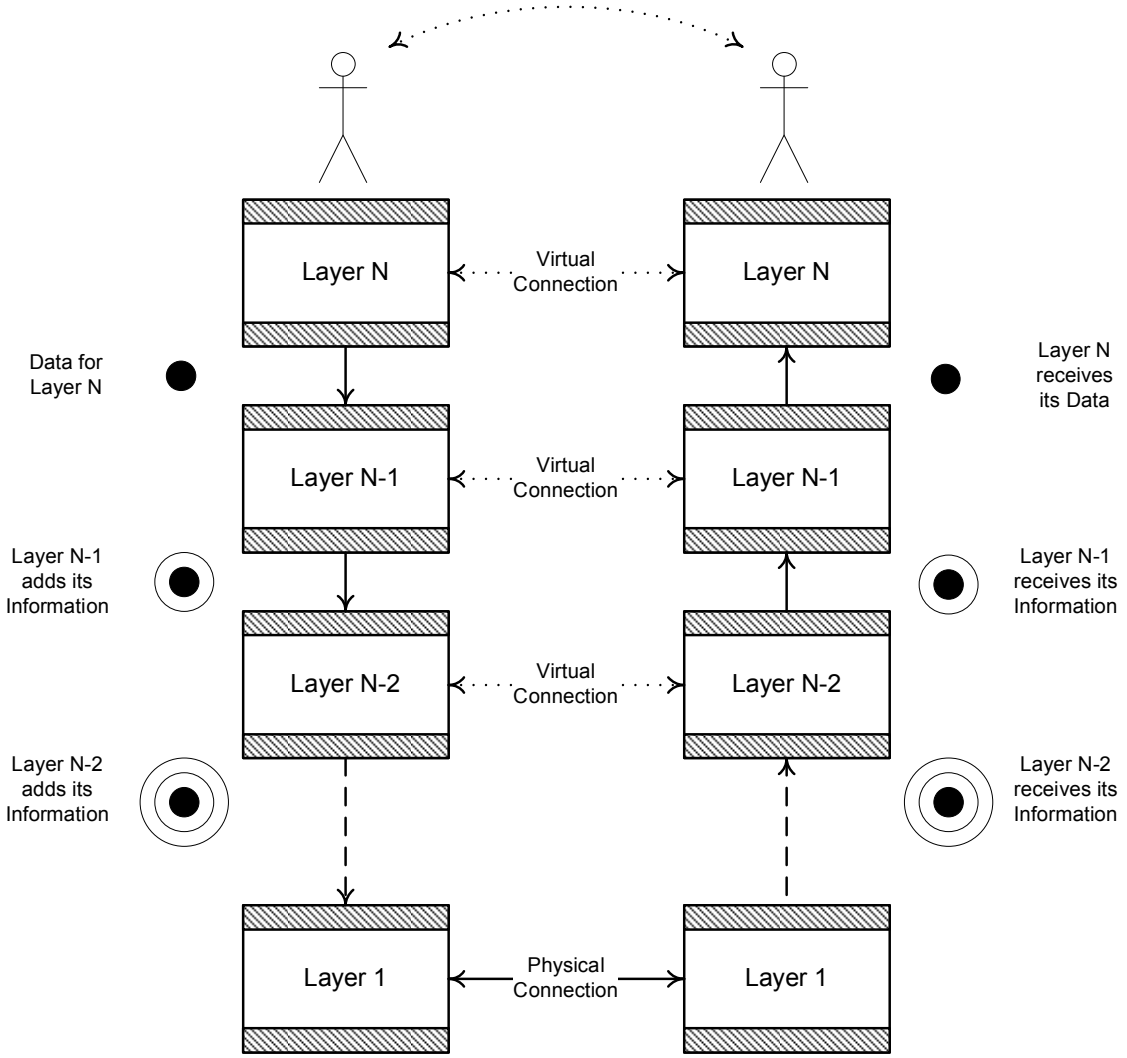
Kap. 1

Anwendungs - Schicht

Schichtenarchitektur: Allgemeine Übersicht



- Schichtenarchitektur Vergleich: Aufbau einer Zwiebel

- Jede Schicht eines Rechners unterhält eine virtuelle Kommunikation.
  Die tatsächliche Datenübertragung findet nur auf der untersten Schicht statt.

-Jede Schicht bildet eine Schale um die Daten der vorherigen Schicht

- Netzwerkarchitektur besteht aus folgenden Komponenten:

  - Schichten
  - Protokolle zwischen den Schichten
  - Schnittstellen zwischen den Schichten

ISO-OSI-Architektur-Modell

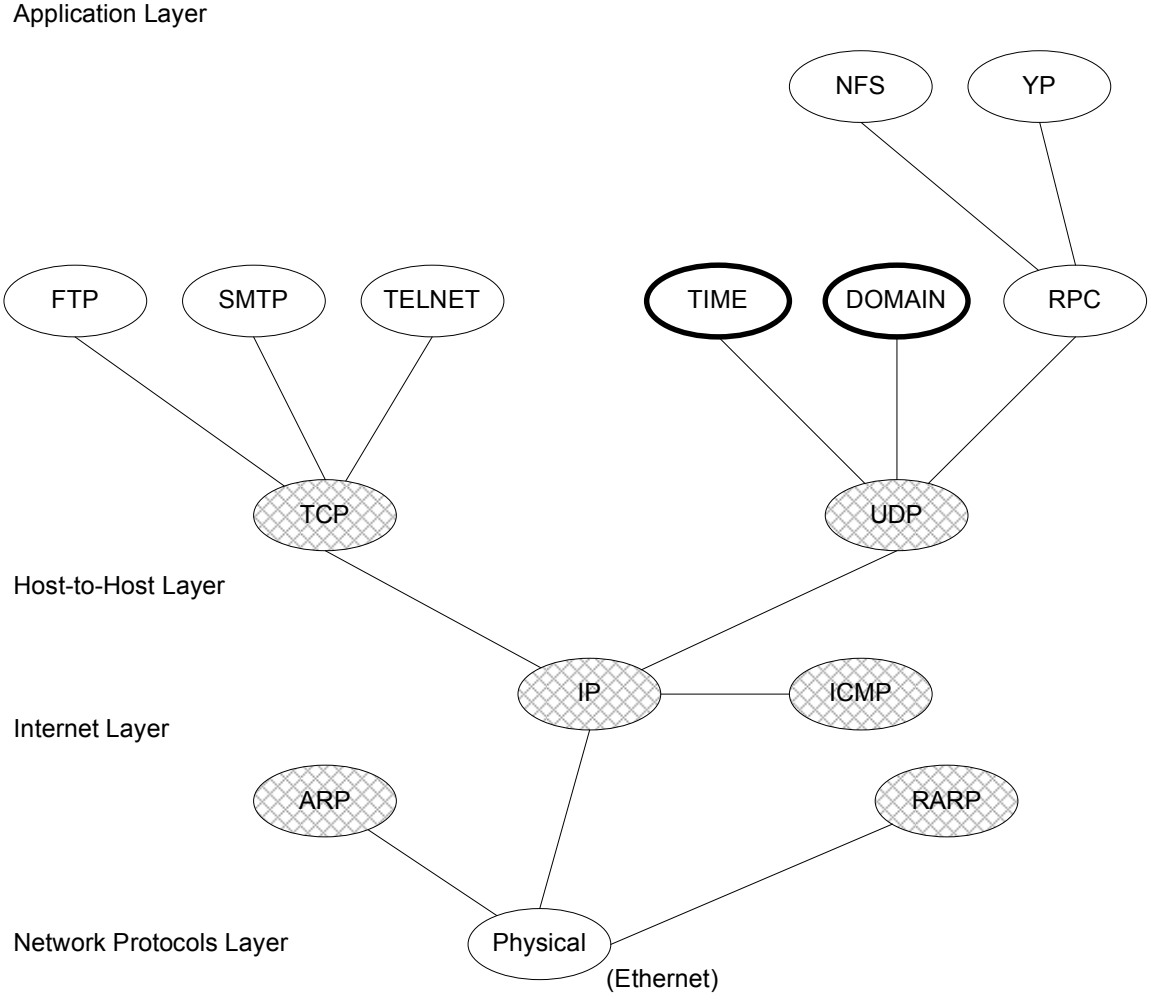
7	**Anwendung** Schicht
6	**Darstellung** Schicht
5	**Sitzung** Schicht
4	**Transport** Schicht
3	**Vermittlung** (Netzwerk) Schicht
2	**Sicherung** (Data Link) Schicht
1	**Physikalische** Schicht (Bitübertragung/Netzwerkhardware)

7-Schichten-Modell

1. Physikalische Schicht (Physical Layer)
2. Sicherungs-Schicht (Data Link Layer)
   - Medium Access Control (MAC-Unterschicht)
   - Logical Link Control (LLC-Unterschicht)
3. Vermittlungs-Schicht (Network Layer)
4. Transport-Schicht (Transport Layer)
5. Sitzungs-Schicht (Session Layer)
6. Darstellungs-Schicht (Presentation Layer)
7. Anwendungs-Schicht (Application Layer)

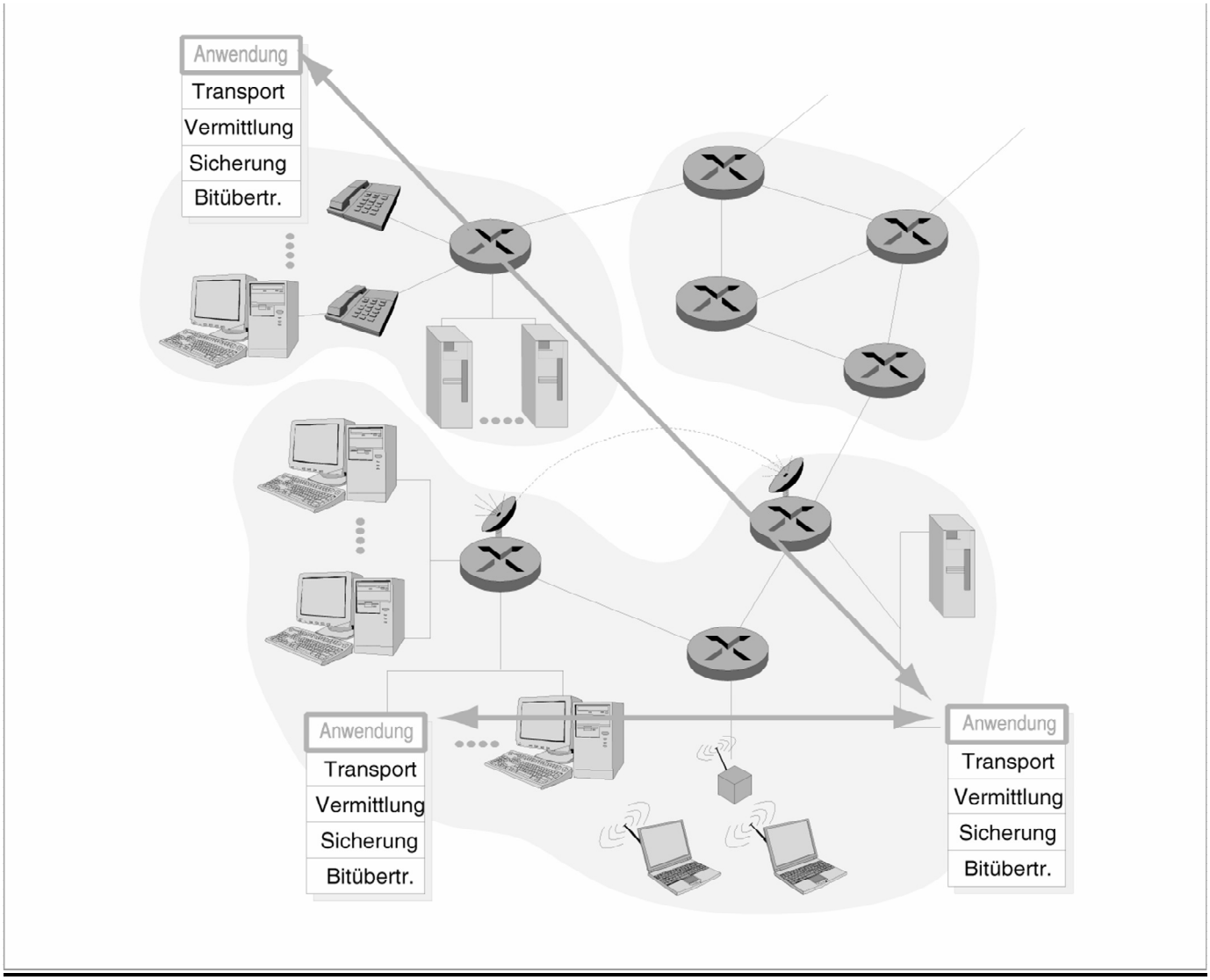
ARPA – DoD Internet (ARPANET)

