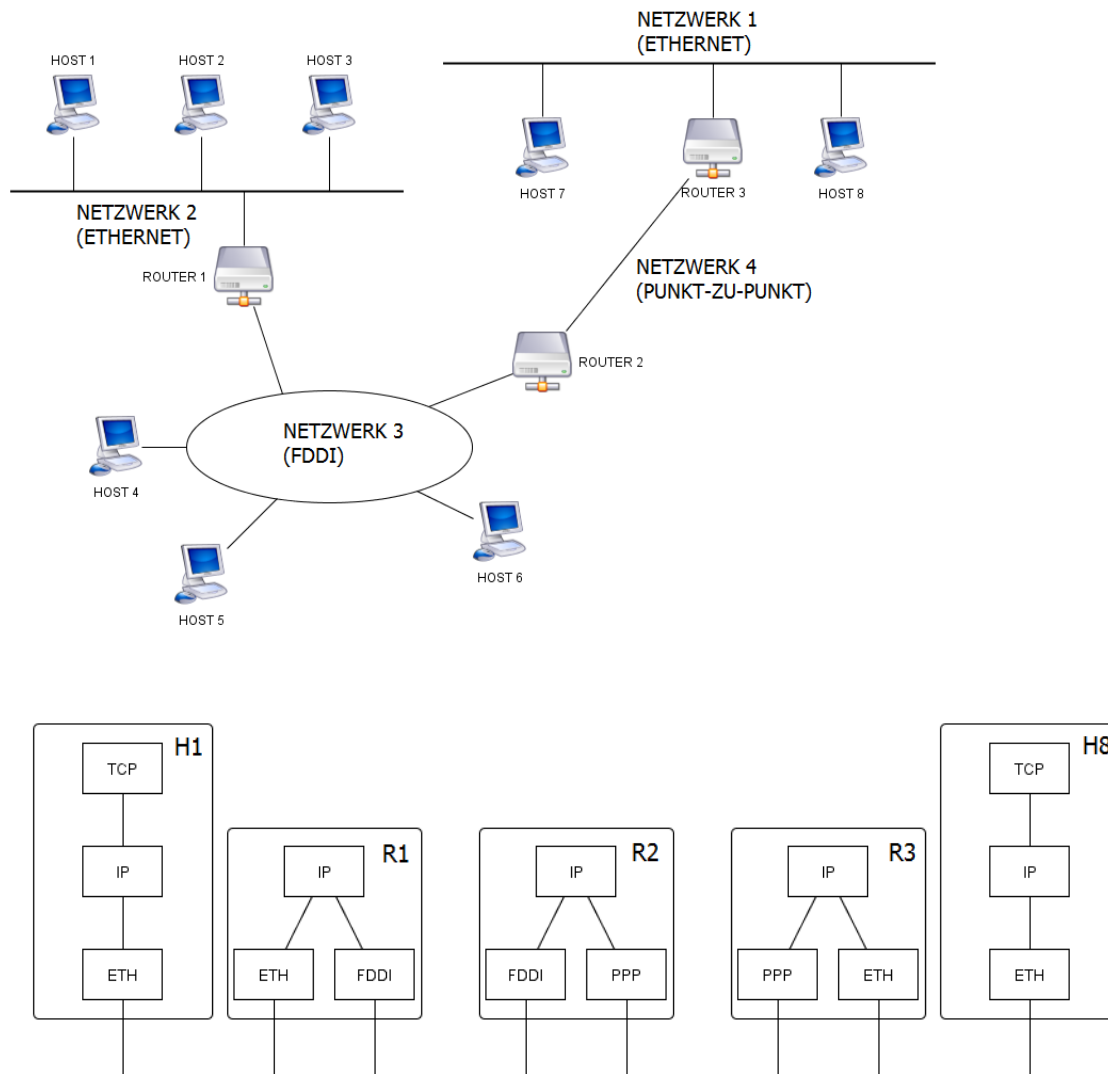


Abb. 4.2 : Einfaches Internetwork mit den Protokollschichten

**Merksätze zu IP:**

- IP ist die Vermittlungsschicht des TCP / IP – Protokoll – Stacks
- IP baut das Internet auf
- IP – Software läuft auf Host und Routern
- Hosts sind Rechner / Computer / Geräte mit einer IP – Adresse, sie sind an einen IP – Netzwerk angeschlossen
- Multihomed Hosts sind Rechner / Computer / Geräte mit mehreren Netzwerkkarten und mehreren IP – Adressen, sie sind an mehrere IP – Netzwerke angeschlossen, aber haben keine Routerfunktionalität

Router / Gateways sind „Spezial – Rechner“, die mehrere Netzwerkkarten besitzen, also mehrere IP – Adressen besitzen, an mehrere IP – Netzwerke angeschlossen sind und IP – Pakete zwischen diesen Netzwerken austauschen können. Sie haben Routing – Funktionalität.

### 3.2.1 Das IP - Dienstmodell

aufgeteilt in 2 Teile

- Adressierungsschema
- verbindungsloser Datagramm - Dienst

Datagramm ist ein Datenpaket mit ausreichend Steuerinformationen um durch das Netz von Sender zum Empfänger zu gelangen

→ „Best Effort“ – Dienst

#### Best Effort – Dienst:

- IP bemüht sich ein Datagramm vom Sender zum Empfänger zu befördern
- IP gibt **keine** Zustellgarantie und macht **keine** Fehlerkorrekturen, das heißt ein IP – Datagramm kann gar nicht ankommen, es kann mehrfach ankommen, IP – Datagramme können in falscher Reihenfolge bzw. verzögert ankommen

### 3.2.2 Die IP - Adresse

- 32 Bit – Adresse – 4 Byte
- wird, damit der Mensch sie lesen kann, in der Punkt – Dezimalschreibweise angegeben

134.96.116.100  
↑      ↙  
Byte    Byte

- jedes Netzwerkgerät, dass im Internet kommunizieren will, besitzt mindestens **eine**, seine IP – Adresse, die wird **statisch** (/etc /hosts) oder **dynamisch** (DHCP) zugewiesen
- jedes IP – Datagramm, das versendet wird, trägt in seinem Kopf (Header) die Sende – und die Empfänger – IP – Adresse
- will ein Computer an einen anderen ein IP – Datagramm senden, muss er dessen IP – Adresse kennen

### 3.2.3 IP – Adresshierarchie

jede IP – Adresse wird logisch in 2 Teile gegliedert:

- einen Präfix = Netzwerkadresse
- eine Suffix = Hostadresse

#### Erklärung der Aufteilung der IP - Adresse:

- jedem Netzwerkgerät kann lokal eine Hostadresse gegeben werden
- die Netzwerkadressen werden weltweit vergeben

#### gewünschte Folge:

- kleine Routing – Tabellen weltweit



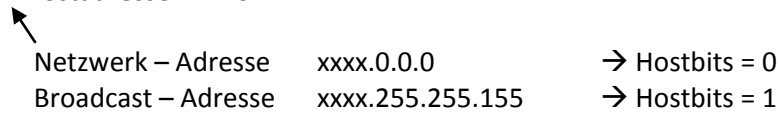
### 3.2.3.1 IP – Adressen Entwicklungsstufe 1

- Classful IP – Addressing
- Primary Address Classes
  - A / 8 Präfix – Bits
  - B / 16 Präfix – Bits
  - C / 24 Präfix – Bits

#### Class A:

- 8 Bit Network – Präfix
- oberste Bit = 0 + 7 Bit bilden die Netzwerkadresse → Netzwerkadresse 8 Bit
- 24 Bit – Hostadresse
- 128 / 8 Networks ( $2^7$ )
 

0.0.0.0	reserviert (old Broadcast)
127.0.0.0	reserviert (loopback – Netzwerk)
- 1.0.0.0 bis 126.0.0.0
- $2^{24} - 2$  Hostadressen = 16777214
 



Netzwerk – Adresse	xxx.0.0.0	→ Hostbits = 0
Broadcast – Adresse	xxx.255.255.155	→ Hostbits = 1
- 50 % aller IP – Adressen :=  $2^{31}$ 

7 Bit
24 Bit

0	Netz - ID	Host – Identifikator (Host – ID)
0.0.0.0 – 127.255.255.255		

#### Class B:

- 16 Bit Network – Präfix
- 2 obersten Bits = 10 + 14 Bits bilden die Netzwerkadresse → Netzwerkadresse 16 Bit  
→ 16 Bit Hostadressen
- $2^{14}$  - Netzwerke → 16384  
mit  
 $2^{16} - 2$  – Hosts → 65534 Hostadressen  
128.0.0.0 → 191.255.255.255
- 25 % aller IP – Adressen
 

14 Bit
16 Bit

1	0	Netz - ID	Host – Identifikator (Host – ID)
128.0.0.0 – 191.255.255.255			

#### Class C:

- 24 Bit Network – Präfix
- die ersten 3 Bit haben 110 + 21 Bits → 24 Netzwerk – Bits  
8 Host – Bits
- $2^{22}$  – Netzwerke – 2097152 in jedem Netzwerk
- $2^8 - 2 = 254$  Hostadresse
- 12,5 % aller IP – Adressen

			21 Bit	8 Bit
1	1	0	Netz - ID	Host – Identifikator (Host – ID)
192.0.0.0 – 223.255.255.255				

**Class D:**

Multicast – Gruppen

28 Bit

1	1	1	0	Identifikator der Multicast - Gruppe
---	---	---	---	--------------------------------------

224.0.0.0 – 239.255.255.255

**Class E:**

RESERVIERT

27 Bit

1	1	1	1	0	reserviert (für die Zukunft)
---	---	---	---	---	------------------------------

240.0.0.0 – 247.255.255.255

### 3.2.3.2 Probleme – unvorhergesehene Beschränkungen der Classful – Adressierung

- ungünstige Vergabepraxis der IP – Adressen – Blöcke  
→ IP – Adressenmangel  
[im Internet sind ca. 69 Millionen IP – Adressen aktiv]
- die Routing – Tabellen wachsen exorbitant

1990	2190	Routen
1995	> 30000	Routen
2000	> 100000	Routen

**Lösungen:**

- früh → Subnetting mit fester Netzwerkmaske
- heute → VLSM = variable length subnet masks  
→ CIDR = Classless Internet Domain Routing
- morgen → IPv6 = 128 Bit – Adressen = 16 Bytes  
=  $2^{128} - 2$

### 3.2.3.3 Subnetting

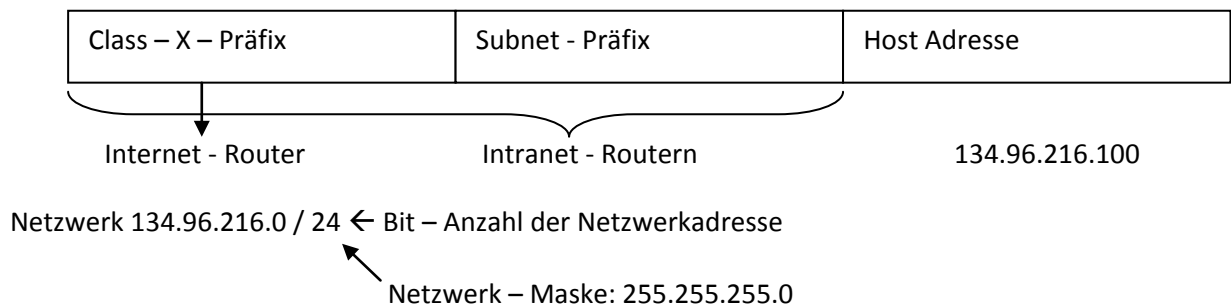
- beschrieben in RFC 950 1985

**Aufgabe:**

Aufteilung von Class A / B / C – Netzwerken in kleinere Einheiten

**Ziel:**

- das Routingtabellen - Wachstum begrenzen
- von außen (aus dem Internet) **nicht** sichtbar

**Lösung:****Extended Network Präfix**

Host – Adresse	134.96.216.100
&	255.255.255.0
Netzwerk – Adresse	134.96.216.0

Routingprotokolle      → Organisationintern

- RIP – 1
- BGP – 4

**Subnet Beispiel 1:**

**gegeben:**                      134.96.216.0 / 24

**gesucht:**                      6 Subnets mit mindestens 25 Hosts

**Lösung:****Schritt 1:**

wähle die zusätzlich notwendige Anzahl von Bits für die Netzwerkadresse

- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$                       ← 3 zusätzliche Netzwerk – Adress – Bits
- $2^4 = 16$

**Schritt 2:**

ich wähle 3 Bits → **8 Netzwerke**

134.96.216.0 / 27

Netzwerk – Maske: ALT / 24      → 255.255.255.0

Netzwerk – Maske: NEU / 27      → 255.255.255.xxx.0000    (xxx = 224, weil  $2^5 + 2^6 + 2^7$ )

→ 255.255.255.224

**Schritt 3:**

Host – Bit: 5  
 $2^5 - 2 = 30$

0 0000  
 0 0001  
 0 0010  
 0 0011  
 0 0100  
 ...  
 1 1111

30

**Schritt 4: Festlegung der Subnet – Adressen**

1. Netzwerk:	134.96.216.0 / 27	0000	0000
2. Netzwerk:	134.96.216.32 / 27	0010	0000
3. Netzwerk:	134.96.216.64 / 27	0100	0000
4. Netzwerk:	134.96.216.96 / 27	0110	0000
5. Netzwerk:	134.96.216.128 / 27	1010	0000
6. Netzwerk:	134.96.216.192 / 27	1100	0000
7. Netzwerk:	134.96.216.224 / 27	1110	0000

**Schritt 5: Festlegung der Broadcast - Adressen**

1. Netzwerk:	134.96.216.31	0001	1111
2. Netzwerk:	134.96.216.63	0011	1111
3. Netzwerk:	134.96.216.95	0101	1111
4. Netzwerk:	134.96.216.127	0111	1111
5. Netzwerk:	134.96.216.159	1011	1111
6. Netzwerk:	134.96.216.191	1101	1111
7. Netzwerk:	134.96.216.223	1111	1111

**Schritt 6: Vergegenwärtigen der Hostadressen**

Netzwerk:	134.96.216.128 / 27	1000	0000
30	134.96.216.129 / 27		0001
	134.96.216.130 / 27		0010
	.		.
	.		.
	.		.
	.		.
	.		.
	.		.
	134.96.216.158		1110
Broadcast:	134.96.216.159	1001	1111

### 3.2.3.4 IP - Adressen Entwicklungsstufe 2

Problem von einfachen Subnetting mit festem Netzwerk – Präfix

- ineffektive IP – Vergabe  
das größte Subnetzwerk bestimmt die Netzwerk – Bit – Anzahl

**Beispiel:**

134.96.216.0 / 24

254 → / 25

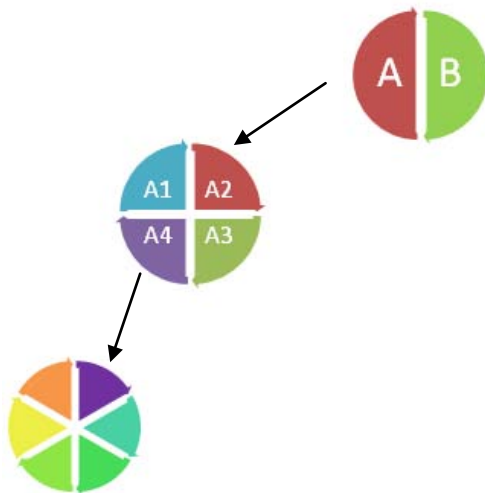
es gibt ein Netzwerk → 120 Hosts → 134.96.216.0 / 25 → 2 \* 128 Hosts  
 → 20 Hosts  
 → 50 Hosts  
 → 3 Subnetze sind nicht möglich!!

#### 3.2.3.4.1 VSLM (variable length Subnet Mesh)

- beschrieben 1987 in RFC 1009
- nötige Routing – Protokolle – Organisationintern  
RIP – 2 / OSPF / I – IS – IS

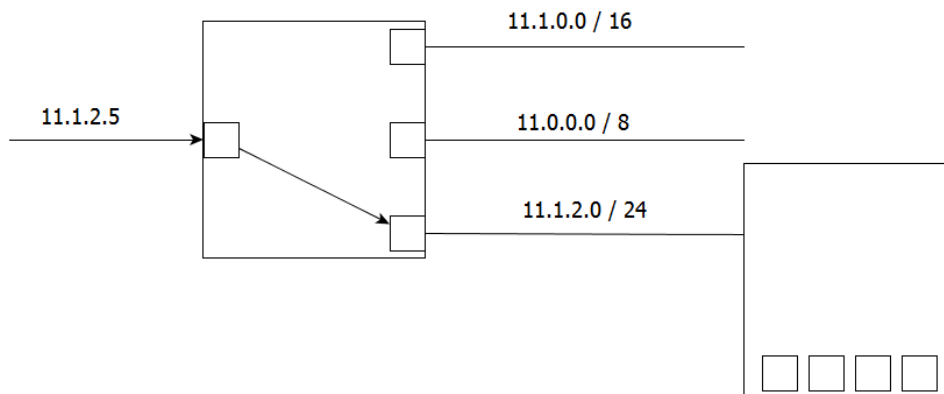
#### Routen – Bündelung

VSLM ermöglicht das rekursive Teilen von Netzwerken und erlaubt das Bündeln von Routen zu den entstandenen Teilnetzwerken



#### 3.2.3.4.2 Voraussetzungen für VLSM

- Die Routing – Protokolle müssen die Extended – Network – Präfix - Information weiterreichen
- Alle Router müssen beim Wegesuchen den Algorithmus der „größten Übereinstimmung“ realisieren
- Die IP – Adressen, damit die Routenbündelung funktioniert, die Topologie des Netzes widerspiegeln



### 3.2.3.5 IP – Entwicklungsstufe 3

CIDR → Classless InterDomain Routing

- definiert in RFC 1517 / 1518 / 1511 / 1520

#### Warum?

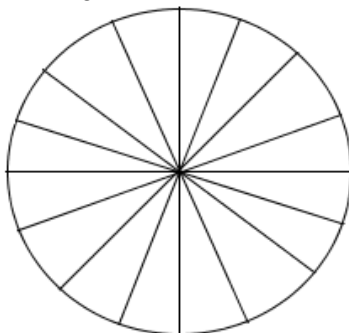
- Klasse B – Netzwerke ‚fast‘ aufgebraucht
- rasantes Routingtabellen – Wachstum
- das IP – Adressen Klassenkonzept wird für die Klassen A / B / C aufgelöst
- erlaubt die effektive Ausnutzung des IP – Raumes weltweit
- jede Routing – Information enthält die Präfix – Länge als Zahl IP – Adresse / 20
- Internet Provider oder Firmen kaufen keine Class A / B / C – Netzwerke sondern / 13 / 14 / 18 .... / 27 Netzwerke

#### Beispiel:

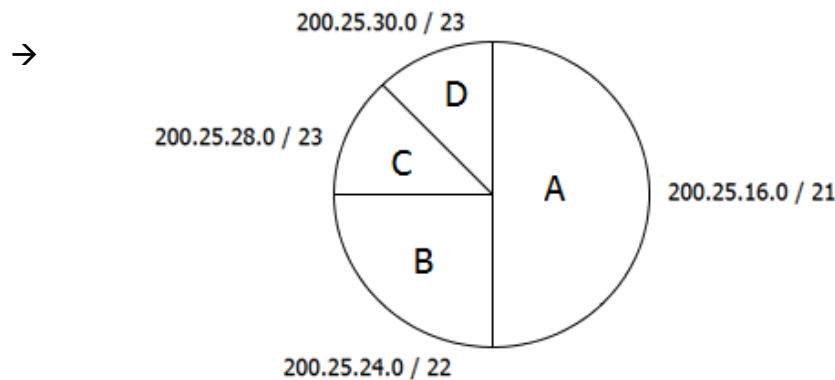
**gegeben:** 200.25.0.0 / 16

Teilung in 16 Teilnetzwerke festgelegt

→ 120



200.25.16.0 / 20 (Teilnetzwerk von oben)



