Netzwerksicherheit
Anwendungsprotokolle

Stephen Wolthusen

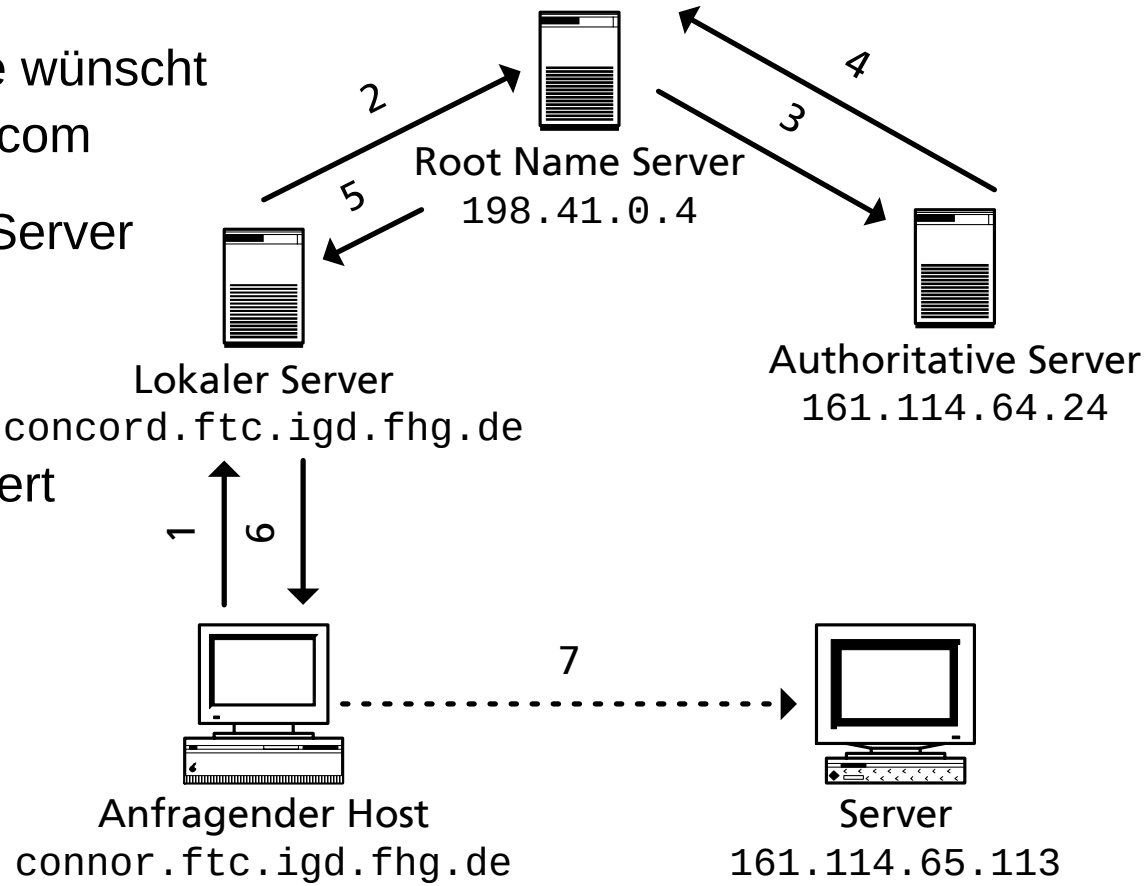
Fraunhofer IGD Institut Graphische Datenverarbeitung

Domain Name Service

- Verteilung der Host-Tabelle (`HOSTS.TXT`) via FTP wurde unhandlich
- Verteilte Datenbank zur Auflösung symbolischer Namen in IP-Adressen und umgekehrt
  - Erste Implementierung (Jeeves, 1983)
  - De facto Standard: BIND (Berkeley Internet Name Domain)
    - Wird vom ISC gepflegt. Aktuell: Bind 8.2.5, 9.1.3
- Adreßraum wird von ICANN (zuvor: Network Solutions im Auftrag der NSF) gepflegt
  - Registrars dürfen Einträge für bestimmte Top-Level Domains entgegennehmen

Einfache DNS-Anfrage

- Host connor.ftc.igd.fhg.de wünscht Adresse von www.digital.com
- Kontaktiert lokalen DNS-Server concord.ftc.igd.fhg.de
- Adresse nicht im Cache, Root Server wird kontaktiert
- Adresse nicht im Cache, Authoritative Server wird kontaktiert
- Antwort wird über lokalen Server an Host weitergeleitet

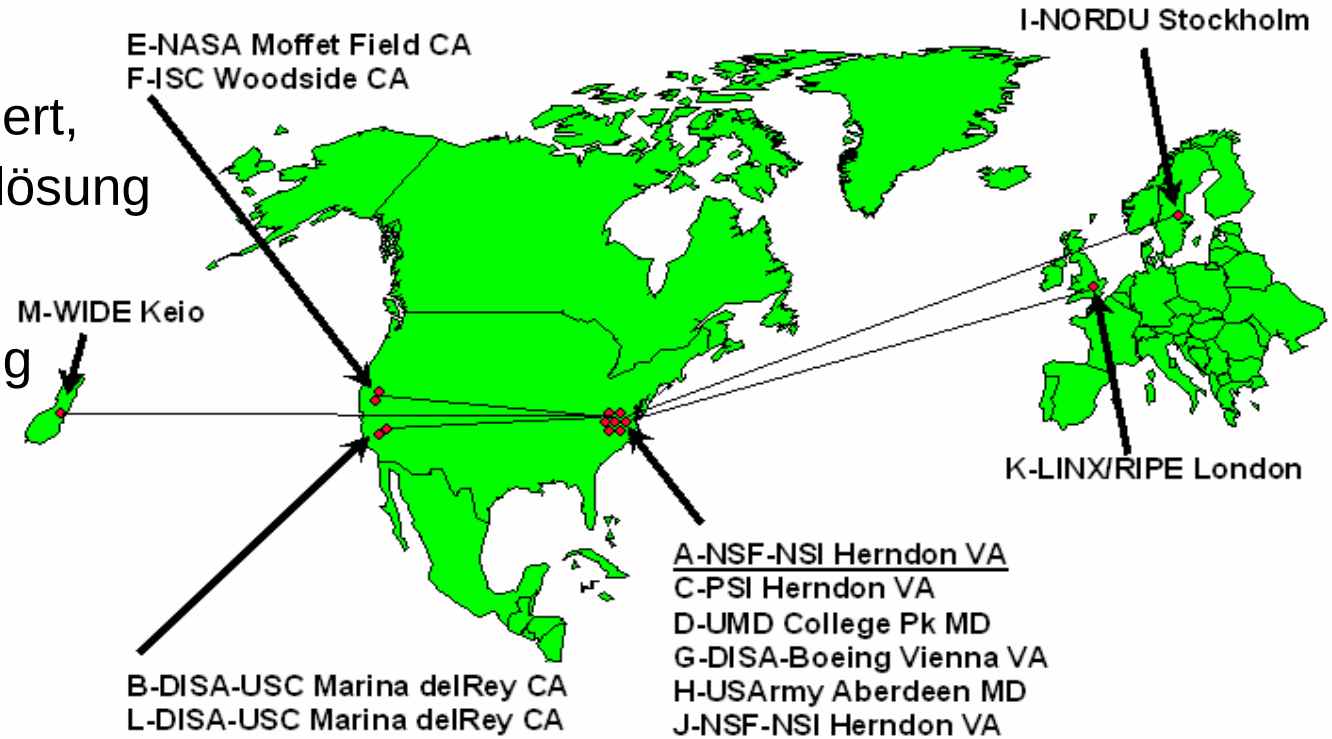


Fraunhofer IGD Institut Graphische Datenverarbeitung

DNS Root Server

- Werden kontaktiert, wenn lokale Auflösung erfolglos war
- Stellen Auflösung anhand „authoritative servers“ fest

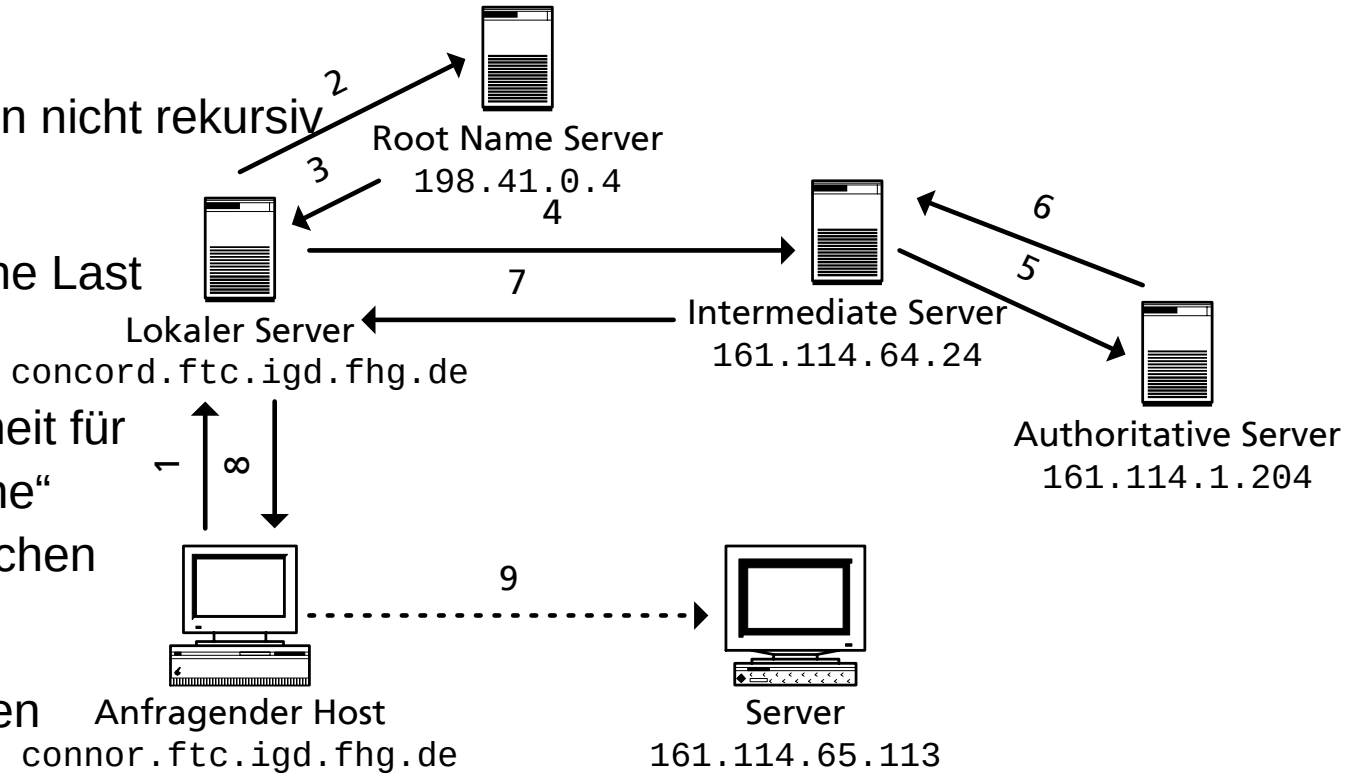
  Auflösung wird anfragendem Server mitgeteilt



Quelle: World Internetworking Alliance, 1998

Adreßauflösung in DNS (2)

- Anfragen werden nicht rekursiv gestellt:
- Würde erhebliche Last verursachen
- Keine Gelegenheit für Server, „nützliche“ Adressen zu cachen
- Statt dessen: Iterative Anfragen
- Anfragen werden an zwischengeschaltete Server geleitet