Referenz Modell



TCP:	Transmission Control Protocol
IP:	Internet Protocol
ICMP:	Internet Control Massage Protocol
UDP:	User Datagram protocol
FTP:	File Transfer Protocol
SMTP:	Simple Mail Transfer Protocol
RPC:	Remote Procedure Call
YP:	Yellow Pages
NFS:	Network File System
ARP:	Address Resolution Protocol
RARP:	Reverse Address Resolution Protocol

Anwendungs-Schicht



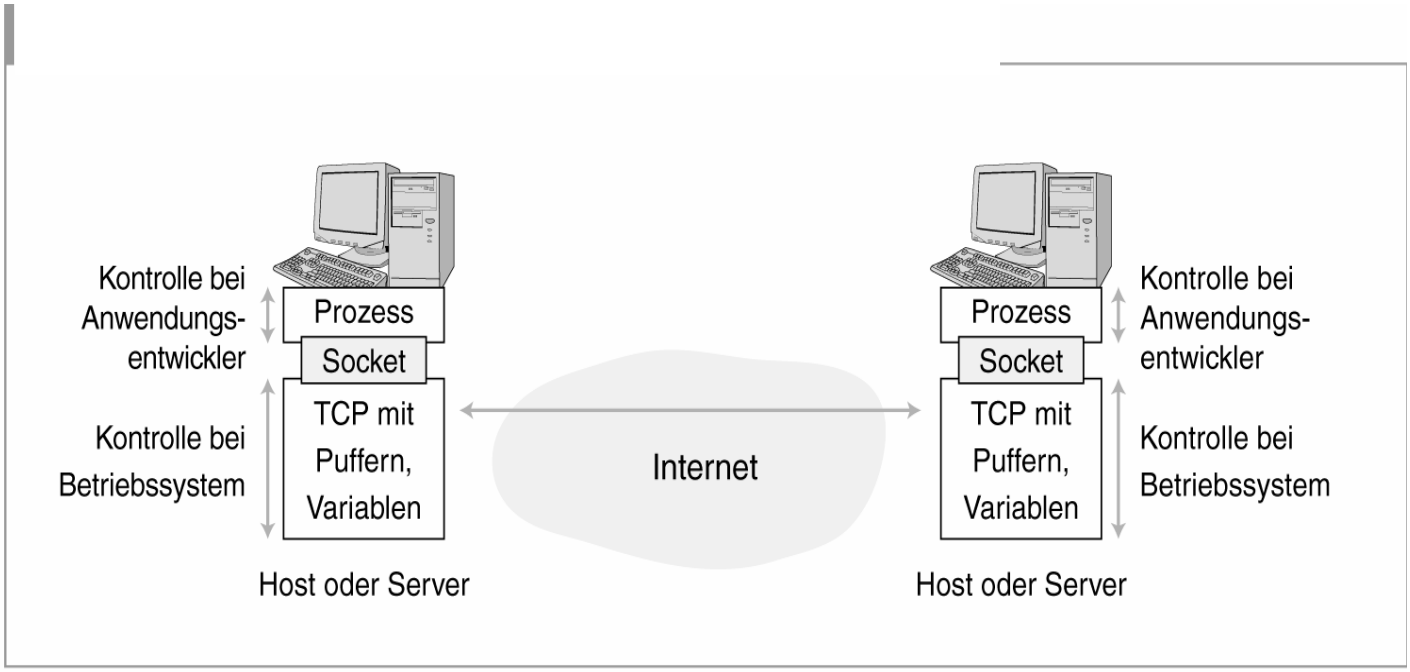
Operations-Prinzip

- Kommunizierende Prozesse / Endsystem tauschen Nachrichten aus
- Dieser Austausch findet unter Verwendung eines dafür definierten Protokolls statt.

Anwendungs-Varianten mit dazugehörigen Protokollen

- FTP → Server-Prozess + Client Prozess
- TELNET → Server-Prozess + Client Prozess
- SMTP → Server-Prozess + Client Prozess
- HTTP → Server-Prozess (WEB-Server) + Client Prozess
  Client-Prozess (WEB Browser) + Client Prozess

Kommunikation zwischen Prozessen



Prinzip:

- Prozesse senden und empfangen mittels SOCKETS
- SOCKET = > Schnittstelle zwischen Anwendungs- und Transport-Schicht
- SOCKET = > API
- Kontrolle aus der Sicht der Anwendungs-Schicht:
  - Socket-Auswahl
  - Transport-Protokoll – Auswahl (z. B. TCP oder UDP)
- Prozess-Identifizierung im jeweiligen Endsystem erfolgt mittels:
  - Adresse/Name des Endsystems
  - Prozess-ID innerhalb des Endsystems

Dienste für Netzwerk-Anwendungen

Anwendung	Datenverlust	Bandbreite	zeitsensitiv
Filetransfer	Kein Verlust	Elastisch	Nein
E-Mail	Kein Verlust	Elastisch	Nein
Web-Dokumente	Kein Verlust	Elastisch (wenige Kbps)	Nein
Echtzeitaudio/-video	Verlusttolerant	Audio: wenige Kbps bis 1 MB Video: 10 KB bis 5 MB	Ja, einige hundert Millisekunden
Gespeichertes Audio/Video	Verlusttolerant	wie oben	Ja: wenige Sekunden
Interaktive Spiele	Verlusttolerant	Wenige Kbps bis 10 KB	Ja: einige hundert Millisekunden
Finanzanwendungen	Kein Verlust	Elastisch	Ja und nein

- **Datenverlust:**
  - Verlustfreie Anwendungen: z. B. E-Mail, FTP, Finanzanwendung
  - Verlusttolerante Anwendungen: z. B. Multimedia-Anwendung

- **Bandbreite:**
  - Bandbreite sensitive Anwendungen: z. B. Multimedia Anwendung
  - Elastische Anwendung: so wenig Bandbreite nutzen wie verfügbar
    z. B. E-Mail, FTP

- **Zeit:**
  - Strenge Zeitbeschränkungen, d.h. keine schwankenden Verzögerungen
    z. B. Internet-Telefonie, Telefon-Konferenzen, Multimedia-Games
  - Schwache Zeitbeschränkungen, z. B. FTP, E-Mail

Vorhandene Transportdienste für Anwendungen

Anwendung	Protokoll der Anwendungsschicht	Zugrunde liegendes Transportprotokoll
E-Mail	SMTP (RFC 821)	TCP
Remote-Login	Telnet (RFC 854)	TCP
Web	H TTP (RFC 2616)	TCP
Filetransfer	FTP (RFC 959)	TCP
Remote File-Server	NFS [McKusik 1996]	UDP oder TCP
Streaming Multimedia	Proprietär (z. B. Real Networks)	UDP oder TCP
Internet-Telefonie	Proprietär (z. B. Vocaltec)	Normalerweise UDP

TCP-Dienste:

- Verbindungsorientierter Dienst:
  Analogie Telefongespräch
  - Verbindungs-Aufbau vor der Datenübertragung
  - Verbindungs-Abbau nach der Datenübertragung
  In dieser Phase wird die Verbindung Aufbau/Abbau zwischen End-zu-End-Prozessen via Sockets durchgeführt.

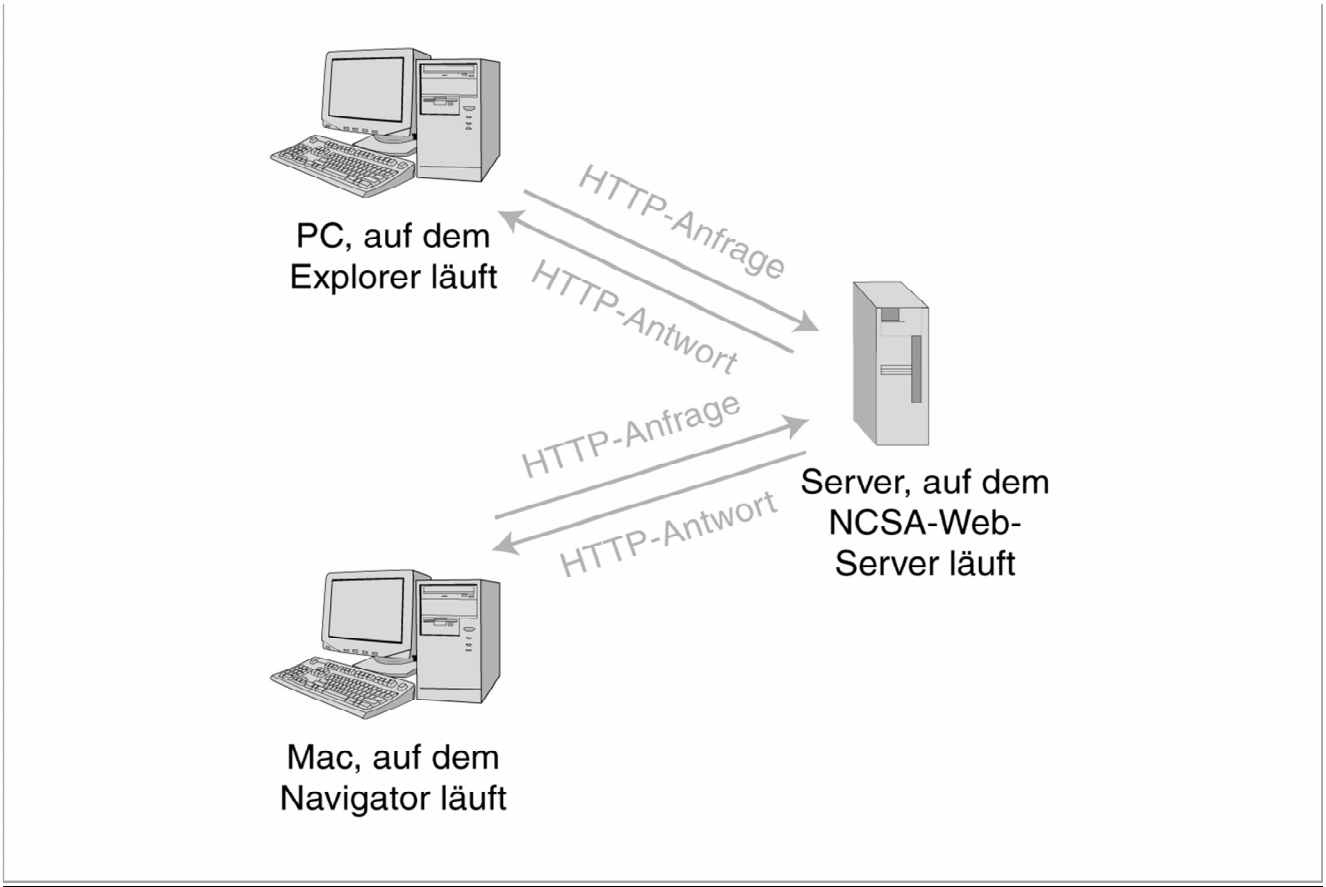
- Zuverlässiger Transport Dienst:
  - Bestätigung der Datenübertragung/Fehlerkontrolle
  - Überlast-Kontrollmechanismus mittels Fenster-Technik