**Das Vorgehen zur rekursiven Aufteilung einer IP – Adressen – Bereiches bei VSLM (intern) bzw. CIDR (Internet)**

**Schritt 1: Teile den zugewiesenen Adressen – Bereich in 2 gleich große Teile**

200.25.16.0 / 20                      0001 xxxx. 0000 0000

A: 200.25.16.0 / 21                      0001 0xxx. 0000 0000

200.25.24.0 / 21                      0001 1xxx. 0000 0000

**Schritt 2: Teilblock wieder in 2 Blöcke**

200.25.24.0 / 31                      0001 1000. 0000 0000

B: 200.25.24.0 / 22                      0001 1000. 0000 0000

200.25.28.0 / 22                      0001 1100. 0000 0000

**Schritt 3: Teile erhaltenen IP – Bereich in 2 Teilbereiche**

200.25.28.0 / 22                      0001 1100. 0000 0000

C: 200.25.28.0 / 23                      0001 1100. 0000 0000

D: 200.25.30.0 / 23                      0001 1110. 0000 0000

**3.2.3.6 Neue Lösung für das IP – Adressen – Problem**

gute Nachricht                      → CDIR – arbeitet

schlechte Nachricht                      → die Routingtabellen wachsen exponentiell

RFC 1917 → verlangt die Rückgabe von ungenutzten IP – Adressraum an die IANA

RFC 1918 → Adressblocks für private Netzwerke, die nicht direkt mit dem Internet kommunizieren können

10.0.0.0 – 10.255.255.255 / 8

172.16.0.0 – 172.31.255.255 / 12

192.168.0.0 – 192.168.255.255 / 16

### 3.2.4 Spezielle IP – Adressen

#### 3.2.4.1 Netzwerk – Adressen

- ist die IP – Adresse des Netzwerks
- $\text{Netzwerk – Adresse} = f(\text{Netzwerk – Maske})$
- alle Host – Bits = 0

	Netzwerk – Adresse	Netzwerk - Maske	
UNI – SD	134.46.0.0	255.255.0.0	/ 16
STL	134.96.216.0	255.255.255.0	/ 24
Campus	134.96.208.32	255.255.255.224	/ 27

#### 3.2.4.2 gerichtete Broadcast – Adresse

- soll alle Host eines IP – (Sub) – Netzwerkes erreichen
- alle Host – Bits = 1
- $\text{gerichtete Broadcast – Adresse} = f(\text{Netzwerk – Maske})$
- **IP – Pakete mit der gerichteten Broadcast – Adresse als Ziel – IP, werden von Routern weitergeleitet**

	Netzwerk – Adresse	Netzwerk - Maske	gerichtete BC - Adresse
UNI – SB	134.46.0.0	255.255.0.0	234.96.255.255
STL	134.96.216.0	255.255.255.0	134.96.216.255
Campus	134.96.208.32	255.255.255.224	134.96.208.63

#### **Befehl:**

ping –s gerichtete Broadcast - Adresse

#### 3.2.4.3 begrenzte Broadcast – Adresse

- adressiert die Hosts des lokalen IP – Netzwerkes
- IP – Pakete mit **Zieladresse = begrenzte Broadcast – Adresse** werden von Routern **nicht** weitergeleitet

#### **Befehl:**

ping –s 255.255.255.255

ping –s 134.96.216.255

#### 3.2.4.4 Schleifenadresse – loopback

- 127.0.0.1 – localhost
- 127.0.0.0 / 8



### 3.3 DHCP – Dynamic Host Configuration Protocol

**Frage:**

Was muss ein Host alles kennen, um in einem IP – Netzwerk und dem Internet arbeiten zu können?

**Antwort:**

- Standardgateway / Router
- seine IP – Adresse
- seine Subnet – Mask
- einen DNS – Server

Wie bekommt er die Daten?

- manuelle Konfiguration
- dynamisch → **DHCP** (beschrieben in RFC 2131)

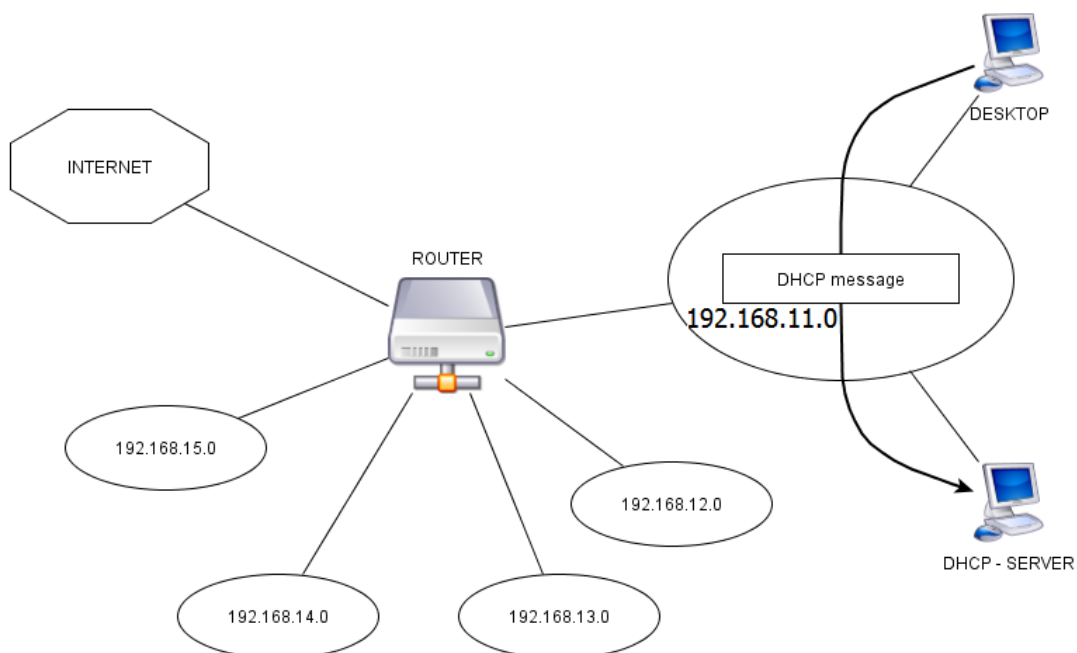
Ein DHCP – Server kennt alle nötigen Daten, die ein Host benötigt um im Netz des Servers arbeiten zu können.

**Problem:**

Wie findet der Host (s)eine DHCP – Server?

**Lösung:**







- Der Host sendet seine DHCP – Anfrage an die begrenzte IP – Broadcast – Adresse
- alle Hosts des IP – Netzwerkes werden diese Anfrage entgegennehmen; die DHCP – Server werden antworten
- DHCP benutzt UDP als Transportdienst (DHCP ist ein Nachfolger – Protokoll von BOOT P)



### 3.3.1 DHCP – Kommunikationsablauf

#### Kommandos:

Solaris: snoop  
Win XP: wireshark

1. Client sendet ein DHCP – Discover mit seinen Transaction – ID  
 via UDP / IP – BROADCAST / Ethernet – Broadcast
  2. alle erreichten Server senden ein DHCP – Offerpaket mit der Transaction – ID des Clients  
 via UDP / IP – BROADCAST / Ethernet – Broadcast
  3. der Client sendet ein DHCP – Request an den, von ihm ausgewählten, Server  
 via UDP / IP – BROADCAST / Ethernet – Broadcast
  4. der Server sendet ein DHCP – Acknowledgement  
 via UDP / IP – BROADCAST / Ethernet – Broadcast
- ↓
- ← Client konfiguriert sein Netzwerk – Interface mit den vom Server erhaltenen Daten
5. Client sendet zyklisch ein DHCP – Request um sein Lease zu erhalten  
 via UDP / IP – UNICAST / Ethernet – UNICAST
  6. der Server sendet zyklisch ein DHCP – Acknowledgement  
 via UDP / IP – BROADCAST / Ethernet – Broadcast

#### **Wir wissen:**

IP – gerichtete BC / IP – begrenzte BC Adresse → Ethernet – BC – Adresse als Ziel – Ethernet – Adresse

IP – UNICAST Adresse als Zieladresse → Ethernet – UNICAST – Adresse als Ziel – Ethernet – Adresse

MULTICAST – IP – Adressen erreichen eine Gruppe von Hosts → welche Ethernet – Adresse?

## 3.4 Multicasting

#### Anwendungen:

- Videospiele über Internet
- Fernsehen
- Software – Auslieferung

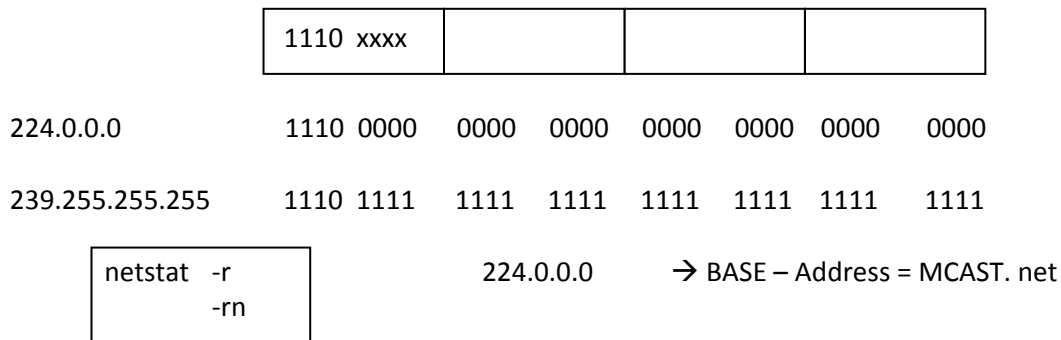


**ein und dieselbe Nachricht geht an viele Hosts**

#### Merkmale:

- Quelle sendet an eine variable Gruppe von Hosts
- Hosts können eine Multicast – Gruppe beitreten / verlassen (IGMP: Internet Group Management Protocol)
- Hosts können in mehreren Multicastgruppen aktiv sein
- der Sender / Quelle muss nicht Mitglied der Gruppe sein, um ihr etwas zu senden

- eine Multicastgruppe kann sich über viele IP – Netzwerke erstrecken
- Multicast – Adresse → Klasse D – IP – Adresse

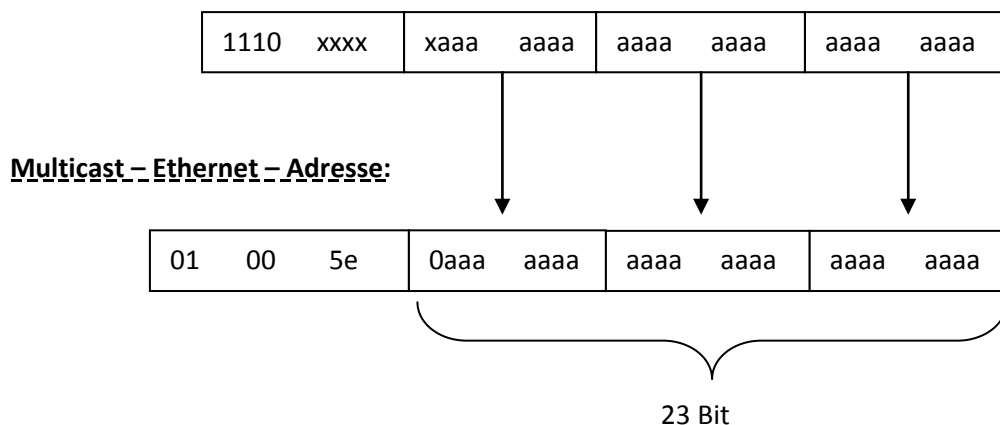


- es gibt **permanente Multicastgruppen** von der IANA festgelegt

### 3.4.1 IP – Multicast → Ethernet – Multicast

- die IANA besitzt einen Ethernet – Adressen – Block **00 : 00 : 5e**
- IP – Multicast – Adressen werden auf die IANA – Ethernet – Adresse **01 : 00 : 5e : 00 : 00 : 00** bis **01 : 00 : 5e : 7f : ff : ff** abgebildet

#### Klasse D – IP – Adresse:



### 3.4.2 Aufgaben eines Netzwerk - Interfaces

Eine Netzwerkkarte filtert aus dem Rahmenstrom des Netzwerkes alle Rahmen des Netzwerkes heraus, die als Ziel – MAC – Adresse

- die eigene MAC – Adresse
  - die Broadcast – Adresse
  - eventuell konfigurierten MAC – Adressen
  - eventuell die fordere Multicast – MAC – Adressen
- enthalten.

### 3.5 Die Verbindung einer IP – UNICAST – Adresse zu einer Ethernet – UNICAST – Adresse

#### Frage:

Wie adressiert man in INTERNET Hosts?

#### Antwort:

via IP – Adresse

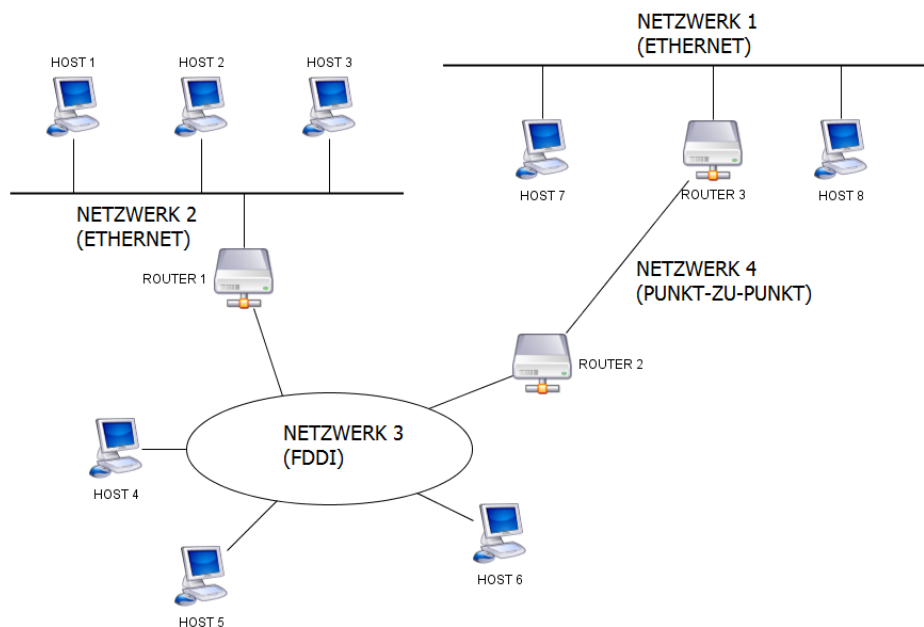
#### ABER:

der physikalische Transport der Daten erfolgt auf Layer 0, gemanagt von Layer 1, dort gelten MAC – Adressen.

→ es muss ein System her, dass **wenn notwendig** zu einer IP – Adresse die MAC – Adresse findet

#### Wann notwendig?

Nur, wenn Kommunikationspartner im gleichen LAN, das heißt im gleichen IP - Netzwerk



H<sub>1</sub> will mit H<sub>3</sub> kommunizieren. Er berechnet:

$$\begin{aligned} IP_{H_1} \& \text{ Netzwerk} - \text{Maske}_{H_1} &= \text{Netzwerk} - IP_{H_1} \\ IP_{H_3} \& \text{ Netzwerk} - \text{Maske}_{H_3} &= \text{Netzwerk} - IP_{H_3} \end{aligned} \quad \left. \vphantom{\begin{aligned} IP_{H_1} \& \text{ Netzwerk} - \text{Maske}_{H_1} &= \text{Netzwerk} - IP_{H_1} \\ IP_{H_3} \& \text{ Netzwerk} - \text{Maske}_{H_3} &= \text{Netzwerk} - IP_{H_3} \end{aligned}} \right\} 0 \rightarrow H_3 \text{ im gleichen IP - Netzwerk wie } H_1$$

$$\begin{aligned} 134.96.216.99 \& 255.255.255.0 &\rightarrow 134.96.216.0 \\ 134.96.216.47 \& 255.255.255.0 &\rightarrow 134.96.216.0 \\ IP_{H_8} \& \text{ Netzwerk} - \text{Maske}_{H_1} &= \text{Netzwerk} - IP_{H_8} \\ 134.208.17 \& 255.255.255.0 &\rightarrow 134.96.208.0 \end{aligned} \quad \left. \vphantom{\begin{aligned} 134.96.216.99 \& 255.255.255.0 &\rightarrow 134.96.216.0 \\ 134.96.216.47 \& 255.255.255.0 &\rightarrow 134.96.216.0 \\ IP_{H_8} \& \text{ Netzwerk} - \text{Maske}_{H_1} &= \text{Netzwerk} - IP_{H_8} \\ 134.208.17 \& 255.255.255.0 &\rightarrow 134.96.208.0 \end{aligned}} \right\} \begin{array}{l} = \\ \neq \end{array}$$

#### Berechnungsergebnis:

Kommunikationspartner im gleichen IP – Netzwerk (= VLAN)

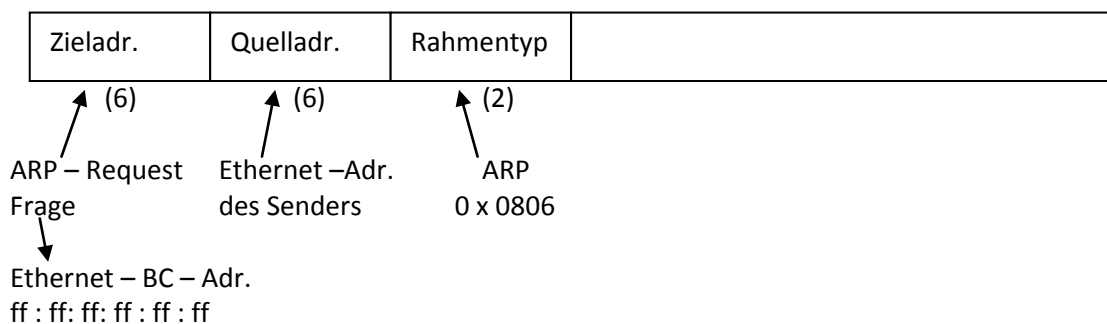
**Wie findet man zu den Ziel – IP – Adressen die Ziel – Ethernet – Adressen?**

- a) Berechnung (wie bei MULTICAST)
- b) Übersetzungstabelle manuell erstellen, jedem Host geben
- c) der Host baut seine Übersetzungstabelle selbständig über eine Kommunikation auf

└─→ ARP = Address Resolution Protocol

**3.5.1 ARP - Address Resolution Protocol**

- Adressauflösungsprotokoll
- arbeitet mit 2 Nachrichten:
  - Frage
  - Antwort
- beschrieben in RFC 825
  - ARP findet zu gegebener IP – Adresse die Ethernet / MAC – Adresse
  - RARP findet zu gegebener Ethernet – Adresse die IP – Adresse

**ARP Datenpaket - Format**

Zieladr.	Quelladr.	Rahmentyp	1	2	3	4	5	6	7	8	9
(6)	(6)	(2)	(2)	(2)	(1)	(1)	(2)	(6)	(4)	(6)	(4)

- |                     |                             |
|---------------------|-----------------------------|
| 1: Hardware – Typ   | 6: Ethernet Quell - Adresse |
| 2: Software – Typ   | 7: IP – Quell – Adresse     |
| 3: Hardware – Größe | 8: Ethernet – Zieladresse   |
| 4: Software – Größe | 9: IP Zieladresse           |
| 5: Operation        |                             |

HW – Typ            = 1 wenn Ethernet

SW –Typ            = 0 x 0800 = IP

HW – Größe = Adresslänge in Bytes    = 6    ← Ethernet

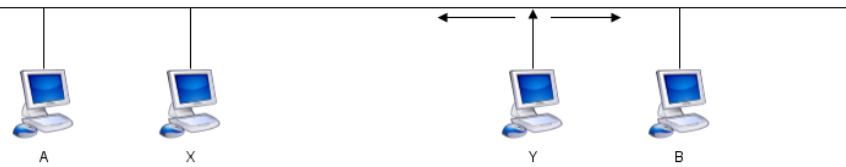
SW – Größe                                    = 4    ← IP