Fraunhofer IGD Institut Graphische Datenverarbeitung

Resource Records (RRs) im DNS (Auswahl)

- A Records
  - Abbildung symbolischer Namen in IP-Adressen
- PTR-Records
  - Abbildung von IP-Adressen in symbolische Namen
- NS-Records
  - Enthält Name Server für eine Domain
- CNAME-Records
  - Aliase („canonical names“) für andere Einträge
- MX-Records
  - Mail Exchange Hosts (mehr dazu später)

Nachrichten in DNS

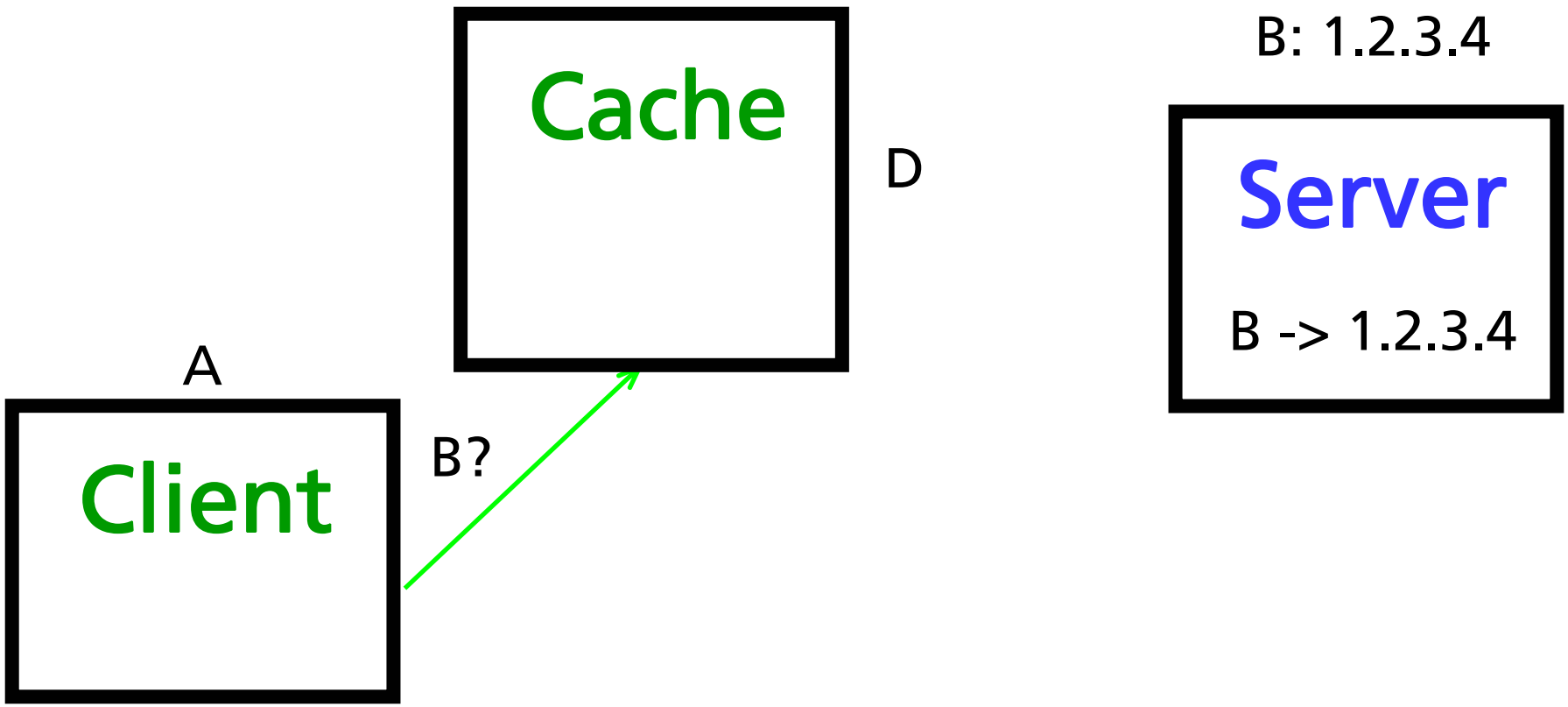
- Anfrage und Antwort weisen das gleiche Format auf
- Anfrage, Antwort werden anhand der 16 Bit Identification aufeinander abgebildet
- Flags geben an, ob
  - Anfrage oder Antwort
  - Rekursion erwünscht ist
  - Rekursion möglich ist
  - Antwort “authoritative” ist

Identification	Flags
# Anfragen	# Antwort-RRs
# Authority-RRs	# Zusatz-RRs
Anfragen (Anzahl variabel)	
Antworten (Anzahl variabel)	
Authority (Anzahl variabel)	
Zusatzinformation (Anzahl variabel)	

Problemes im DNS

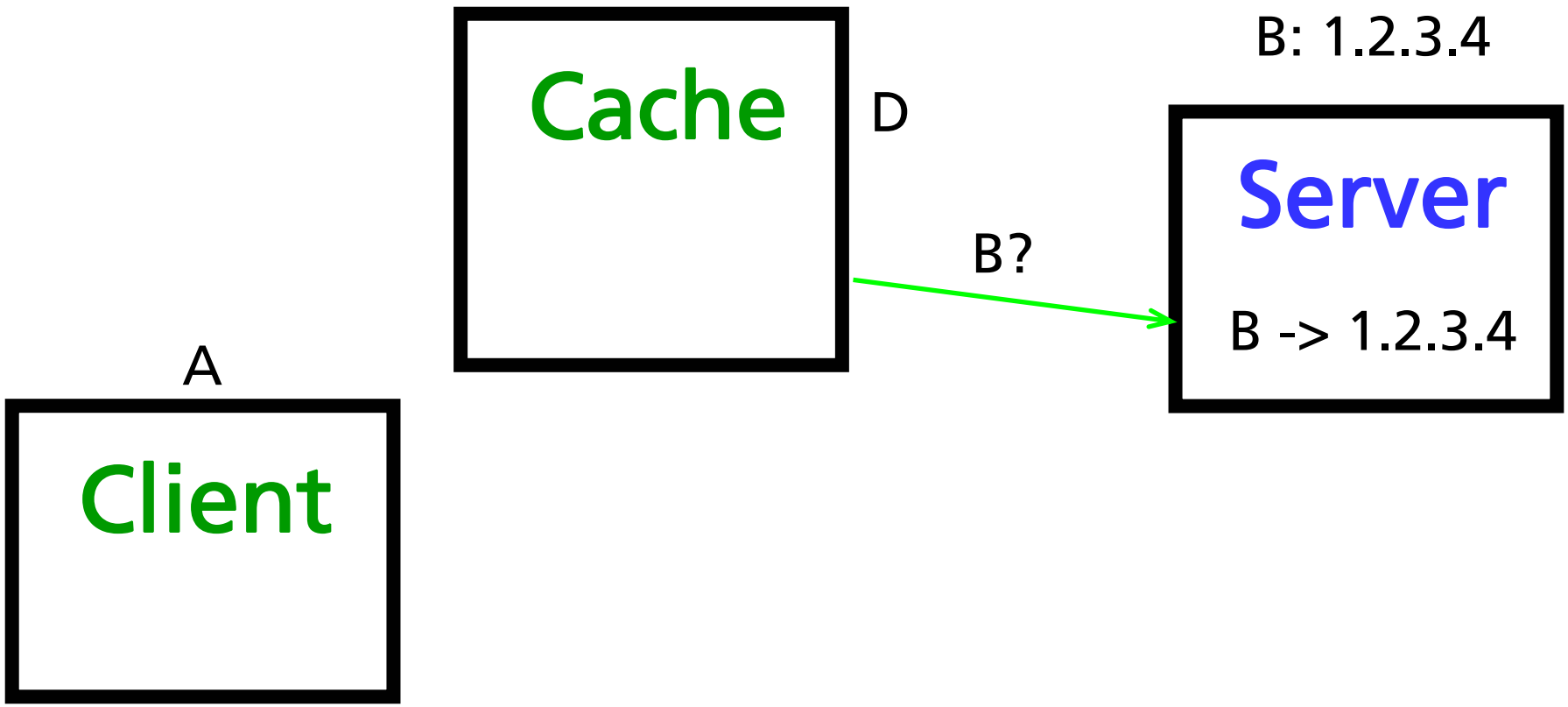
- Root Name Servers sind verwundbar
  - 1997 eindrucksvoll von einem Mitarbeiter des InterNIC bewiesen
- Zur Optimierung von Caches verwendete Verfahren zum „piggybacking“ von weiteren Antworten können mißbraucht werden: DNS Cache Pollution
  - Zwischengelagerter Server versendet zusätzlich falsche Reply-RRs
  - Andere Server, Hosts nehmen dies als gültige Antwort hin
  - Dies geschieht nicht nur bei Angriffen...