- Nicht-unterstützte Dienste:
  - keine minimale Übertragungsrate
  - keine Verzögerungs-Sicherung

UDP-Dienste:

- Verbindungsloser Dienst
- Unzuverlässiger Datentransfer
- Kein Überlastungs-Kontrollmechanismus
  (vorteilhaft für die Echtzeit-Anwendungen, wie z. B. Internet-Telefonie
  Diese Anwendung verkraftet Datenverluste)
- Keine Verzögerungs-Sicherung

HTTP-Anwendung



- HTTP = Protokoll des Web für die Anwendungs-Schicht
- HTTP besteht aus Client-Prozess und Server-Prozess, die lt. HTTP-Vorschriften Informationen austauschen

Begriffe/Definitionen

Web-Seite	= Summe von Objekten
Objekt	= Eine Datei vom Typ .HTML, .JPEG, .GIF, JAVA-Applet,etc. HTML-Basis Datei referenziert die übrigen Objekte (z. B. .GIF) mittels sog. URL
URL	= 2 Komponenten => Host Name des Servers .=> Pfad-Name des Objekts Bsp. www.tu-muenchen.de/linformatik/picture.gif
Browser	= User Agent für das WEB. Er stellt die angezeigte Webseite dar
WEB-Server	= enthält WEB-Objekte, auf die per URL zugegriffen werden kann

HTTP-Protokoll-Ablauf

Varianten:

- ***HTTP Vers.1.0*** = verwendet nicht persistente Verbindung basierend auf TCP-Dienst: Port 80 (Well known Port)

- ***HTTP Vers. 1.1***.= (ab 1998)/ RFC 2616) verwendet persistente Verbindungen per Default.
  Kompatibel mit Vers. 1.0

Eigenschaften:

- ***Zeitlicher Ablauf*** wird mittels RTT beurteilt
  Round-Trip-Time (RTT) = Übertragungszeit eines Pakets vom Client zum Server und zurück
  RTT = Ausbreitungsverzögerung + Warteschlangenverzögerung + Verarbeitungsverzögerung

- ***Persistente Verbindung (PSV)***
  Um die gesamte RTT im Falle von HTML-Anfragen zu verkürzen, verwendet man persistente Verbindungen

  Persistente Verbindung (PSV) : für jedes angeforderte Objekt werden keine neuen Verbindungen aufgebaut
  D. h. die TCP-Vebindung bleibt offen.
  Die TCP-Verbindung wird vom Server nach einem bestimmten Time-Out-Intervall (konfigurierbar) geschlossen

- ***Versionen von PSV:***

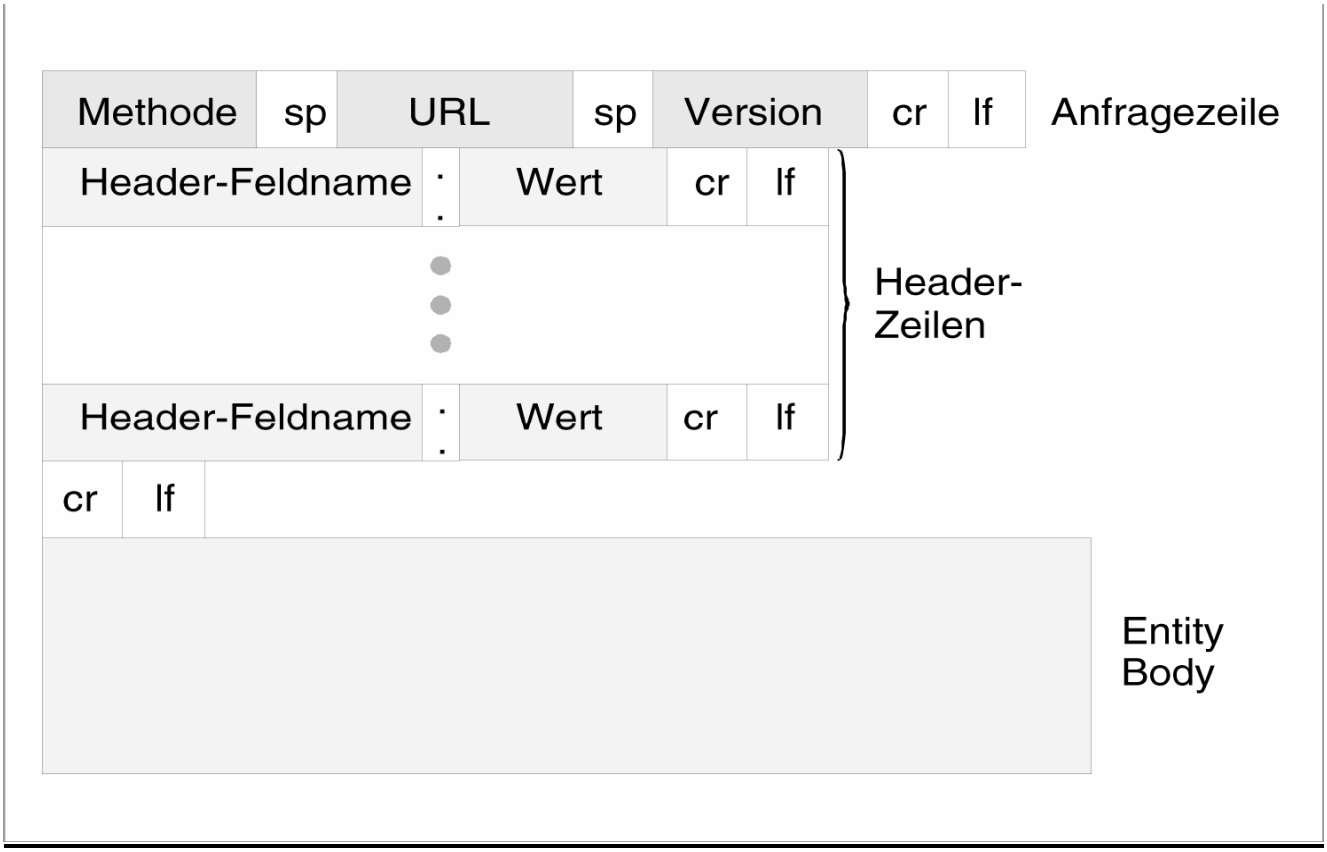
  1) ***ohne Pipelining:***
     Client sendet neue Anfragen nur dann, wenn die vorherige Antwort nicht empfangen wurde, d.h. pro Objekt = 1xRTT

  2) ***mit Pipelining***
     Client sendet die Anfragen aus, ohne auf die Antwort für die vorherigen Objekte zu warten. D.h. der Verbindungs-Kanal Client- Server wird effizienter genutzt.
     HTTP Vers. 1.1 nutzt per Default persistente Verbindungen mit Pipelining.
     D.h. der Server sendet die Objekte nacheinander für alle angekommenen Anfragen

Protokoll-Ablauf:

- siehe auch verschiedene Protokoll-Abläufe (Anhang)

HTTP-Nachrichtenformat (1)



Anfrage:

- Nachricht → .ASCII-Text
  - Versch Anzahl von Zeilen mit Txt + Leerzeile am Ende.
  - Alle Zeilen mit CR+LF abgeschlossen

- 1 Zeile → Request Line (Anfrage Zeile)
  - Methode → Anfrage Befehl: z. B. GET, POST, HEAD
  - URL → Objekt Identifizierung
  - Version Nr.

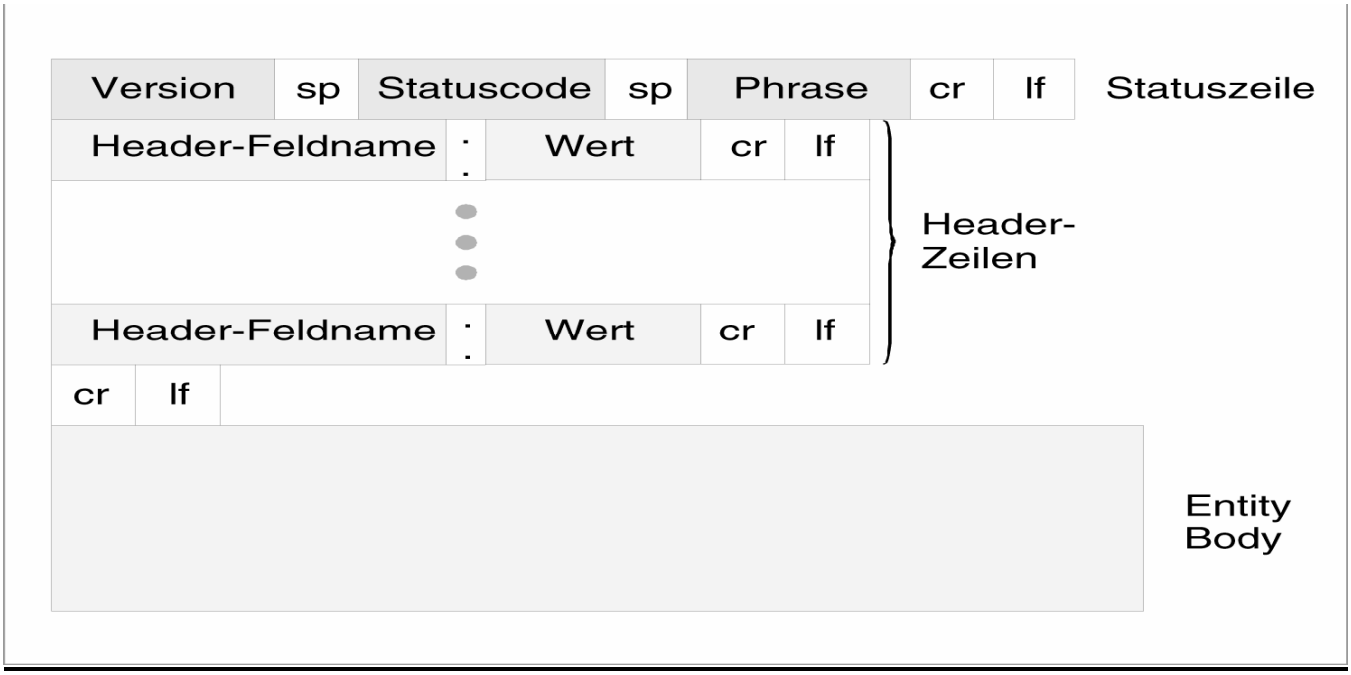
- Rest Zeilen → Header Zeilen
  - Host → Adresse des Servers
  - Connection → persistente oder nicht persistente Verbindung
  - User Agent → Browser-Typ:

- Accept Language → Sprachversion des Objekts auf dem Server

- Zusätzliche spezifische Header sind auch vorhanden
  (s. Bsp. im Labor)

- Entity Body → nur im Falle von POST-Anfrage (Formular Bearbeitung)

HTTP-Nachrichtenformat (2)



Antwort

- Statutszeile:
  - Protokollversion (1.0 oder 1.1)

  - Phrase => Ergebnis der Anfrage:
    Mögliche Varianten:
    - 200 OK
    - 301 Moved Permanently: angefragte Objekt wurde permanent verschoben
    - 400 Bad Request: Server kann die Anfrage nicht interpretieren
    - 404 Not Found: Dokument auf diesem Server nicht vorhanden
    - 505 HTTP Version NOT Supported (vom Server)