**Op = Operation:**

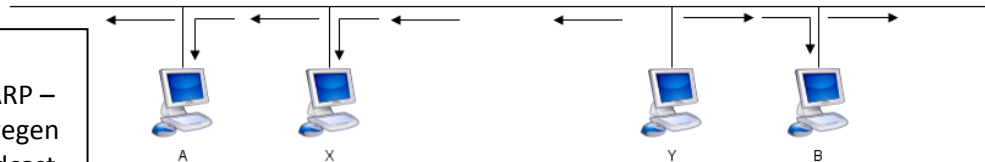
- 1 = ARP – Request / Anfrage
- 2 = ARP – Reply / Antwort
- 3 = ARP – Request
- 4 = ARP – Reply

**Experiment:****1. Schritt:**

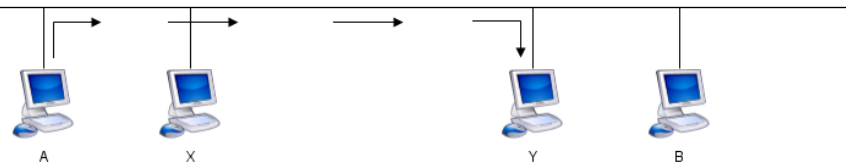
ARP – Anfrage  
senden mit  
Ethernet - Broadcast

**2. Schritt:**

alle Cs werten ARP –  
Nachricht aus wegen  
Ethernet - Broadcast

**3. Schritt:**

der C mit der  
gesuchten IP – Adr.  
antwortet mit ARP –  
Antwort und  
speichert in seiner  
ARP – Tabelle das  
IP – Ethernet –  
Adresspaar  
→ ARP - Cache



1. suche Host mit dem der Arbeitshost in letzter Zeit noch nicht kommuniziert hat  
arp -a  
zeigt den Inhalt der momentanen ARP – Tabelle / des ARP – Caches
2. starte snoop -v arp | tee / tmp / xx
3. ping -s stl - c09 50 1
  - um die ARP – Kommunikation zu begrenzen speichert sich die ARP – Software alle ARP – Antworten in ihrer ARP – Tabelle / Cache
  - die Eintragungen im ARP – Cache unterliegen einer Alterung; sie werden nach spätestens 20 – 30 min aus der Tabelle gelöscht, wenn sie nicht durch ein ARP – Request oder ARP – Reply aufgefrischt wurden

**Wie verarbeitet die ARP – Software ankommende ARP – Pakete?**

- ARP – Requests **werden in der Regel** als LAN – Broadcasts versendet
- ARP – Replies **werden in der Regel** als LAN – UNICAST versendet

**Wann verändert die ARP – Software die ARP – Tabelle eines Hosts?**

1. der Empfängerhost ändert einen **bestehenden** ARP – Tabellen – Eintrag **immer** ab, egal ob es sich bei der eingetragenen Ziel – IP – Adresse (Target IP – Adresse) um seine eigene IP – Adresse handelt oder nicht
2. Das Zielsystem erzeugt einen neuen ARP – Tabellen – Eintrag, falls es sich bei den Target – IP – Adressen um seine IP – Adresse handelt.  
Auch hier ist es unerheblich, ob es sich um ein ARP – Request oder ein ARP – Reply handelt.

**Sicherheitsproblem:**

Die ARP – Software übernimmt ohne weitere Prüfung Informationen aus ankommenden ARP – Paketen, die eigentlich **nicht** für diesen Host erzeugt wurden.

ARP ist zustandslos.

Speziell werden auch Antworten bearbeitet, zu denen nie eine Frage gestellt wurde.

**3.5.2 Gratuitous – ARP****1) Entdeckung von doppelt benutzten IP – Adressen**

Ein Host sendet bei Systemstart via Ethernet – Broadcast einen ARP – Request mit seiner IP – Adressen als Target IP – Adresse ins LAN.

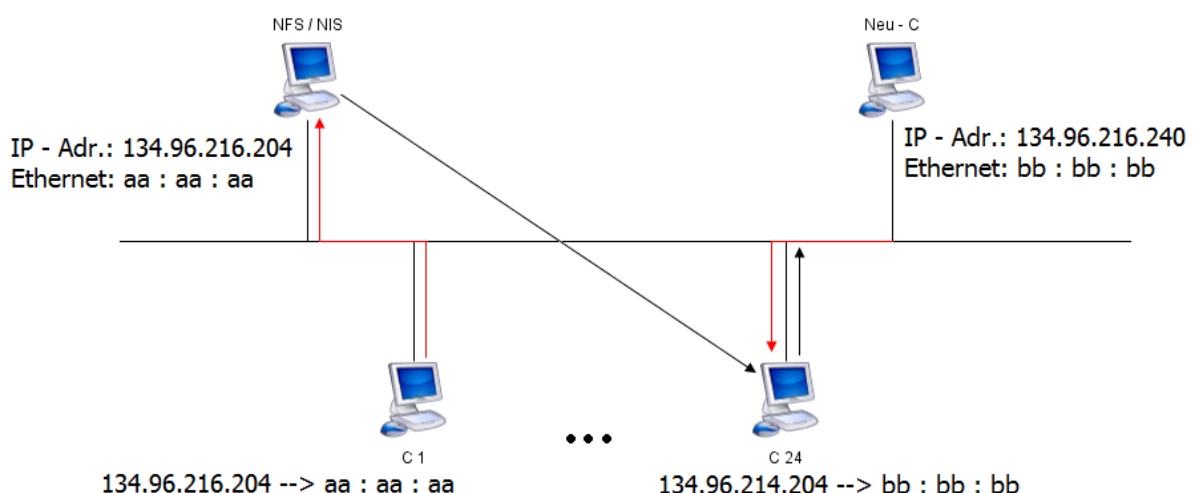
Er erwartet **keine** Antwort.

Aber falls eine Antwort kommt, benutzt noch ein Host diese / seine IP – Adresse und er wird folgende Fehlermeldung ausgeben:

duplicate IP – Address sent from Ethernet:      xx : xx : xx : xx : xx

**Beispiel:**

stl – s – stud



## 2) Aktualisierung der ARP – Tabellen aller Host eines LAN

Ein Server besitzt 2 Netzwerkkarten, eine ist aktiv, die andere ist im Standby – Modus.

Geht die erste Karte kaputt, so wird die 2. Karte aktiviert.

Früher waren die Netzwerkkarten **nicht** konfigurierbar was die Ethernet – Adresse betrifft.

Der Server gibt im Falle des Kartenwechsels via ARP – Reply Ethernetbroadcast seine neue

Ethernet – Adresse bekannt.

## Einschub: Angriffe aus dem lokalen Netz

70 % aller Netzwerkangriffe kommen von innen!

- **Mac Spoofing:**  
der Angreifer benutzt eine fremde MAC – Adresse  
**Abwehrmaßnahme:**  
Port - Security
- **MAC – Flooding (CAM Flooding):**  
der Angreifer versendet tausende MAC – Adressen an den Switch; dessen CAM – Tabelle läuft über → Switch schaltet eventuell in den HUB – Modus
- **ARP – Sopofing (ARP – Poisoning):**  
der Angreifer versendet falsche ARP – Nachrichten

## Statische ARP – Einträge

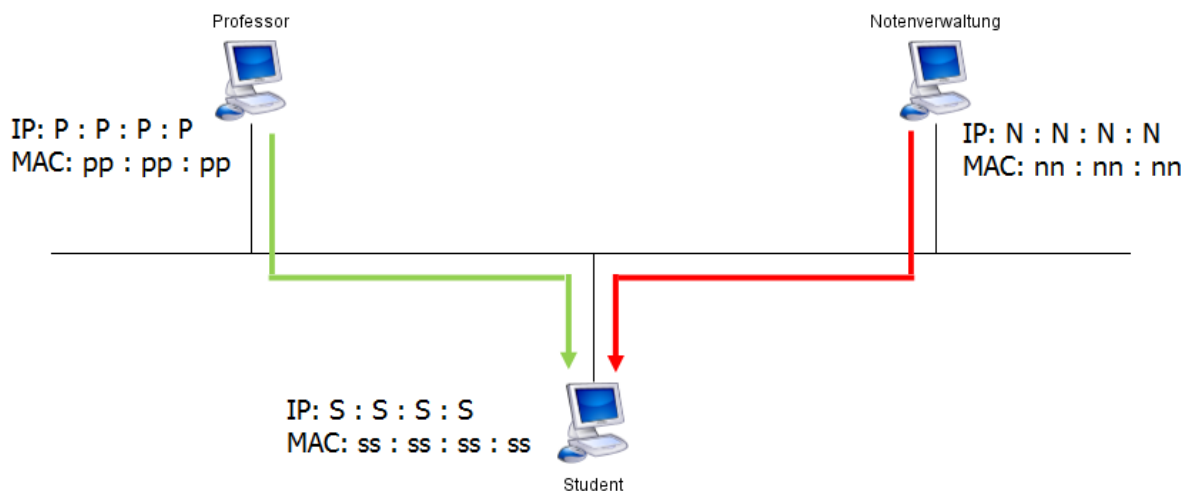
Administrator trägt für alle wichtigen Server und Router das IP – MAC – Adresspaar in den ARP – Cache statisch ein

→ **SCHEINLÖSUNG!!**

Bei den meisten IP – Implementationen überschreiben die ARP – Reply – oder Request – Pakete auch manuell angelegte statische ARP – Einträge. Daher bieten statische ARP – Definition in der Regel keinen Schutz gegen *ARP Spoofing*.

## „Man in the Middle Attack“

ettercap





- 1) P will Noten an N senden
- 2) S sendet ARP – Replies  $\rightarrow$  an P  $\rightarrow$  N : N : N : N = ss:ss:ss  
 $\rightarrow$  an N  $\rightarrow$  P : P : P : P = ss:ss:ss
- 3) P schaut in seinem ARP – Cache nach N:N:N:N, findet ss:ss:ss, sendet IP – Datagramm an S

### 3.6 IP – Datagramme, Routing, Frequentierung

#### Erinnerung:

der TCP / IP – Protokollstack hat die Aufgabe dafür zu sorgen, dass Anwendungsprogramme die auf verschiedenen Rechnern laufen, Daten austauschen können **ohne** sich die Hardware – Implementierungsdetails der zwischen ihnen liegenden Netzen zu kümmern.

IP = Internet – Protocol

**TCP / IP – Software benutzt einmal verbindungslose Protokolle (IP / UDP) oder verbindungsorientierte Protokolle (TCP).**

Der grundlegende Zustelldienst ist verbindungslos, er ist nur für den Datentransport und das „Wege finden“ (Routing) zuständig  $\rightarrow$  IP

IP ist ein unzuverlässiger, verbindungsloser Datagramm – Transportdienst.

#### **unzuverlässig heißt:**

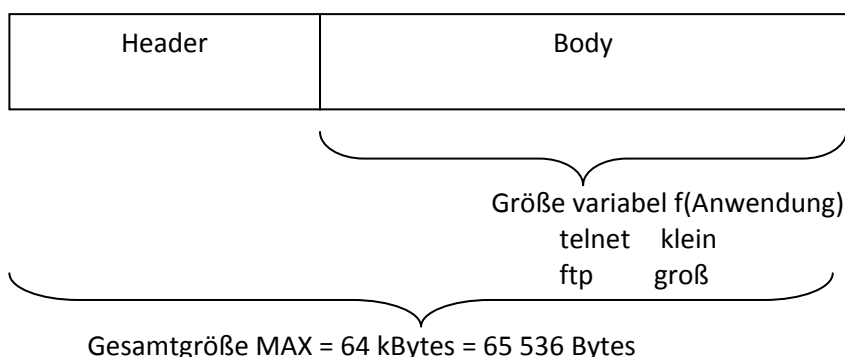
IP gibt keine Garantie, dass ein IP – Datagramm beim Ziel ankommt. Wird das Datagramm irgendwo verworfen, wird der Sender mittels ICMP (Internet Control Message Protocol) darüber informiert.

#### **verbindungslos heißt:**

es gibt keine Statusinformationen über IP - Datagramme

IP ermöglicht es einem Sender einzelne Datenpakete ins Netz zu übertragen, wobei jedes IP – Datagramm unabhängig mit seiner Information durch das Netz zum Empfänger läuft.

#### 3.6.1 Der Aufbau des IP – Datagramms



### 3.6.2 Das Paketformat

- 32 Bit – Zeilen von oben nach unten gelesen

#### Warum?

- 32 Bit – Zeilen vereinfachen die programmtechnische Verarbeitung des Datenpaketes
- das erste Wort ist dasjenige, das als erstes übertragen wird
- das höchstwertige Byte ist das erste Byte eines jeden Wortes

0	4	8	16	19	24	31
Vers	H. Len	Service Type	Total Length			
Identification			Flags	Fragment Offset		
Time to live		Type	Header Checksum			
Source IP Address						
Destination IP Address						
IP Options (können weggelassen werden)					Padding	
Beginn der Nutzdaten						
.						
.						
.						

VERS – 4 Bit → Version der benutzten IP Software : IPv4

H. LEN – 4 Bit → Anzahl der 32 Bit – Zeilen im Header  
 $2^4 = 15 \times 4 \text{ Bytes} = 60 \text{ Bytes}$  maximale Größe des IP – Header  
 (20 Bytes ist IP – Standard – Header)

SERVICE TYPE (Tos –Bit) - 8 Bit = 1 Byte

die ersten 3 Bit ungenutzt

dann die 4 TOS – Bits:

immer nur ein Bit gesetzt

- Verzögerung minimieren
- Durchsatz maximieren
- Zuverlässigkeit maximieren
- Kosten minimieren

das letzte Bit ist immer = 0

wird von den meisten IP – Implementierungen **nicht** unterstützt!

TOTAL LENGTH 2 Bytes → Länge des gesamten IP – Datagramms  
 Länge MAX =  $2^{16} = 65\,535 \text{ Bytes}$

IDENTIFICATION 3 Bytes → Zahl, die das IP – Datagramm eindeutig kennzeichnet

FLAGS / FRAGMENT OFFSET später

TIME TO LIVE 1 Byte → Zahl, die die Obergrenze der möglichen Routerübergänge angibt  
 Ein Router dekrementiert beim Weiterleiten dieses Feld. Wird die Null erreicht, wird das IP – Datagramm weggeschmissen und eine ICMP an den Sendehost geschickt.  
 Zahl = f(Betriebssystem)

P – TYPE 1 Byte → ↑ /etc/ protocols	es gibt an, an welches höhere Protokoll die IP – Daten im Body weiterzureichen sind
HEADER CHECKSUM →	Berechnung siehe RFC 1071 nur über die Header – Zeilen die transportierenden Protokolle, wie UDP / TCP / ICMP haben eigene Prüfsummen
SOURCE – IP – Address →	Sender – IP – Adresse
DESTINATION – IP – Address →	Empfänger – IP – Adresse
IP – OPTIONS →	optional, selten benutzt
PADDING →	füllt nach den Optionen eine 32 Bit Zeilen vollständig auf

### 3.6.3 Entstehung eines IP – Datagramms

#### Fall A:

Ein Anwendungsprogramm produziert einen Datenstrom, dieser wird der TCP – Software übergeben, die TCP – Software zerteilt den Datenstrom in sogenannte TCP – Segmente, die von der IP – Software jeweils in ein IP – Datagramm verpackt werden. Das IP – Datagramm wird dann versendet.

Die TCP – Software erzeugt TCP – Segmente die so groß sind, dass die in eine IP – Datagramm hinein passen, das in einen Hardware – Rahmen versendet werden kann.

$$\text{TCP – Segmentlänge} = f(\text{Hardware – Technologie})$$

|  
mtu 1500

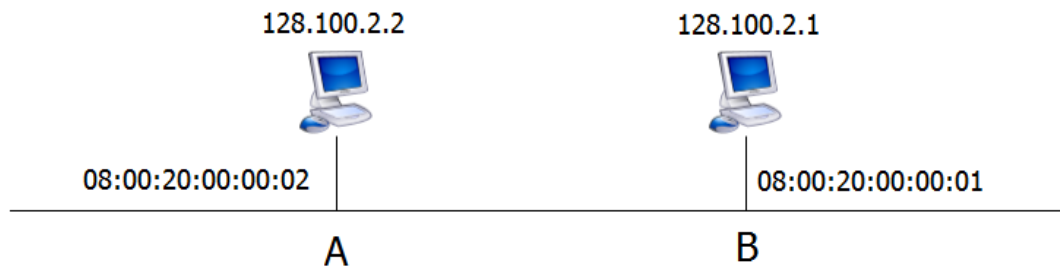
#### Fall B:

Ein Anwendungsprogramm produziert Nachrichten, die über die UDP – Software versendet werden.

Die UDP – Software erstellt pro Nachricht ein UDP – Datagramm, dieses wird von der IP – Software in ein IP – Datagramm verpackt und dann eventuell „stückchenweise“ das heißt fragmentiert via Hardware – Rahmen versendet.

**Wohin versendet der Host die Hardware – Rahmen?****Fall 1:**

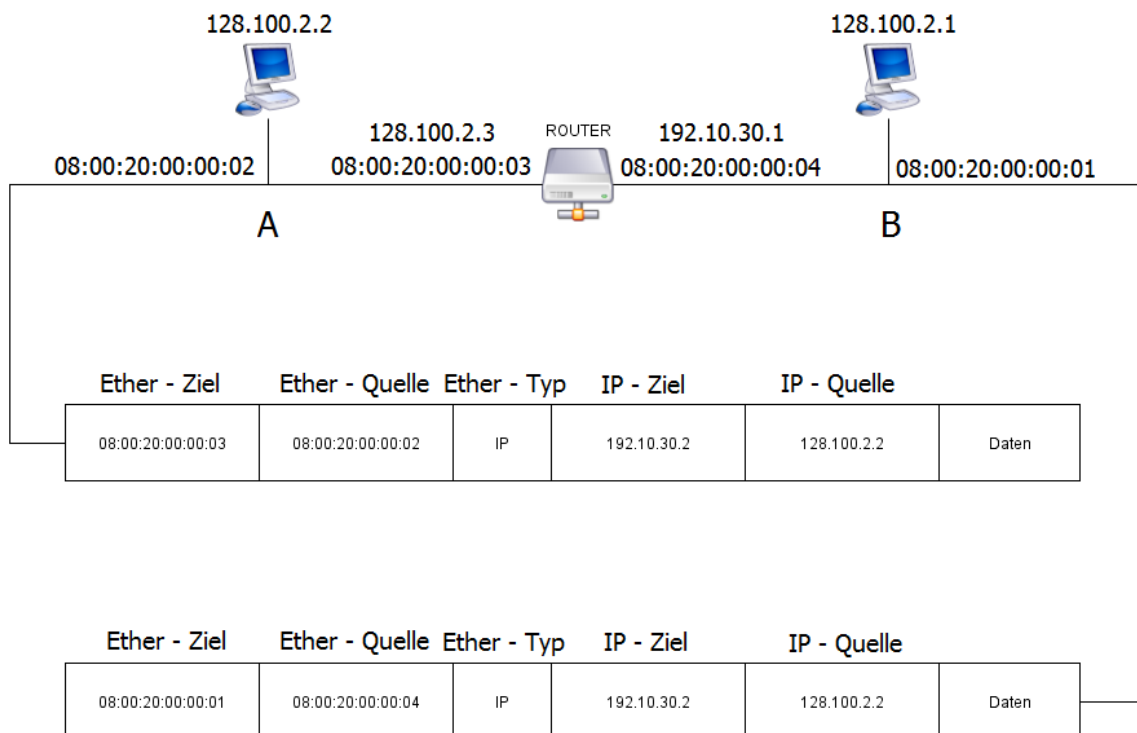
Zielhost im gleichen IP – Netzwerk = gleiches LAN



Ether - Ziel	Ether - Quelle	Ether - Typ	IP - Ziel	IP - Quelle	
08:00:20:00:00:01	08:00:20:00:00:02	IP	128.100.2.1	128.100.2.1	Daten

**Fall2:**

Zielhost in einem anderen IP – Netzwerk = anderes LAN



Der Host trifft die 1. Routingentscheidung selbst.

```

1
2  if (Empfänger in meinem LAN)
3  {
4      eventuell ARP - Kommunikation
5      IP - Datagramm direkt via Ethernet - Frame an Ziel senden
6  }
7  else
8  {
9      eventuell ARP - Kommunikation
10     IP - Datagramme an DEFAULT - Router senden
11 }

```

**Router:**

verbinden verschiedene IP – Netzwerke die eventuell über verschiedene Hardware – Technologien verfügen

**Problem:**

jede Hardware – Technologie hat ihre eigenen Vorstellungen über die beste Rahmengröße  
ATM – MTU → 41 Bytes

**Lösung:**

- A) IP – Datagramme klein genug wählen
- B) IP – Datagramme werden aufgeteilt und wieder zusammengefügt
- C) Ermittlung einer Path – MTU

**Wer fragmentiert wann?**

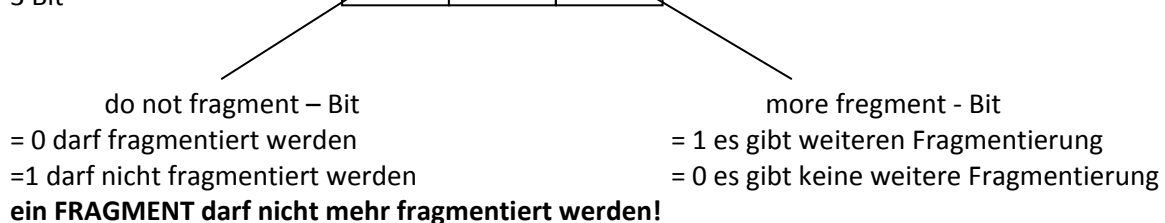
- Ein Host teilt, falls nötig, IP – Datagramme gemäß seiner lokalen Hardware – Technologie (Ethernet MTU = 1500 Bytes) auf, der Zielhost fügt das IP – Datagramm wieder zusammen (UDP).
- Ein Router fragmentiert ein IP – Datagramm, wenn die Netzwerke zwischen denen er routet unterschiedliche MTU's benutzen.
- Ein Router fügt nie ein IP – Datagramm wieder zusammen
- Jedes IP – Datagramm Fragment ist ein vollständiges IP – Datagramm, das unabhängig von seinen Geschwister – Fragmenten durch das Netz wandert

**3.6.4 Fragmentierungsgrundlagen**

- jedes Fragment hat das gleiche Format wie das ursprüngliche IP – Datagramm; das heißt **alle** Headerfelder, die ursprünglichen IP – Datagramme werden unverändert in den IP – Header der Fragmente übernommen. Das FLAGS und das FRAGMENT OFFSET Feld werden geändert.