DNS-Anfrage: A fragt D nach Adresse von B

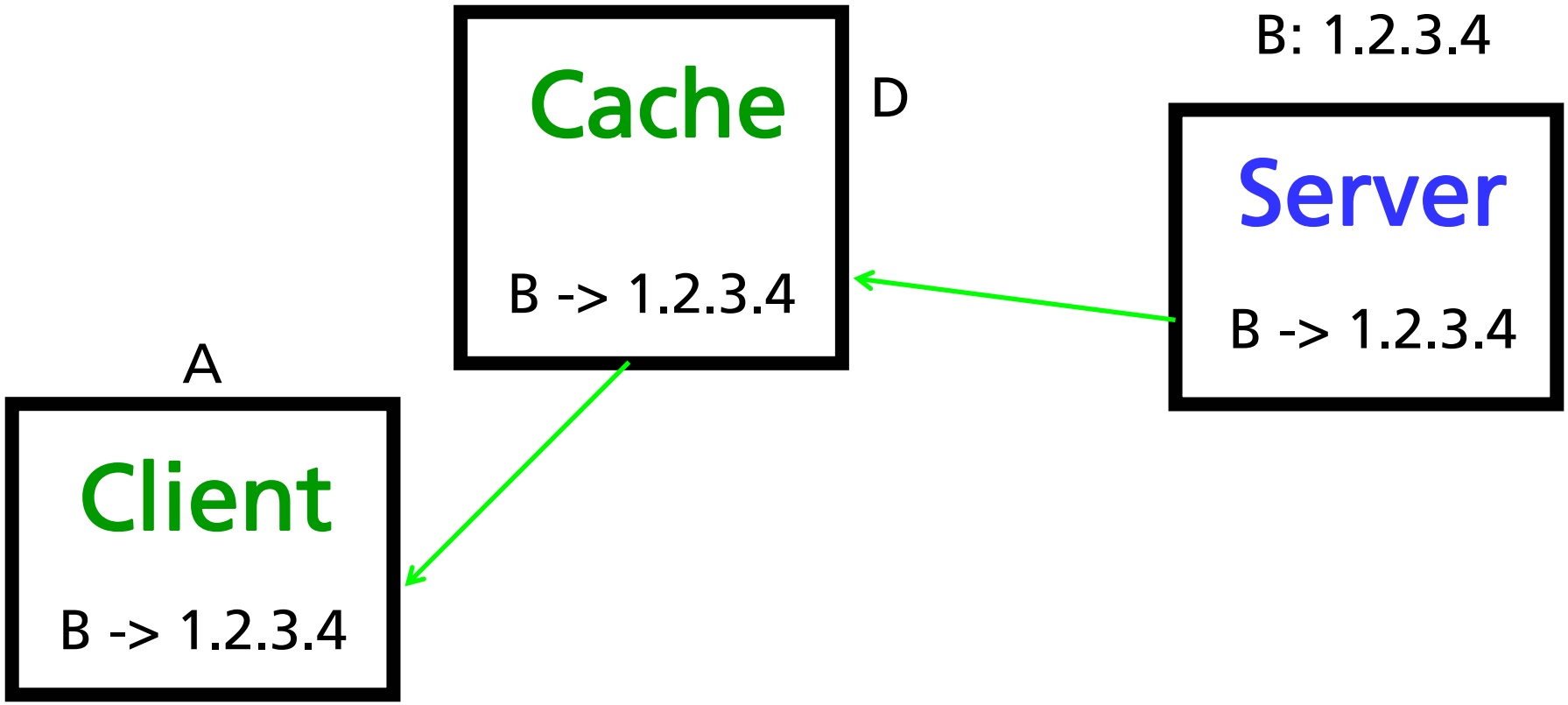


Fraunhofer IGD Institut Graphische Datenverarbeitung

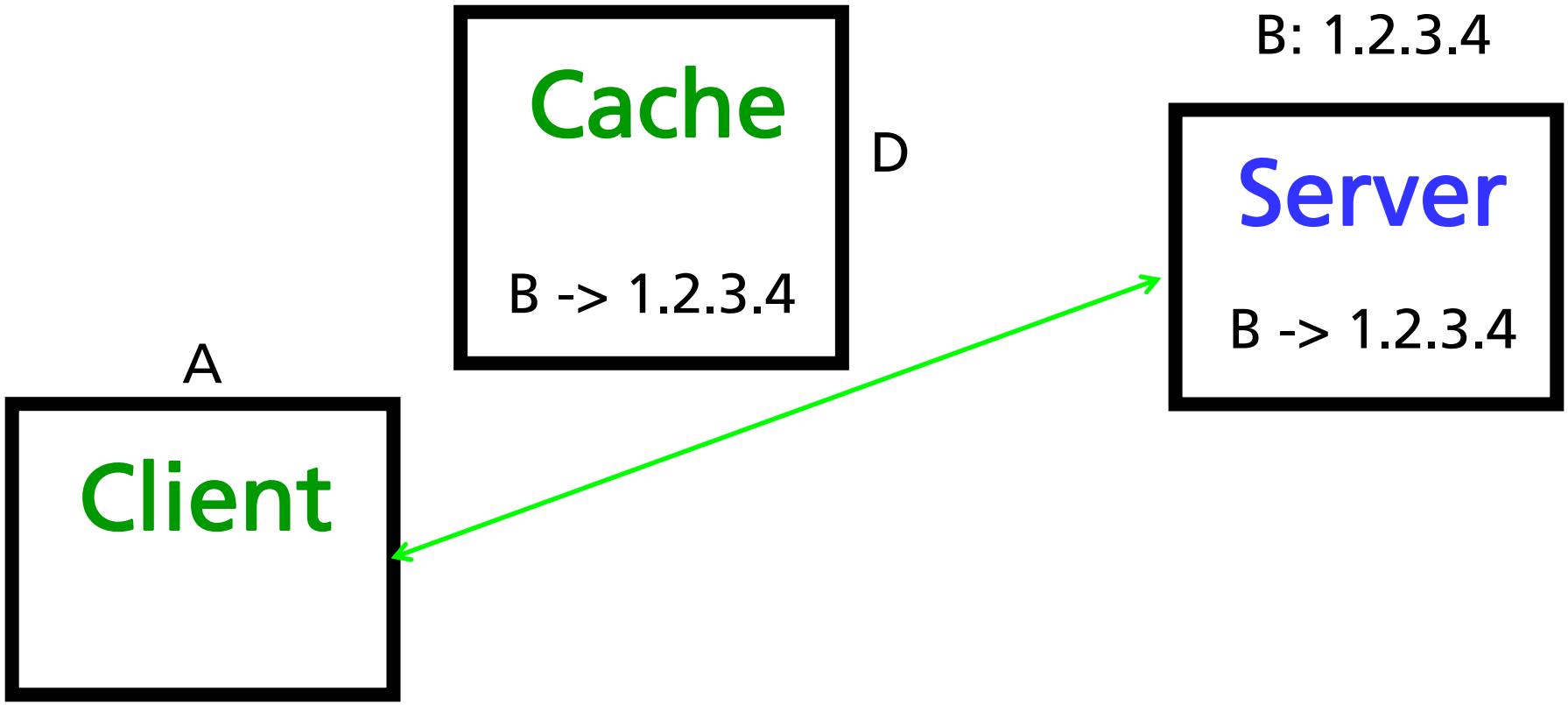
DNS-Anfrage: D fragt B oder Autorität für B