  - Connection => Server teilt dem Client mit, ob die TCP-Verbindung geschlossen oder offen bleibt

  - Datum => der gesendeten Antwort

  - Server => WEB-Server-Typ

  - Last Modified => Datum, wann das Objekt erstellt oder geändert wurde
  - Content Length => # im Objekt enthaltene Bytes

  - Content Type => Objekt-Typ (z. B. .html)

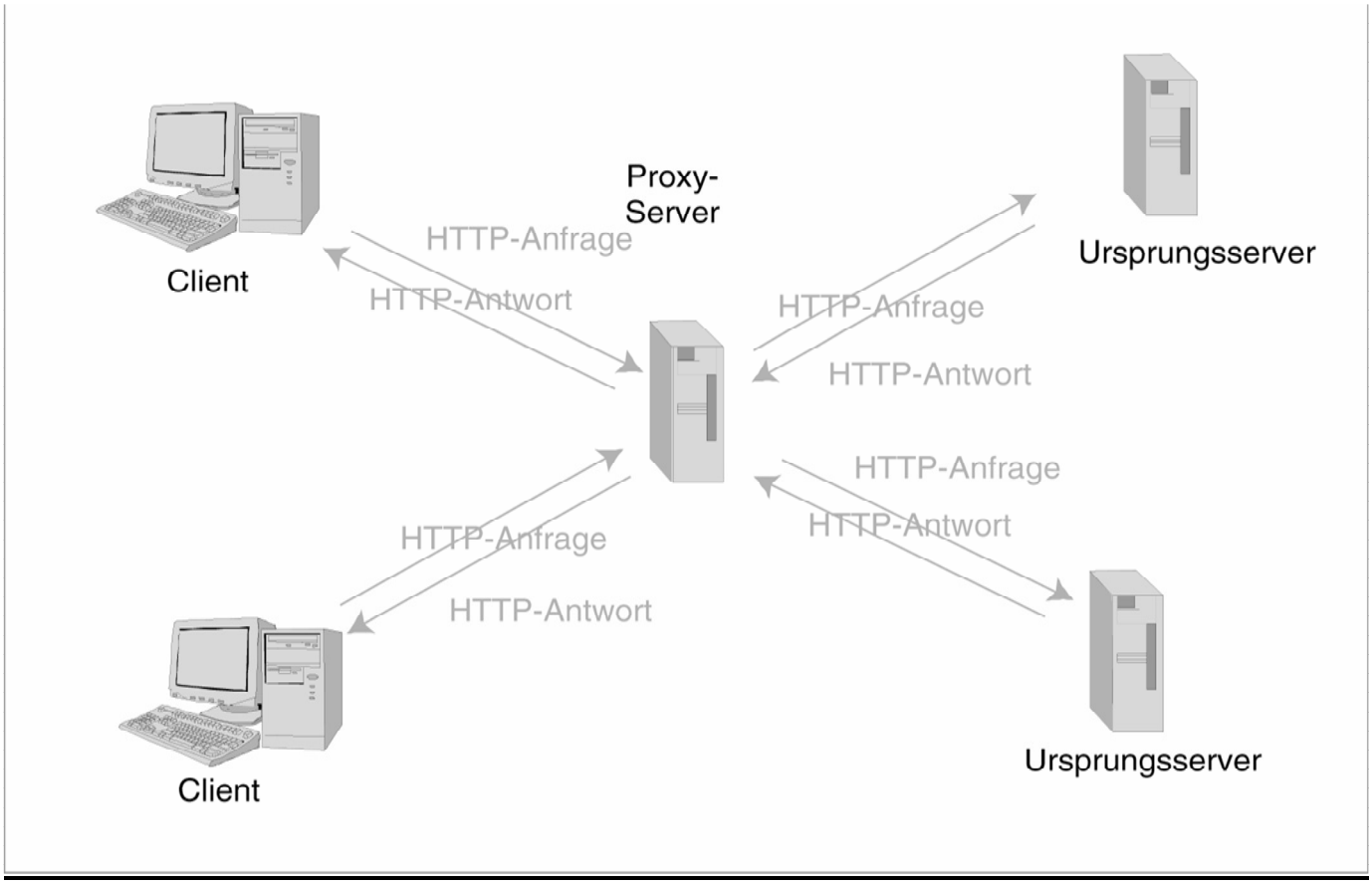
Benutzer-Authentifizierung beim WEB-Server

Varianten:

1. ***Authentifikation***
   - Identifizierung erfolgt per: User-Name und Passwort
   - HTTP unterstützt speziellen Status-Code und Header-Zeilen dafür
   - Server verlangt per Status-Code 401 Authorization Required
     Client verlangt vom Benutzer Name und PW und sendet innerhalb der Anfrage eine Header-Zeile: Authorization
     Client behält im Cache diese Information und sendet sie zusammen mit den nachfolgenden Anfragen

2. ***Cookies (RFC 2109)***
   Identifizierung des Benutzers erfolgt per sog. Cookie
   Cookie => Dez. Zahl (z. B. 1678453)
   Server verlangt: Set Cookie: xx . . xx
   Client antwortet: Cookie: xx . . xx
   Vorteil:
   - Server kennt den Benutzer, auch wenn er später mit dem Server arbeitet
   - Ersetzt den Overhead immer, bei jeder Anfrage Name und PW zu schicken
   - Der Server kann dem Benutzer zusätzliche Dienste liefern, da er per Cookie die Benutzer-Anfragen verfolgen kann.

WEB-Caches

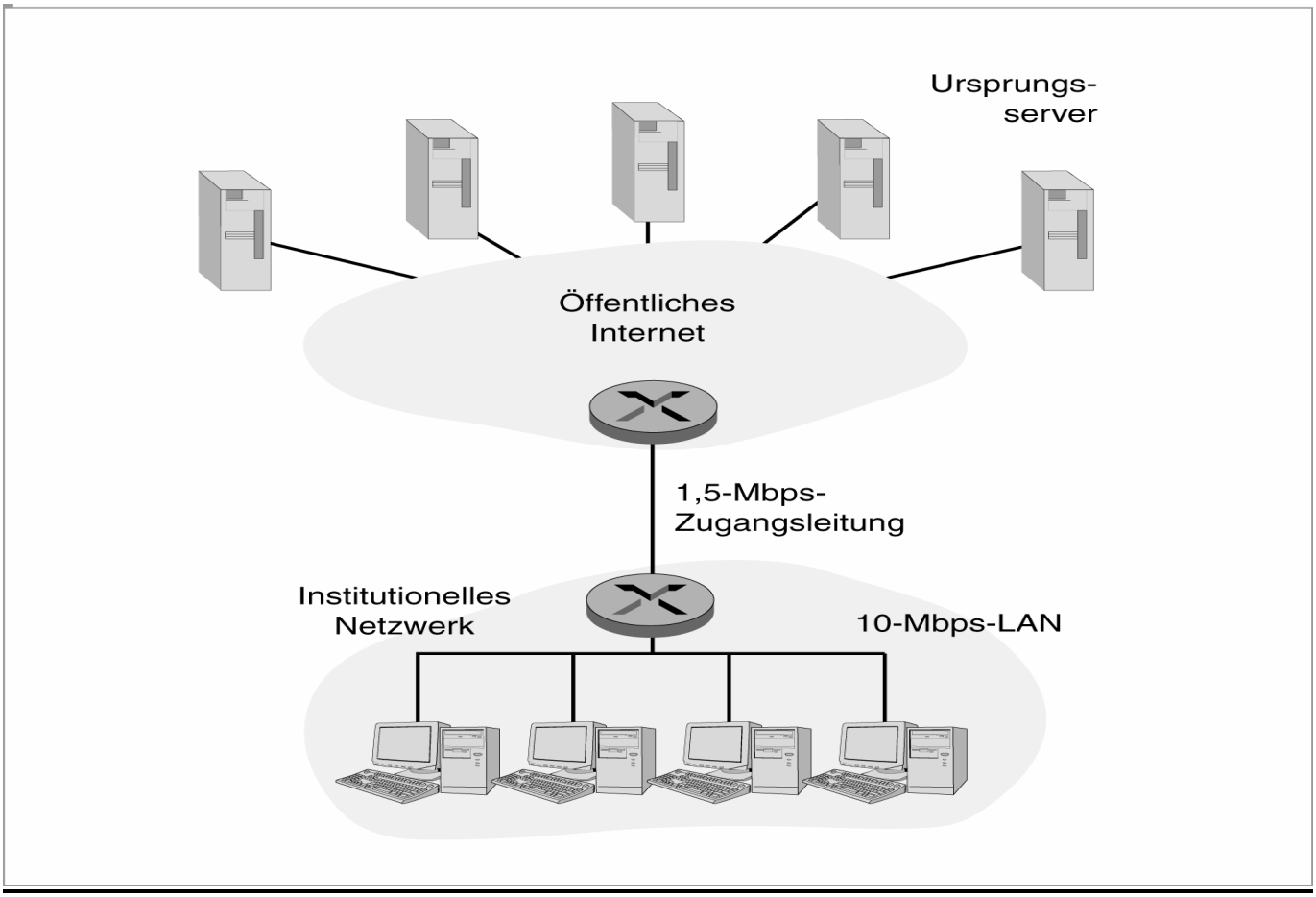


Rolle:

- Web-Cache = Proxy Server
- Proxy Server → Erfüllt Anfragen im Auftrag eines Clients
- Er speichert eine Kopie des angefragten Objektes lokal und sendet auch eine Kopie an den Client

- Vorteil:
  - Reduzierung der Zugriffe auf den WEB-Server, falls ähnliche Objekte öfter angefragt werden.

Einsatz von Proxy-Servern (1)