**FLAGS Feld**

3 Bit



**Fragment Offset Feld:**

bezeichnet die genaue Lage der Fragment – Daten im Original – IP – Datenbereich.  
Anhand der Byte – Nummern des ersten Bytes im Fragment.

alle Fragmente tragen die gleiche **Identification**.

**3.6.5 Zusammenbau eines IP – Datagramms (defragmentieren)**

der Zielhost sammelt alle Fragmente und zählt die Anzahl der empfangenen Datenbytes. sind alle Datenbytes da, dann wird das Original – IP – Datagramm erzeugt.

letztes IP – Fragment = Fragment Offset + Aktuelle (Totallength – H – Leng)

**Experiment:**

ping –s ZIELHOST 4500 1

ICMP

HW	4500
----	------

IP - Header	BODY
-------------	------

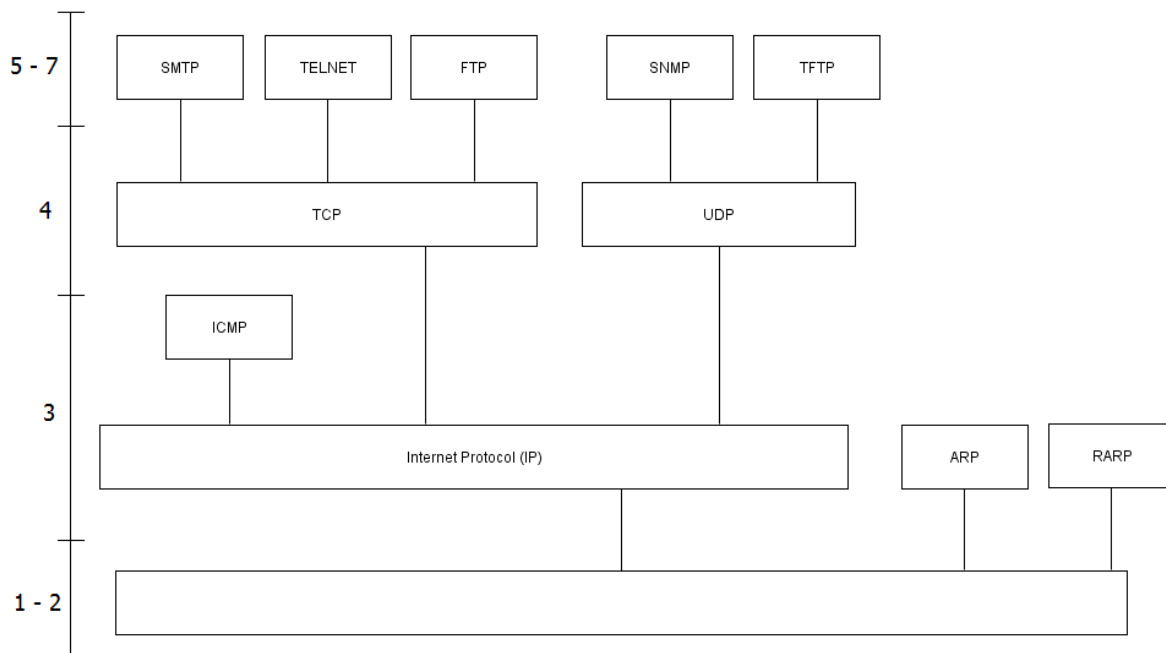
Eth. - Header	////////////////////////////////////
---------------	--------------------------------------

snoop – ta – v ZIELHOST | tee /tmp / proto

Gesamtlänge	Headerlänge	Fragment - Offset	More – Fragment	Do not Fragment
1500	- 20 = 1480	0	1	1
1500	- 20 = 1480	1480	1	1
1500	- 20 = 1480	2960	1	1
88	- 20 = 68	4440	0	1
Gesamt:	<b>4508</b>			

**3.7 Das ICMP – Internet Control Message Protocol**

- beschrieben in RFC 792
- ist Bestandteil der IP – Schicht (Layer 3)
- bietet der IP – Quelle, dem sendenden Host, die Möglichkeit zu erfahren, **warum** ein
- IP – Datagramm **nicht** zugestellt wurde
- hilft der IP – Schicht Fehler zu erkennen
- ICMP wird über IP transportiert



### 3.7.1 Das ICMP – Datenpaket

Typ – Feld = 1 Byte :

Typkennzeichen 0 ... 15

Code – Feld = 1 Byte:

Codekennzeichen zum Type

Checksum = 4 Bytes:

Prüfsumme über das ICMP – Paket

Datenbereich = 4 Bytes:

$f(\text{typ} + \text{code})$

↓  
IP – Header + 8 Daten Bytes

↓  
TCP und UDP – Header

### 3.7.2 Die wichtigsten ICMP – Nachrichten

**Type 4** source quench (Quelle dämpfen)

Code – 0

- ein Router sendet diese Meldung, wenn seinen Puffer ein Überlauf droht  
→ die empfangenden Hosts werden ihre Senderate senken
- ein Host sendet die Meldung, wenn IP – Datagramme schneller ankommen als sie verarbeitet werden können

Beispiel:      netio    stl-s-stud      → Client  
                                 1 GBit                      → 100 MBit

**Type 11** time exceeded

Code 0:

- ein Router sendet diese Meldung, wenn er das Time to Live – Feld eines weiterzuleitenden IP – Datagramms auf 0 dekrementiert hat und deshalb das IP – Datagramm wegwirft

Code 1:

- ein Host sendet diese Meldung, wenn sein intern Defragmentierungstimer abläuft bevor alle IP – Datagramm – Fragmente eingetroffen sind

**Type 3 destination unreachable**

Code 0 / 1:

- sendet ein Router, wenn er das Zielnetzwerk oder den Zielhost nicht erreichen kann

Code 3:

- sendet ein Host, wenn der angesprochene Server – Dienst nicht läuft

**Type 4 fragmentation required**

- sendet ein Router, wenn die MTU seines Zielnetzes kleiner der MTU des Sendenetzes ist, er das IP – Datagramm Fragmentieren müsste aber das Do – Not – Fragment – Bit gesetzt ist

**Type 8**Code 0: **echo request****Type 0**Code 0: **echo reply**

- sendet ein Host ein Echo – Request, so sollte der Empfänger – Host mit einem Echo – Reply antworten (ping)

**3.7.3 ICMP - Anwendungen**

- das Kommando ping → man ping

ping Host *is alive*

→ die IP – Software auf dem Host läuft und funktioniert fehlerfrei

→ es gibt einen Netzweg zum Host

*no answer from*

→ die IP – Software läuft nicht

→ es gibt keinen Weg

**Solaris:**

ping -s	host	paketgröße	anzahl
ping -s		4500	1

```

3 for TTL is 1 2 ...
4 do
5     ping -s -t TTL host 50 1
6     ↑
7     Zahl
      1 2
done

```

**3.7.4 Das Kommando traceroute (Unix / Linux)**

- funktioniert oberflächlich betrachtet wie die for – ping – Schleife
- ist UDP basiert, arbeitet mit den beiden ICMP – Meldungen
  - Time Exceeded
  - Port Unreachable
- traceroute sendet UDP – Datagramme via IP – Datagramme mit TTL = 1...n an die Zieladresse, dort an unbenutzte UDP – Ports > 33434



1. UDP – Dg	TTL = 1	Zielport	33434
2. UDP – Dg	TTL = 1	Zielport	33435
3. UDP – Dg	TTL = 1	Zielport	33436
4. UDP – Dg	TTL = 2	Zielport	33437
5. UDP – Dg	TTL = 2	Zielport	33438
6. UDP – Dg	TTL = 2	Zielport	33439

- traceroute sendet solange UDP – Datagramme mit aufsteigender TTL, solange die Meldung „Time Exceeded“ zurückgesendet wird
- kommt ein „Destination Unreachable“ Meldung ist der Zielhost erreicht  
→ auf dem Zielhost wurde ein UDP – Port angesprochen, hinter dem keine UDP – Server – Software läuft

### 3.7.5 Feststellen der Path – MTU

man versendet IP – Datagramme mit gesetztem Do – not – Fragment – Bit und reduziere deren Größe, falls eine Fragmentation - Required – Meldung von einem Router eintrifft

## 3.8 Routing – Begriffe, Verfahren, Protokolle

- Router und Routingtabellen arbeiten auf Layer 3

### Direct Routing

die kommunizierenden Hosts liegen im gleichen IP – Netzwerk ((V)LAN)

### Indirect Routing

die kommunizierenden Hosts liegen in verschiedenen IP – Netzwerken ((V)LANs)

### Default Routing

in der Routing –Tabelle des Hosts / Routers existiert ein Default – Router / Gateway

### statisches Routing

- an der HTW genutzt
- die Routing – Tabelle wird beim Systemstart nach Administratormaßgabe aufgebaut und nicht mehr verändert

( ifconfig + /etc / default router  
ipconfig + Standard Gateway )

### **Wann:**

- Netz klein
- es gibt keine redundante Routen
- es gibt nur einen Router im Netzwerk über den man ins Internet kommt

### dynamisches Routing / adaptives Routing

die Routingtabelle wird nach dem Startup durch die Kommunikation mit anderen Routern immer wieder angepasst

RIP 1	}	Routing Information Protocol	(RFC 1058)
RIP 2			
OSPF		Open Shortest Path First	(RFC 1247)
BGP		Border Gateway Protocol	(RFC 1771)
			[65]

### 3.9 Die Transportschicht und ihre Ende – zu – Ende Protokolle

Aufgabe der Transportschicht ist es mit Hilfe der Internetschicht (Layer 3) dem Host – zu – Host – Datagramm Übertragungsdienst, einen Prozess – zu – Prozess – Kommunikationsdienst aufzubauen.

**Prozess:**

ein im Ablauf befindliches Anwendungsprogramm

**Probleme der Internetschicht / IP - Software**

- verwerfen von IP – Datagrammen
- Umstellung in der Reihenfolge der IP – Datagramme
- Übertragen von mehreren Datagramm – Kopien
- Einschränkung bei der Datagramm – Größe
- beliebig lange Verzögerung

**Was möchte der Anwendungsprogrammierer?**

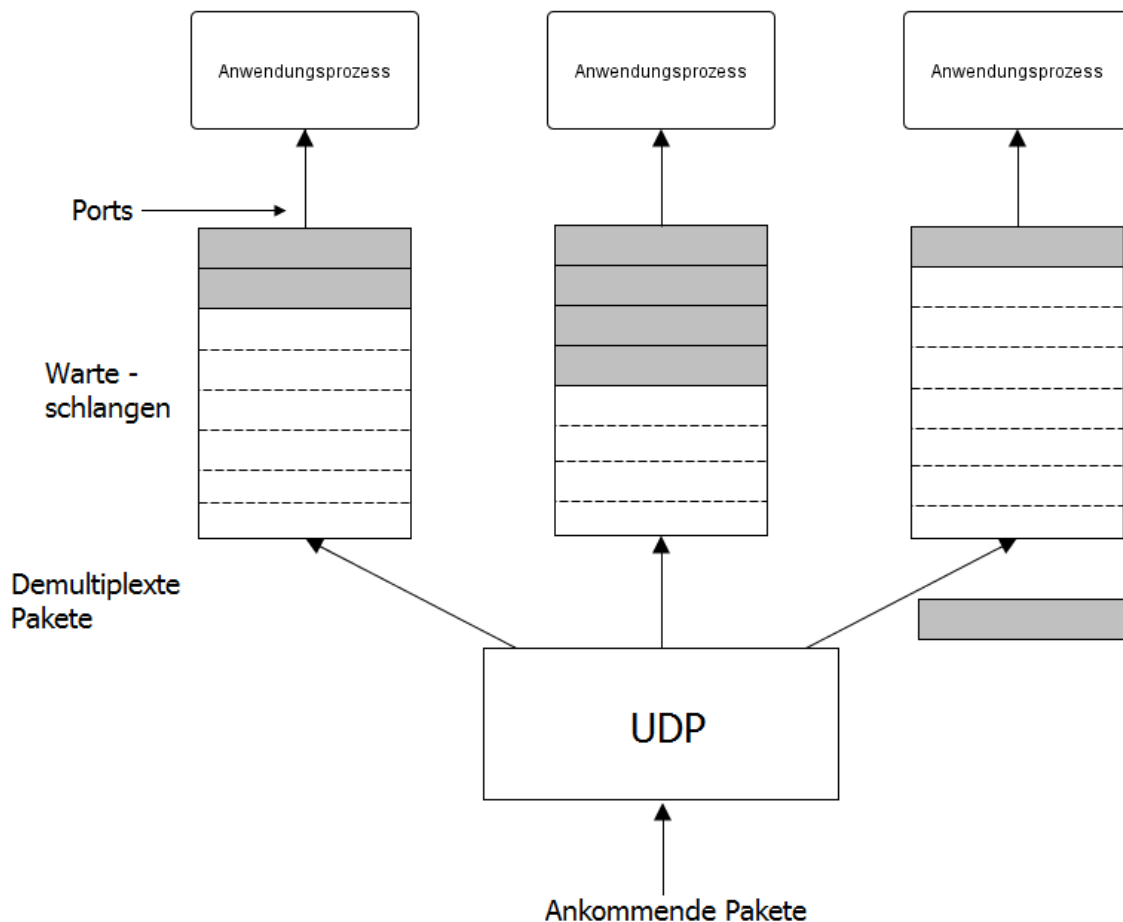
- garantierte Übertragung der Daten
- Übertragen der Daten in der richtigen Reihenfolge
- Unterstützung beliebig „großer“ Datenmengen
- Unterstützung der Synchronisation von Sender und Empfänger
- Unterstützung mehrerer Netzwerkprozesse auf einem Host

**Mögliche Transportsdienste im TCP / IP – Protokollstack**

- einfacher asynchroner Demultiplexdienst → UDP
- zuverlässiger Bytestromdienst → TCP
- Frage - / Antwortdienst → RPC – Protokolle (SUN RPC, DCE RPC)

#### 3.9.1 UDP - Der Verbindungslose Transportsdienst

- beschrieben in RFC 768 – User Datagram Protocol
- UDP erweitert den Host – to – Host – Datagrammdienst von IP auf einen Prozess – zu – Prozess – Datagrammdienst
- UDP sorgt dafür, dass mehrere Prozesse die auf einem Host laufen quasi parallel mit anderen Prozessen auf entfernten Hosts kommunizieren können
- UDP Multiplext (beim Senden) ein Demultiplext (beim Empfangen) der Datenströme der Anwendungsprogramme
- UDP ist unzuverlässig und verbindungslos wie IP
- UDP wird über IP transportiert
- die Prozesse werden über Portnummern identifiziert



## EXKURS: die Begriffe Port und Socket

Port = Hafen / Tor → Ziel

ist eine 16 – Bit – Adresse zu Identifizierung eines eindeutigen Zugangspunktes zwischen einem Prozess und der darunter liegenden Transportprotokollsoftware (TCP / UDP).

Diese sogenannten Protokoll – Portnummern ( $2^{16}-1$ ) stehen auf einem Rechner pro Transportprotokoll zur Verfügung.

TCP 1...65535

UDP 1...65535

Stdin Kanal 0

Dateidescriptor

Stdout Kanal 1

Stderr Kanal 2

## Protokollnummern – Aufteilung

### 1) well known parts:

1 bis 1023 „Server – Standard – Ports“ von der IANA festgelegt

### **Besonderheiten:**

- unter UNIX / LINUX nur durch Superuser – Prozess nutzbar
- Ports 256 bis 1023 sind UNIX spezifischen Diensten vorbehalten  
Beispiel:
  - rlogin
  - ssh

**2) registrierte Ports: 1024 – 49151**

Sollten nicht ohne Registrierung bei der IANA als Server – Ports benutzt werden

**3) dynamische / private Ports: 49152 – 65535**

frei nutzbare Ports ohne Registrierung bei der IANA

vorzugsweise Clientports

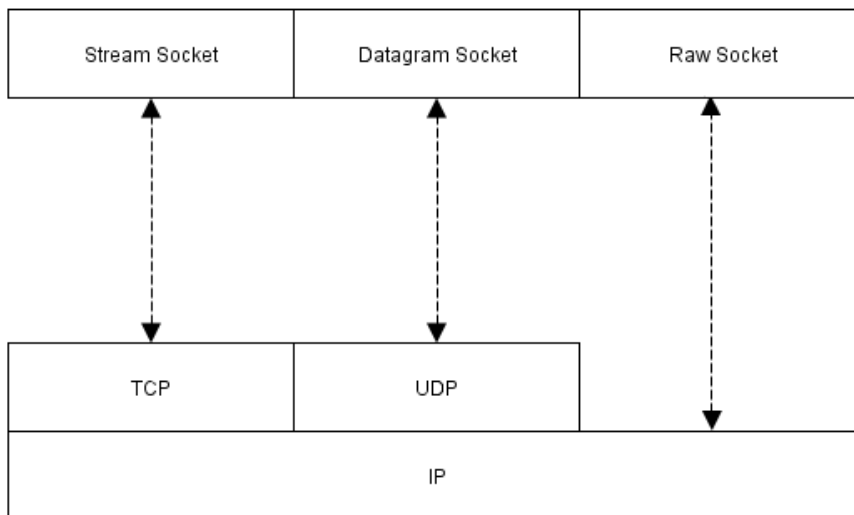
**Socket**

ist das weltweit eindeutige Tripel aus IP – Adresse + Transport – Protokoll + Portnummer, das auf Layer 4 einen Kommunikationsendpunkt beschreibt.