DNS-Anfrage: B antwortet, D cached, A erhält Antwort



DNS-Anfrage: A verwendet Antwort



DNS-Caching: D speichert B für Eintrags-Lebensdauer

B: 1.2.3.4

Server

B -> 1.2.3.4

D

Cache

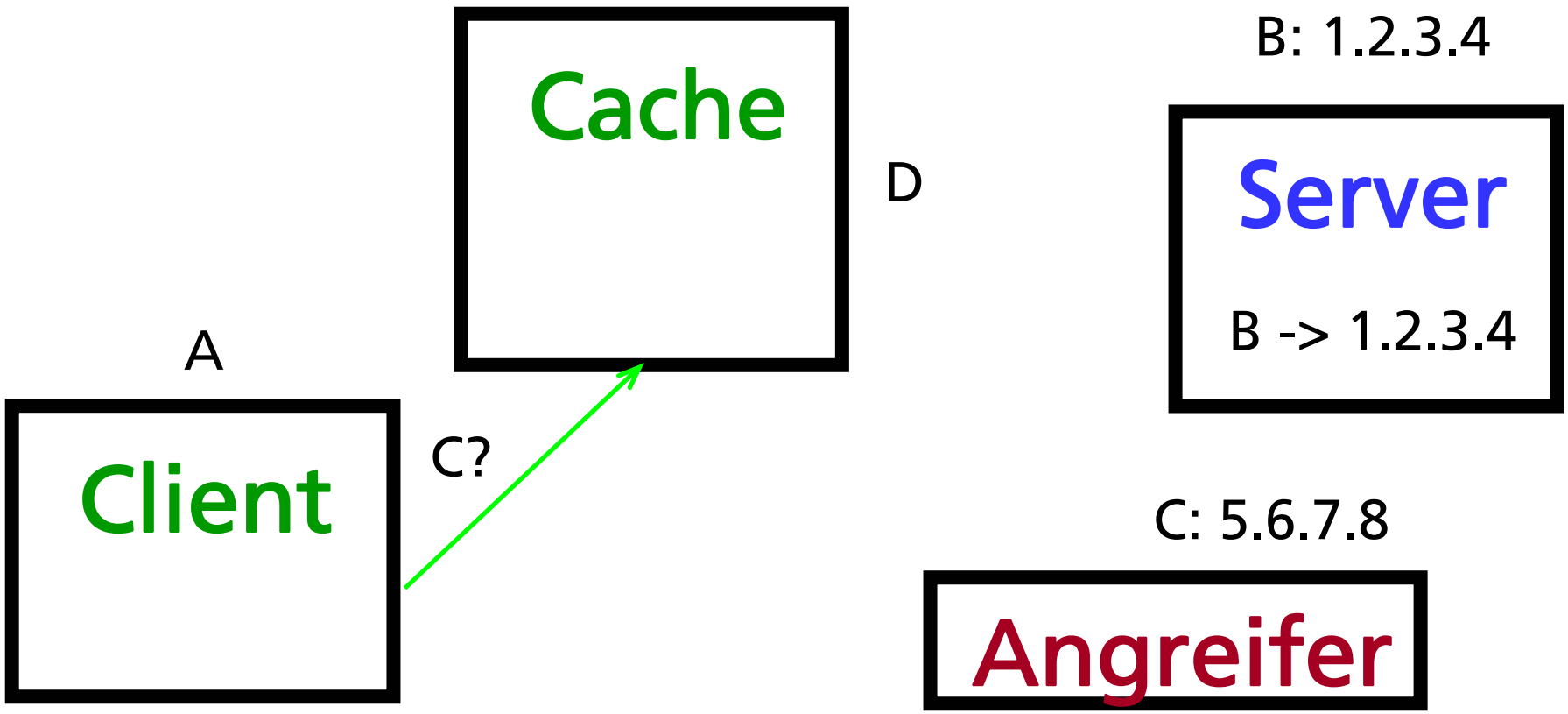
B -> 1.2.3.4

A

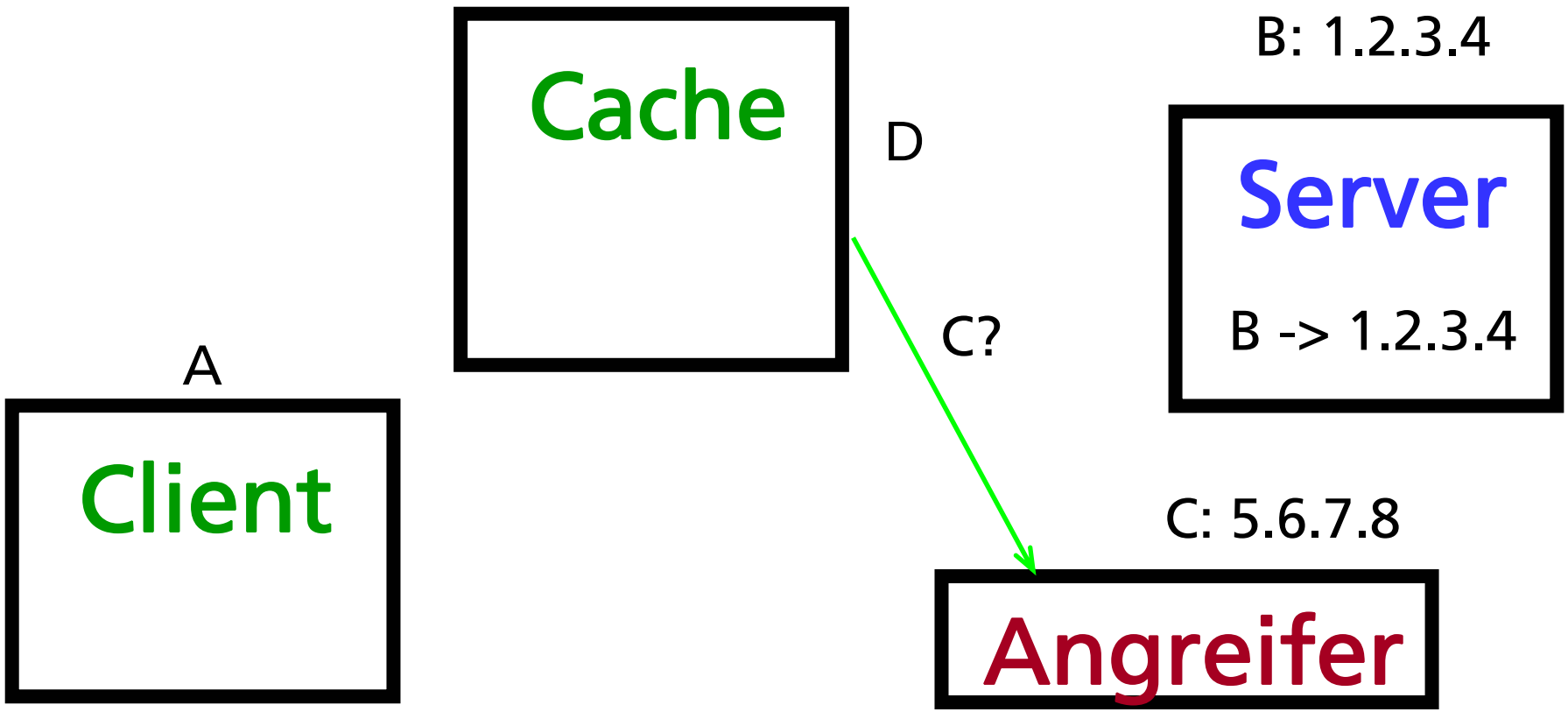
Client

Fraunhofer IGD Institut Graphische Datenverarbeitung

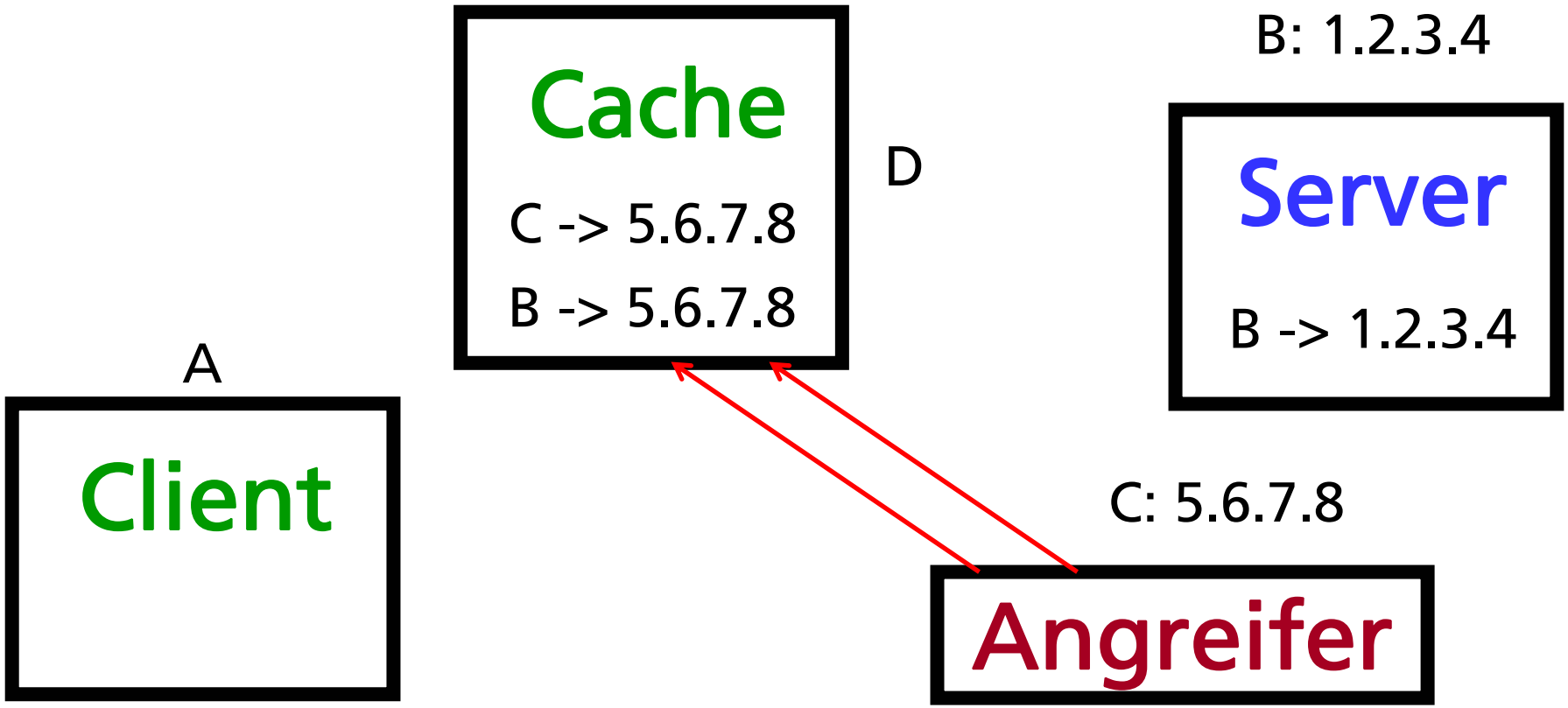
DNS Cache Poisoning: C erreicht, daß D nach ihm fragt



DNS Cache Poisoning: C erreicht, daß D nach ihm fragt



Angreifer gibt Antwort - und mehr!



Fraunhofer IGD Institut Graphische Datenverarbeitung

A erhält Antwort und nutzt diese...

B: 1.2.3.4

Cache

C -> 5.6.7.8

C -> 5.6.7.8

D

Server

B -> 1.2.3.4

A

Client

C -> 5.6.7.8

C: 5.6.7.8

Angreifer

Fraunhofer IGD Institut Graphische Datenverarbeitung

Cache von D enthält nun zusätzliche Antwort...

B: 1.2.3.4

Cache

C -> 5.6.7.8

B -> 5.6.7.8

D

Server

B -> 1.2.3.4

A

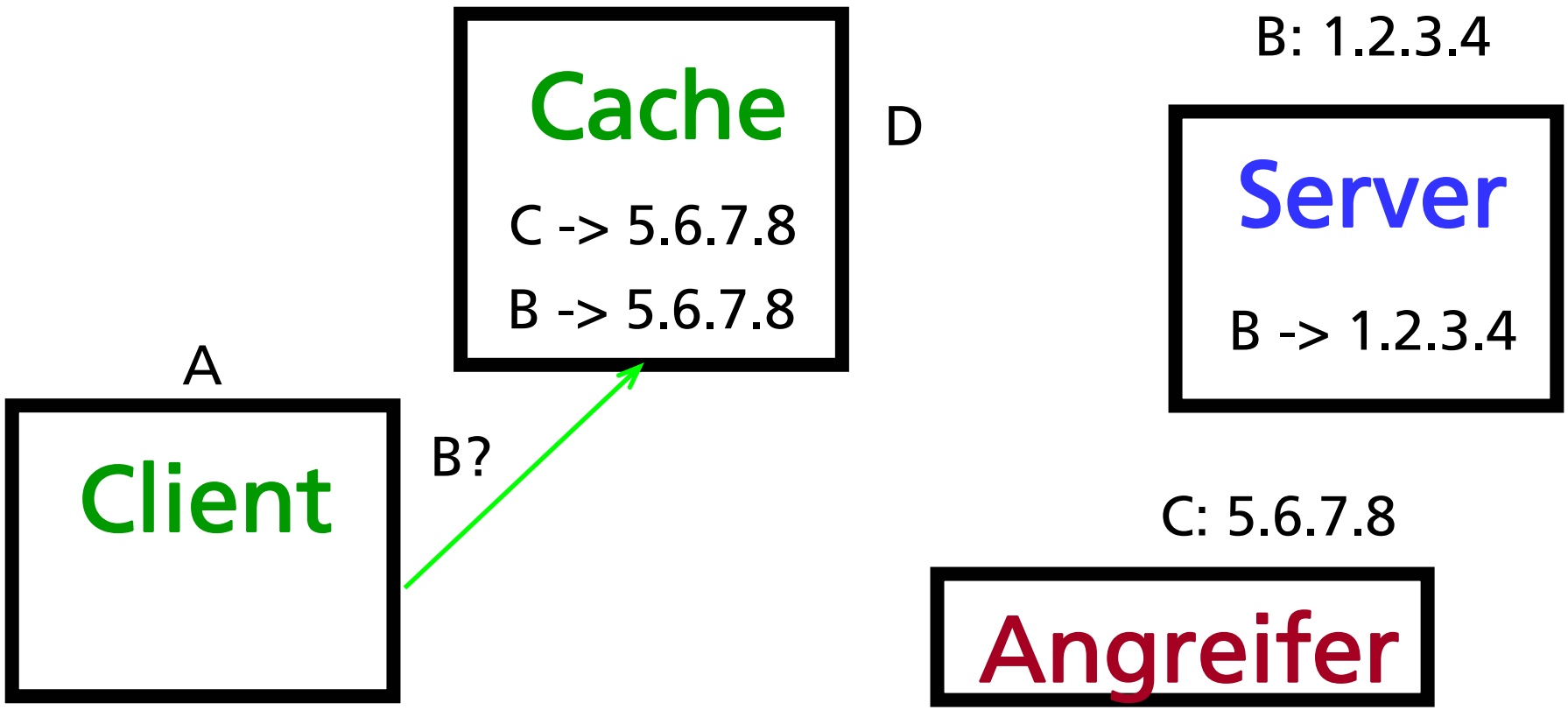
Client

C: 5.6.7.8

Angreifer

Fraunhofer IGD Institut Graphische Datenverarbeitung

A (oder anderer Host) fragt nun nach B...



D kennt die Antwort bereits...