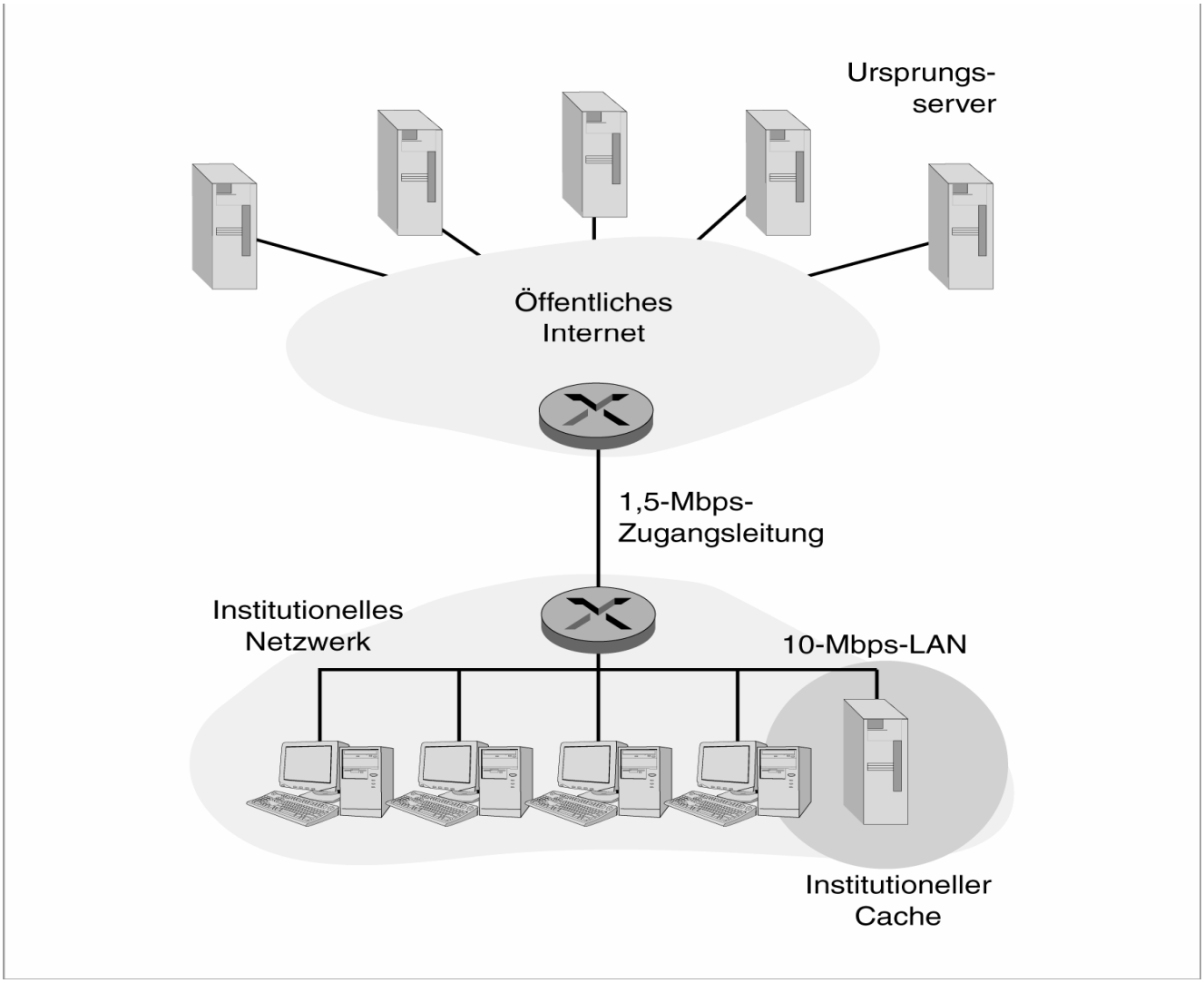
Verkehrslast

Bsp: LAN → 15 Anfragen/sec x (100KBit/Anfr)/ 10 Mbps = 0.15

Router-Louter-Leitung → 15 Anfr/secx(100KBit/Anfr)/1.5 Mbps = 1

Folgerung: → 0.15 innerh. LAN → Verzögerung von ca. 0.1 ms
→ 1 innerh. RRLeit → Verzögerung fast unbegrenzt, da keine Reserve mehr da
ca. Minutenbereich

Einsatz von Proxy-Servern (2)

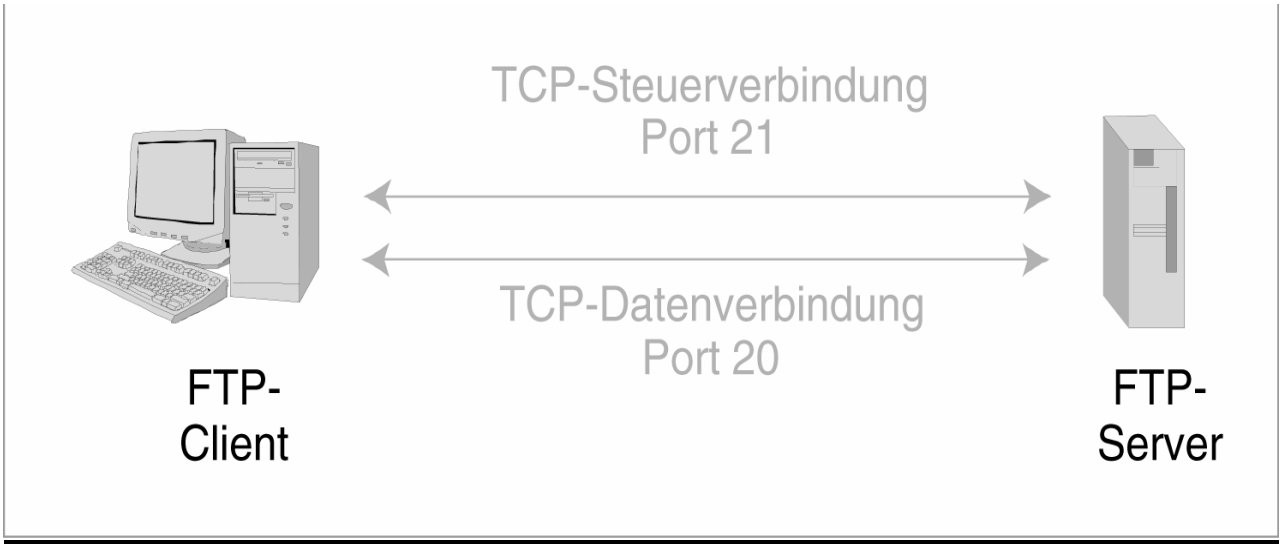


Lösung

1) RR-Leitungs-Geschwindigkeit → Erhöhen

2) Einführung eines Proxy-Servers auf LAN-Ebene.
   Die Treffer-Rate: Anfragen aus dem Cache zu erhalten und nicht vom WEB-Server ist auf 0,2 bis 0,7 geschätzt:

FTP-Anwendung (1)



Operation basierend auf RFC 959:

- Komponenten: FTP-Client (User Agent) und FTP-Server
- User Agent tritt mit dem Server in Verbindung mittels einer sog. ***„Steuerverbindung“*** (***Port 21-TCP***)
- Server überprüft den Agent, bzw. die Durchführbarkeit des angeforderten Dienstes
- Falls Dateiüberragung verlangt, dann wird eine sog. ***Datenverbindung (Port 20-TCP***) eröffnet
- Am Ende der DÜ wird diese Verbindung geschlossen
- Am Ende der Übertragung von Steuerkommandos via Port 21 wird auch diese Verbindung geschlossen.

FTP-Anwendung (2)

- **FTP-Befehle Client -> Server:**

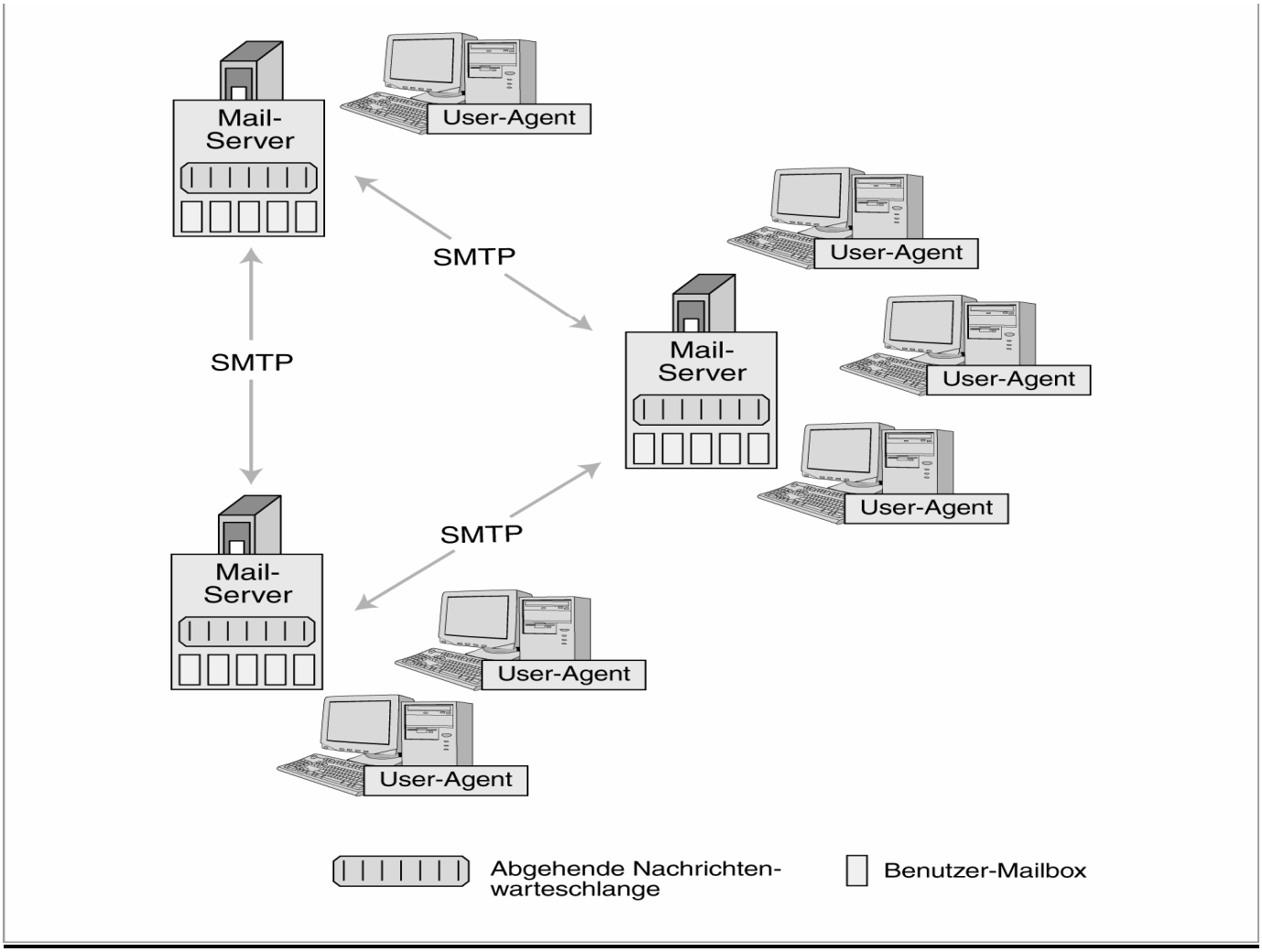
  - USER username
  - PASS password
  - LIST: Liste aller Dateien im entfernten Directory

- **FTP-Antworten Server -> Client**

  - Die Antworten tragen alle Nummern
  - Die Antworten enthalten auch oder keine Nachrichten im klaren Text

- **FTP-Sniffer-Abläufe (siehe Labor-Listings)**

E-Mail Dienst



Komponenten

- User Agents/UA (Clients) → Mail-Reader/Writer
- Mail Server (Servers)
- Übertragungs-Protokolle (SMTP , POP3)

Operation

- UA-Sender → sendet Mail zu einer Warteschlange innerhalb seiner Mailbox des Mail-Servers
- UA-Reader → holt Mail vom Mail-Server aus seiner Mail-Box

Protokolle

1. UA → Mail Server erfolgt per ***SMTP (RFC 821): siehe Labor-Listing***
2. Mail Server → UA erfolgt per ***POP3 (RFC 1939)***

POP3-Protokoll (RFC 1939)

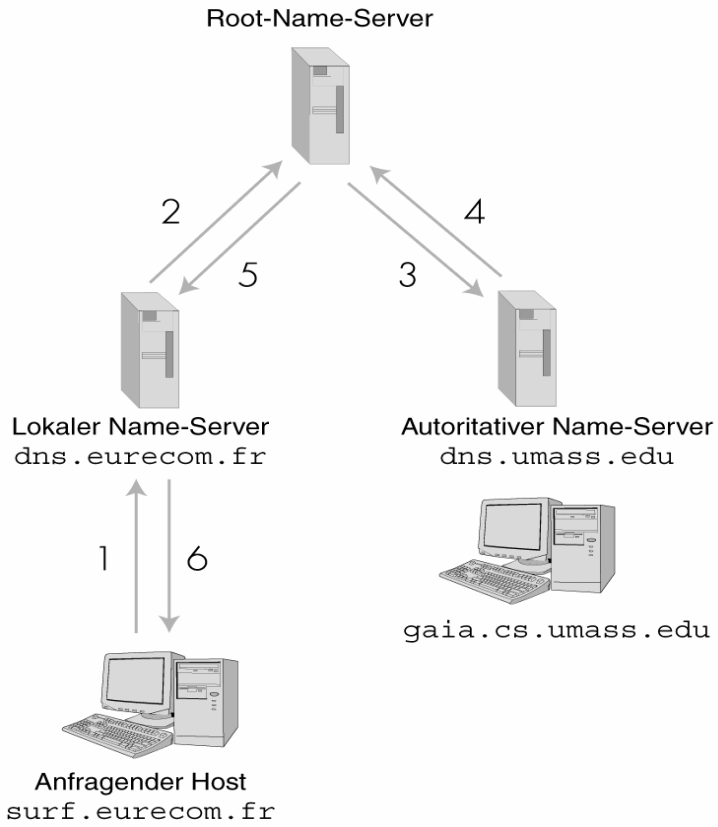
- Mail Zugangsprotokoll
- Es wird ***Port 110/TCP*** verwendet (auf Server-Seite)
- Phasen: - Authorisation: UA → UserName + PW
  - Transaction: UA → kann Mails zum Löschen markieren oder Statistiken abrufen
  - Aktualisierung: Beendigung der Sitzung erfolgt per „quit"-Befehl vom Client
    Die markierten Mails werden gelöscht
- POP3-Protokoll-Analyse: – siehe Sniffer Protokollablauf im Labor