[ die weltweit eindeutige Adresse eines laufenden Programmes ]

Bei Programmierung von Netzwerkprogrammsystemen, bestehend aus Client – und Server – Software, wird oft die sogenannte Socket – Programmierschnittstelle verwendet.

**Ein Socket dort ist ein Kommunikationskanal!**



**Zur Abbildung:**

**Stream Socket:**

Sie greifen auf den Transportdienst des TCP – Protokolls zu, wobei der stattfindende Datenfluss auf einer sicheren Verbindung zwischen zwei Sockets realisiert wird.

**Datagram Socket:**

Analog erfolgt hier der Transport von Datennach den Gesetzmäßigkeiten des UDP – Protokolls: schneller und weniger zuverlässig.

**Raw Socket:**

Dieser Typ ermöglicht den direkten Zugriff auf die Netzwerkschicht Internet Protocol.

Version	Headerlänge	Type of Service(TOS)	Länge des Datagramms	
Identifikator			Flags	Offset des Fragments
Time to Live (TTL)		Protokolltyp	Prüfsumme des Headers	
IP- Adresse des Absenders				
IP – Adresse des Empfängers				
IP – Optionen				
Portnummer des Senders			Portnummer des Empfängers	
UDP - Länge			UDP - Prüfsumme	
UDP – DATEN				

IP Header

UDP Header

### 3.9.1.1 Aufbau eines UDP – Datagramms

Sendeporrtnummer = 2 Bytes

Empfängerportnummer = 2Bytes

UDP – Länge → Längenangabe für das UDP – Datagramm in Anzahl Datenbytes  
 $2^{16} - 1 = 65535 - \text{IP – Header} - \text{UDP – Header}$   
 $\qquad\qquad\qquad 20 \qquad\qquad\qquad 8$   
 $= 65507 \text{ Datenbytes}$

UDP – Prüfsumme = 2 Bytes

→ wird gebildet über UDP – Header + UDP – Body  
+ IP – Quell – Adresse  
+ IP – Ziel – Adresse  
+ IP – Protokolltypfeld

UDP ist am häufigsten von Fragmentierung betroffen, denn ein UDP – Datagramm wird in ein IP – Datagramm verpackt und versendet  
die UDP – Datagramm – Größe = f (Anwendung)

#### Beispiel:

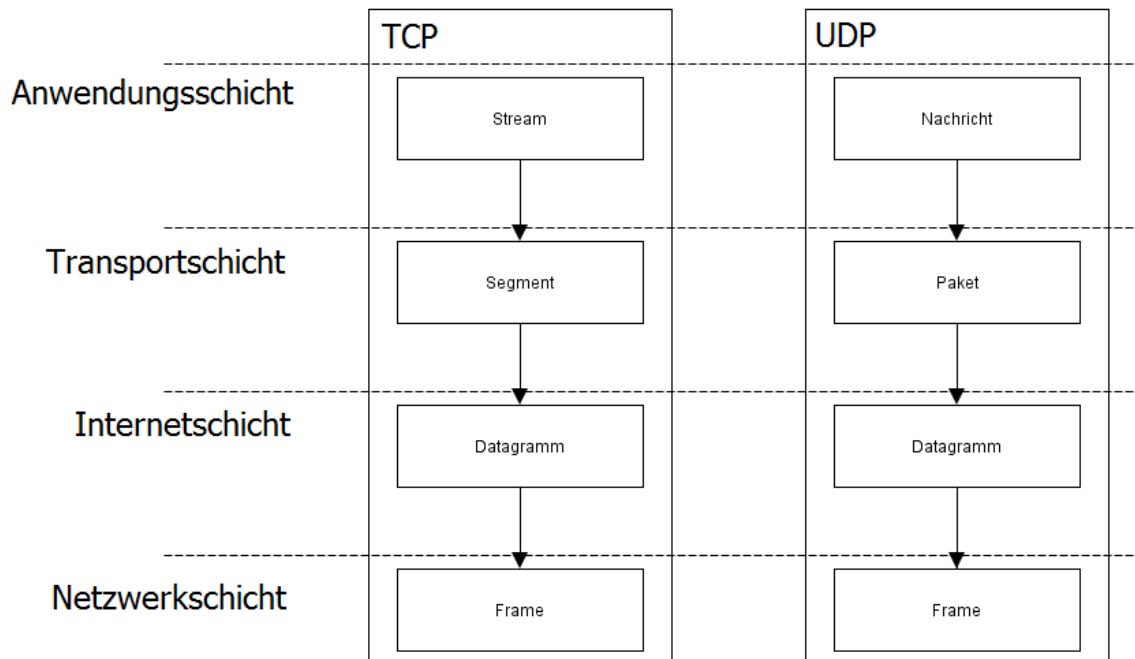
DNS versendet 512 Bytes – Daten via UDP

NFSv2 versendet 8192 Byte – Daten via UDP

### Warum nutzen Anwendungsprogramme UDP als Fragmentdienst?

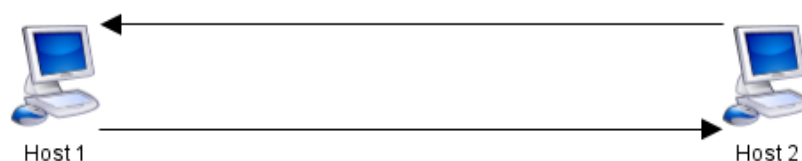
- UDP ist schnell? → geringe Datenmengen mittels eines Datagramms versendbar (bei TCP mindestens 6 Datenpakete)
- werden nur geringe Datenmengen auf einmal versende (Nachrichten)
- Anwendung regelt die Zuverlässigkeit selbst (NFSv1/2)
- wenn mehrere Anwendungsprogramme im Netz die gleiche Nachricht erhalten sollen (Multicast / Broadcastanwendung)

- Anwendung arbeitet mit einem Frage – Antwort – Schema  
**Antwort = positive Bestätigung der Frage**  
 kommt **keine** Antwort, so stellt man die Frage wieder



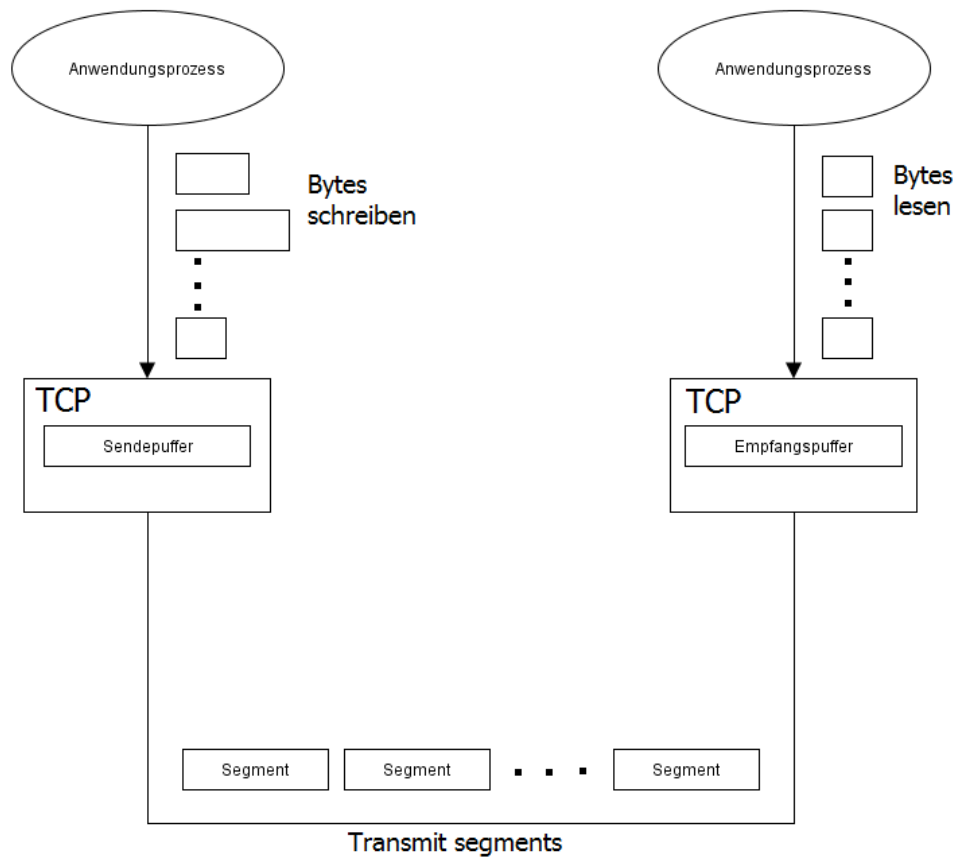
### 3.9.2 TCP – der zuverlässige Bytestrom – Dienst

- Transmission Control Protocol
- RFC 793, 761, 675...
- TCP besitzt einen zuverlässigen verbindungsorientierten, geordneten Bytestrom – Dienst
- TCP ist ein **Vollduplex – Protokoll**, jede TCP – Verbindung hat zwei Byteströme in jede Richtung einen.

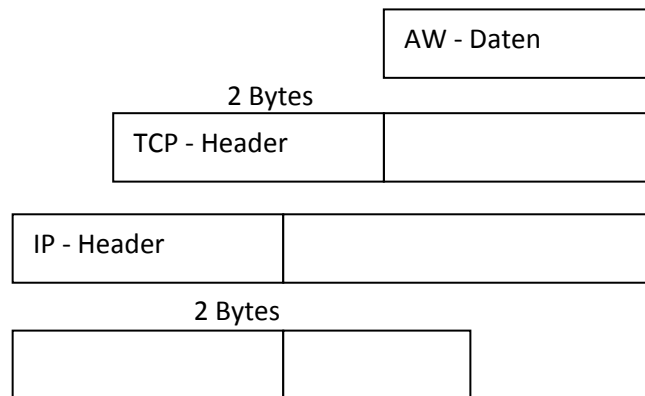


Die Byteströme können einzeln geschlossen werden, dann spricht man von halb – offenen / halb – geschlossenen TCP – Verbindungen.

- TCP ist auch ein Multiplex - / Demultiplex – Dienst (wie UDP)
- TCP implementiert eine Datenflusskontrolle, es hindert den Sender daran, den Empfänger mit Daten zu überfluten (Netio)
- TCP implementiert eine Netzwerk – Überlast – Kontrolle und verhindert somit, dass zu viele Daten ins Netz gesendet werden, die die Router überlasten würden
- Mittels TCP kommunizieren immer genau **zwei Anwendungsprogramme**
- TCP kann Broadcasting / Multicasting **nicht** nutzen



### 3.9.3 Der TCP - Header



Source Port – Nr.	2 Bytes = 16 Bit Die Portnummer des sendenden Anwendungsprogramms
Destination Port – Nr.	2 Bytes = 16 Bit Die Portnummer des empfangenden Anwendungsprogramms
Sequence – Nr.	4 Bytes = 32 Bit Byte – Nummer des ersten Datenbytes im Datenbereich des TCP – Segments, wenn Daten gesendet werden

Acknowledge – Nr.

4 Bytes = 12 Bit

Bestätigungsnummer des als nächstes erwarteten Empfangsbyte

Window – Size

momentan verfügbare sogenannte Fenstergröße

= momentane Größe des Empfangspuffer



- Sequence – Number
- Sende Daten
- Bestätigungs – Number
- Window – Size

Header – Length (4 Bit)

Anzahl der 32 Bit – Zeilen in TCP – Header  $2^4 - 1 = 15 \times 4$  Bytes  
= TCP – Header Länge Max = 60 Bytes

TCP – Checksumme

- 2 Bytes - wird berechnet über TCP – Header + TCP – Body
- Stimmen die vom Empfänger berechnete Checksum nicht mit der des Senders überein. so wird das TCP – Segment vom Empfänger verworfen, das heißt nicht bestätigt.

URG – Pointer

2 Bytes = 16 Bit

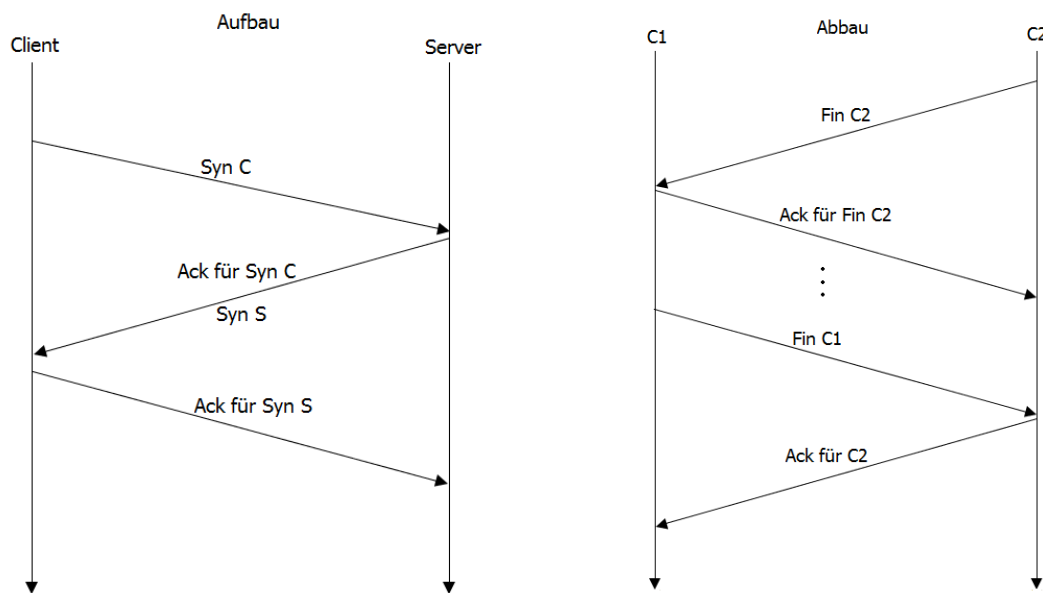
### 3.9.3.1 die 6 TCP – Flags je 1 Bit

- ACK  
Acknowledge – Number ist gesetzt
- PSH  
Daten möglichst schnell an Anwender freigeben
- SYN  
Synchronisations – Bit → TCP – Verbindungsaufbau
- FIN  
Ende – Segment → TCP – Verbindungsabbau
- RST  
Reset – Segment
- URG  
Urgent - Pointer gesetzt



### 3.9.3.2 Grundsätzliches zum Verbindungs- Auf / Abbau

- **aktives Öffnen** von TCP – Verbindung durch einen **Client**, das heißt der Client leitet eine TCP – Verbindung ein
- **passives Öffnen** einer TCP – Verbindung durch einen **Server**, das heißt der Server nimmt eine TCP – Verbindung an
- Der Verbindungsaufbau ist **asymmetrisch**
  - er besteht aus 3 TCP – Segmenten
- Der Verbindungsabbau ist **symmetrisch**
  - jeder Seite schließt ihren Write – Kanal
  - es werden 4 TCP – Segmente benötigt



in Syn C gibt der Client seine Sequence – Number und seine Window – Size und seine genutzte MSS an

MSS = Maximum Segment Size

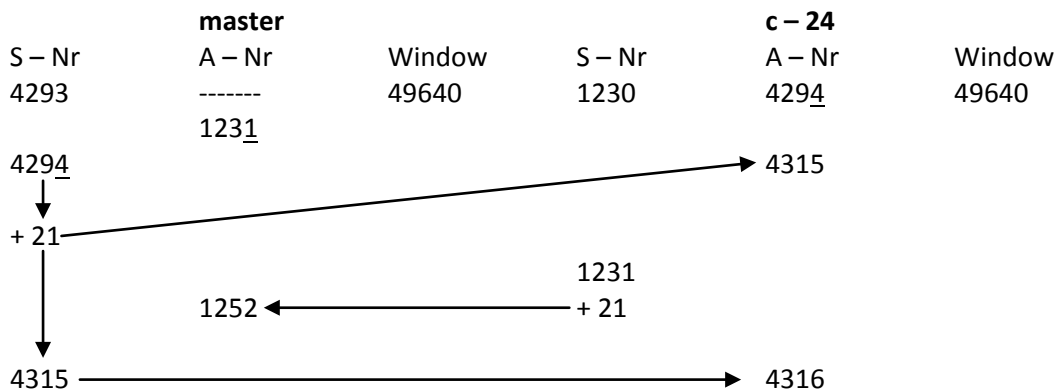
→	MTU	IP – Header	TCP – Header	= MSS
	↓	↓		
	1500	20	20	1460 Byte

### 3.9.3.3 Experiment zu TCP – Verbindungsauf - / abbau

Benutze echo – Server

strg + Altgr + ]

quit



### Wie erreicht TCP Zuverlässigkeit?

**benutzte Technik:**

Neuübertragung von nicht bestätigten TCP – Segmenten nach Sendetime – Ablauf

**Problem:**

wie lange muss der Sende – Timer laufen?

1 ms → 200 ms → 1000 ms ?

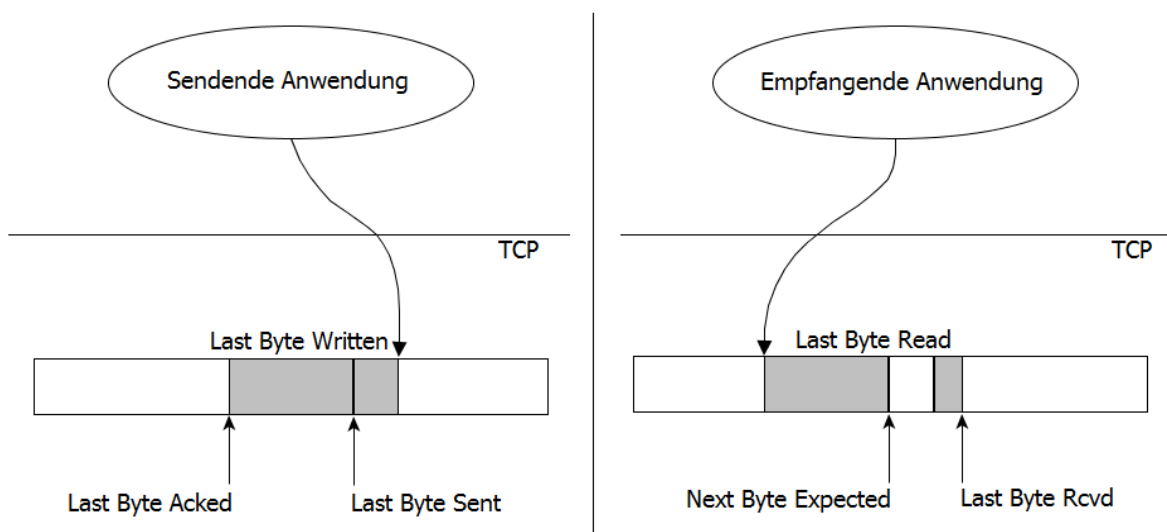
TCP schätzt die sogenannte Umlaufzeit, setzt sie am Anfang „ziemlich“ hoch an und berechnet bei jedem Bestätigungsempfang eine neue Timerlaufzeit.

$$\text{geschätzte } RTT_{NEU} = \alpha \cdot \text{geschätzte } RTT_{ALT} + (1 - \alpha) \cdot \text{geschätzte } RTT$$

### 3.9.3.4 Die TCP - Datenflusskontrolle

- Beim Verbindungsaufbau geben sich die Kommunikationspartner gegenseitig ihre Empfangspuffergröße bekannt
- Der Sender sendet so viele Daten wie er hat und in den Empfangspuffer des Empfängers hineinpassen, so schnell er kann **ohne** auf die Bestätigungen zu warten

Die TCP - Überlastkontrolle



**Was passiert, wenn der Empfänger ein Zero – Window empfängt?**

Der Empfänger sendet ein Zero – Window, wenn sein Empfangspuffer voll ist.

Der Sender stellt daraufhin seine Sendetätigkeit ein und sendet zyklisch 1 Byte – Segmente, die der Empfänger beantwortet, jeweils mit der aktuellen Fenstergröße. Wird diese > 0, so sendet der Sender wieder Daten.

**Wann sendet TCP ein Segment?**

- 1) Wenn genug Sendedaten im Sendepuffer vorliegen um ein TCP – Segment vollständig zu füllen  
MSS = MTU --- TCP – Header --- IP – Header
- 2) Wenn die sendende Anwendung es ausdrücklich fordert (TCP – Push – Operation)
- 3) Wenn der Sendetimer abgelaufen ist
- 4) zyklisch, wenn ein Zero – Window empfangen wurde

MSL = Maximum Sequent Livetime

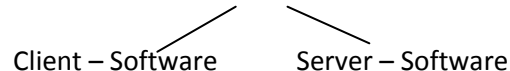
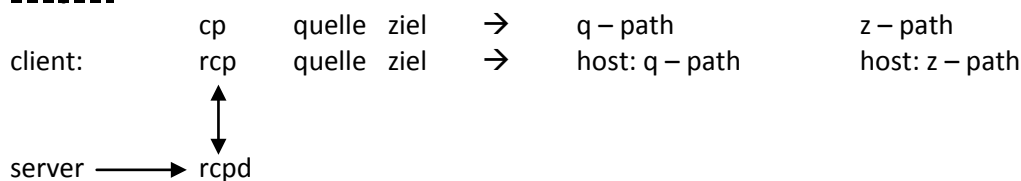
SUN OS 4.x / BSD / AIX → MSL = 30 sec

Solaris → 2 Min

TCP sendet, wenn ein unbesetzter TCP – Port via SYN – Segment angesprochen wird ein RST – Segment an den anfragenden Host

## 4 Netzanwendungen im UNIX – Umfeld

Netzanwendungen bestehen immer aus 2 Teilen

**Beispiel:**

### 4.1 einfache Standard – Dienste / - Server

Client: telnet → **Anwendungsprotokoll:**  
 sende String  
 empfangen und zeige String an

telnet host port – nr

**echo**

- beschrieben in RFC 862
- TCP / UDP Port 7 ← /etc/services

Der Server gibt die vom Client gesendeten Daten an den Client zurück, der Client muss die Verbindung beenden.

**chargen**

- beschrieben in RFC 804
- TCP / UDP Port 19

Der TCP – Server sendet einen ununterbrochenen Zeichenstrom bis der Client die Verbindung beendet.

Der UDP – Server sendet als Antwort auf ein Client – UDP – Datagramm ein UDP – Datagramm mit einer beliebigen Anzahl von Zeichen.

**discard**

- beschrieben in RFC 863
- UDP / TCP Port 9

Der Server liest alle Client – Daten, sendet nichts zurück.

Der Client beendet die Verbindung.

**daytime**

- beschrieben in RFC 867
- UDP / TCP Port 13

Der Server liefert bei Kontakt Uhrzeit und Daten und beendet die Verbindung.

**time**

- beschrieben in RFC 868
- UDP / TCP Port 37

Der Server gibt bei Kontakt die Systemzeit als 32 Bit Binärzahl zurück und beendet die Verbindung.

Die Binärzahl benennt die Anzahl der verflossenen Sekunden seit 01.01.1990 0.0 Uhr.

## 4.2 Das Client – Server – Paradigma

***Paradigma:***

Gesamtheit von Eigenschaften und Konzepten, die das Client – Server – Computing betreffen

Client → aktive Partner  
stellt Anfragen  
Server → passive Partner  
gibt Anfragen

**1. Form**

Client stellt Anfrage  
Server antwortet  
TP → TCP + UDP  
Anwendungen → www

**2. Form**

Client stellt mehrere Fragen im gleichen Kontext  
TP → TCP + UDP  
Anwendungen → ssh / rlogin / telnet

**3. Form**

Server bietet laufend Daten an  
Client nimmt Verbindung auf  
TP → UDP  
Anwendungen → Videokonferenz

**Merkmale der Client – Software**

- wird vom Benutzer aktiviert / deaktiviert
- läuft (meist) lokal auf dem Rechner des Benutzers
- leitet **aktiv** den Kontakt mit dem Server ein
- benötigt keine spezielle Hardware und kein spezielles Betriebssystem
- benutzt temporär zugewiesene Ports
- kann **mehrere Anwendungsprotokolle** unterstützen (Web – Browser: http / https / ftp...)

**Merkmale der Server – Software**

- wird automatisch bei Systemstart oder bei Bedarf gestartet
- läuft (meist) auf einem gemeinsam benutzten Rechner (Server)
- wartet **passiv** auf ankommende Kontaktierungen
- sollte auf einer leistungsfähigen Hardware und einem sicheren, stabilen Betriebssystem laufen
- benutzen normalerweise einen well known – Port beziehungsweise einen bei der IANA registrierten Port
- realisiert **ein** Anwendungsprotokoll

### 4.3 Protokolle, Ports & Sockets

Wie identifiziert TCP / IP einen Kommunikationskanal?

**Problem 0:**

Wie identifiziert TCP / IP einen Host weltweit eindeutig?

**Lösung 0:**

im TCP / IP – Header existieren Ziel - / und Quell – IP – Adresseinträge

**Problem 1:**

Identifizierung des Transportprotokolls TCP und UDP?

**Lösung 1:**

im IP – Header existiert ein Feld: Protokoll – Typ

Info – Quelle: /etc / protocols oder

IANA – Web – Seite

**Problem 2:**

auf einer Server – Maschine laufen mehrere Server – Programme / - Dienste.

Wie werden die einzelnen Dienste durch die Transportprotokolle adressiert / identifiziert?

**Lösung 2:**

jedes Server – Programm meldet sich beim startup mit seiner Protokollnummer beim entsprechenden **Transport – Protokoll** an.

Info : /etc / services oder

IANA – Web – Seite

**Problem 3:**

mehrere Clients greifen quasi parallel auf eine Serversoftware zu.

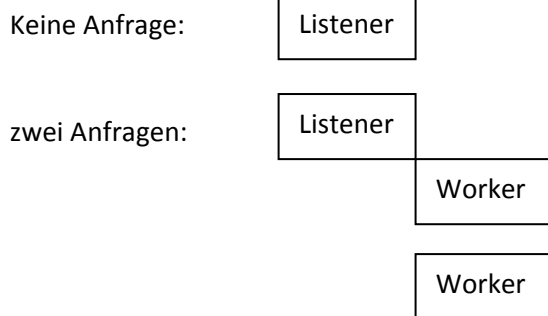
Der Server muss die Anfragen parallel bearbeiten.

**Lösung 3:**

→ ein Server besteht immer aus zwei Komponenten

- **Listener**
- **Worker**

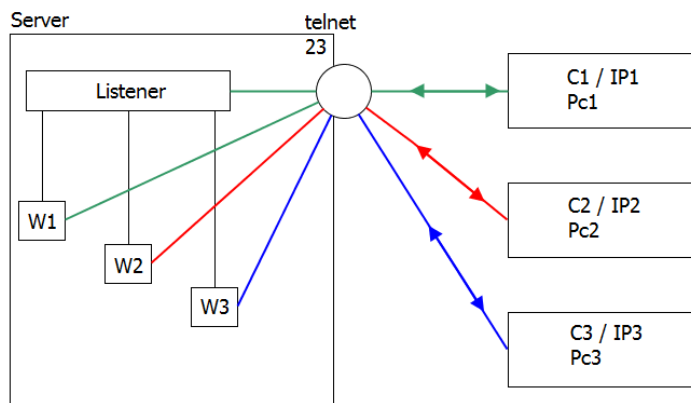
Der Listener wartet am Serverport auf Client – Verbindungswünsche. Trifft ein Verbindungswunsch ein, so startet der Listener einen Worker, übergibt diesen dem Client – Kommunikationskanal. Der Worker bearbeitet die Clientanfrage und beendet sich nach Erfüllung der Aufgabe.



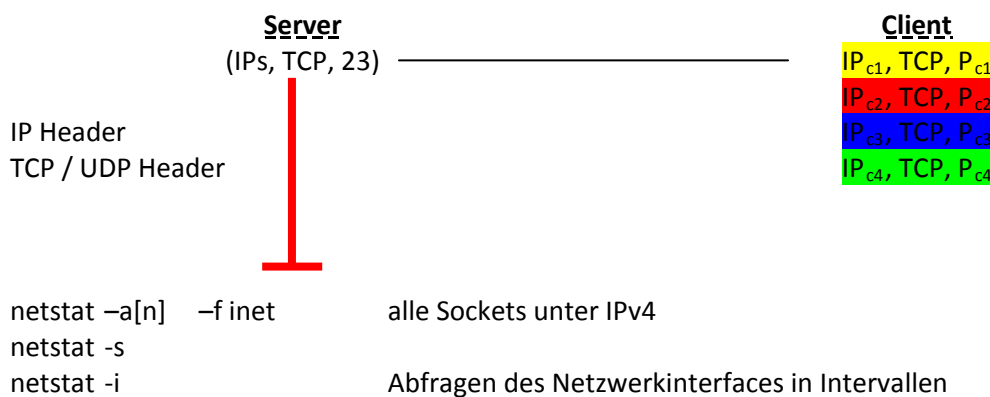
#### Problem 4 (folgt aus Lösung 3):

Es läuft auf einem Host ein **Listener** und n **Worker**.

Wie kann die Transportprotokollsoftware die einzelnen Kommunikationskanäle unterscheiden?



**Ein Kommunikationskanal wird durch seine Kommunikationsendpunkte weltweit eindeutig identifiziert!**



## 4.4 Dämonen und ihre Meister

Daemon → Dienst / Server – Programme

### Welche Serverprogramme / Daemons laufen auf einem Solaris- Rechner?

Es gibt einen Super – Daemon, der stellvertretend für viele Standard – Dienste **Listener** ist.

Es gibt Dienste, die werden bei Systemstart gestartet.

Linux → über rc – Prozeduren /etc / rcd

Solaris → svcs

Viele Dienste werden nur von Zeit zu Zeit benötigt, die laufen nicht dauernd, sondern werden bei Bedarf durch einen Super – Daemon gestartet.

Solaris → inetd + svcs

Linux → xinetd

Windows → Task Planung

} → konfigurierbar

### Beispiel:

Welche Sockets verwaltet der inetd?

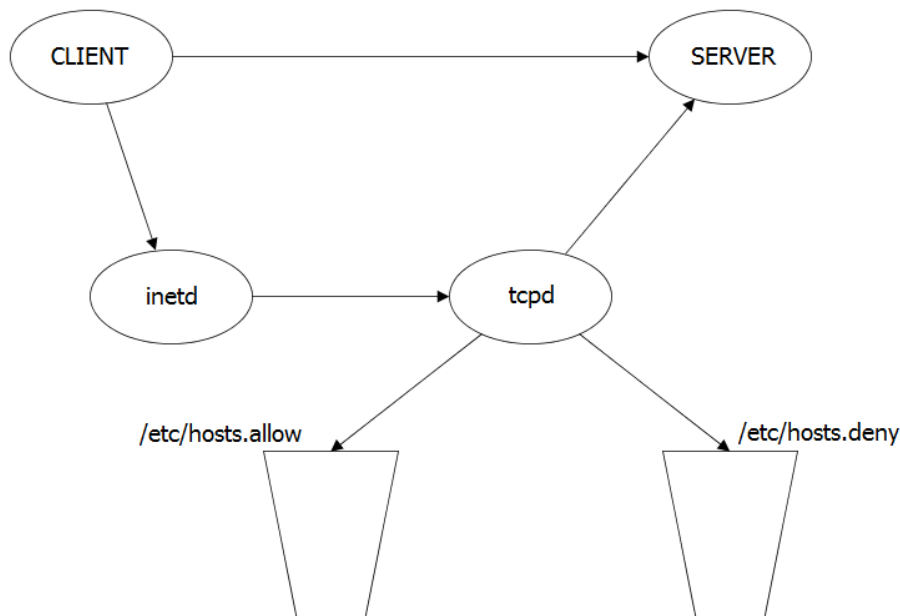
netstat -a -f inet → Listen

pgrep inet → Prozess – ID (PID)

pfiles PID → /tmp / xx

### Exkurs: Sicherheit - tcpwrapper

Alle Standarddienste arbeiten unverschlüsselt (/ rlogin / telnet...) und wie bei Serverprogrammen üblich nehmen sie jeglichen Verbindungswunsch an!



## 4.5 Die Socket – Programmierschnittstelle / Socket - API

Die Socket – API (= Application Programming Interface) ist eine Sammlung von C – Funktionen / Prozeduren / Java – Klassen, die es dem Programmierer erlaubt schnell effektive Clientserverprogramme zu schreiben.

Sockets → im Betriebssystem eingebaute Funktionen (ab 1983 in Unix realisiert)

Socket – Liberaires → Windows

### 4.5.1 Das Unix Ein - / Ausgabe – Konzept

#### 1. open ( ):

öffnet die Datei / Gerät, gibt einen (Datei - ) **Descriptor** zurück

#### 2.

##### a) read ( ):

liest via Descriptor Daten von Datei

##### b) write ( ):

schreibt via Descriptor Daten auf Datei

#### 3. close ( ):

schließt die Datei, gibt Descriptor frei

### 4.5.2 Sockets und das Unix Ein - / Ausgabe – Konzept

#### 1. socket ( ):

öffnet einen Socket, gibt ein **Handle** zurück

#### 2. read ( ) / receive ( ):

liest / empfängt Daten via Handle

#### 3. write ( ) / send ( ):

schreibt / sendet Daten via Handle

#### 4. close ( ):

schließt den Socket, gibt Handle frei

### 4.5.3 Serversoftware – Aufbau

#### socket ( ):

- liefert einen Socket – Descriptor ( = Handle)
- definiert einen Kommunikationsendpunkt (siehe → man –s 3N socket)

#### Mögliche Aufrufparameter

- Protokollfamilie
  - AF\_INET → PF\_INET
  - AF\_IP4
  - AF\_UNIX
- Transport – Art
  - SOCK\_STREAM
  - SOCK\_DGRAM
  - SOCK\_RAWbenutzte Transportprotokolle (TCP, UDP , IP)



**bind ( ):**

verbindet das Socket (über das Handle) mit dem realen, lokalen Kommunikationsendpunkt (siehe man –s 3N bind)

**Mögliche Aufrufparameter**

- die lokale IP – Adresse oder any
- die lokal benutzte Port - Nummer

**listen ( ):**

- versetzt die Serversoftware in den Listen – Modus
- sie wartet auf Clientanfragen (siehe man –s 3N listen)
- Angabe einer Client – Warteschlangenlänge

**except ( ):**

- liefert einen **Bearbeitungssocket** für einen eingegangenen Clientverbindungswunsch, wird an den Worker weitergegeben
- Der Bearbeitungssocket hat zwei Kommunikationsendpunkte, den des Servers und den des Clients

**read ( ) / recieve ( ):**

lesen / empfangen

**write ( ) / send ( ):**

schreiben / senden

**close ( ):**

schließen der Verbindung

**connect ( ):**

baut aktiv eine Verbindung zum Server auf

## 4.6 Client – Server – Beispiel mit Sockets in C

~paulu / INTERNET – VORLESUNG / client-server

**Anwendungsprotokoll:**

Bei Client – Kontakt folgt eine Antwort

**Server:**

führt einen Kontaktzähler, als Relation auf einen Client – Kontakt, gibt der Server einen String an den Client mit der Anzahl der bisherigen Client – Kontakte zurück.

**Client:**

nimmt Kontakt zum Server auf, gibt Serverstring an.

**Server – Start:**

server [Port – Nr]

Standard: 5193



- r – remote
- entfernt

#### **zwei Steuerdateien:**

- \$HOME / .rhosts
- /etc / hosts.equiv      stehen alle vertrauenswürdigen Rechner

#### **Starten eines GUI – Programms auf einem Remote – Host, die Anzeige und Bedienung auf dem lokalen Host**



xhost + stl – c – 24

rsh stl – c – 24 "export DISPLAY = stl – c – master: 0.0; /usr / bin / firefox"&

## **4.8 RPC – Remote Procedure Call**

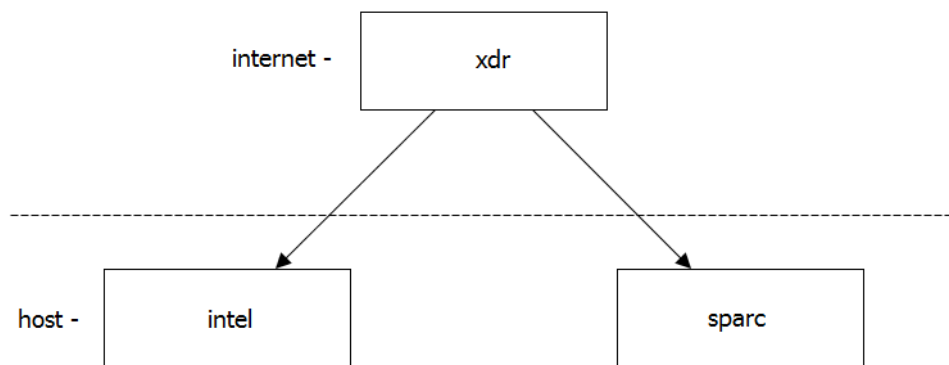
- die Socket – Programmierung ist ungewohnt denn:
  - man sendet und empfängt Nachrichten
  - man überträgt Zeichen **keine** Daten wie integer / real
- schön wär es, wenn man mit Daten arbeiten könnte wie in C / C++ gelernt.

### **4.8.1 Werkzeuge zur RPC - Programmierung**

- es gibt eine Schnittstellen – Beschreibungssprache RPCL (= Remote Procedure Call Language) mit der man die RPC – Funktionen / Prozeduren mit ihren Parametern und Rückgabewerten beschreibt
- der dazugehörige Compiler rpcgen erzeugt aus dieser Beschreibung in einer beliebigen Programmiersprache die nötigen Funktionen.
  - Client – Stub
  - Server – Stub (Skelett)
  - xdr – Funktionen

XDR = eXternal Data Representation

- intel    → little endian      → add (100)
- sparc    → big endian         → add (001)



## 4.8.2 Die RPC – Paket - / Nachrichtenstruktur

### **Transaktions – ID:**

identifiziert die Art der RPC – Nachricht

- 0: RPC – Aufforderung
- 1: RPC – Antwort

### **RPC – Version – 2**

### **Programmnummer / Versionsnummer / Prozedurnummer:**

identifiziert die aufzurufende Prozedur des Servers  
(Identifikator des Clients)

### **bisher:**

- ein Server arbeitet auf einem wellknown Port
- er ist über sein Socket (IP – Adresse, Protokoll, Port) weltweit eindeutig adressierbar

### **Was tun bei einem RPC – Server mit 500 Funktionen?**

#### **RPC – Lösung:**

ein Server auf RPC – Basis meldet sich beim Startup bei einem Service – Server, **Port – Mapper** genannt, unter Angabe seiner Programm – und Versionsnummer, sowie seiner Prozedurnummer an. Der **Port – Mapper** (rpcbind) weist bei Bedarf, das heißt bei Prozeduraufruf, der Server – Prozedur zu.

## 4.8.3 Informationen zu RPC – Servern abfragen

```

rpc info      - p    [host]
              - s    [host]
              - T tcp [host] 100 005 ← mount
              - T udp [host] 100 003 ← nfs
  
```

## 4.9 Ein RPC – Client – Server Beispiel

```
rls [host] Pfadname = rsh host ls -a pfad
```

## 5 Internetdienste

Internet – Vordergrundprotokolle		Internet – Hintergrundprotokolle	
Telnet	zeichenorientierte Terminalemulation zu einem entfernten Rechner mit standardisierter Datenrepräsentation	RPC	Remote Procedure Call: zuverlässiger Client – Server – Interaktionsmechanismus
FTP	File Transport Protocol: erlaubt den Zugriff zu einem entfernten Dateisystem und das Kopieren	XDR	eXternal Data Representation: Darstellung von Datentypen über RPC
SMTP	Simple Mail Transfer Protocol: Electronic Mail System für Textnachrichten verschiedener Rechner	MIME	Multipurpose Internet Mail Extensions: System zur Übertragung von Graphik und nichttextueller Information mit E – Mails
Gopher	Informationssuchsystem mit Menüs für Ressourcen, die auf anderen Gopherservern im Internet vorliegen	BOOTP	Bootstrap Protocol: ein IP / UDP – Protokoll zum Anschließen von Computern ohne Festplatte im Internet
HTTP	Hypertext Transfer Protocol: WWW nutzt es zur hypermedialen Seitenübertragung über das Netz	DNS	Domain Name Service: Abbildung von Internetnamen und Internetadressen
Finger	Finger User Information Protocol: liefert Information über Personen aufgrund Ihrer Internetadresse	SNMP	Simple Network Management Protocol: Protokoll zum Management von Internetknoten

### 5.1 Das DNS – Domain Name System / Service

#### Aufgabe:

- Menschenlesbare Hostnamen in maschinenlesbare IP – Adressen zu übersetzen
- die TCP / IP – Server arbeitet mit IP – Adressen

#### Informationsquellen am Hostname → IP – Adresse umwandeln

- Rechner – Lokal:        / etc / hosts
- Organisation – Lokal:    nis – Tabelle : hosts
- weltweit:                DNS → nslookup / host / dig

jedes „Internet – Client – Programm“ übersetzt mit Hilfe einer „gethostbyname( )“ – Funktion einer angegebenen Hostnamen in einer IP – Adresse, damit er über TCP / IP kommunizieren kann

#### **Geschichte:**

das Network Information Center der USA führte eine zentrale Hosts – Datei, in der alle aktiven Internet – Hosts aufgeführt waren mit den Kommandos

- gettable
- htable

musste man sich diese Datei von Zeit zu Zeit herunterladen.

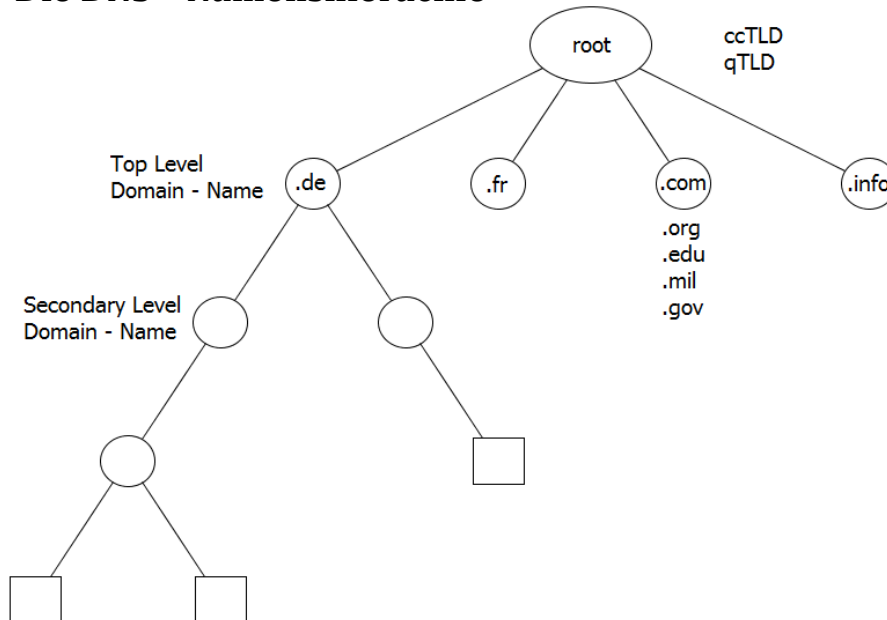
#### **Heute:**

Es existiert eine weltweit verteilte Hosts – Datenbank:

das DNS – System (beschrieben in RFC 1034 / 1035)

- die DNS – Server, die gemeinsam die Hosts – Datenbank aufbauen, verwalten nur ihren Teil der DNS – Datenbank, zum Beispiel den DNS – Server ns.htw-saarland.de verwaltet die Rechnernamen / IP – Adressen der HTW

## 5.2 Die DNS – Namenshierarchie



### Informationen zu DNS:

- [www.iana.org](http://www.iana.org)
- [www.denic.de](http://www.denic.de)

will man eine eigene Domain erwerben, so wendet man sich an

- seinen Provider
- das nationale NIC
- Internationaler Anbieter ([www.alldomain.com](http://www.alldomain.com))

**WICHTIG:** man sollte Eigentümer der Domain sein!

## 5.3 Das DNS Client – Server – Modell

- der wichtigste Aspekt des DNS: die Autonomie seiner Teilbereiche  
= jede Domain ist unabhängig von den anderen Domains
- DNS ist so angelegt, dass jede Organisation, die eine DNS – Domain besitzt, unterhalb dieser beliebige Subdomains einrichten kann ohne übergeordnete zentrale Stellen informieren zu müssen
- Die DNS – Serverhierarchie entspricht im Allgemeinen der Namenshierarchie
- Die DNS – Server sind für ihre Domain die sogenannte Autorität. An der Spitze der DNS – Server stehen die Root – Server  
(13 Organisationen betreiben im Moment Root – Server)

- Ein DNS – Server kennt nicht alle Rechnernamen seiner Domains, aber erkennt DNS – Server, die die Namen der Rechner verwalten

**zum Beispiel:**

ns.denic.de → kennt → alle Second – Level DNS – Server der Domain .de  
 ns.uni – sb.de → kennt → alle DNS – Server von uni-sb.de

- jeder DNS – Server ist so konfiguriert, dass er alle seine Subdomain – Server kennt und auch mindestens die ihm übergeordneten root – Server sowie eventuell sein direkt übergeordneten Server

## 5.4 DNS – Server Typen

### 5.4.1 primary Server

- er lädt seine Domain – Daten direkt aus Dateien, die von Domain – Administrator verwaltet werden
- er ist der sogenannte autoritative Server, die Autorität der Domain
- er gibt immer korrekte Antworten

### 5.4.2 secondary Server

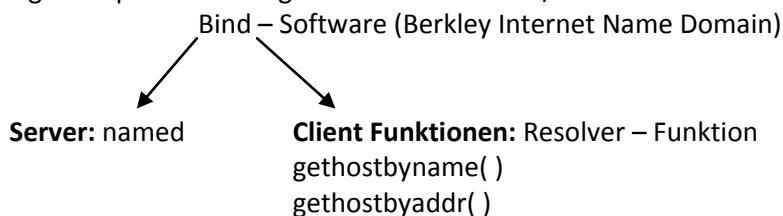
- er lädt seine Domain – Daten von zeit zu zeit vom primary – Server und speichert sie in Dateien auf Festplatte
- auch gibt er autoritative Antworten
- allerdings nicht immer 100 % aktuell

### 5.4.3 caching – only – Server

- besitzt keine Domain – Daten
- er speichert Anfrage – Ergebnisse und gibt an diesen Cache von Antworten bei Anfragen eventuell Antworten

## 5.5 Elemente des DNS unter UNIX / LINUX

die häufigste Implementierung des DNS unter UNIX / LINUX ist die



### 5.5.1 Client – Konfigurationsdateien

- / etc / nsswitch.conf → Datenquellen für die Namensauflösung festlegen
- / etc / resolv.conf → die zuständigen Namensserver werden bekannt gegeben

### 5.5.2 Server – Konfigurationsdateien

oberste Konfigurationsdatei:

- named.conf
- named.boot



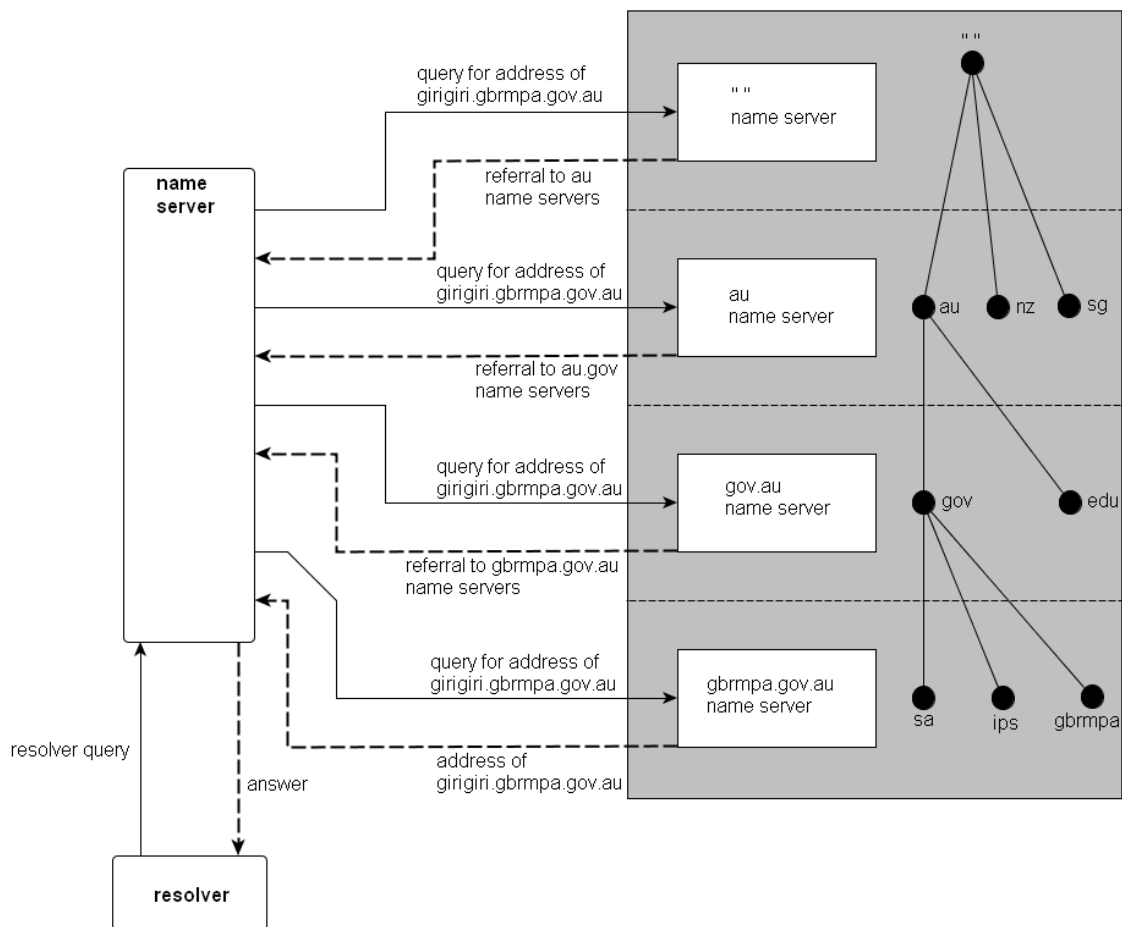
definiert die Rolle des Servers und die Namen aller weiteren Konfigurationsdateien

Name des Resource – Record	Record – Typ	Funktion
Start of Authority	SOA	Makiert den Anfang der Daten für eine Zone und definiert Parameter, die für die gesamte Zone gelten
Name Server	NS	Identifiziert den Name – Server einer Domain
Address	A	Wandelt einen Rechnernamen in eine IP – Adresse um
Pointer	P	Wandelt eine IP – Adresse in einen Rechnernamen um
Mail Exchange	MX	Bennent den E – Mail – Server für eine Domain
Canonical Name	CNAME	Definiert einen Alias – Namen für einen Rechner
Host Information	HINFO	Beschreibt Hardware und Betriebssystem eines Rechners
Well Known Service	WKS	Listet vorhandene Netzdienste auf

### 5.6 Name – Resolution - Prozess

1. Der User tippt in einen Rechnernamen in ein Internetclient – Programm ein (ftp / Browser...)
2. die Resolver – Funktion des Client – Programms stellt die Namensauflösungsanfrage an ihren zuständigen DNS – Server
3. dieser löst den Namen auf, das heißt findet via DNS – Kommunikation zu den dem angegeben Rechnernamen die IP – Adresse heraus
4. sendet diese an die Resolver – Funktion



**Probleme:**

- Flaschenhals → Root – Server (13 Stück)

↑  
Vervielfachung der Server  
**Anycast**

- jeder DNS – Server führt einen DNS – Cache mit früheren Antworten, er ist bestrebt aus diesem Cache die Antwort zu geben

**5.7 DNS – technisch**

- beschrieben in RFC 1034 / 1035
- UDP + TCP Port 53
- Kommunikationsart: stelle Frage – erhalte Antwort

→ die Resolver – Funktionen fragen standardmäßig via UDP den DNS – Servern und erhalten auch via UDP ihre Antwort

**Achtung:**

eine DNS – Antwort darf maximal 512 Datenbytes enthalten, sie wird tatsächlich falls eigentlich länger gekürzt.

- der Zone – Transfer, das heißt der Ladevorgang primary auf secondary – Server geschieht via TCP, auch die Nachfrage bei genutzten DNS – Antworten wird via TCP abgehandelt.

## 5.8 Das DNS – Nachfrageformat

Identifikation	FLAGS
Anzahl der Fragen	Anzahl der Antworten
Anzahl der autoritativen RR	Anzahl der zusätzlichen RR
Frage(n) Antwort(en) Autorität(en) Zusatzinformation(en)	

- die **Identifikation** wird von Client gestartet, vom Server bei seiner Antwort genutzt
- **Flags:**

QR	opcode	AA	TC	RD	RA	zero	rcode
1	4	1	1	1	1	3 = 000	4

### QR – 1Bit:

- 0 = Frage (Query)
- 1 = Antwort v(Response)

### Opcode – 4 Bit:

- 0000 = Standard – Query
- 0001 = Invers – Query
- 0010 = Server – Status – Query

### AA – 1 Bit:

- 1 = Autoritative Antwort
  - 0 = not Autoritative Antwort
- Antwort aus dem DNS – Cache des Servers

### TC – 1 Bit:

- 1 = Antwort gekürzt (truncated)
- 0 = Antwort ungekürzt

### RD – 1 Bit:

- 1 = „Rekursion gewünscht“ (Recursion Desired)  
der angesprochene Server soll die Namensauflösung vollständig durchführen
- 0 = falls keine autoritative Antwort möglich, gib mögliche Ansprechpartner zurück

### RA – 1 Bit:

- 1 = „Rekursion verfügbar“ (Recursion available)  
in der Server – Antwort gibt der Server bekannt, dass er eine komplette Namensauflösung durchführen könnte

### rcode = return code – 4 Bit:

- 0000 = kein Fehler aufgetreten
- 0011 = Namensfehler

### 5.8.1 den DNS „von Hand“ abfragen

**Tools:**

- host
- dig
- nslookup

**Beispiel:**

host -r -v [www.paris.fr](http://www.paris.fr) ns1.htw-saarland.de  
- r = keine Rekursion

### 5.8.2 Whois

- beschrieben in RFC 812 / 954 / 3912
- TCP: 43

**Sinn & Zweck:**

Abfrage von Domain – und IP – Informationen

**Protokoll:**

- Frage als String übermitteln
- Antwort als String bekommen

**Client – Software:**

UNIX / LINUX : whois

whois [-h whois-server] identifier

## 5.9 Elektronische Post – E – Mail

- einer der wichtigsten / meistbenutzen Dienste des Internet
- ursprünglich zum Versenden kurzer 7 – Bit – Ascii Nachrichten entwickelt
- heute vollwertiges Kommunikationsmittel mit Texten / Bildern / Tönen  
→ Multimedia
- normalerweise steht hinter einer E – Mail Adresse eine Organisation / Mensch, der die Mail liest und antwortet
- **Auch das E – Mail System** ist ein „normales“ Client – Server - System

**User Agent – UA:**

- das E – Mail Programm des Benutzers
- hilft E – Mails lesen / Schreiben / verwalten
- er versendet E – Mails via SMTP
- er empfängt E – Mails via POP / POPs / IMAP / file

**Message Transfer Agent:**

- SMTP – Server
- Mail – Server  
sind beide Outgoing Mail – Server
- POP3 – Server
- IMAP – Server  
sind Incoming Mail – Server

### 5.9.1 Aufgaben des SMTP – Servers

- wartet auf eingehende Nachrichten von einem UA oder anderen MTA
- wertet die E – Mails aus:
  - sendet Mail an nächsten MTA
  - speichert in **Mailbox** des Users ab

### 5.9.2 Die Mailbox

ist nichts weiter als eine ganz normale Textdatei in der die eintreffenden Mails durch eine Leerzeile getrennt hintereinander abgespeichert werden.

Ort:

/ var /mail → / var /mail / pauly

### 5.9.3 Aufbau einer E – Mail - Adresse

**Früher:**

<name>@<host>

Pauly@stl-s-stud.htw-saarland.de

**Heute:**

generische E – Mail – Adressen im Domain Name Service definiert

<name>@<domain>

Wolfgang.Pauly@htw-saarland.de

**Mail – Alias – Tabelle:**

Wolfgang.Pauly → wpy

Wolfgang Pauly <wpy@htw-saarland.de>

### 5.9.4 Aufbau einer E – Mail

Wie jedes Informations-Paket im Internet ist auch die E – Mail ein Datenpaket mit dem allgemeinen Aufbau: **Header + Body**.

- Email-HEADER – enthält Steuerinformationen
- Email-BODY – enthält die eigentliche Nachricht, evtl. gegliedert nach Datentyp (7 Bit ASCII-Zeichen)

**Header:**

- besteht aus einzelnen Headerzeilen, die ein bestimmtes Format haben müssen, deren Anzahl und Inhalt nicht festgeschrieben ist.
- Die Anzahl der Header – Zeilen hängt vom User Agent, von den Verarbeitungsprogrammen und dem Weg der E-Mail durch das Internet ab

### 5.9.5 Aufbau einer E – Mail – Header – Zeile

<Schlüsselwort>: < Argumente/werte>

Es gibt nötige Schlüsselwörter: FROM, TO ...

Es gibt optionale Schlüsselwörter: Date, Subject, CC, BCC...

Es gibt Verarbeitungssoftwarewörter: X-

**Wie kann man Word- / PDF Dokumente oder Bilder (.jpg..), Videos oder Töne via E – Mail versenden?****Antwort:**

- man wandelt die Daten vor dem Versenden in 7-Bit ASCII Darstellung um
- nach Erhalt wird die ursprüngliche Kodierung wieder hergestellt

Die IETF hat eine Auszeichnungssprache für den Aufbau und den Inhalt von E – Mail definiert:

**MIME = Multipurpon Internet Mail Extensions**

- beschrieben in RFC 1521 / 1522 / 1524

**5.9.6 MIME – Header – Zeilen**

- MIME-Version: 1.0
- Content-Type: Typ / SubTyp ; Seperator
- Content-Length: Anzahl der Bytes
- Content-Transfer-Encoding: Kodierung
- weitere Anwendung von MIME: [www](http://www)

SMTP: Simple Mail Transport Protocol

- wird zum Versenden von E – Mails benutzt

**UW** sendet per **SMTP** zu **MTA**

**MTA** sendet per **SMTP** zu **MTA**

Die Übertragung einer E – Mail erfolgt mittels SMTP in 5 Schritten.

**1) Beim SMTP-Server anmelden**

Client: HELLO domain

Server: please to meet you

**2) Angabe des Senders**

Client: MAIL FROM: Donald.Duck@entenhausen.de

Server: 250 ok

**3) Angabe des Empfängers**

Client: RCPT TO: ki-st07@htw-saarland.de

Server: 220 ...

**4) Angabe der zu sendenden Daten**

Client: DATA (enter)

Server: 345 Enter mail

Client: ... (Mail eingeben)

...

Client: . (enter)

Server: 250 ok

**5) vom Server abmelden**

Client: Quit

Server: 221

### 5.9.7 POPv3 over SSL

- beschrieben in RFC 1939
- POPv3 → TCP Port 110
- PPv3 over SSL → TCP Port 995
- Ein Dialog orientiertes ASCII Protokoll

#### Vorgehen

1. Anmelden am POP Server mit USERID + PASSWORT
2. Herunterladen der Mails
3. Löschen der Mails auf dem Server
4. beenden der Verbindung zum Server
5. offline Lesen und verarbeiten der Mails

telnet stl-pop 110

SSL = Secure Socket Layer

### 5.9.8 E – Mail Weiterleitung

via .forward Datei im User-home

-rwxr-xr-x

#### Beispiel 1:

[w.pauly@web.de](mailto:w.pauly@web.de)

leitet E – Mail an die aufgeführte Adresse weiter

#### Beispiel 2:

\pauly

[w.pauly@web.de](mailto:w.pauly@web.de)

speichert Mail im Postfach von Pauly

leitet email weiter

#### Beispiel 3:

\pauly . | /usr/bin/vacation



Automatische Beantwortung einer Mail

#### Beispiel 4:

[w.pauly@web.de](mailto:w.pauly@web.de)

/export/home\_pm/pauly/mail.log

|~pauly/myprog

## 5.10 Spam <-> Ham

### Spam:

- Bedeutung in der EU: unverlangt zugestellte E - Mail
- Junk-mail
- Bulk-mail
- UBE: unsolicited bulk mail
- UCE: unsolicited commercial mail

### Ham:

- erwünschte Mail

### Warum wird so viel gespamt?

- Um einfach Geld zu verdienen
- weil es so billig ist

### Die Rechtslage:

EU-Parlament (12.07.2002)

„Richtlinie für den Schutz persönlicher Daten und der Privatsphäre auf dem Feld der elektronischen Kommunikation“

RL 2002/58/EG

- Opt in –Lösung: E – Mail – Werbung nur nach **vorheriger** Zustimmung des Empfängers.
- Zuwiderhandlungen sind rechtswidrig und im Gesetz so festgeschrieben

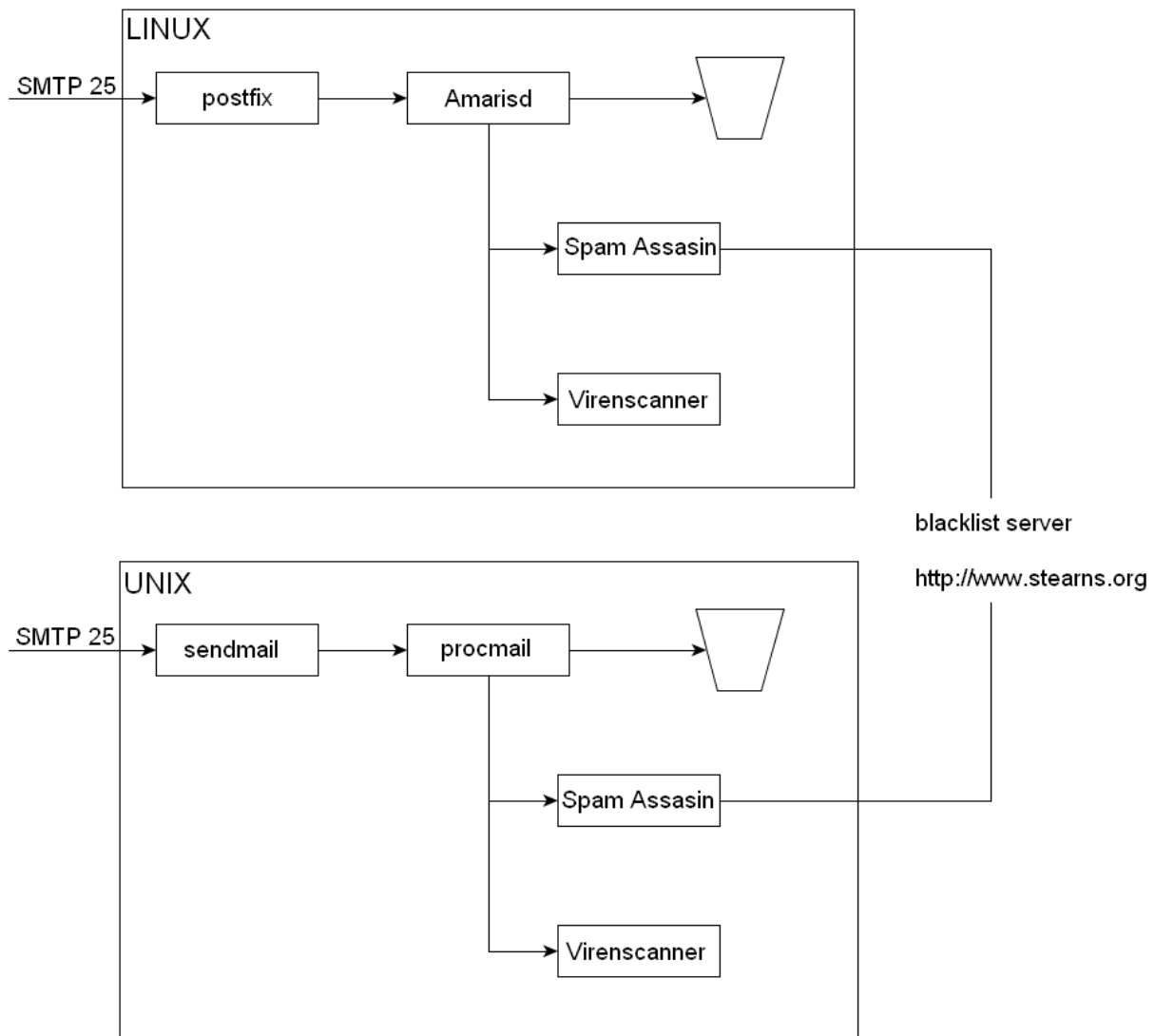
### **Spam – Was tun?**

- 1) Schütze deinen PC mit aktueller Viren – Schutz – Software und einer Firewall
- 2) eigenen Mail-Server, **kein** Mail – Relay
- 3) Alternativer E – Mail – Account
- 4) Einsetzen von Filterprogrammen
  - User mit seinem UA
  - frei benutzbare Filterprogramme
  - AntiSpamWare
  - Procmail

### **Sankmail:**

serverbasiert → SamAssassin

Eu.spamassassin.org



## 5.11 Greylisting

- Die Spamttools sind auf Massenmailversendung ausgelegt
- Sie arbeiten nach dem Prinzip: „fire and foget“

### 5.11.1 Das Vorgehen

Der empfangende SMTP – Server speichert folgende Merkmale des SMTP – Dialoges

- IP – Adresse des kontaktaufnehmendes Hosts
- Die SMTP – Dialog – Adresse des Absenders
- Die SMTP – Dialog – Adresse des Empfängers

In seiner Datenbank.

- Beim ersten Auftreten des Tripels wird der Zustellungsversuch mit folgender Meldung abgelehnt:  
450 you are greylisted – try again later
- Im SMTP ist dies Verhalten erlaubt, es wird von einem sendenden SMTP – Server erwartet, dass er nach kurzer Zeit einen erneuten Zustellversuch unternimmt (Zeit ca. 30 min)



- Der empfangende SMTP – Server wird bei einem erneuten Zustellversuch innerhalb von 12 Stunden die Mail annehmen, das Tripel für 36 Tage freigeschaltet
- Bei jeder neuen Mail mit dem Tripelkennzeichen, wird die Freischaltzeit auf 36 Tage eingestellt

### 5.11.2 Probleme

- Aufwendige Überprüfung, auch bei bekannten „Nicht – Spammern“  
→ Whitelists
- Verdächtige Rechner schließt man per Namensfrage aus  
Zum Beispiel: > 4 Ziffern = Spammer

## 5.12 Fallstricke für Spam – Filter – Betreiber

- §206 Abs. 2 Nr. 2 STGB verbietet es Inhabern oder Beschäftigten eines Unternehmens das geschäftsmäßig Post- oder Telekommunikationsdienste erbringt, unbefugt eine dem Unternehmen zur Übermittlung anvertraute Sendung zu unterdrücken.
- Sinn und Zweck ist der Schutz des Post- und Fernmeldegeheimnissen und der Schutz von Zensur (Freiheitsstrafe bis zu 5 Jahre)

#### Achtung:

Auch Firmen, die eine - auch zeitweise - private Nutzung von Internetdiensten erlauben fallen unter diesen Paragraphen; sie sind Telekommunikationsprovider für ihre Mitarbeiter, aber: Firmen, die dies verbieten nicht.

#### Lösung:

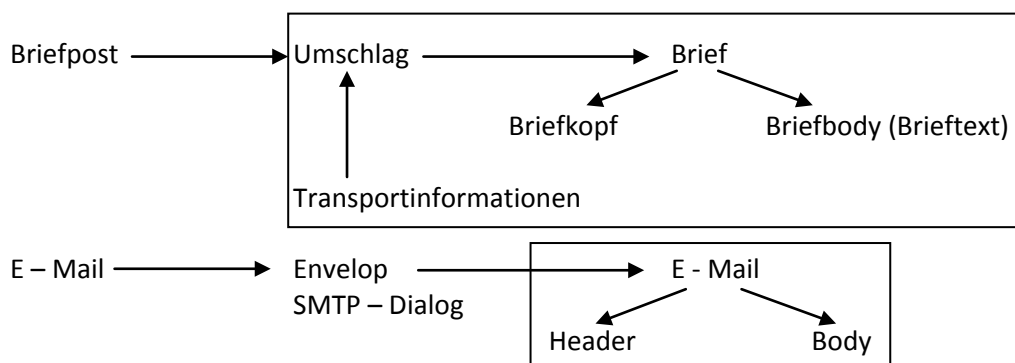
- Klauseln im Vertrag über die Nutzung (aol.com)
- Konfigurierbare Eigenschaft des Mailtools (web.de → bekannt, unbekannt, Spam)
- E-Mails markieren

§303a STGB - verbietet allgemein das Löschen und unterdrücken von Daten  
ist nicht auf Telekommunikationsdienstleister begrenzt

#### Achtung:

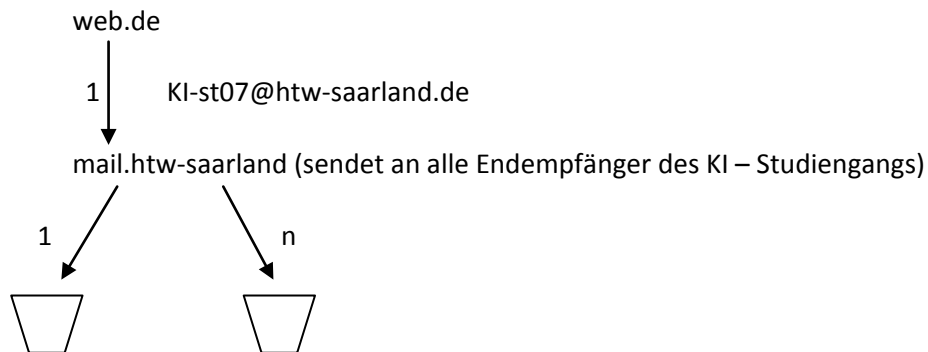
selbst **virenbehaftete** E-Mails fallen darunter

## 5.13 E – Mail – Header Analyse



**Nützlich:**

## E – Mail Verteiler



## 5.14 Return -Path

Existiert diese Zeile und steht ganz am Anfang des E – Mail – Headers, so ist sie die **Mail – From – Zeile** des SMTP – Dialoges.

**Vorsicht:**

Auch diese kann gefälscht sein

Die Recieved – Zeilen sind die einzigen Zeilen im E – Mail – Header, auf die ein Fälscher keinen Einfluss hat. Sie geben den SMTP – Server – Weg vom Sender zum Empfänger an, wenn man von unten nach oben liest. Jeder SMTP – Server, der die Mail weitergeleitet hat, hinterlässt eine **Recieved – Zeile**.

**Aber Vorsicht:**

- Es dürfen keine Lücken in der Serverliste sein
- Es dürfen keine anderen Zeilen zwischen den Recieved – Zeilen stehen

**Recieved:**

from [Mail - Server] (Name <IP - Adresse>)

Versender der Mail IP - Adresse des Rechners

mit diesem Namen hat  
er sich im SMTP - Dialog  
angemeldet

by [Mail - Server] ← Empfänger Mail - Server

for <E - Mail - Adresse> ← Empfänger

## 6 Das WWW – World Wide Web

### 6.1 Geschichte

- entstanden 1989 in den Forschungslaboren des CERN
  - alle Forscher sollten innerhalb der Organisation Forschungsberichte finden und verteilen können
  - Tim Berners – Lee (März 1989)  
Idee war ein Netzwerk von verknüpften Dokumenten
- 18 Monate später ein textbasierter, zeilenorientierter Prototyp als Client & Server
- öffentliche Demo 1991 in San Antonio, Texas
- Hypertext 1991
- Februar 1993 erster „Browser“ = **Mosaic**
- Marc Andreessen – Netscape
- 1994: das W3C wird gegründet
- März 1994 bekommt das STL Internetanschluss
- Ende 1994: experimenteller Web – Server der HTW
- 1996: Redesign – 3 PI – Studenten
- April 1998: beste FH – Webseite in Deutschland
- Oktober 1998: zweitbeste Webseite aller Hochschulen in Deutschland

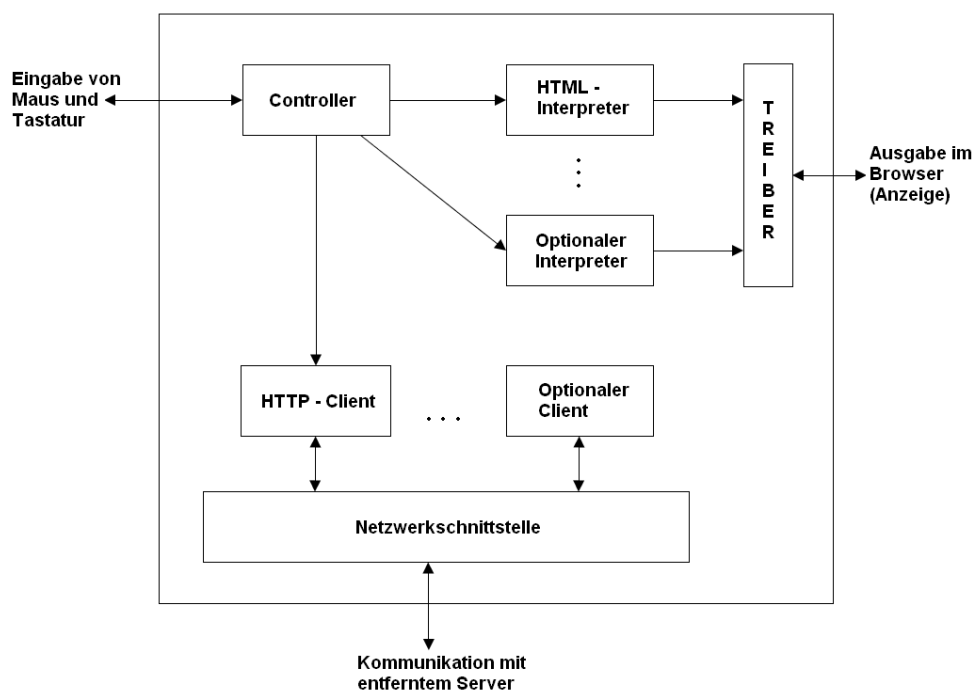
### 6.2 Adressierung im World Wide Web

auf Basis der allgemeinen URI (Universal Resource Identifier), beschrieben in RFC 1030, wird im WWW mit http zur Adressierung von Inhalten die URL (Uniform Resource Locator), beschrieben in RFC 1808, benutzt.

<Protokoll> : // <host> [:<port>] / Zugriffspfad  
 http://www.htw-saarland.de / \_\_\_\_\_

↑  
https,ftp,telnet

↑  
wenn leer, dann wird versucht die index.html Seite aufzurufen



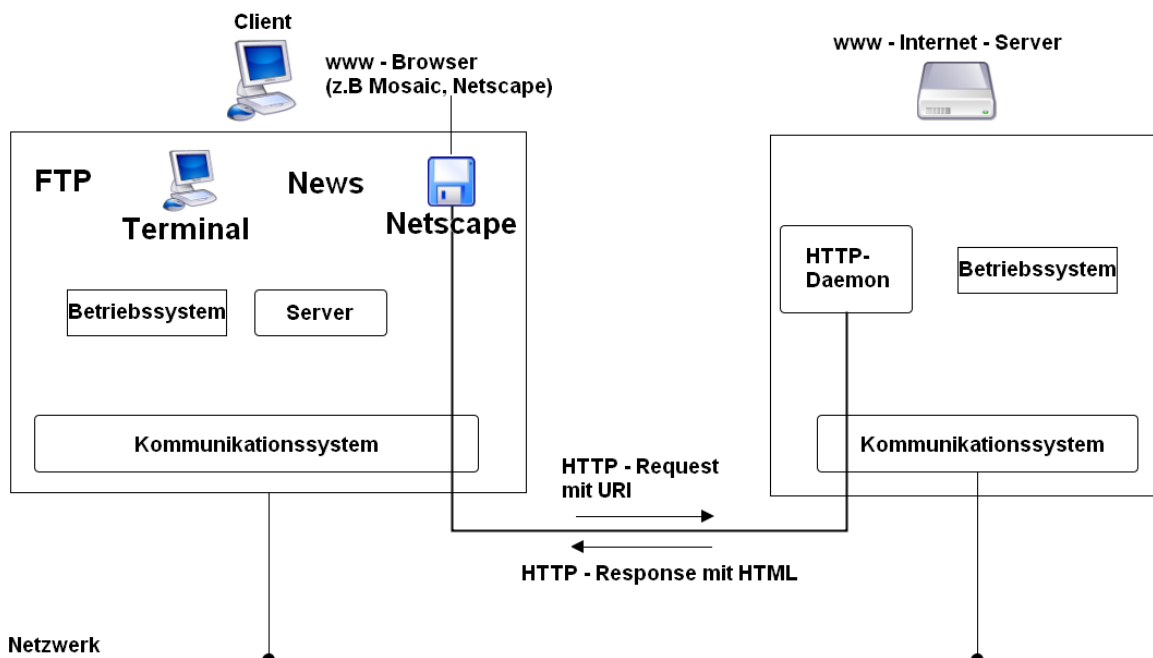
## 6.3 HTTP

- beschrieben in RFC 1955 / HTTP 1.0
- beschrieben in RFC 1966 / HTTP 1.1
- ist ein ASCII – basiertes, einfaches Dialogprotokoll um HTML – Dateien zu übertragen
- ein **Frage – Antwort – Protokoll**

## HTTP – Request

## HTTP – Response

- wird über TCP transportiert
- ist **zustandslos**



## 6.4 HTTP - Kommunikation

- Client ---> Server:
  - Aufbau einer TCP – Verbindung
  - Übertragung des HTTP – Requests
- Server ---> Client:
  - HTTP – Response  
Client analysiert Response, stellt eventuell Nachfrage an den Server  
(Zum Beispiel: Nachladen von Images)
- Client ---> Server:
  - Beenden der TCP – Verbindung
- Server ---> Client:
  - Beenden der TCP – Verbindung

**Vorteil:**

- früher Offline – Lesen der Webseiten
- Serversoftware sehr einfach, keine Sessionverwaltung für den Grundserver

GET	Aufforderung zum Kopieren der durch URI adressierten Daten in den Nachrichtentext der Antwort
HEAD	Wie GET, aber nur für Testzwecke. Es werden keine Daten in den Nachrichtentext der Antwort kopiert
POST	Senden von Daten im Nachrichtentext der Anforderung zu dem durch URI adressierten Server
PUT	Senden von Daten im Nachrichtentext der Anforderung, um sie unter URI zu speichern
DELETE	Löschen der durch URI adressierten Daten
LINK	Herstellen von Links zwischen den durch URI adressierten Daten und einer anderen Ressource
UNLINK	Auflösen von Links zwischen den durch URI adressierten Daten und einer anderen Ressource