B: 1.2.3.4

Cache

C -> 5.6.7.8

B -> 5.6.7.8

D

Server

B -> 1.2.3.4

A

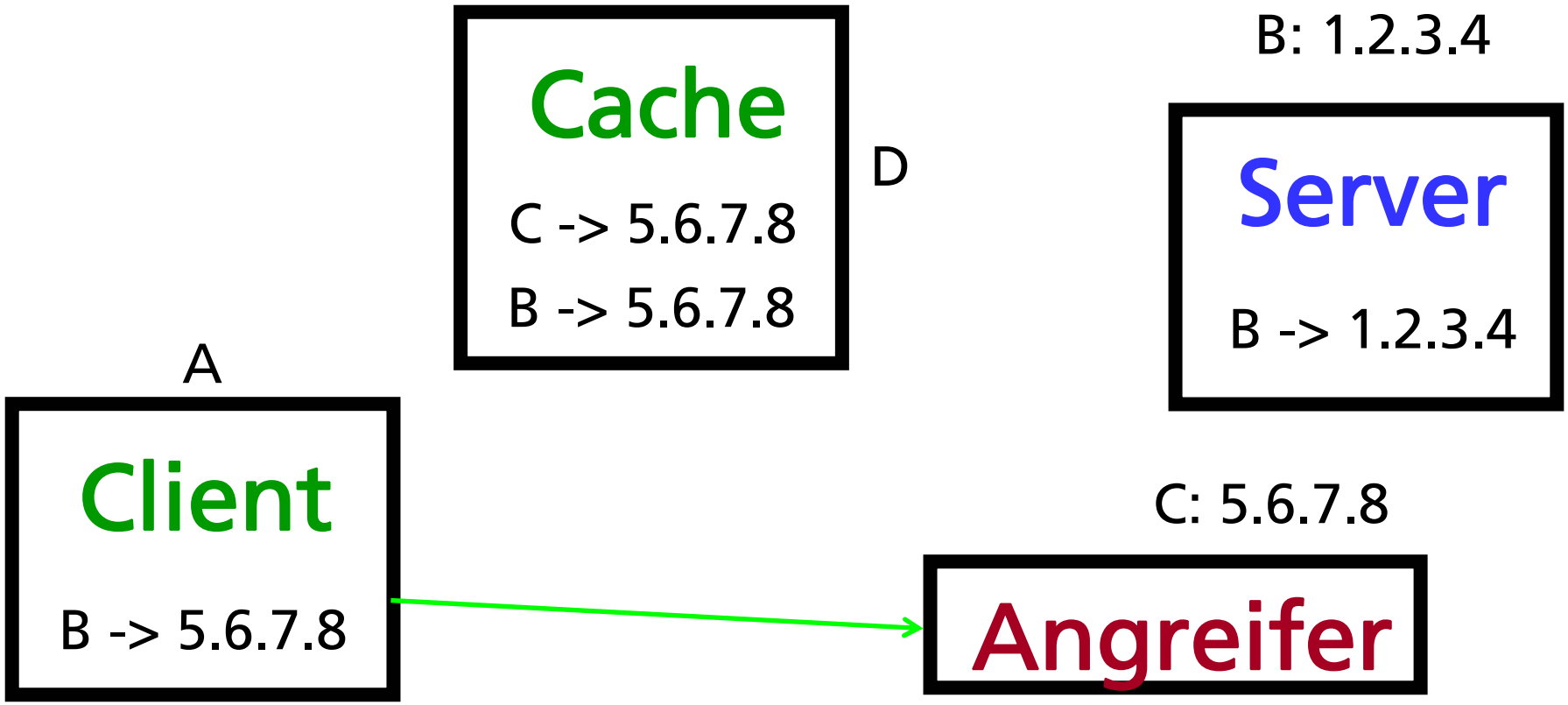
Client

B -> 5.6.7.8

C: 5.6.7.8

Angreifer

A greift auf „B“ zu - Cache Poisoning erfolgreich



Fraunhofer IGD Institut Graphische Datenverarbeitung

DNS Cache Poisoning

- Angriff funktioniert nur, wenn Caching DNS Server zusätzlich gelieferte Informationen ebenfalls akzeptiert
- Eigener Server kann „paranoid“ konfiguriert werden
- Aufgrund der verteilten Architektur von DNS mit vielen Caches genügen wenige „schwarze Schafe“, um Angriff erfolgreich sein zu lassen
- Anfrage auf Zone Authority zu beschränken ist nicht praktikabel
- DNS ist ohnehin nicht authentisiert (DNSSec...)

Firewalling-Strategien

- Eigenen DNS-Server als „Split-Brain-Konfiguration“ betreiben
- „Erste Antwort gewinnt“
  - Auflösung erfolgt über UDP (Port 53)
  - Erfordert virtuelle Verbindungen, zusätzliche Antworten anderer Server oder Angreifer werden zurückgewiesen
- Weitere Strategien - auf Kosten der Effizienz, Zuverlässigkeit:
  - Reformatierung der Antworten durch DNS-Proxies
    - Zusätzliche Informationsfelder in Antworten ignorieren
    - Akzeptanz nur von Authoritative Servers

Firewalling-Strategien II

- Neben der Adreßauflösung existiert für die Übertragung z.B. zwischen Authoritative und Slave Servers ein weiteres Protokoll
  - Ebenfalls Port 53, aber TCP
  - Wird für „Zone Transfers“ benötigt
  - Es existiert kein Grund, außer (im Prinzip bekannten) anderen DNS-Servern diesen Dienst nach außen anzubieten
  - Zone Files enthalten häufig für Angreifer interessante Informationen
- Probleme auch im Bereich DoS/DDoS-Angriffe - angefangen bei Root-Servern

Simple Mail Transfer Protocol (SMTP)

- Erstes Mail-System verband lokale 1970 Mailboxes von TENEX-Systemen (auf DEC PDP-10) via Dateitransfer
  - 1971 im ARPANET im Betrieb „SNDMSG“, verwendet „CPYNET“
  - Die „Strudel-Notation“ stammt von SNDMSG (Ray Tomlinson)
  - Erste Ansätze zur Übertragung via FTP mit besonderem MAIL-Kommando und Nutzerkonten (NETMAIL)
  - Heute noch gültiger Standard: RFC 822 (August 1982)
    - Basierend auf MTP (1980)
    - Notwendige Änderungen/Verbesserungen, da TCP anders als NTP Vollduplex-fähig ist

MTP-Sitzung

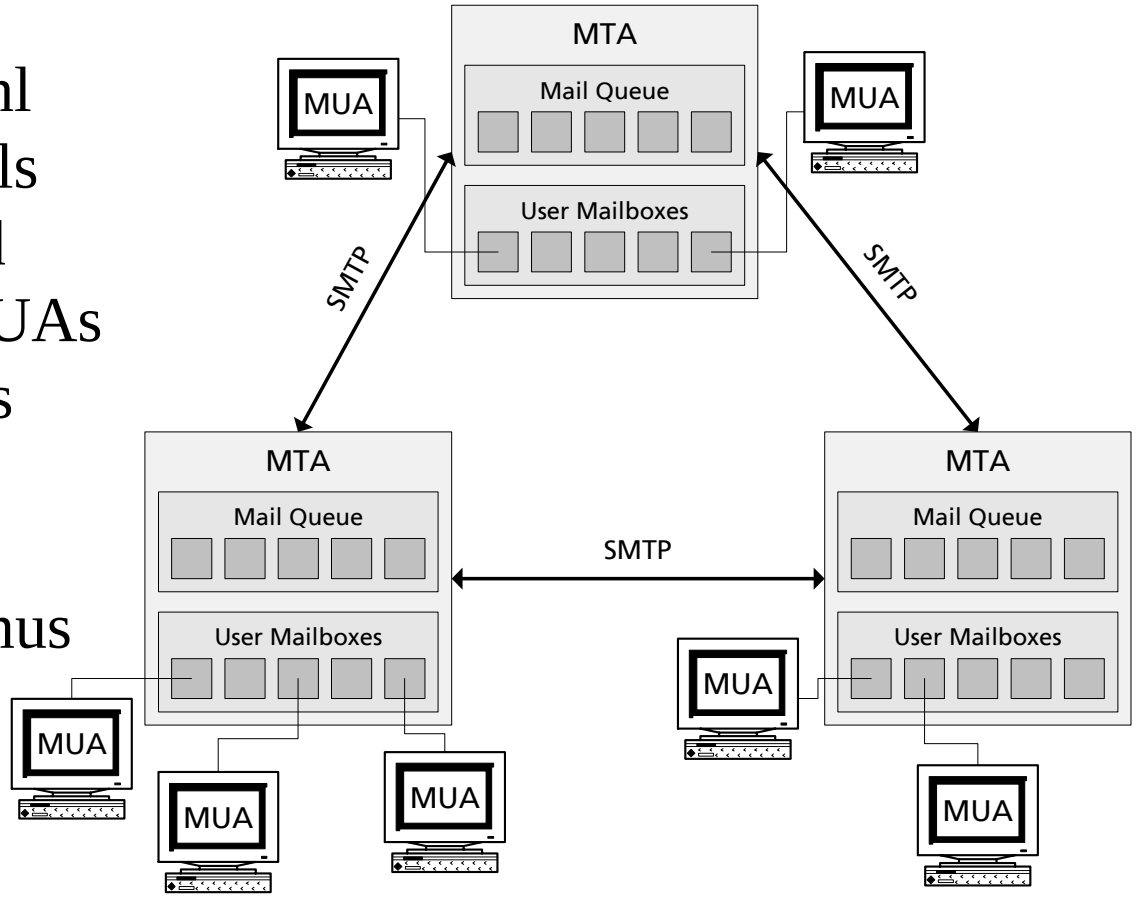
- **`MAIL FROM:<Bar@A.NET> TO:<Foo@B.ORG> <CRLF>`**
  `354 Start mail input; end with <CRLF>.<CRLF>`
  `badzoing`
  `foobar`
  `.`
  `250 Mail sent`
- Die MTP-Kommandos entsprechen denen in SMTP
- Nur wenige Anwendungsprotokolle sind nach über 20 Jahren noch unverändert im Einsatz
  - Einfachheit und Konzentration auf Wesentliches

Komponenten in SMTP

- Mail Transport Agents (MTA)
  - Nehmen eingehende SMTP-Verbindungen an und verteilen empfangene Mails weiter an MUAs
  - Nehmen Verbindungen von MUAs für ausgehende Mails an
- Mail User Agents (MUA)
  - Präsentieren dem Nutzer eingegangene Mails
  - Leiten ausgehende Mails an MTAs weiter

Komponenten in SMTP

SMTP wird sowohl zwischen MTAs als auch zum Versand ausgehend von MUAs in Richtung MTAs verwendet.
SMTP beinhaltet keinen Mechanismus zur Auslieferung



Weitere Eigenschaften von SMTP

- SMTP kann in einer Sitzung mehrere Nachrichten für unterschiedliche Anwender empfangen
- Es wird von 7 Bit ASCII ausgegangen
- Es wird nur ein TCP-Port (25) für Empfang benötigt
- Übertragung weiterer Daten durch geschichtete Protokolle: MIME-Erweiterungen
- Kompatible Erweiterungen zur Leistungssteigerung, erweiterte Funktionen: ESMTP (1995)

ESMTP-Sitzung

- 220 concord.ftc.igd.fhg.de ESMTP Sendmail 8.10.2+Sun/8.9.3; Wed, 15 Aug 2001 09:28:51 +0200 (MET DST)
  EHLO cetus.igd.fhg.de
  250-concord.ftc.igd.fhg.de Hello cetus.igd.fhg.de [146.140.8.115], pleased to meet you
  250-EXPN
  250-VERB
  250-8BITMIME
  250-SIZE
  250-DSN
  250-ONEX
  250-ETRN
  250-XUSR
  250 HELP
  MAIL FROM:<wolt@igd.fhg.de> SIZE=51
  250 <wolt@igd.fhg.de>... Sender ok
  RCPT TO:<wolt@ftc.igd.fhg.de>
  250 <wolt@ftc.igd.fhg.de>... Recipient ok
  DATA
  354 Enter mail, end with "." on a line by itself
  .
  250 JAA01922 Message accepted for delivery
  QUIT
  221 concord.ftc.igd.fhg.de closing connection

Auslieferung von Mails

- Traditionell erfolgt Auslieferung unter Unix über Mailbox-Dateien
  - erfordert root-Zugriff auf Dateisysteme
  - root-Rechte ebenfalls für Zugriff auf privilegierten Port
- In heterogenen Systemen kommen meist separate Protokolle zur Auslieferung zum Einsatz:
  - POP3 (RFC 1939)
  - IMAP (RFC 1730)
  - WWW-basierte, anwendungsspezifische Protokolle (Notes...)