HTTP- Mail Dienst

- User Agent => ein gewöhnlicher WEB-Browser
- Kommunikation zwischen UA und Mail-Server erfolgt per HTTP und nicht per SMTP, d.h. Mails senden/empfangen erfolgt per HTTP-Protokoll
- Mail-Server tauscht Nachrichten mit anderen Mail-Servers mittels SMTP aus
- Vorteil: - Für reisenden Benutzer sehr praktisch
  - Platz: Hotel, Internet-Cafés, WEB-TV

- HTTP- Mail Dienst Protokoll-Analyse: – siehe Sniffer Protokollablauf (Anhang)

Domain Name Service (DNS)-Dienst (1)
(RFC 1034 und RFC 1035)



Rolle

1. Hauptaufgabe => Umwandlung Host Namen → IP-Adressen

2. Host-Aliasing => Ein Host kann mehrere Alias-Namen haben

3. Mail-Server-Aliasing=> Mail-Anwendung kann vom DNS-Server den kanonischen Host Name für einen bestimmten Alias-Host Name abholen

4. Lastverteilung => Für replizierte WEB-Server existiert eine Gruppe von IP-Adressen. DNS enthält diese Gruppen von IP-Adressen. Wenn Clients per DNS diese Namen verlangen, erhalten sie alle Adressen in der Gruppe. Die 1. Adresse wird angesprochen. DNS-Server rotiert aber die Adressen in der Gruppe.

Domain Name Service (DNS)-Dienst (2)

Funktionsweise

- DNS-Server sind hierarchisch organisiert
- DNS-Server sind über die ganze Welt verteilt

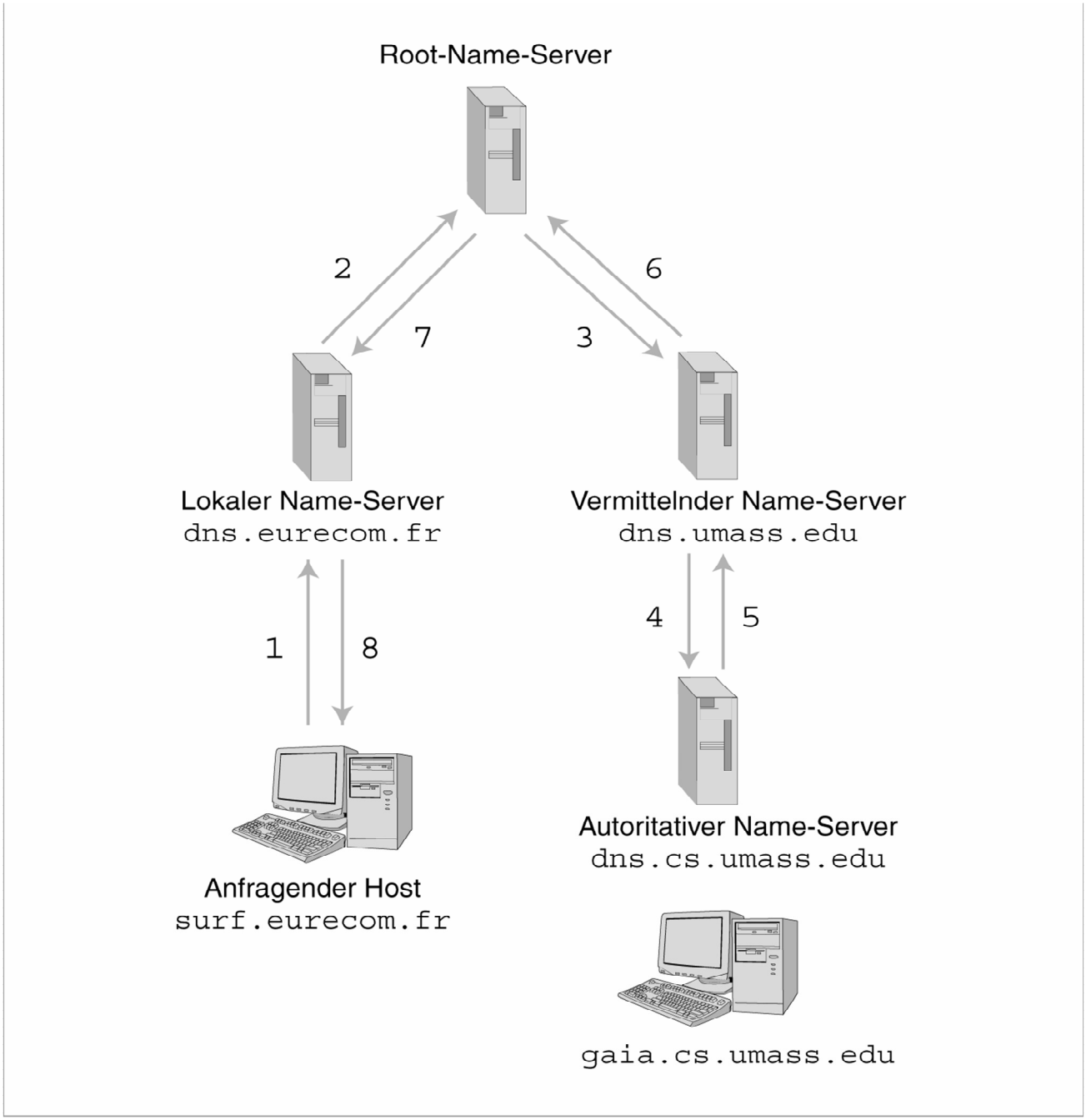
- ***DNS-Typen**:*

1. **Lokaler Name Server (LNS)**
   - jede Institution verfügt über ein LNS
   - Er wird als Default NS betrachtet
   - Die Adresse des LNS wird manuell in jedem Host konfiguriert.

2. **Root Name Server (RNS)**
   - Es gibt ca. 12 RNS im Internet
   - Falls LNS die Anfrage die Anfrage nicht beantworten kann, dann wir die Anfrage an den RNS weitergeleitet
   - RNS kennt die IP-Adresse eines Name-Servers (sog. Autoritativen Name Server (ANS) der die Anfrage beantworten kann. D.h. RNS leitet die Anfrage an ANS weiter.

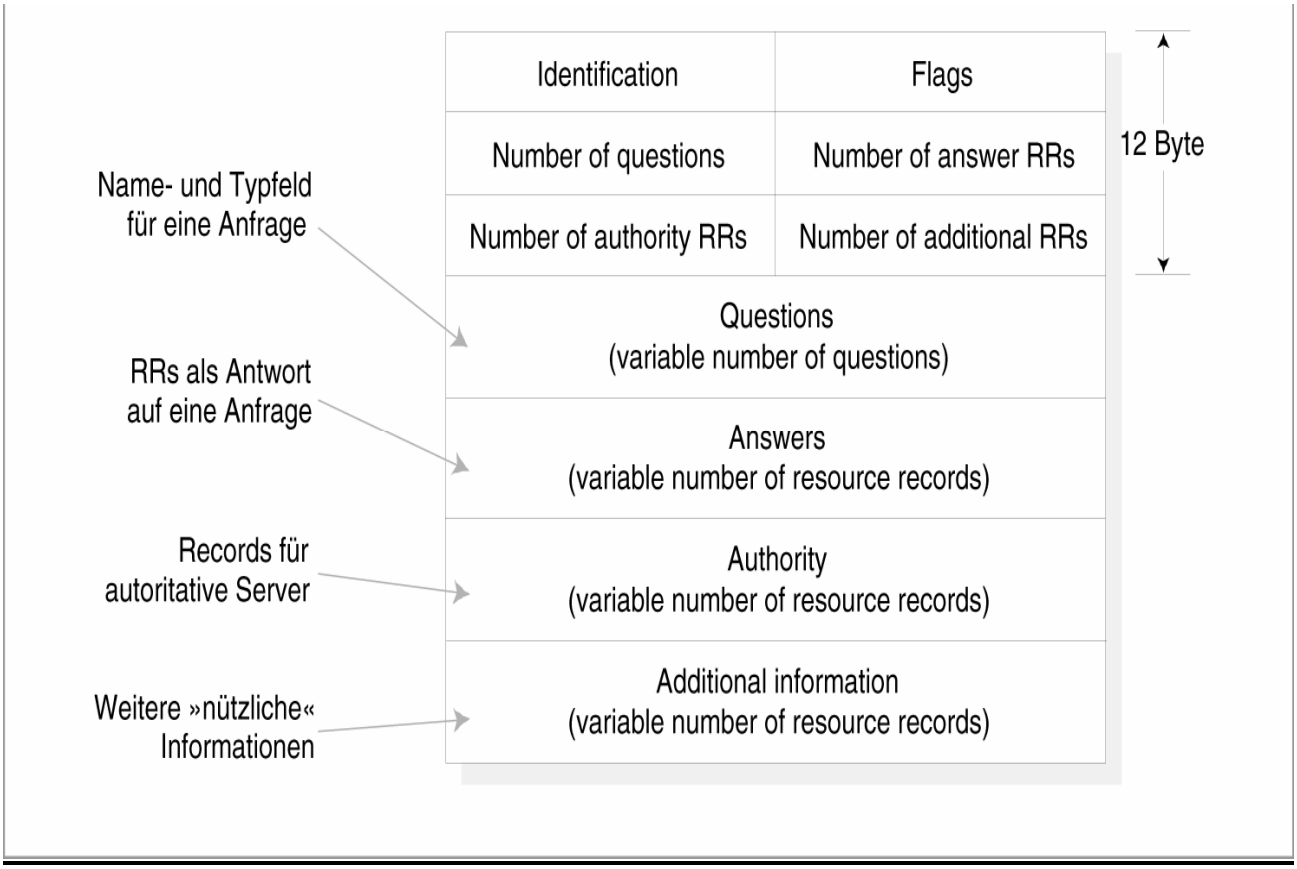
3. **Autoritativer Name Server (ANS)**
   - Jeder Host im Internet ist bei einem ANS registriert
   - ANS reagiert immer mit einer Antwort, in der sich die angefragte Umwandlung des Namen in der IP-Adresse befindet.