Firewalling-Strategien

- SMTP ist als Store-and-Forward-Protokoll für Firewalling (Proxies) gut geeignet
  - Komplexe MTAs sind häufig Angriffspunkte: Fehlkonfigurationen (Relays...), Verwundbarkeiten
- Schutzmechanismen an Firewalls allein reichen nicht
  - Cisco PIX Mailguard-Feature
    - Wird ausgiebig dokumentiert... aber in Verbindung mit bestimmten anderen Erlaubnisregeln bleibt es völlig wirkungslos (korrigiert im September 2000)

Telnet

- Erster Entwurf bereits 1971 (SRI)
- Persistente TCP-Verbindung (Port 23)
- Definiert einen Network Virtual Terminal (NVT)
  - Prinzipiell bidirektional, wird aber als gepufferter Halbduplex-Kanal genutzt
  - Bei Verbindungsaufbau findet Optionsverhandlung statt
    - WILL und WON‘T geben von Sender angebotene und verweigerte Optionen wieder
    - DO und DON‘T teilen gefordertes Verhalten dem Gegenüber mit

Telnet-Optionen

- Es existieren derzeit 52 von der IANA verwaltete Optionen. Bsp:
  - Horizontale Tabstops (RFC 653)
  - Binäre Datenübertragung (RFC 856)
  - X Display Location (RFC 1096)
  - IBM TN3270-Unterstützung (RFC 1647)
  - Zeichensatz (RFC 2066)
  - Kermit-Übertragung (RFC 2840)
  - Authentisierung (RFC 2941)
  - Verschlüsselung (RFC 2946)

Verwundbarkeiten

- Telnet ist ein offenes Protokoll, das nach über 30 Jahren immer noch erweitert wird
- Noch im Juli 2001 wurde ein Buffer Overrun in vielen BSD-basierten Implementierungen der Option Negotiation gefunden
- Verwundbar unter anderem: Solaris, IRIX, FreeBSD, Linux,...
  - Selbst wenn ein Protokoll „trivial“ und wohlbekannt ist, bleiben derartige Überraschungen nicht aus
- Häufigstes Problem jedoch bleibt Verwendung identischer Accountnamen/Paßwörter für interne/externe Hosts und die Übertragung im Klartext

Firewalling-Strategie

- Sofern Alternativen (z.B. SSH oder Sicherung über VPN) bestehen sollte Telnet deaktiviert werden.
  - Gilt sowohl für eingehenden als auch ausgehenden Verkehr
- Telnet ist in der Lage fast beliebige Datenströme zu übertragen
  - Gefahr der Nutzung durch Trojaner
- Sofern Proxies zum Einsatz kommen kann Verwendung von Verschlüsselung nach RFC 2946 und Authentisierung nach RFC 2941 überprüft werden
  - Beseitigt auch Problem des Session Hijacking

File Transfer Protocol

- Erste Versuche ebenfalls 1971
- Nach verschiedenen Änderungen und Korrekturen 1985 mit RFC959 derzeit gültiger Standard
  - Mußte z.B. verschiedene Dateisysteme, Verzeichnisstrukturen und die Länge eines Bytes (TOPS-20: 6 Bit) handhaben
  - Bis zum Aufkommen von HTTP das Protokoll mit dem höchsten Datenvolumen
- Erfordert stets 2 TCP-Verbindungen
  - Kontrollverbindung (Port 21)
  - Datenverbindung (beliebig, meist Port 20)

File Transfer Protocol (2)

- Kontrollverbindung folgt Telnet-Protokoll und muß während Datenübertragung bestehen bleiben
  - geschlossene Kontrollverbindung kann als Aufforderung zum Abbruch der Übertragung verstanden werden
- Verbindung kann auch abgebrochen werden, wenn
  - Alle Daten übertragen wurden und Protokoll Abbruch als EOF akzeptiert
  - Server erhält ABORT von Client
  - Port-Spezifikation ändert sich nach Kommando von Client
  - Nicht behebbarer Fehler auftritt

Kommandos der FTP-Kontrollverbindung im FTP

- USER: Nutzername (insb. `ftp` oder `anonymous`)
- PASS: Paßwort (nur nach USER)
- ACCT: Rollenangabe für Operationen
- CWD: Veränderung des aktuellen Verzeichnisses, systemspezifisch
- CDUP: Wechel in übergeordnetes Verzeichnis
- SMNT: Mount von anderem Dateisystem auf Server
- REIN: Vervollständige Übertragung, Rücksetzung vor USER
- QUIT: Vervollständige Übertragung, schließe Kontrollkanal

Weitere FTP-Kommandos (1)

- PORT: Angabe von Host und Port für Datenverbindung
- PASV: Passive Mode
- TYPE: Datentyp (z.B. ASCII, Bytestrom, EBCDIC)
- STRU: Daten-Struktur (Bytestrom, Records, Seiten)
- MODE: Strom-, Block-, oder komprimierte Übertragung
- RETR: Kopie von Datei in CWD auf Client
- STOR: Kopie von Datei in Client-CWD in Server-CWD
- STOU: Wie STOR, Server vergibt eindeutigen Namen
- APPE: Anfügen an Datei auf Server

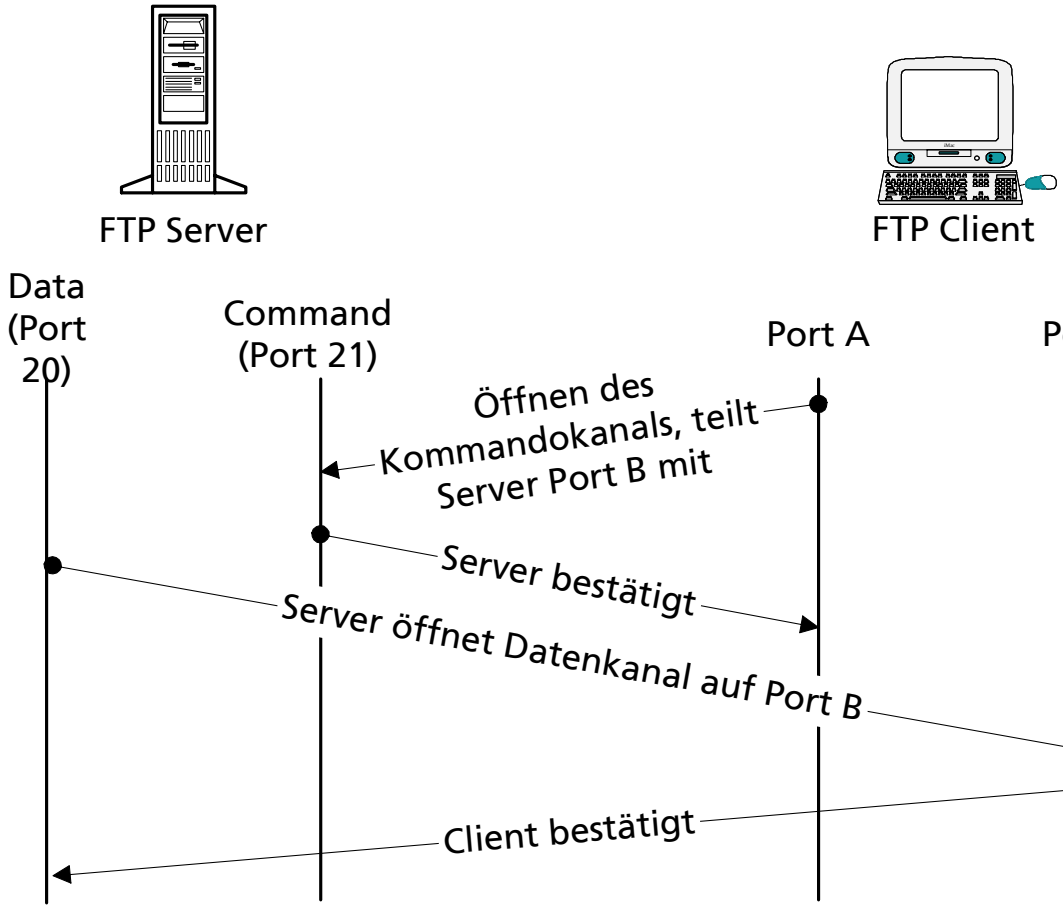
Weitere FTP-Kommandos (2)

- ALLO: Gibt im folgenden Kommando versandte #Bytes an
- REST: Wiederaufnahme von Dateitransfer (RETR)
- RNFR: Umbenennen Angabe Quelldatei (zusammen mit RNTO)
- RNTO: Umbenennen Angabe Zieldatei (zusammen mit RNFR)
- ABOR: Abbruch Datenverbindung (OOB, Telnet SYNCH)
- DELE: Lösche Datei auf Server
- RMD: Lösche Verzeichnis auf Server
- MKD: Lege Verzeichnis auf Server an
- PWD: Aktuelles Verzeichnis auf Server

Weitere FTP-Kommandos (3)

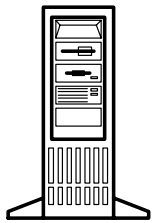
- LIST: Attribute von Datei oder Liste von Dateien in Argument
- NLST: Nur Liste von Dateien (keine Attribute bei Datei-Arg.)
- SITE: Setzen von site-/systemspezifische Optionen
- SYST: Vom Server verwendetes Betriebssystem
- STAT: Statusmeldungs-Anforderung von Server
- HELP: Systemspezifische Informationen für Client
- NOOP: Keine Operation (Keepalive)
- AUTH: Authentisierungs-Erweiterung (RFC 2228)
- ADAT: Authentisierungs-Erweiterung (RFC 2228)

FTP Active Mode



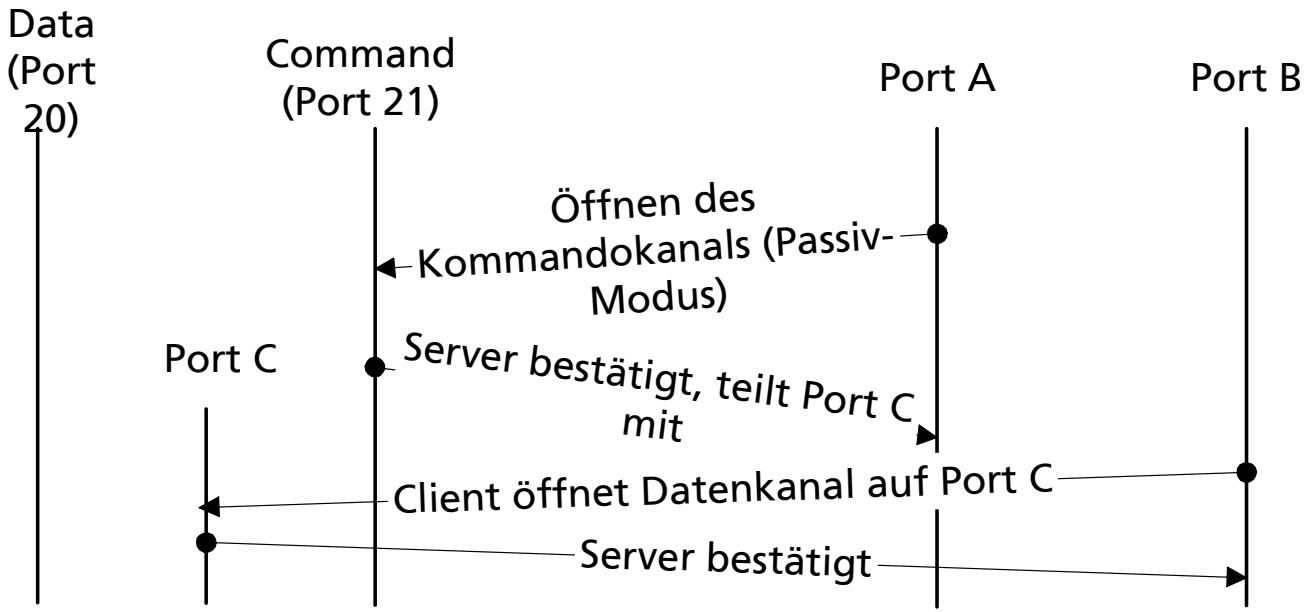
Fraunhofer IGD Institut Graphische Datenverarbeitung

FTP Passive Mode



FTP Server

FTP Client



Eine FTP-Sitzung


```
$ ftp simtel20.army.mil
Connected to simtel20.army.mil.
220 WSMR-SIMTEL20.ARMY.MIL FTP Server Process 5Z(50)-7 at Sun 14-Feb-
88
Name (simtel20.army.mil:davy): anonymous
Password (simtel20.army.mil:anonymous):
331 User name ok. Password, please.
230 User ANONYMOUS logged in at Sun 14-Feb-88 15:48-MST, job 13.
ftp> cd "pd6:<unix-c>“
250 Connected to PD6:<UNIX-C>.
ftp>dir
200 Port 6.246 at host 1.2.3.4 accepted.
150 List started.
PD6: 000-INTRO-UNIX-SW.TXT.1
(...)
226 Transfer completed.
669 bytes received in 0.08 seconds (8.2 Kbytes/s)
ftp>
```


Firewalling-Strategie

- Keine Verwendung von Active Mode FTP
  - Potential für Umgehung a la Firewall-1
- Anonymous FTP nur auf vollständig separatem System
  - Anonyme Uploads verbieten
  - `chroot(1M)` oder `jail(8)`
- Sonstige Transfers entweder mit vollständig getrennten Konten (Wiederverwendung von Paßworten)
  - Transitive Verwundbarkeiten
  - Einschleppen von Trojanern etc. von kompromittierten Hosts

Hypertext Transfer-Protokoll (1)

- Am CERN 1991/92 entwickelt
- Einfaches Request/Reply-Protokoll auf TCP-Basis (Port 80)
- Aktueller Stand: HTTP 1.1 (RFC 2616, Juni 1999)
  - Abwärtskompatibilität ist zwingend
- Anfragen bestehen aus
  - Request Method
  - Uniform Resource Identifier
  - Protokollversion
  - Request Modifiers, Client-Information, Nutzdaten

Hypertext Transfer-Protokoll (2)

- HTTP stellt Byte-orientierten „8 bit clean“ Kanal bereit
- Antworten sind Statuscodes (dreistellige Dezimalzahl) + Daten
- Führende Dezimalstelle:
- 1: Zur Information; Anfrage wurde erhalten und wird bearbeitet
- 2: Erfolgsmeldungen
- 3: Umleitung; Anfrage bedarf weiterer Aktionen
- 4: Fehler auf Client-Seite (Syntax-Fehler), kann nicht bearbeitet werden
- 5: Fehler auf Server-Seite; Anfrage ist gültig

Hypertext Transfer-Protokoll (3)

- Request-Methoden:
- GET: Eine Zeile, enthält URI, kann aber auch weiteren Code enthalten
- HEAD: Nur Header, sonst wie GET
- POST: Übergebene Entity wird unterhalb URI angeordnet (z.B. Rückgabe von Formularen, aber auch Uploads)
- PUT: Exakte Übertragung einer Entity an angegebene URI
- DELETE: Löschen einer Entity mit angegebener URI
- TRACE: Exakte Spiegelung der vom Client versandten Daten

HTTP-Beispielsitzung

- telnet draco.igd.fhg.de 7071

```
Trying 146.140.8.6...
Connected to draco.igd.fhg.de.
Escape character is '^]'.
GET / HTTP/1.1

HTTP/1.1 200 Sending cached data
MIME-version: 1.0
Server: OSU/3.8alpha4;UCX
Content-type: text/html
Content-transfer-encoding: 8bit
Content-length: 1021
Last-Modified: Tue, 17 Apr 2001 14:42:32 GMT
Date: Wed, 24 Oct 2001 12:58:40 GMT

<HTML>
<HEAD>
...
</HTML>
Connection closed by foreign host.
```

Erweiterungen zu HTTP

- Ursprünglich reines Request/Reply-Protokoll
  - Server mußte nach Versand der Antwort Kanal schließen
  - Ineffizient, da TCP-Handshakes, FIN-Wait, etc.
- Keepalive
  - Apache-spezifischer Mechanismus (da bis HTTP 1.0 kein Mechanismus vorhanden): Verbindung zu Server bleibt für weitere Anfragen bestehen. Implizit in HTTP 1.1 enthalten
- Pipelining
  - Paralleles Absetzen mehrerer Anfragen an Server; reduziert z.B. bei mehreren Links (Bilder etc.) die Latenzzeiten

Firewalling-Strategie

- Die Option, HTTP zu blockieren existiert de facto nicht
- Wichtig ist die Verwendung eines guten Proxies
  - Verifikation der Syntax von HTTP
  - Gefahren durch Werkzeuge wie z.B. HTTPtunnel
    - Keepalive ermöglicht sonst unkontrollierten bidirektionalen Kanal
  - Proxy kann gestaffelt mit Cache integriert werden
  - Caching ist zu komplex, um eine Kombination mit Firewalling/ALGW ratsam zu machen
    - Probleme bei Microsoft ISA Server

XML-RPC

- Entwickelt 1998
  - außerhalb regulärer Standardisierungsgremien entstanden
  - wird nicht mehr weiterentwickelt
- Ziel: Abbildung von RPC-Mechanismen auf XML via HTTP
- MIME `Content-Type` ist `text/xml`
- Methodenaufruf wird mit `<methodCall>`, Antwort mit `<methodResponse>` XML-Tags abgebildet
- Definition erfolgte ohne XML DTD oder Schema
  - Keine Indikation, daß ausführbarer Code vorliegt

XML-RPC Beispielaufruf


- ```
  POST /RPC2 HTTP/1.0
  User-Agent: Frontier/5.1.2 (WinNT)
  Host: host.somewhere.net
  Content-Type: text/xml
  Content-length: 181
  ```

- ```
  <?xml version="1.0"?>
  <methodCall>
   <methodName>examples.getStateName</methodName>
    <params>
     <param>
      <value><i4>41</i4></value>
     </param>
    </params>
  </methodCall>
  ```


XML-RPC Beispielantwort


- ```
  HTTP/1.1 200 OK
  Connection: close
  Content-Length: 158
  Content-Type: text/xml
  Date: Fri, 17 Jul 1998 19:55:08 GMT
  Server: UserLand Frontier/5.1.2-WinNT
  ```
- ```
  <?xml version="1.0"?>
   <methodResponse>
    <params>
     <param>
      <value><string>South Dakota</string></value>
     </param>
    </params>
  </methodResponse>
  ```

Fraunhofer IGD Institut Graphische Datenverarbeitung

Simple Object Access Protocol

- Erster Draft im Anschluß an XML-RPC als Internet Draft 1999
  - Ziele analog zu XML-RPC
  - Weiterentwicklung wurde in der XML Protocol Working Group des W3C angesiedelt
  - Aktuelle Version ist SOAP 1.2 (W3C Recommendation, Juni 2003)
  - Spezifikation gegenüber XML-RPC deutlich strukturierter
  - Nachrichten werden in Form von XML-Namensräumen definiert; es darf keine DTD/Schema angegeben werden.

Bestandteile der SOAP-Spezifikation

- Codierungsregeln für die Serialisierung anwendungsspezifischer Datentypen
- RPC-Repräsentation für die Abbildung von RPC-Aufrufen / Antworten in Transportprotokoll (bisher nur HTTP)
- Struktur eines „SOAP Envelope“, enthält Inhalt, Ziel einer Nachricht
  - Ermöglicht Umleitung von Aufrufen über Relaisstationen
- Bindung an bestehendes Transportprotokoll

SOAP-Beispielaufruf

- POST /StockQuote HTTP/1.1
  Host: www.stockquoteserver.com
  Content-Type: text/xml; charset="utf-8“
  Content-Length: 477
  SOAPAction: "http://example.org/2001/06/quotes“
- <env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
  <env:Body>
  <m:GetLastTradePriceDetailed
  env:encodingStyle="http://www.w3.org/2001/06/soap-encoding“
  xmlns:m="http://example.org/2001/06/quotes">
  <Symbol>DEF</Symbol>
  <Company>DEF Corp</Company>
  <Price>34.1</Price>
  </m:GetLastTradePriceDetailed>
  </env:Body>
  </env:Envelope>

SOAP-Beispielantwort

- HTTP/1.1 200 OK
  Content-Type: text/xml; charset="utf-8“
  Content-Length: nnnn
- <env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
  <env:Body>
  <m:GetLastTradePriceResponse
  env:encodingStyle="http://www.w3.org/2001/06/soap-encoding“
  xmlns:m="http://example.org/2001/06/quotes" >
  <PriceAndVolume>
  <LastTradePrice>34.5</LastTradePrice>
  <DayVolume>10000</DayVolume>
  </PriceAndVolume>
  </m:GetLastTradePriceResponse>
  </env:Body>
  </env:Envelope>

Firewalling-Strategien

- Die Verwendung von SOAP auf Servern ist nicht gefährlicher als es CGI,NSAPI,ISAPI etc. auch waren/sind
- Problematisch ist die Weiterleitung von Anfragen an nicht als exponiert identifizierte Server (Relais-Funktion)
- Durch den Transport via HTTP und die mangelnde Identifikationsmöglichkeiten (der Content-Type text/xml ist de facto nicht zu blockieren) kaum mittels Firewalling in den Griff zu bekommen
- Diese Umgehung von Firewall-Mechanismen wird als **„firewall compatibility“** beworben.

Network Time Protocol

- Korrekte, konsistente Zeit ist zwingend notwendig (Revisionsdaten, Authentisierungsmechanismen wie Kerberos)
- NTP (1985 vorgestellt) erlaubt die Synchronisation mit externen Zeitquellen via UDP
  - Aktuelle Version 3 (RFC 1305) wurde 1992 vorgestellt
- NTP ist zwar recht robust, aber aufgrund der Nutzung von UDP leicht Angriffen ausgesetzt; Verschlüsselung, Authentisierung sind zwar möglich aber häufig nicht aktiviert
  - Zeitverteilung nur über internes Verwaltungsnetz oder direkt über externe Zeitgeber wie DCF77, GPS

Real Audio

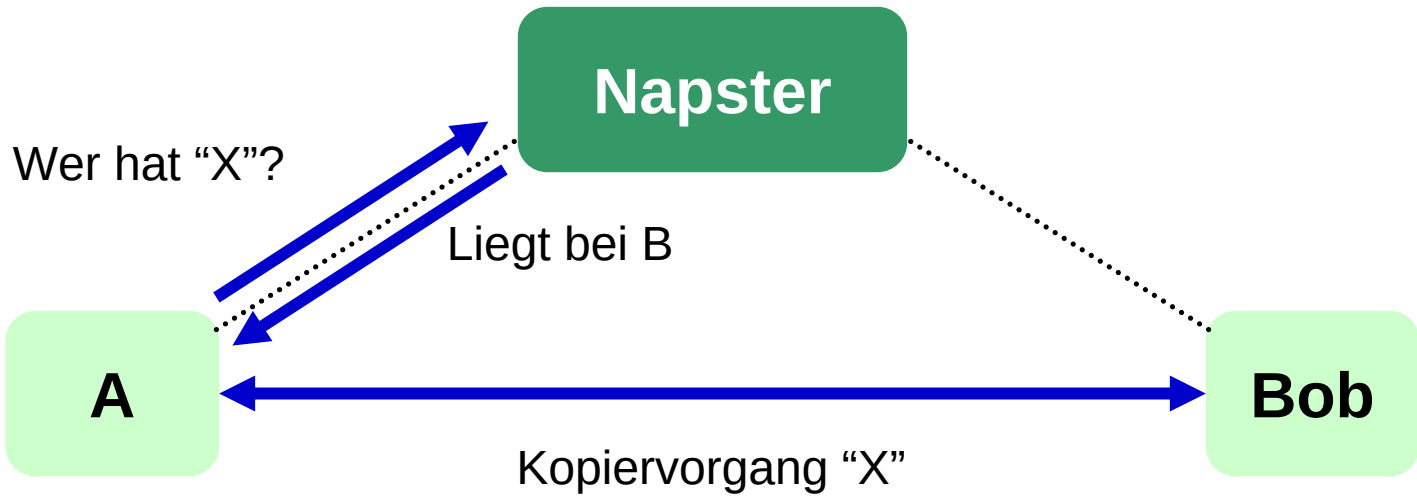
- Protokolle für Audio, Video, Präsentationsmedien
- Teile der Protokolle wurden veröffentlicht, andere sind proprietär
- Nach vermehrter Blockierung des Ur-Protokolls wurde ein flexibles Protokoll entwickelt, daß mehrere Transportmedien zulässt (TCP, UDP, UDP mit Wiederanforderung)
- Real Networks veröffentlichte zwar ein Proxying-Protokoll, meist wird jedoch einfach über Port 80 getunnelt
  - Proxies bzw. semantische Analyse können dies jedoch identifizieren

Peer-to-Peer Protokolle

- Verteiltes System aus „Peer“ (gleichberechtigten) Knoten
- Jeder Knoten ist gleichzeitig Client und Server
  - Abgrenzung ist oft willkürlich, da viele P2P-Systeme einige Teil-Dienste auch zentralisiert abwickeln
  - Nicht besonders neu: Bis zum Auftauchen von Windows-Systemen waren meist alle Hosts im ARPANET/Internet mit ähnlichen Diensten nach außen hin erreichbar
- P2P-Systeme erlangen wieder Popularität während der Technologie-Spekulationsphase (ca. 1999)
  - Napster (Shawn Fanning, 1999): Primär zum illegalen Austausch von Musikdaten
  - Januar 2000: Mehr als 1 Million Nutzer
  - November 2000: Mehr als 23 Mio. Nutzer
  - Februar 2001: Mehr als 50 Mio. Nutzer

Napster-Architektur

- Zentraler Server, der den eigentlichen Datenverkehr zwischen Peer-Knoten steuert
- Für jede Anfrage nach einer Datei wird vom Server eine Liste derzeit mit Napster verbundener Nutzer erstellt, deren Dateilisten diese enthalten



Fraunhofer IGD Institut Graphische Datenverarbeitung

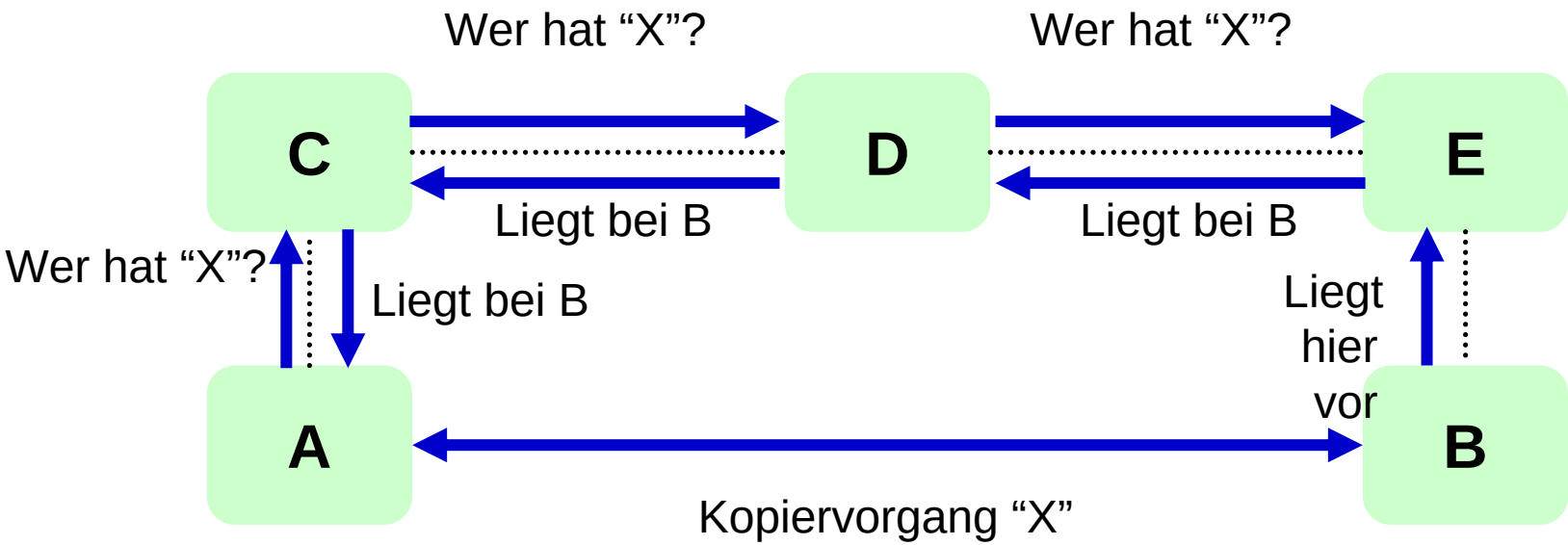
Protokolldaten (Ur-Version 2000)

- Einfaches Binär-Protokoll:
  - Paketgröße (16-Bit-Wert)
  - Paketinformation (2 Bytes, 1. Byte gibt Kommando oder Rückgabewert an)
  - Nutzdaten
- Übertragung zwischen Peer-Nodes kann sowohl im Push- als auch im Pull-Betrieb erfolgen
- Mittlerweile mehrfach überarbeitet, z.B. nach OpenNap-Spezifikation
- Blockieren dieser Dienste auf Firewall-Ebene war trivial
  - Sperren des Zugangs zu zentralen Napster-Servern
  - Spezifisches Protokoll läßt sich ebenfalls leicht filtern

Architektur-Varianten

- Napster ist Beispiel für eine Broker-Architektur
  - Single Point of Failure im Netz
  - Begrenzung der Verfügbarkeit und Bandbreite durch zentralen Server
  - Ähnliche Architektur z.B. für SETI@Home, Protein Folding, etc.
- Eliminierung des zentralen Systems
  - Ersetzen durch verteilten Index der verfügbaren Daten
  - Resultat: Erhöhte Verfügbarkeit und Robustheit, aber auch Verlust einer konsistenten globalen Sicht auf das Netz
  - Beispiele: Freenet, Gnutella, KaZaA

Reine Peer to Peer-Architektur



Fraunhofer IGD Institut Graphische Datenverarbeitung

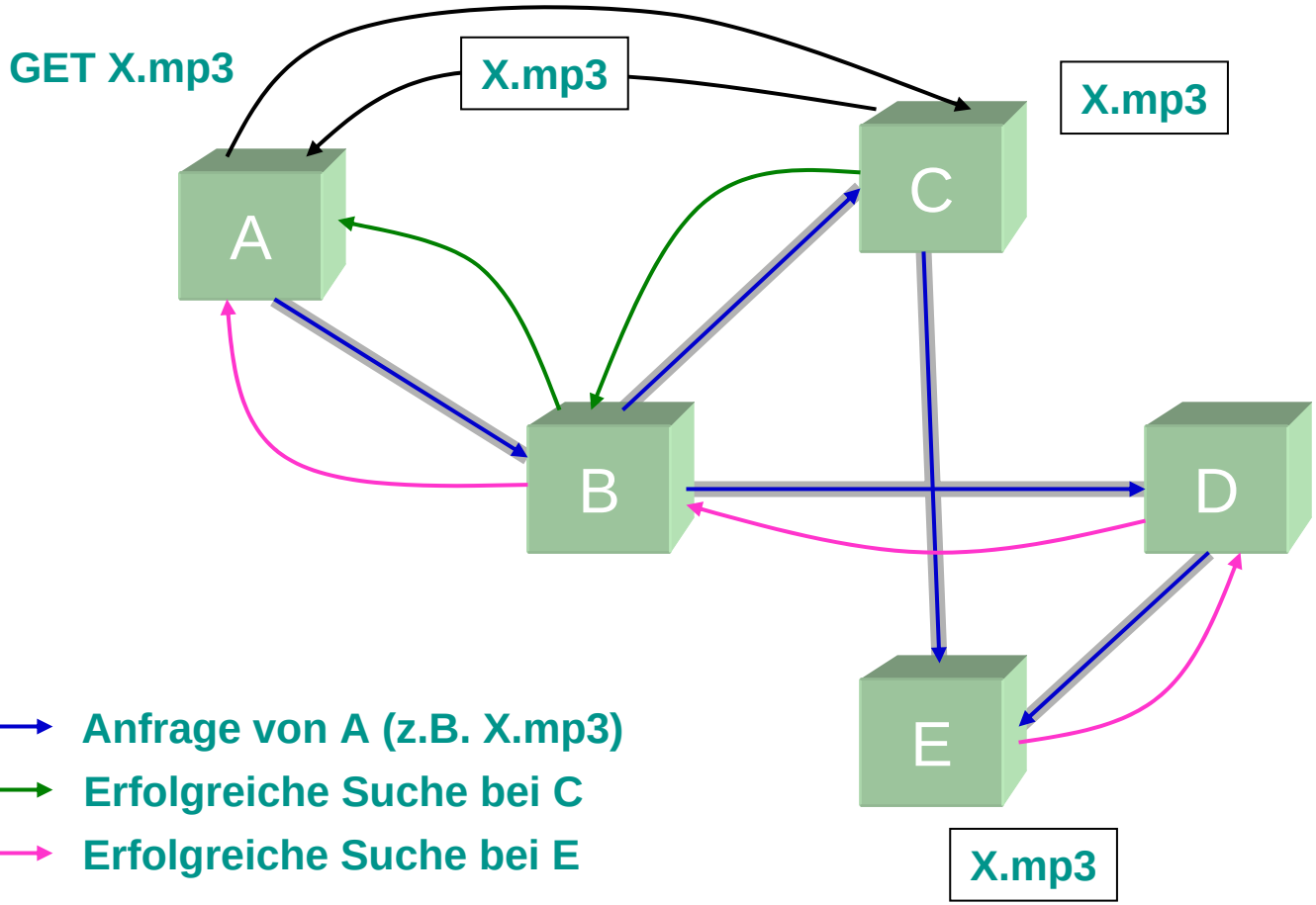
Gnutella

- Protokoll für verteilte Dateisuche
  - Von Nullsoft (Tochter von AOL) 2000 definiert und implementiert
  - AOL beendete Entwicklung wegen Potential für Urheberrechtsverletzung
  - Seither: Reverse Engineering, Implementierung als Open Source Projekte mit einer Reihe von Client/Server-Systemen („Servents“)
  - Kontrollstrukturen ähnlich einfach wie bei Napster, eigentliche Datenübertragung via HTTP (ebenfalsl Push- und Pull-Betrieb)
- Integration in Gnutella-Netz erfordert Kenntnis mindestens eines existierenden Knotens
  - Erlangung dieser Adresse ist nicht Teil des Gnutella-Protokolls
- Naives Protokoll konterkariert die (theoretischen) Vorteile
  - Strategien zur Minimierung der Skalierungsprobleme: Caching, hierarchische Netzwerkstrukturen

Gnutella-Nachrichtenstruktur

- Nachrichten bestehen aus
  - Typfeld:
    - PING, PONG
    - QUERY, QUERYHIT
    - PUSH
  - Time-to-Live-Feld
    - Analog zu TTL im Internet Protocol
  - 16-Byte-Identifikation (zufällig erzeugt), die den Absender identifiziert
- Verbreitung der Nachrichten über Broadcasts zu allen verbundenen Knoten (außer PONG- und QUERYHIT-Nachrichten, diese müssen über denselben Kanal zurückgeleitet werden, auf dem sie empfangen wurden)
  - Massive Netzwerk-Auslastung (exponentielles Wachstum in TTL und Konnektivitätsgrad)

Suche im Gnutella-Netz



Fraunhofer IGD Institut Graphische Datenverarbeitung