Domain Name Service (DNS)-Dienst (3)



- Rekursive Abfragen für die Auflösung der IP-Adresse aus der Namens-Adresse
- In der Kette zwischen RNS und ANS können mehrere sog. vermittelnde Name-Server sein

DNS-Nachrichten (1)



1. ***Header Abschnitt:***
   - ID → Anfrage / Antwort – ID
   - FLAG → Query/Reply: d.h. Anfrage oder Antwort
   - Nr of Questions) Häufigkeit
   - Nr. of Answer Resource Records (RR)) der anderen
   - Nr. of Authority RRs) 4 Datenabschnitte
   - Nr. of Additional RRs) des Headers
   - Questions→ Namensfeld + Typfeld (A oder MX)
   - Answer→ - RR der Anfrage und
     - die RR der Antwort, die aus ein oder mehreren RRs (ein Host Name) besteht, kann mehrere IP-Adressen enthalten
   - Authority→ RR anderer ANS
   - Additional:

DNS-Nachrichten (2)

2. ***Resource Records (RR)***
   Besteht aus 4 Feldern:
   - Name
   - Wert
   - Typ
   - TTL

   Abhängig vom Typ haben die RRs folgende Bedeutung:

   Typ = A: - Name = Host Name
   - Wert = IP-Adresse des Hosts
   - Bsp: .....

   Typ = NS: - Name = Domain
   - Wort = Host Name eines ANS
     ANS kann die IP-Adresse des Hosts innerhalb der Domain finden

   Typ = CNAME: - Name = Alias Host Name
   - Wert = kanonischer Host Name

   Typ = MX: - Name = Alias Host Name
   - Wert = Host Name eines Mail-Servers
     d.h. man kann für Mail-Server Aliasnamen verwenden

   Bsp. Ausdruck aus dem DNS-Server der FH-Regensburg

DNS-Protokollablauf

Sniffer Network Analyzer data from 11-Oct-102 at 09:30:48, unsaved capture data, Page 1
.....
.....
- - - - - - - - - - - - - - - - - Frame 3 - - - - - - - - - - - - - - - - -
SUMMARY:

	Delta T	Destination	Source	Summary
3	**0.0001**	**DNS**	**[194.95.109.136)**	**DLC Ethertype=0800, size=80 bytes** **IP D=[194.95.104.1] S=[194.95.109.136] LEN=46 ID=86** **UDP D=53 S=1035 LEN=46** ***DNS C ID=6 OP=QUERY NAME=ftp.uni-paderborn.de***

DLC: ----- DLC Header -----
DLC:
DLC: Frame 3 arrived at 09:30:52.5481; frame size is 80 (0050 hex) bytes.
DLC: Destination = Station 00A08E30D27F
DLC: Source = Station 00065B75C343, RFHPCI136
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 66 bytes
IP: Identification = 86
IP: Flags = 0X
IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops
IP: Protocol = 17 (UDP)
IP: Header checksum = E00C (correct)
IP: Source address = [194.95.109.136]
IP: Destination address = [194.95.104.1], DNS
IP: No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 1035
UDP: Destination port = 53 (Domain)
UDP: Length = 46
UDP: Checksum = 7BC2 (correct)

UDP:
DNS: ----- Internet Domain Name Service header -----
DNS:
DNS: ID = 6
DNS: Flags = 01
DNS: 0... .... = Command
DNS: .000 0... = Query
DNS: .... ..0. = Not truncated
DNS: ....1 = Recursion desired
DNS: Flags = 0X
DNS: ...0 .... = Unicast packet
DNS: Question count = 1, Answer count = 0
DNS: Authority count = 0, Additional record count = 0
DNS:
DNS: Question section:
DNS: Name = ftp.uni-paderborn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)
DNS:
DNS: [Normal end of "Internet Domain Name Service header".]
DNS:

==========
- - - - - - - - - - - - - - - - Frame 4 - - - - - - - - - - - - - - - -

	Delta T	Destination	Source	Summary
4	**0.0023**	**[194.95.109.136)**	**DNS**	**DLC Ethertype=0800, size=273 bytes**
				IP D=[194.95.109.136] S=[194.95.104.1] LEN=239 ID=8075
				UDP D=1035 S=53 LEN=239
				DNS R ID=6 STAT=OK NAME=ftp.uni-paderborn.de

DLC: ----- DLC Header -----
DLC:
DLC: Frame 4 arrived at 09:30:52.5504; frame size is 273 (0111 hex) bytes.
DLC: Destination = Station 00065B75C343, RFHPCI136
DLC: Source   = Station 00A08E30D27F
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length   = 259 bytes
IP: Identification = 8075
IP: Flags   = 4X

IP: .1.. .... = don't fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 254 seconds/hops
IP: Protocol = 17 (UDP)
IP: Header checksum = 0216 (correct)
IP: Source address = [194.95.104.1], DNS
IP: Destination address = [194.95.109.136]
IP: No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 53 (Domain)
UDP: Destination port = 1035
UDP: Length = 239
UDP: Checksum = 6FB1 (correct)
UDP:

DNS: ----- Internet Domain Name Service header -----

DNS: ID = 6
DNS: Flags = 81
DNS: 1... .... = Response
DNS: .... .0.. = Not authoritative answer
DNS: .000 0... = Query
DNS: .... ..0. = Not truncated
DNS: Flags = 8X
DNS: ...0 .... = Unicast packet
DNS: 1... .... = Recursion available
DNS: Response code = OK (0)
DNS: Question count = 1, Answer count = 2
DNS: Authority count = 4, Additional record count = 4
DNS:
DNS: Question section:
DNS: Name = ftp.uni-paderborn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)

DNS: Answer section 1:
DNS: Name = ftp.uni-paderborn.de
DNS: Type = Canonical name for alias (CNAME,5)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 85602 (seconds)
DNS: Length = 11
DNS: Canonical name = gigaserv.uni-paderborn.de

DNS: Answer section 2:
DNS: Name = gigaserv.uni-paderborn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 85602 (seconds)
DNS: Length = 4
DNS: Address = [131.234.25.10]

DNS: Authority section 1:
DNS: Name = uni-paderborn.de
DNS: Type = Authoritative name server (NS,2)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 63734 (seconds)
DNS: Length = 2
DNS: Name server domain name = uni-paderborn.de

DNS: Authority section 2:
DNS: Name = uni-paderborn.de
DNS: Type = Authoritative name server (NS,2)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 63734 (seconds)
DNS: Length = 20
DNS: Name server domain name = ws-was.win-ip.dfn.de

DNS: Authority section 3:
DNS: Name = uni-paderborn.de
DNS: Type = Authoritative name server (NS,2)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 63734 (seconds)
DNS: Length = 10
DNS: Name server domain name = info-f1.uni-paderborn.de

DNS: Authority section 4:
DNS: Name = uni-paderborn.de
DNS: Type = Authoritative name server (NS,2)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 63734 (seconds)
DNS: Length = 10
DNS: Name server domain name = ws-han1.win-ip.dfn.de

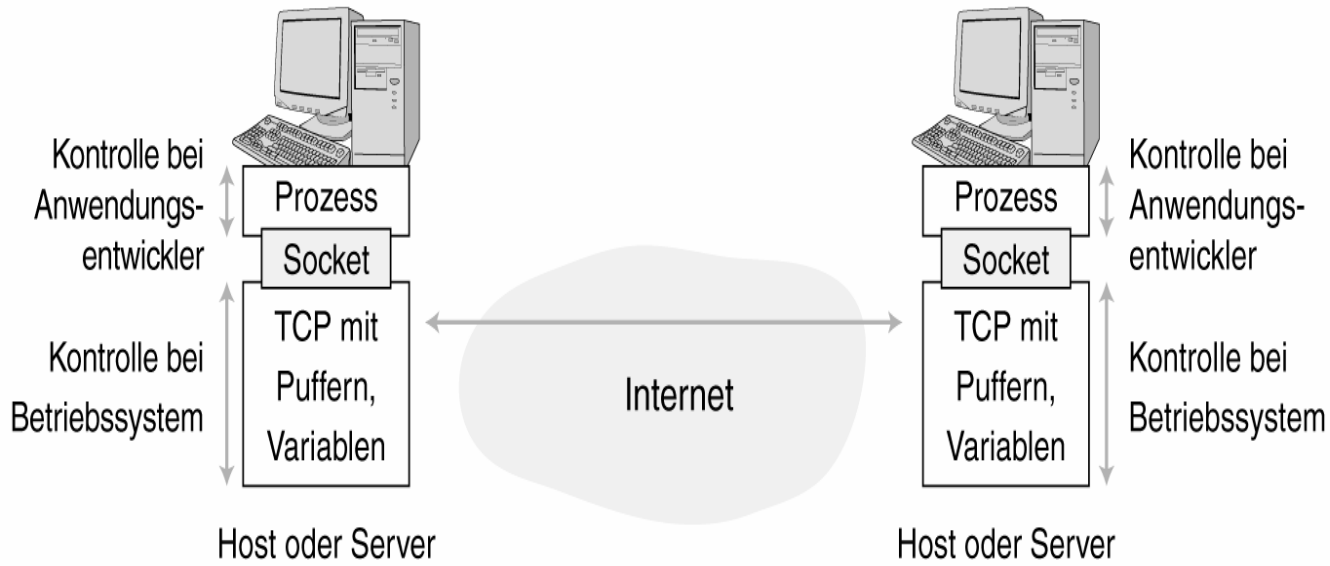
DNS: Additional record section 1:
DNS: Name = uni-paderborn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 63734 (seconds)
DNS: Length = 4
DNS: Address = [131.234.22.30]

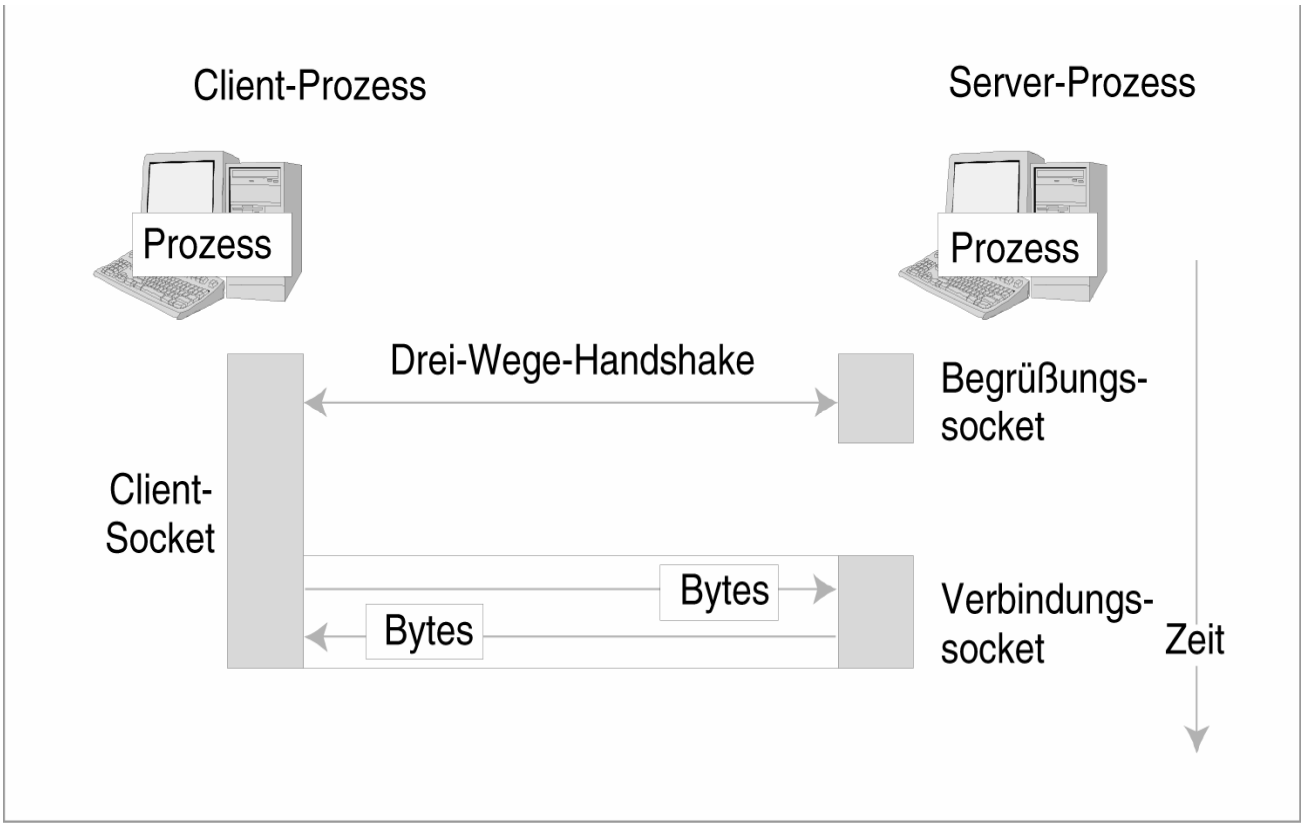
DNS: Additional record section 2:
DNS: Name = ws-was.win-ip.dfn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 86015 (seconds)
DNS: Length = 4
DNS: Address = [193.174.75.110]

DNS: Additional record section 3:
DNS: Name = info-f1.uni-paderborn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 63734 (seconds)
DNS: Length = 4
DNS: Address = [131.234.22.3]

DNS: Additional record section 4:
DNS: Name = ws-han1.win-ip.dfn.de
DNS: Type = Host address (A,1)
DNS: Class = Internet (IN,1)
DNS: Time-to-live = 13023 (seconds)
DNS: Length = 4
DNS: Address = [193.174.75.150]
DNS:
DNS: [Normal end of "Internet Domain Name Service header".]

Anwendungs-Programmierung mit Sockets (1)





Funktionsweise des Client-Server Modells

Client Prozess → leitet den Kontakt zum Server-PR ein.
Voraussetzung: Server-PR ist aktiv

Socket → die Tür (TCP-Port) wird über die Verbindung Client-Server festgelegt. Auf beiden Seiten muss ein solcher Socket verfügbar sein.

Verbindungsaufbau/ Verbindungsabbau → erfolgt mittels sog. ***„Drei Wege Handshake“*** im Falle von TCP-Transport Schicht

Client-Server-Anwendung → wird auch ***Socket-Programmierung*** genannt

TCP-Protokoll-Analyse (Verbindungsaufbau)

Sniffer Network Analyzer data from 11-Oct-102 at 09:30:48, unsaved capture data, Page 1
- - - - - - - - - - - - - - - - - Frame 5 - - - - - - - - - - - - - - - - -

SUMMARY:

	Delta T	Destination	Source	Summary
5	**0.0031**	**gigaserv.uni-..**	**[194.95.109.136)**	**DLC Ethertype=0800, size=62 bytes**
				IP D=[131.234.25.10] S=[194.95.109.136] LEN=28 ID=87
				TCP D=21 S=1036 SYN SEQ=2392861514 LEN=0 WIN=16384

DLC: ----- DLC Header -----
DLC:
DLC: Frame 5 arrived at 09:30:52.5536; frame size is 62 (003E hex) bytes.
DLC: Destination = Station 00A08E30D27F
DLC: Source = Station 00065B75C343, RFHPCI136
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 48 bytes
IP: Identification = 87
IP: Flags = 4X
IP: .1.. .... = don't fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops
IP: Protocol = 6 (TCP)
IP: Header checksum = 2D95 (correct)
IP: Source address = [194.95.109.136]
IP: Destination address = [131.234.25.10], gigaserv.uni-paderborn.de
IP: No options
IP:
TCP: ----- TCP header -----
TCP:
TCP: Source port = 1036
TCP: Destination port = 21 (FTP)
TCP: Initial sequence number = 2392861514
TCP: Data offset = 28 bytes
TCP: Flags = 02
TCP: ..0. .... = (No urgent pointer)
TCP: ...0 .... = (No acknowledgment)

TCP: .... 0... = (No push)
TCP: .... .0.. = (No reset)
TCP: .... ..1. = SYN
TCP: ....0 = (No FIN)
TCP: Window = 16384
TCP: Checksum = B837 (correct)
TCP:
TCP: Options follow
TCP: Maximum segment size = 1460
TCP: No-op
TCP: No-op
TCP: Unknown option 4
TCP: 1 byte(s) of header padding
TCP:

- - - - - - - - - - - - - - - - Frame 6 - - - - - - - - - - - - - - - -

	Delta T	Destination	Source	Summary
6	**0.0243**	**[194.95.109.136**	**gigaserv.uni-..**	**DLC Ethertype=0800, size=62 bytes**
				IP D=[194.95.109.136] S=[131.234.25.10] LEN=28 ID=37079
				TCP D=1036 S=21 SYN ACK=2392861515 SEQ=3168076761 LEN=0 WIN=8760

DLC: ----- DLC Header -----
DLC:
DLC: Frame 6 arrived at 09:30:52.5779; frame size is 62 (003E hex) bytes.
DLC: Destination = Station 00065B75C343, RFHPCI136
DLC: Source = Station 00A08E30D27F
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 48 bytes
IP: Identification = 37079
IP: Flags = 4X
IP: .1.. .... = don't fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 246 seconds/hops
IP: Protocol = 6 (TCP)

IP: Header checksum = 2714 (correct)
IP: Source address   = [131.234.25.10], gigaserv.uni-paderborn.de
IP: Destination address = [194.95.109.136]
IP: No options
IP:
TCP: ----- TCP header -----
TCP:
TCP: Source port   = 21 (FTP)
TCP: Destination port   = 1036
TCP: Initial sequence number = 3168076761
TCP: Acknowledgment number   = 2392861515
TCP: Data offset   = 28 bytes
TCP: Flags   = 12
TCP: ..0. .... = (No urgent pointer)
TCP: ...1 .... = Acknowledgment
TCP: .... 0... = (No push)
TCP: .... .0.. = (No reset)
TCP: .... ..1. = SYN
TCP: ....0 = (No FIN)
TCP: Window   = 8760
TCP: Checksum   = 1540 (correct)
TCP:
TCP: Options follow
TCP: No-op
TCP: No-op
TCP: Unknown option 4
TCP: 5 byte(s) of header padding
TCP:

Sniffer Network Analyzer data from 11-Oct-102 at 09:30:48, unsaved capture data, Page 3

- - - - - - - - - - - - - - - - - Frame 7 - - - - - - - - - - - - - - - - -

	Delta T	Destination	Source	Summary
7	**0.0001**	**gigaserv.uni-..**	**[194.95.109.136**	**DLC Ethertype=0800, size=60 bytes**
				IP D=[131.234.25.10] S=[194.95.109.136]
				LEN=20 ID=88
				TCP D=21 S=1036 ACK=3168076762
				WIN=17520

DLC: ----- DLC Header -----
DLC:
DLC: Frame 7 arrived at 09:30:52.5780; frame size is 60 (003C hex) bytes.
DLC: Destination = Station 00A08E30D27F
DLC: Source   = Station 00065B75C343, RFHPCI136
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00

IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 40 bytes
IP: Identification = 88
IP: Flags = 4X
IP: .1.. .... = don't fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops
IP: Protocol = 6 (TCP)
IP: Header checksum = 2D9C (correct)
IP: Source address = [194.95.109.136]
IP: Destination address = [131.234.25.10], gigaserv.uni-paderborn.de
IP: No options
IP:
TCP: ----- TCP header -----
TCP:
TCP: Source port = 1036
TCP: Destination port = 21 (FTP)
TCP: Sequence number = 2392861515
TCP: Acknowledgment number = 3168076762
TCP: Data offset = 20 bytes
TCP: Flags = 10
TCP: ..0. .... = (No urgent pointer)
TCP: ...1 .... = Acknowledgment
TCP: .... 0... = (No push)
TCP: .... .0.. = (No reset)
TCP: .... ..0. = (No SYN)
TCP: ....0 = (No FIN)
TCP: Window = 17520
TCP: Checksum = 1FCC (correct)
TCP: No TCP options
TCP:

TCP-Protokoll-Analyse (Verbindungsabbau)

Sniffer Network Analyzer data from 11-Oct-102 at 09:30:48, unsaved capture data, Page 1

- - - - - - - - - - - - - - - - Frame 48 - - - - - - - - - - - - - - - -

	Delta T	Destination	Source	Summary
48	**0.0012**	**gigaserv.uni-..**	**[194.95.109.1..**	**DLC Ethertype=0800, size=60 bytes**
				IP D=[131.234.25.10] S=[194.95.109.136] LEN=20 ID=109
				TCP D=21 S=1036 FIN ACK=3168079573
				SEQ=2392861579 LEN=0 WIN=16822

DLC: ----- DLC Header -----
DLC:
DLC: Frame 48 arrived at 09:31:02.9334; frame size is 60 (003C hex) bytes.
DLC: Destination = Station 00A08E30D27F
DLC: Source = Station 00065B75C343, RFHPCI136
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 40 bytes
IP: Identification = 109
IP: Flags = 4X
IP: .1.. .... = don't fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops
IP: Protocol = 6 (TCP)
IP: Header checksum = 2D87 (correct)
IP: Source address = [194.95.109.136]
IP: Destination address = [131.234.25.10], gigaserv.uni-paderborn.de
IP: No options
IP:
TCP: ----- TCP header -----
TCP:
TCP: Source port = 1036
TCP: Destination port = 21 (FTP)
TCP: Sequence number = 2392861579
TCP: Acknowledgment number = 3168079573
TCP: Data offset = 20 bytes

TCP: Flags = 11
TCP: ..0. .... = (No urgent pointer)
TCP: ...1 .... = Acknowledgment
TCP: .... 0... = (No push)
TCP: .... .0.. = (No reset)
TCP: .... ..0. = (No SYN)
TCP: ....1 = FIN
TCP: Window = 16822
TCP: Checksum = 174A (correct)
TCP: No TCP options
TCP:

Sniffer Network Analyzer data from 11-Oct-102 at 09:30:48, unsaved capture data, Page 2

- - - - - - - - - - - - - - - - - Frame 49 - - - - - - - - - - - - - - - - -

	Delta T	Destination	Source	Summary
49	**0.0066**	**[194.95.109.1..**	**gigaserv.uni-..**	**DLC Ethertype=0800, size=60 bytes**
				IP D=[194.95.109.136] S=[131.234.25.10] LEN=20 ID=37099
				TCP D=1036 S=21 FIN ACK=2392861579 SEQ=3168079573 LEN=0 WIN=8760

DLC: ----- DLC Header -----
DLC:
DLC: Frame 49 arrived at 09:31:02.9401; frame size is 60 (003C hex) bytes.
DLC: Destination = Station 00065B75C343, RFHPCI136
DLC: Source = Station 00A08E30D27F
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 10
IP: 000. .... = routine
IP: ...1 .... = low delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 40 bytes
IP: Identification = 37099
IP: Flags = 4X
IP: .1.. .... = don't fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 246 seconds/hops
IP: Protocol = 6 (TCP)
IP: Header checksum = 26F8 (correct)
IP: Source address = [131.234.25.10], gigaserv.uni-paderborn.de
IP: Destination address = [194.95.109.136]
IP: No options
IP:

TCP: ----- TCP header -----

TCP:
TCP: Source port = 21 (FTP)
TCP: Destination port = 1036
TCP: Sequence number = 3168079573
TCP: Acknowledgment number = 2392861579
TCP: Data offset = 20 bytes
TCP: Flags = 11
TCP: ..0. .... = (No urgent pointer)
TCP: ...1 .... = Acknowledgment
TCP: .... 0... = (No push)
TCP: .... .0.. = (No reset)
TCP: .... ..0. = (No SYN)
TCP: ....1 = FIN
TCP: Window = 8760
TCP: Checksum = 36C8 (correct)
TCP: No TCP options
TCP: