Energy Informatics

02 Internet Protocols

Christian Schindelhauer

Technical Faculty

Computer-Networks and Telematics

University of Freiburg

Overview

- IP Packet Forwarding

- IP Routing

- BGP

- DNS

- HTTP

Protocols of the Internet

Application	Telnet, FTP, HTTP, SMTP (E-Mail), ...
Transport	TCP (Transmission Control Protocol) UDP (User Datagram Protocol)
Network	IP (Internet Protocol) IPv4 + IPv6 + ICMP (Internet Control Message Protocol) + IGMP (Internet Group Management Protoccol)
Host-to-Network	LAN (e.g. Ethernet, W-Lan etc.)

TCP/IP Layers

- 1. Host-to-Network
  - Not specified, depends on the local networ,k e.g. Ethernet, WLAN 802.11, PPP, DSL

- 2. Routing Layer/Network Layer (IP - Internet Protocol)
  - Defined packet format and protocol
  - Routing
  - Forwarding

- 3. Transport Layer
  - TCP (Transmission Control Protocol)
    - Reliable, connection-oriented transmission
    - Fragmentation, Flow Control, Multiplexing
  - UDP (User Datagram Protocol)
    - hands packets over to IP
    - unreliable, no flow control

- 4. Application Layer
  - Services such as TELNET, FTP, SMTP, HTTP, NNTP (for DNS), …
  - Peer-to-peer networks

IPv6-Header (RFC 2460)

- Version: 6 = IPv6
- Traffic Class
  - for QoS (priority)
- Flow Label
  - QoS or real-time
- Payload Length
  - size of the rest of the IP packet
- Next Header (IPv4: protocol)
  - e..g. ICMP, IGMP, TCP, EGP, UDP, Multiplexing, ...
- Hop Limit (Time to Live)
  - maximum number of hops
- Source Address
- Destination Address
  - 128 bit IPv6 address

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Source Address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                     Destination Address                       +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Routing Tables and Packet Forwarding

- ■ **IP Routing Table**
  - contains for each destination the address of the next gateway
  - destination: host computer or sub-network
  - default gateway

- ■ **Packet Forwarding**
  - IP packet (datagram) contains start IP address and destination IP address
    - if destination = my address then hand over to higher layer
    - if destination in routing table then forward packet to corresponding gateway
    - if destination IP subnet in routing table then forward packet to corresponding gateway
    - otherwise, use the default gateway

IP Packet Forwarding

- IP -Packet (datagram) contains...

  - TTL (Time-to-Live): Hop count limit

  - Start IP Address

  - Destination IP Address

- Packet Handling

  - Reduce TTL (Time to Live) by 1

  - If TTL ≠ 0 then forward packet according to routing table

  - If TTL = 0 or forwarding error (buffer full etc.):

    - delete packet

    - if packet is not an ICMP Packet then

        - send ICMP Packet with

        - start = current IP Address

        - destination = original start IP Address

Introduction to Future IP

- IP version 6 (IP v6 – around July 1994)
- Why switch?
  - rapid, exponential growth of networked computers
  - shortage (limit) of the addresses
  - new requirements towards the Internet infrastructure (streaming, real-time services like VoIP, video on demand)
- evolutionary step from IPv4
- interoperable with IPv4

Capabilities of IP

- dramatic changes of IP

  - Basic principles still appropriate today

  - Many new types of hardware

  - Scale of Internet and interconnected computers in private LAN

- Scaling

  - Size - from a few tens to a few tens of millions of computers

  - Speed - from 9,6Kbps (GSM) to 10Gbps (Ethernet)

  - Increased frame size (MTU) in hardware

Static and Dynamic Routing

- Static Routing

  - Routing table created manually

  - used in small LANs

- Dynamic Routing

  - Routing table created by Routing Algorithm

  - Centralized, e.g. Link State

    - Router knows the complete network topology

  - Decentralized, e.g. Distance Vector

    - Router knows gateways in its local neighborhood

Intra-AS Routing

- Routing Information Protocol (RIP)
  - Distance Vector Algorithmus
  - Metric = hop count
  - exchange of distance vectors (by UDP)
- Interior Gateway Routing Protocol (IGRP)
  - successor of RIP
  - different routing metrics (delay, bandwidth)
- Open Shortest Path First (OSPF)
  - Link State Routing (every router knows the topology)
  - Route calculation by Dijkstra's shortest path algorithm

Distance Vector Routing Protocol

- Distance Table data structure

  - Each node has a

    - Line for each possible destination

    - Column for any direct neighbors

- Distributed algorithm

  - each node communicates only with its neighbors

- Asynchronous operation

  - Nodes do not need to exchange information in each round

- Self-terminating

  - exchange unless no update is available
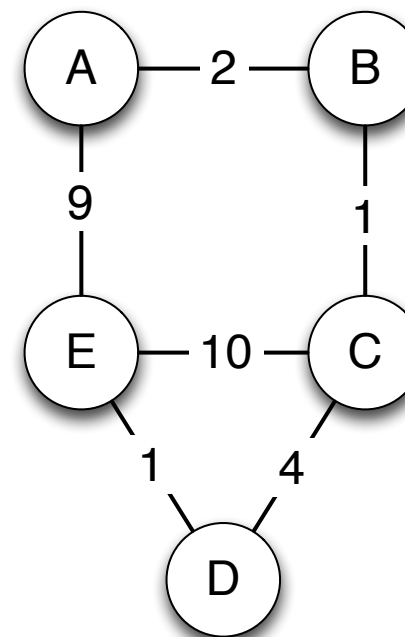
Distance Table for A

	via		Routing Table entry
from A	B	E	
to B	**2**	15	B
C	**3**	14	B
D	**7**	10	B
E	**8**	9	E

Distance Table for C

	via			Routing Table entry
from C	B	D	E	
to A	**3**	11	18	B
B	**1**	9	21	B
D	6	**4**	11	D
E	7	**5**	10	D

Distance Vector Routing Example
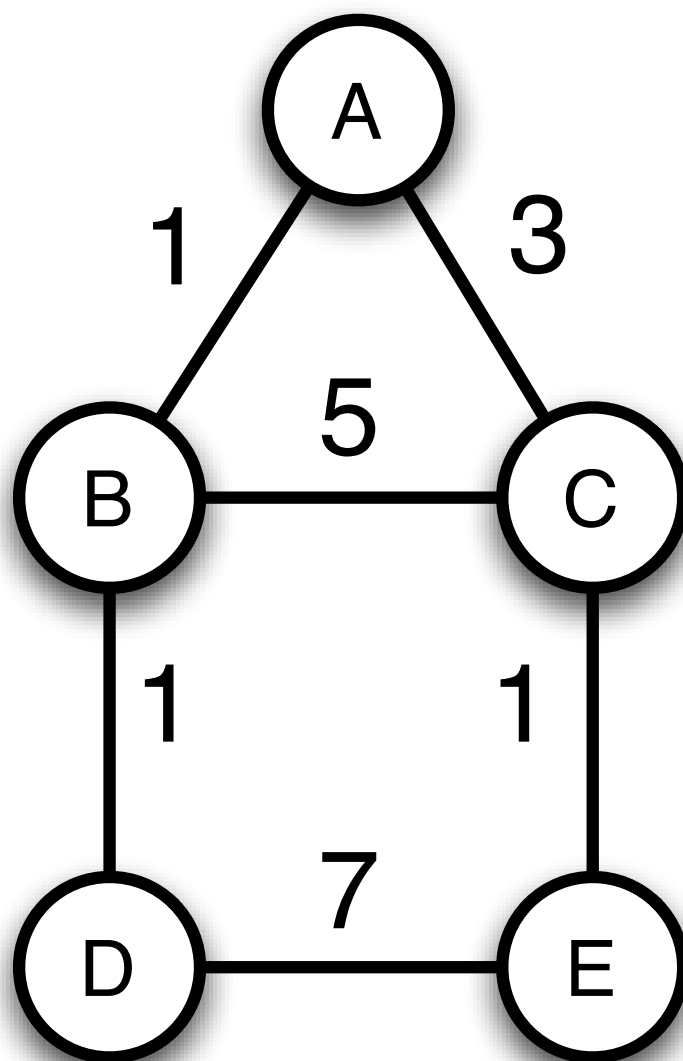
from A to	via		entry
	B	C	
B	1	8	B
C	6	3	C
D	2	9	B
E	7	4	C

Distance Vector Routing
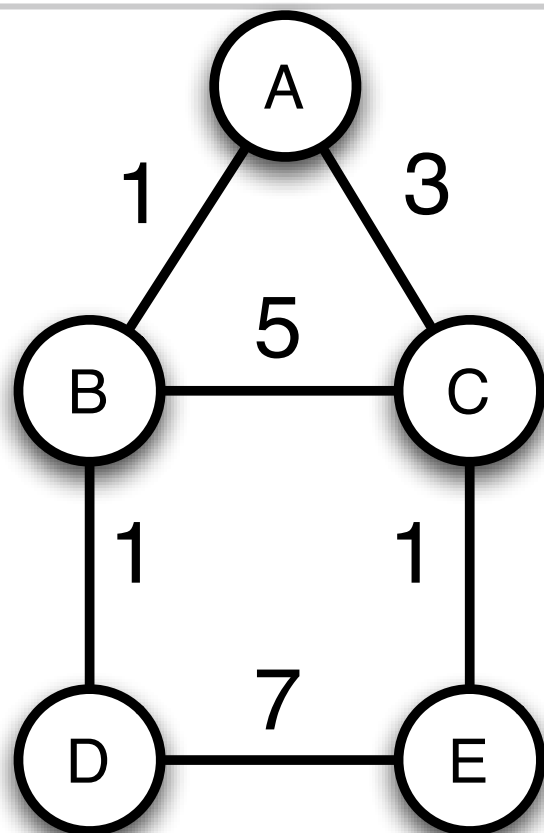
from A to	via		entry
	B	C	
B	1	-	B
C	-	3	C
D	-	-	-
E	-	-	-

from B to	via			entry
	A	C	D	
A	1	-	-	A
C	-	3	-	C
D	-	-	1	C
E	-	-	8	D

from C to	via			entry
	A	B	E	
A	3	-	-	A
B	-	5	-	B
D	-	-	8	E
E	-	-	1	E

Distance Vector Routing

from B to	via			Entry
	A	C	D	
A	1	-	-	A
C	-	5	-	C
D	-	-	1	D
E	-	-	8	D

from C to	via			Entry
	A	B	E	
A	3	-	-	A
B	-	5	-	B
D	-	-	8	E
E	-	-	1	E

from B to	via			Entry
	A	C	D	
A	1	8	-	A
C	-	5	-	C
D	-	13	1	D
E	-	6	8	C
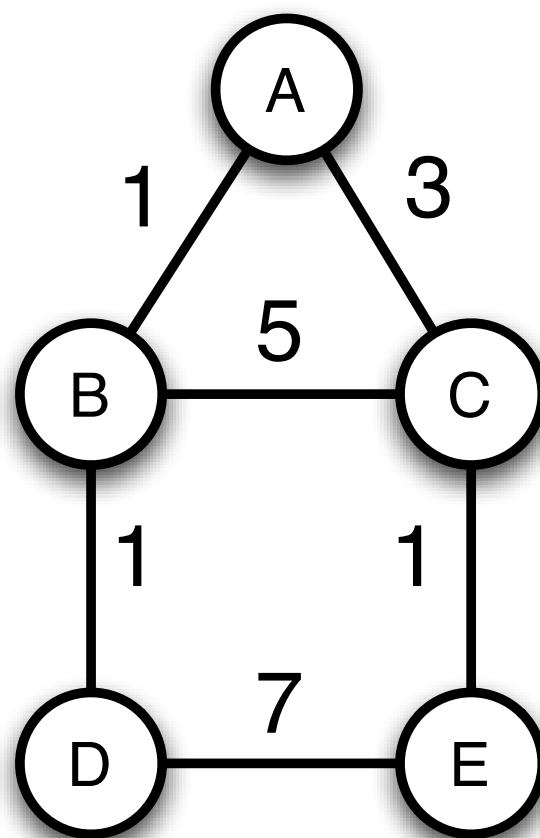
from C to	via			Entry
	A	B	E	
A	3	6	-	A
B	-	5	-	B
D	-	6	8	B
E	-	13	1	E

"Count to Infinity" - Problem

- Good news travels fast

  - A new connection is quickly at hand

- Bad news travels slowly

  - Connection fails

  - Neighbors increase their distance mutally

  - "Count to Infinity" Problem

from A	via B	Routing Table entry
to B	2	B
C	3	B

from B	via A	via C	Routing Table entry
to A	2	-	A
C	5	-	A

from A	via B	Routing Table entry
to B	2	B
C	7	B

from B	via A	via C	Routing Table entry
to A	2	-	A
C	5	-	A

from A	via B	Routing Table entry
to B	2	B
C	7	B

from B	via A	via C	Routing Table entry
to A	2	-	A
C	9	-	A

Link-State Protocol

- Link state routers
  - exchange information using Link State Packets (LSP)
  - each node uses shortest path algorithm to compute the routing table
- LSP contains
  - ID of the node generating the packet
  - Cost of this node to any direct neighbors
  - Sequence-no. (SEQNO)
  - TTL field for that field (time to live)
- Reliable flooding (Reliable Flooding)
  - current LSP of each node are stored
  - Forward of LSP to all neighbors
    - except to be node where it has been received from
  - Periodically creation of new LSPs
    - with increasing SEQNO
  - Decrement TTL when LSPs are forwarded

Inter-AS: BGPv4 (Border Gateway Protocol)

- de facto standard

- Path-Vector-Protocol
  - like Distance Vector Protocol
    - store whole path to the target
  - each Border Gateway advertizes to all its neighbors (peers) the complete path to the target (per TCP)

- If gateway X sends the path to the peer-gateway W
  - then W can choose the path or not
  - optimization criteria
    - cost, policy, etc.
  - if W chooses the path of X, it publishes
    - Path(W,Z) = (W, Path (X,Z))

- Remark
  - X can control incoming traffic using advertisements
  - all details hidden here

BGP-Routing Table Size 1994-2013

http://bgp.potaroo.net/as1221/bgp-active.html
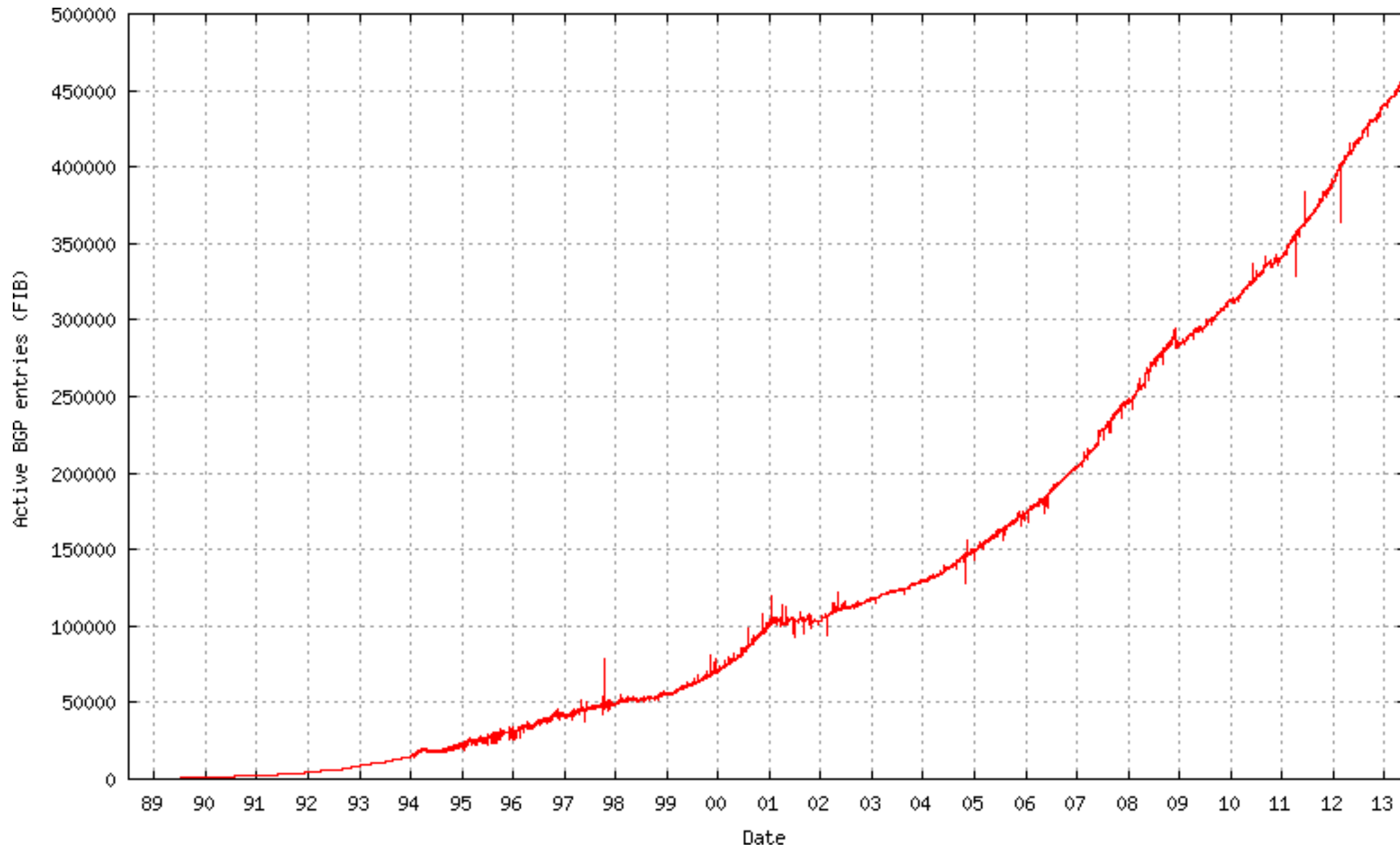
- Nobody can work with IP-Addresses
  - 173.194.113.15 for Google
  - 132.230.2.100 for Uni Freiburg
  - What is the meaning of 46.243.125.34?
- IP addresses change
- The Domain Name System (DNS) translates address in both directions

DNS – Architecture

- DNS maps names to addresses and vice versa
- Names are constructed hierarchically in a name space
  - Max. 63 characters per component, max 255 overall characters
  - in each domain the domain owner controls the name space below
- The mapping is done by Name-Servers

(from Tanenbaum)
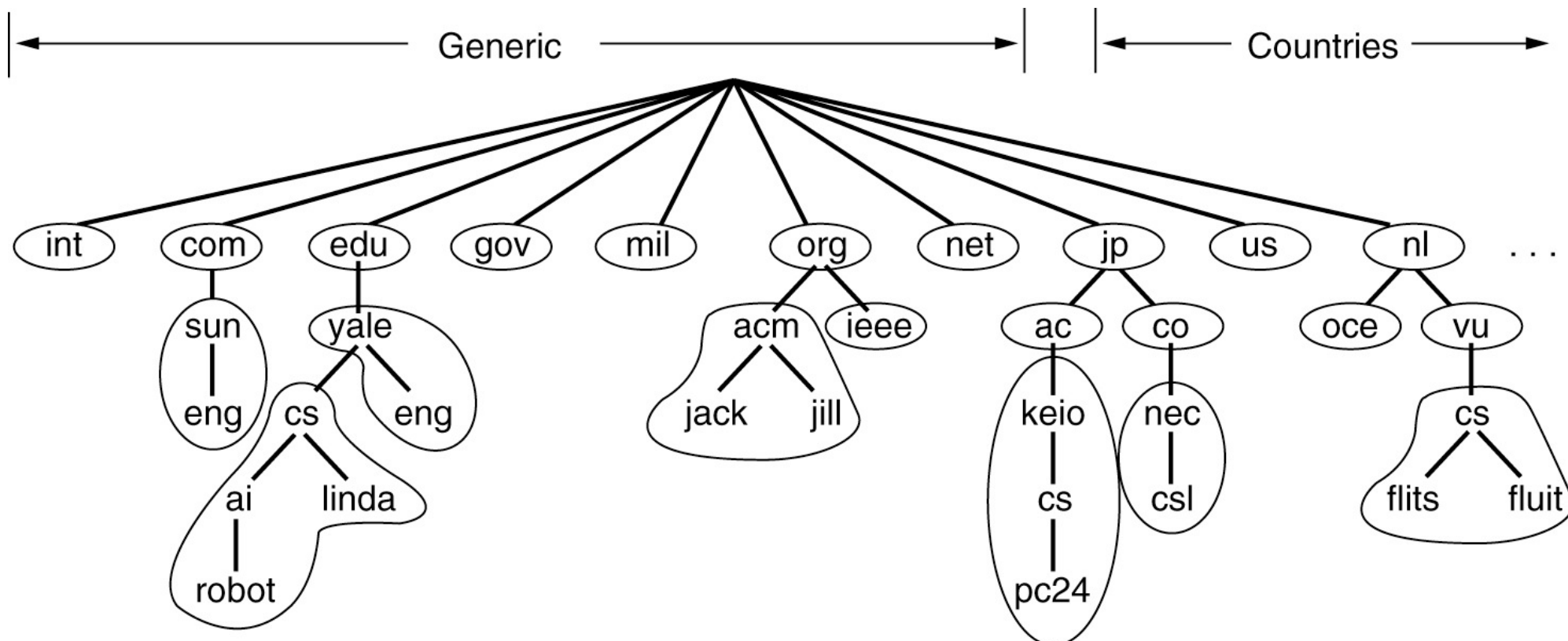
DNS Name Server

- The name space is partitioned in zones
- Each zone has a *Primary Name Server* with the authoritative information
  - in addition *Secondary Name Server* for reliability
- Each Name Server knows
  - his zone
  - name-servers of below
  - sister name servers or at least a server reffering to them

(from Tanenbaum)
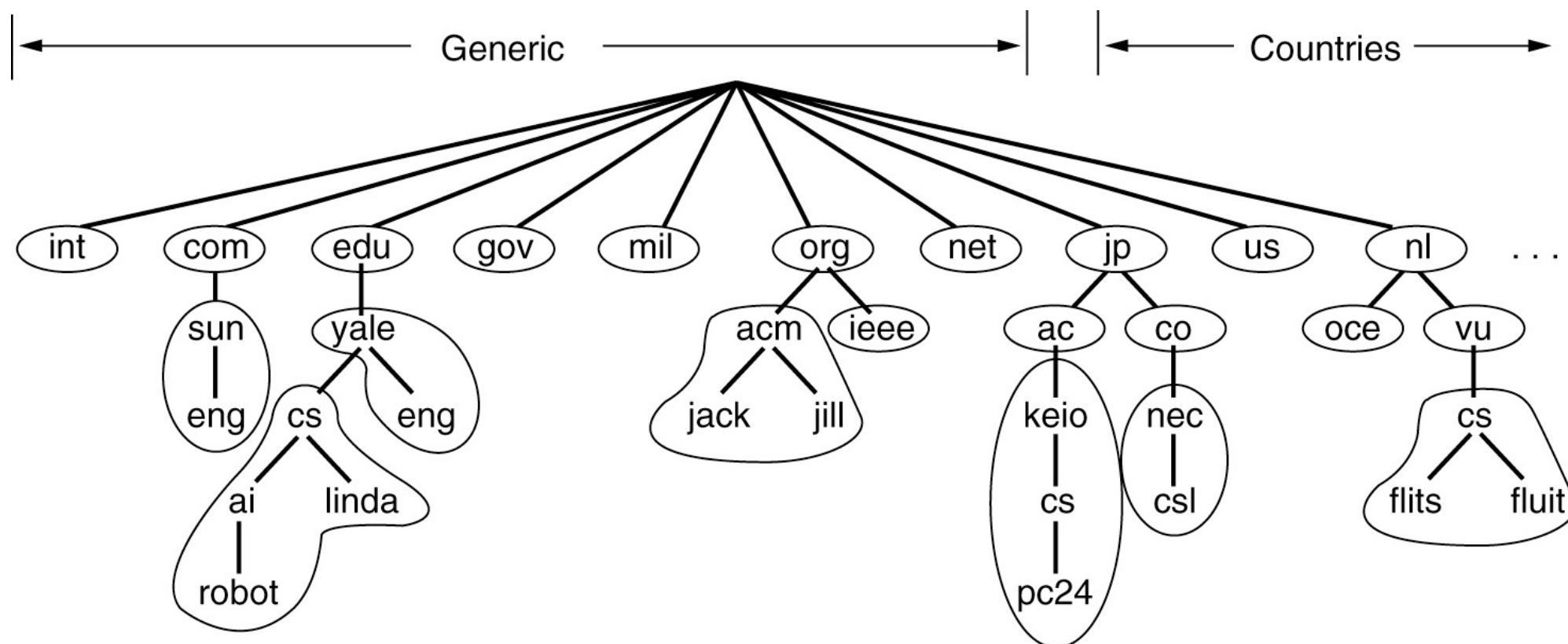
DNS Query Processing

- Requests are send to the pre-configured name-server
  - if possible this name-server resolves it
  - if not, the query is passed on to the best suited name server
  - Answers are passed back by the intermediate servers
- Server may store previous answers (cachen)
  - but only for a pre-determined time
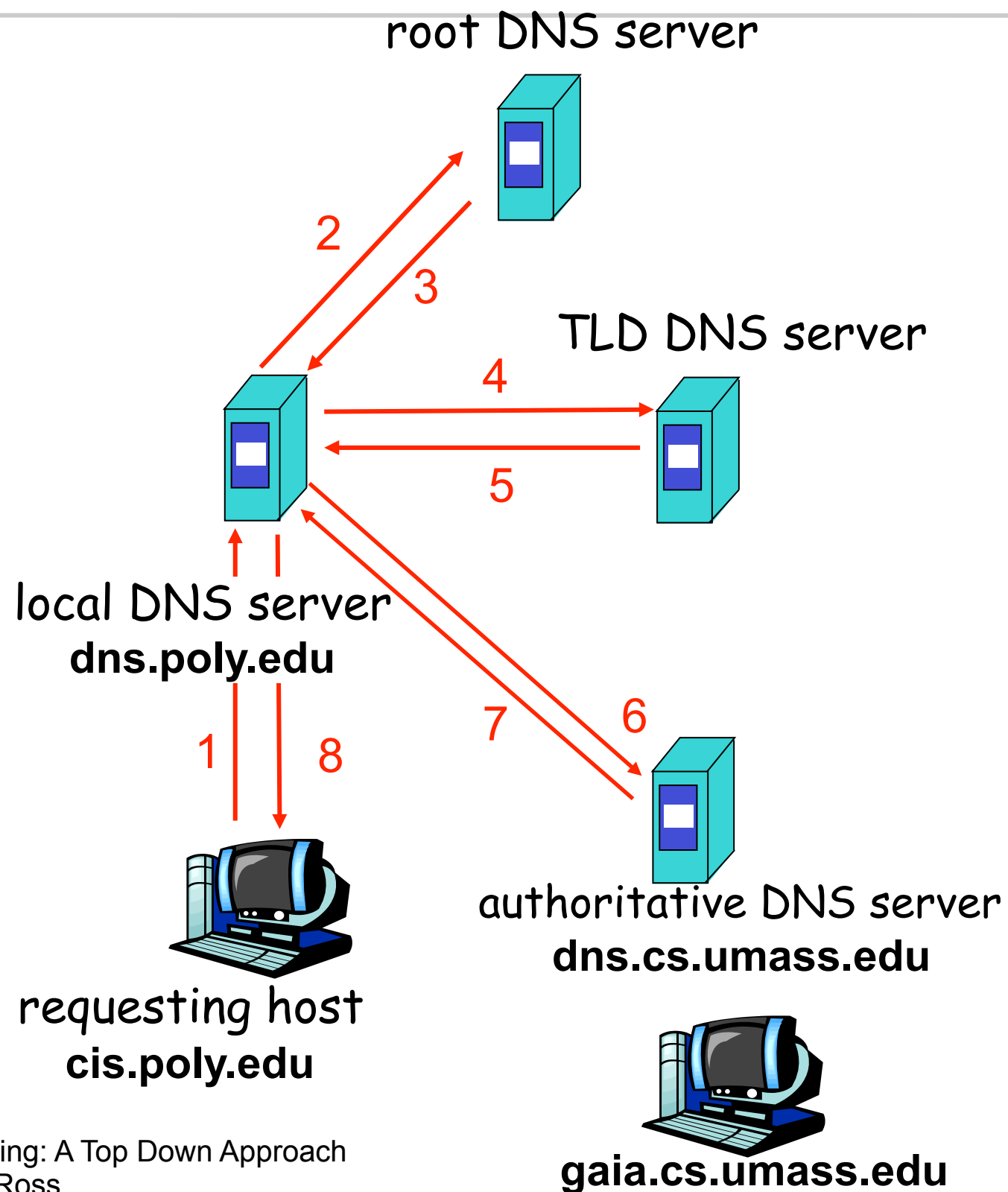
Originator	VU CS name server	Edu name server	Yale name server	Yale CS name server
flits.cs.vu.nl	cs.vu.nl	edu-server.net	yale.edu	cs.yale.edu

1 2 3 4
8 7 6 5

(from Tanenbaum)

Beispiel

(from Tanenbaum)

DNS Root Name Servers

CoNe Freiburg

a Verisign, Dulles, VA
c Cogent, Herndon, VA (also LA)
d U Maryland College Park, MD
g US DoD Vienna, VA
h ARL Aberdeen, MD
j Verisign, (21 locations)

k RIPE London (also 16 other locations)

i Autonomica, Stockholm (plus
28 other locations)

e NASA Mt View, CA
f Internet Software C. Palo Alto, CA
(and 36 other locations)

m WIDE Tokyo (also Seoul,
Paris, SF)

b USC-ISI Marina del Rey, CA
l ICANN Los Angeles, CA

13 root name
servers worldwide

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross

UNI FREIBURG

TLD and authoritative Servers

- **Top-Level Domain (TLD) Server**

  - responsible for com, org, net, edu, etc, and all Top-Level-Country-Domains uk, fr, ca, jp.

  - Network Solutions provides Server for *com* TLD

  - Educause for *edu* TLD

- **Authorized DNS Servers:**

  - DNS-Servers of organizations

    - responsible for the mapping from IP-Adresse to host names

  - ISP, companies, …

DNS Entries

- DNS: distributed data base storing Resource Records (RR)

- RR Format: (Name, Wert, Typ, TTL)

- Contents:

  - Domain_name: Domain(s) of the entry

  - Time_to_live: validity (in seconds)

  - Type: see table

  - Value: e.g. IP-Adresse

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

DNS Iterative Lookup

root DNS server

TLD DNS server

local DNS server
dns.poly.edu

authoritative DNS server
dns.cs.umass.edu

requesting host
cis.poly.edu

gaia.cs.umass.edu

DNS Rekursive Lookup

root DNS server

2 3

7 6

TLD DNS server

local DNS server
dns.poly.edu

5 4

1 8

requesting host
cis.poly.edu

authoritative DNS server
dns.cs.umass.edu

gaia.cs.umass.edu

Slides and Contents from
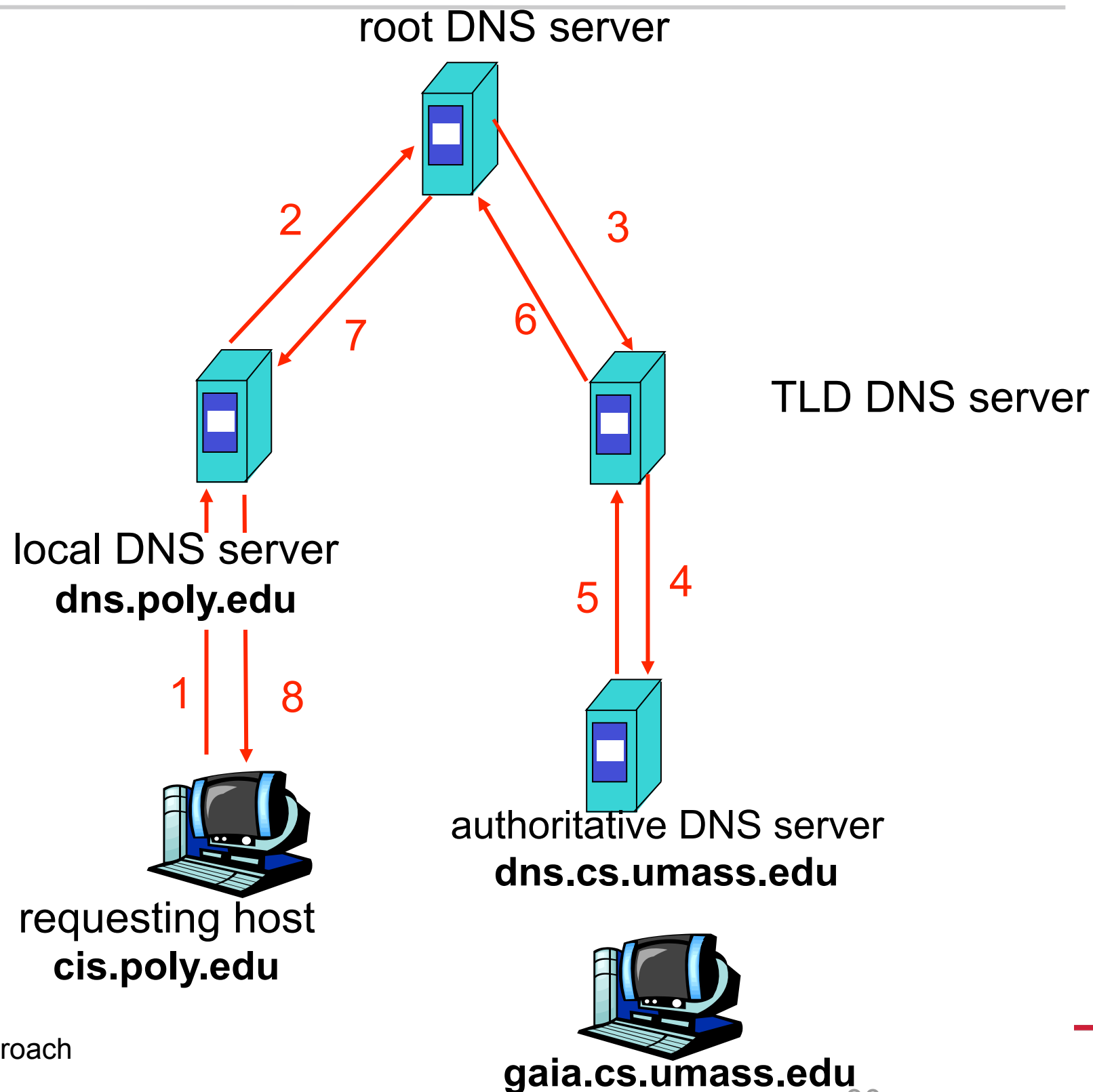
- Computer Networking: A
  Top Down Approach
  5th edition.
  Jim Kurose, Keith Ross
  Addison-Wesley, April
  2009.
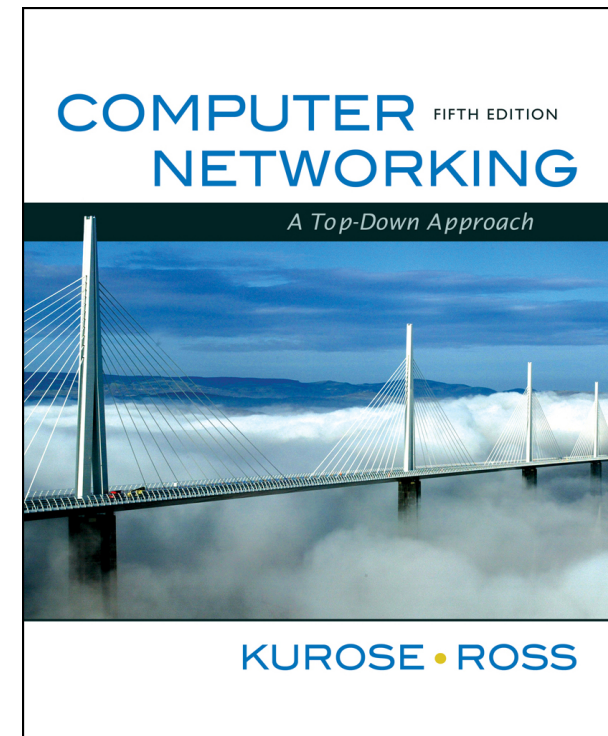
Application Layer

- Goals
  - conceptual, implementation aspects of network application protocols
    - transport-layer service models
    - client-server paradigm
    - peer-to-peer paradigm

- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS

- programming network applications
  - socket API

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- social networks
- voice over IP
- real-time video conferencing
- grid computing

Creating a network app

- Programs that
  - run on (different) end systems
  - communicate over network
  - e.g., web server software communicates with browser software
  - No need to write software for network-core devices

- Network-core devices do not run user applications
  - applications on end systems allows for
  - rapid app development, propagation

application
transport
network
data link
physical

application
transport
network
data link
physical

application
transport
network
data link
physical
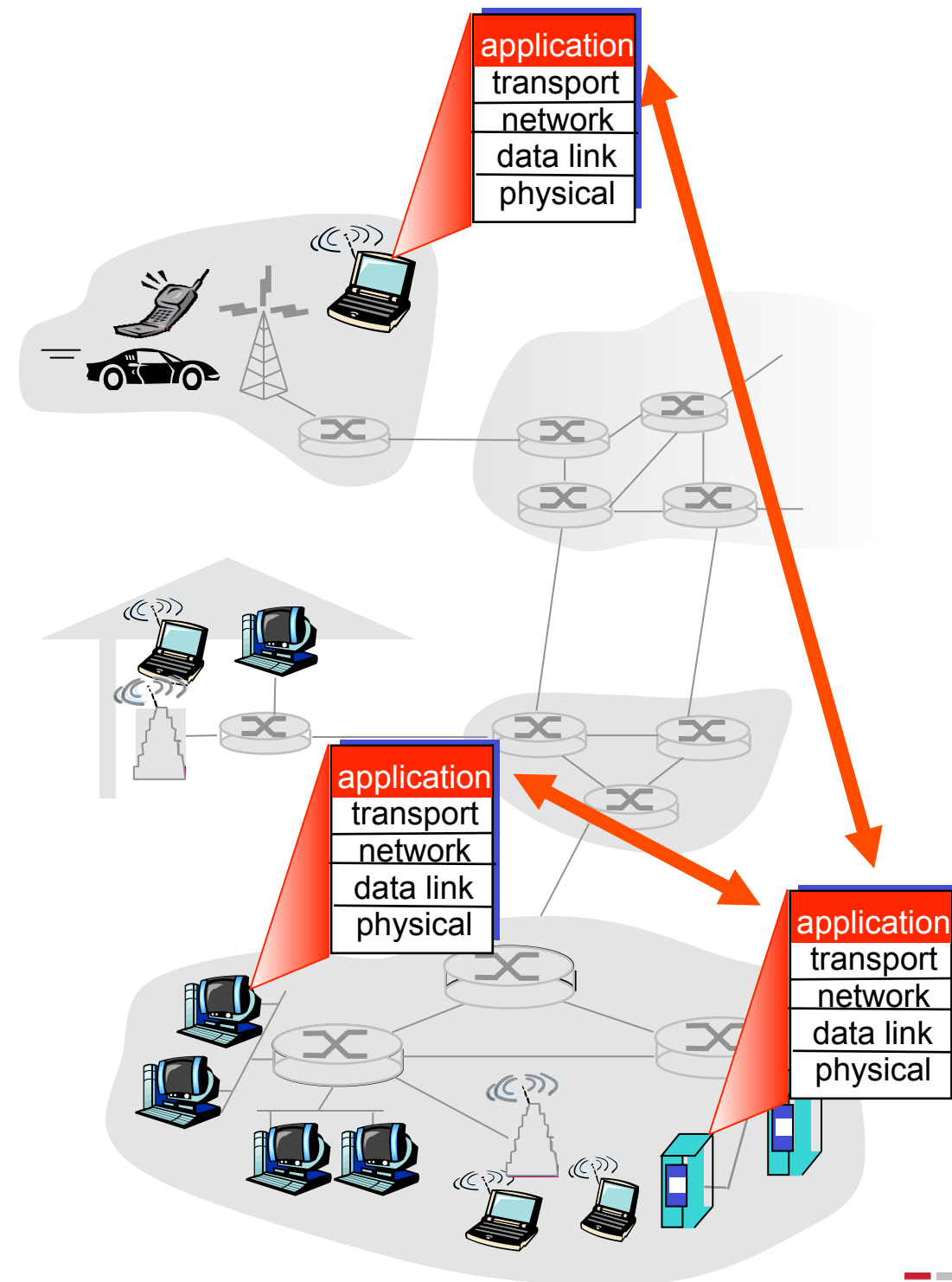
Client-Server Architecture

- Server:
  - always-on host
  - permanent IP address
  - server farms for scaling
- Clients:
  - communicate with server
  - may be intermittently connected
  - may have dynamic IP addresses
  - do not communicate directly with each other
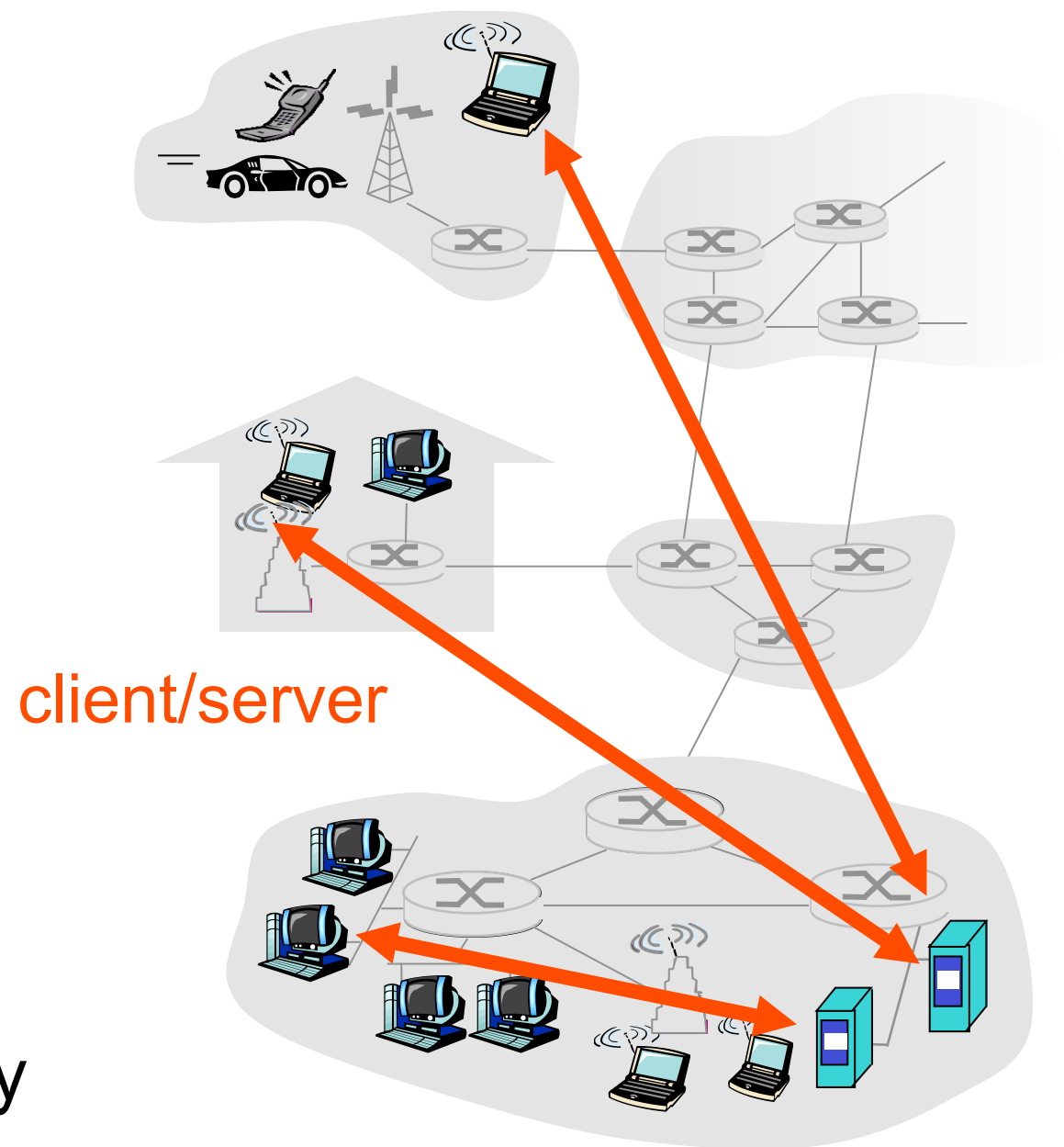
client/server

Peer-to-Peer Architecture

- no always-on server

  - arbitrary end systems directly communicate

  - peers are intermittently connected and change IP addresses

peer-peer

- Highly scalable but difficult to manage
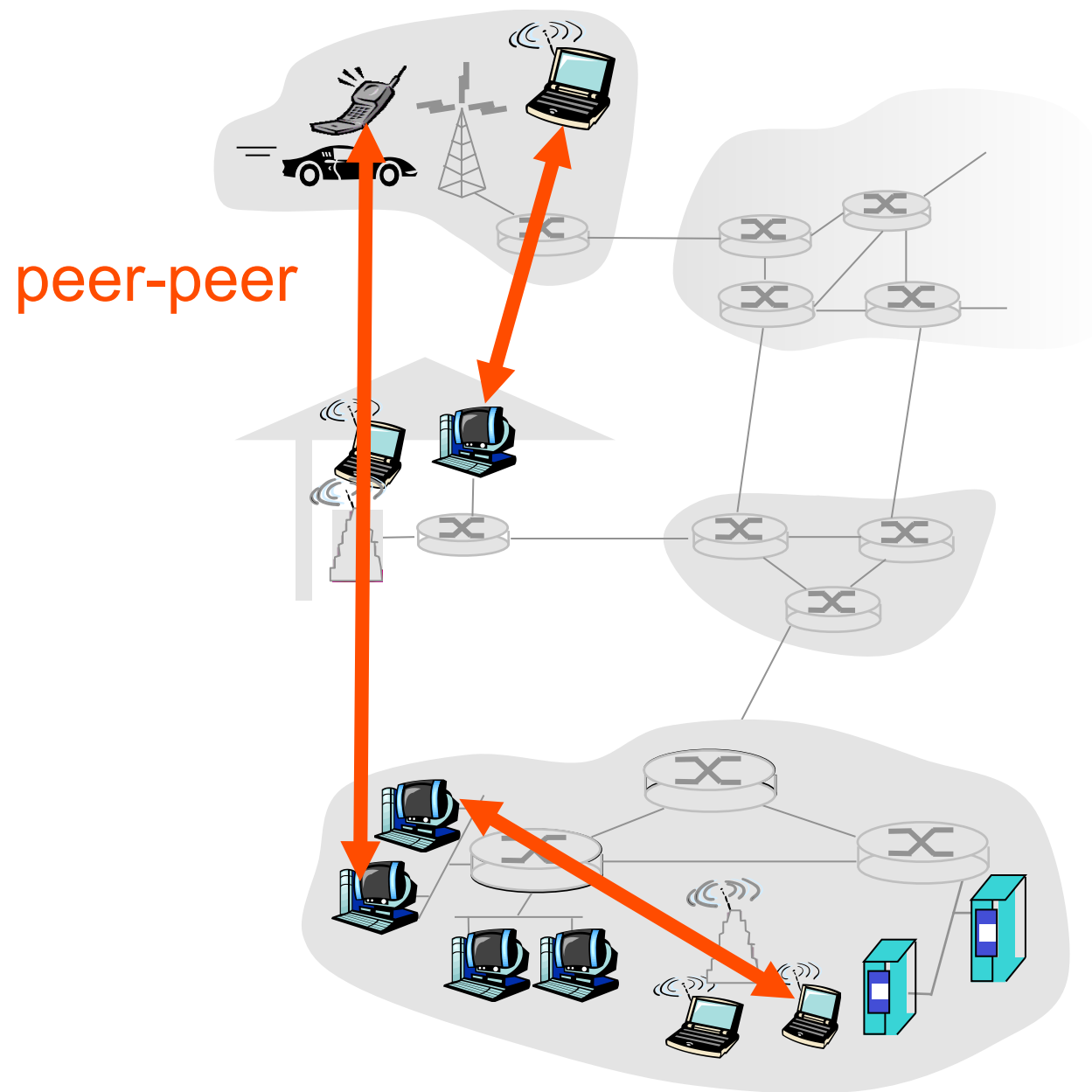
Hybrid of client-server and P2P

- Skype
  - voice-over-IP P2P application
  - centralized server: finding address of remote party:
  - client-client connection: direct (not through server)
- Instant messaging
  - chatting between two users is P2P
  - centralized service: client presence detection/ location
  - user registers its IP address with central server when it comes online
  - user contacts central server to find IP addresses of buddies

Processes communicating

- Process: program running within a host.
  - within same host, two processes communicate using inter-process communication (defined by OS).
  - processes in different hosts communicate by exchanging messages
- Client process: process that initiates communication
- Server process: process that waits to be contacted
- Applications with P2P architectures have both
  - client processes & server processes

Sockets

- process sends/receives messages to/from its socket

  - socket analogous to door

  - sending process shoves message out door

  - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process
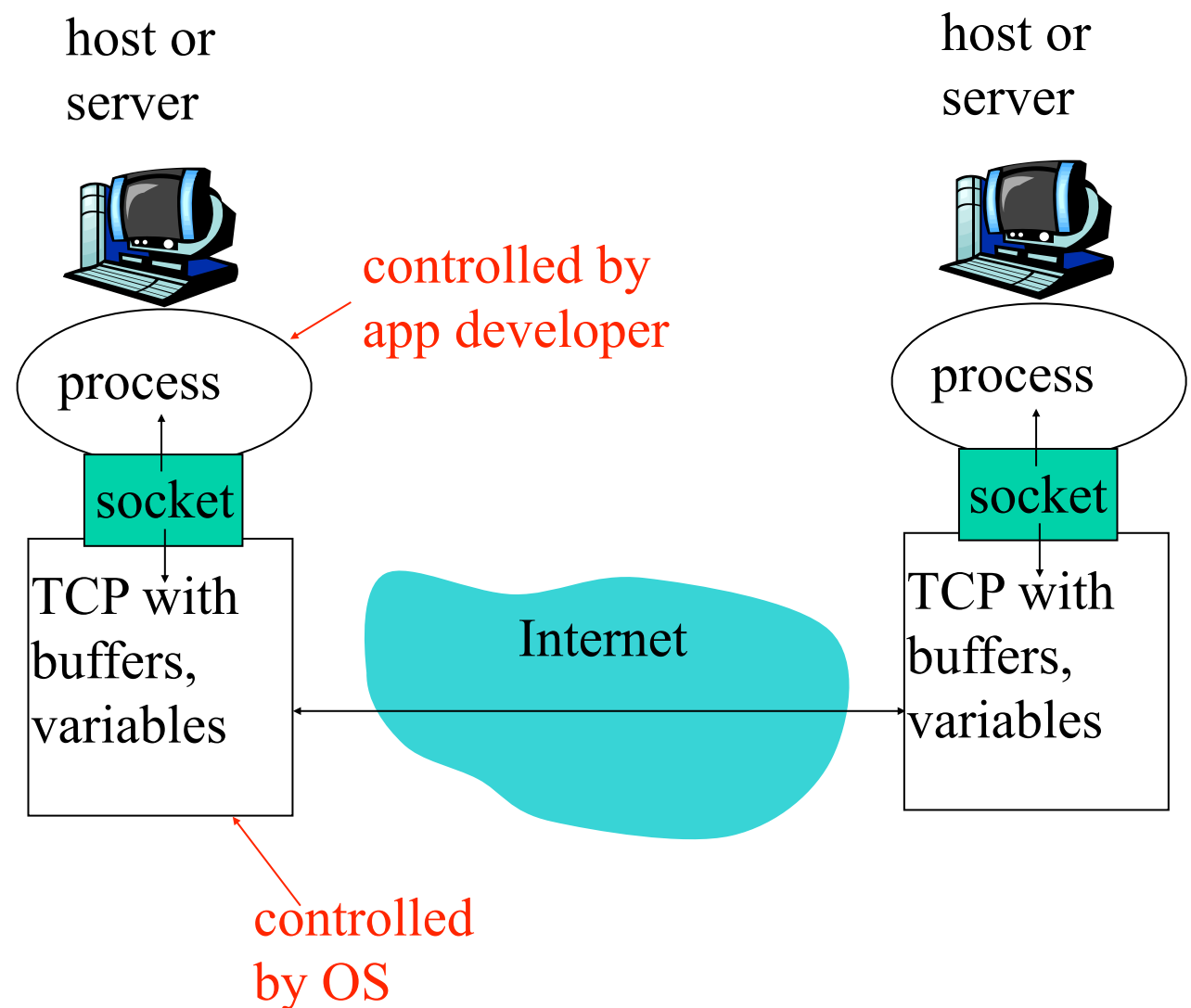
Addressing processes

- to receive messages, process must have identifier
  - host device has unique 32-bit IP address
  - Exercise: use ipconfig from command prompt to get your IP address (Windows)
- Q: does IP address of host on which process runs suffice for identifying the process?
- A: No, many processes can be running on same
  - Identifier includes both IP address and port numbers associated with process on host.
- Example port numbers:
  - HTTP server: 80
  - Mail server: 25

App-layer protocol defines

- Types of messages exchanged,
    - e.g., request, response
- Message syntax:
    - what fields in messages & how fields are delineated
- Message semantics
    - meaning of information in fields
- Rules for when and how processes send & respond to messages
- Public-domain protocols:
    - defined in RFCs
    - allows for interoperability
    - e.g., HTTP, SMTP, BitTorrent
- Proprietary protocols:
    - e.g., Skype, ppstream

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross

What transport service does an app need?

- ## Data loss
  - some apps (e.g., audio) can tolerate some loss
  - other apps (e.g., file transfer, telnet) require 100% reliable data transfer

- ## Timing
  - some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

- ## Throughput
  - some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
  - other apps ("elastic apps") make use of whatever throughput they get

- ## Security
  - Encryption, data integrity, …

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross

Web und HTTP

- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
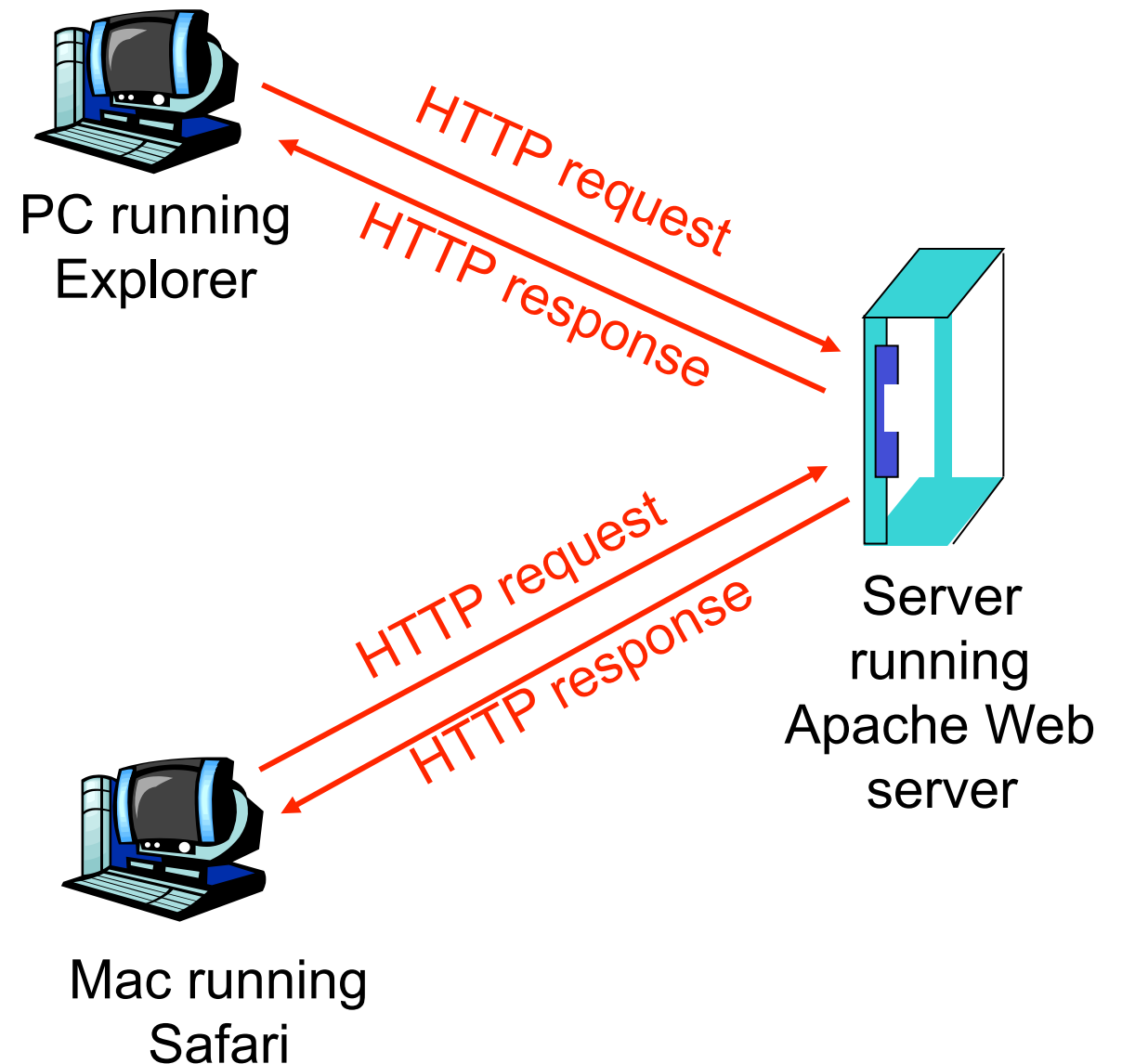- Each object is addressable by a URL

  - Example URL:

`www.someschool.edu/someDept/pic.gif`

host name   path name

HTTP overview

- **HTTP: hypertext transfer protocol**
  - Web's application layer protocol

- **client/server model**

  - client: browser that requests, receives, "displays" Web objects

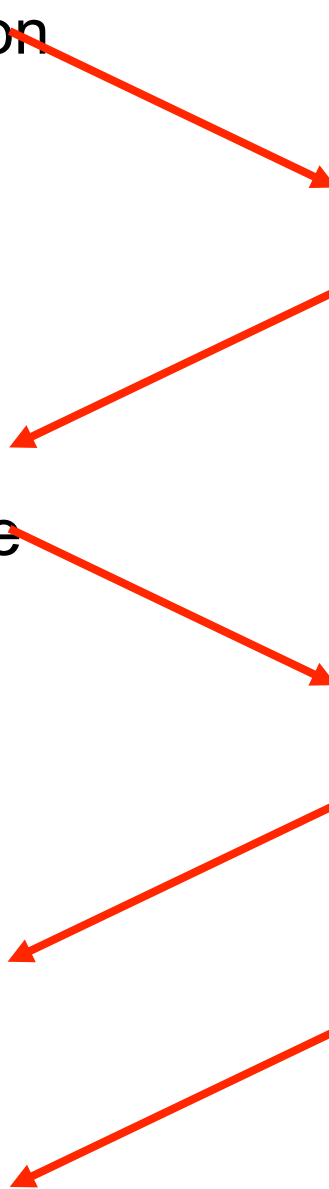 - server: Web server sends objects in response to requests

PC running Explorer

HTTP request
HTTP response

HTTP request
HTTP response

Server running Apache Web server

Mac running Safari

HTTP overview

- **Uses TCP:**
  - client initiates TCP connection (creates socket) to server, port 80
  - server accepts TCP connection from client

- **HTTP messages (application-layer protocol messages) exchanged**
  - between browser (HTTP client)
  - and Web server (HTTP server)

- **TCP connection closed**

CoNe Freiburg

- ## HTTP is "stateless"

  - server maintains no information about past client requests

- ## Why

  - Protocols that maintain "state" are complex!

  - past history (state) must be maintained

  - if server/client crashes, their views of "state" may be inconsistent, must be reconciled
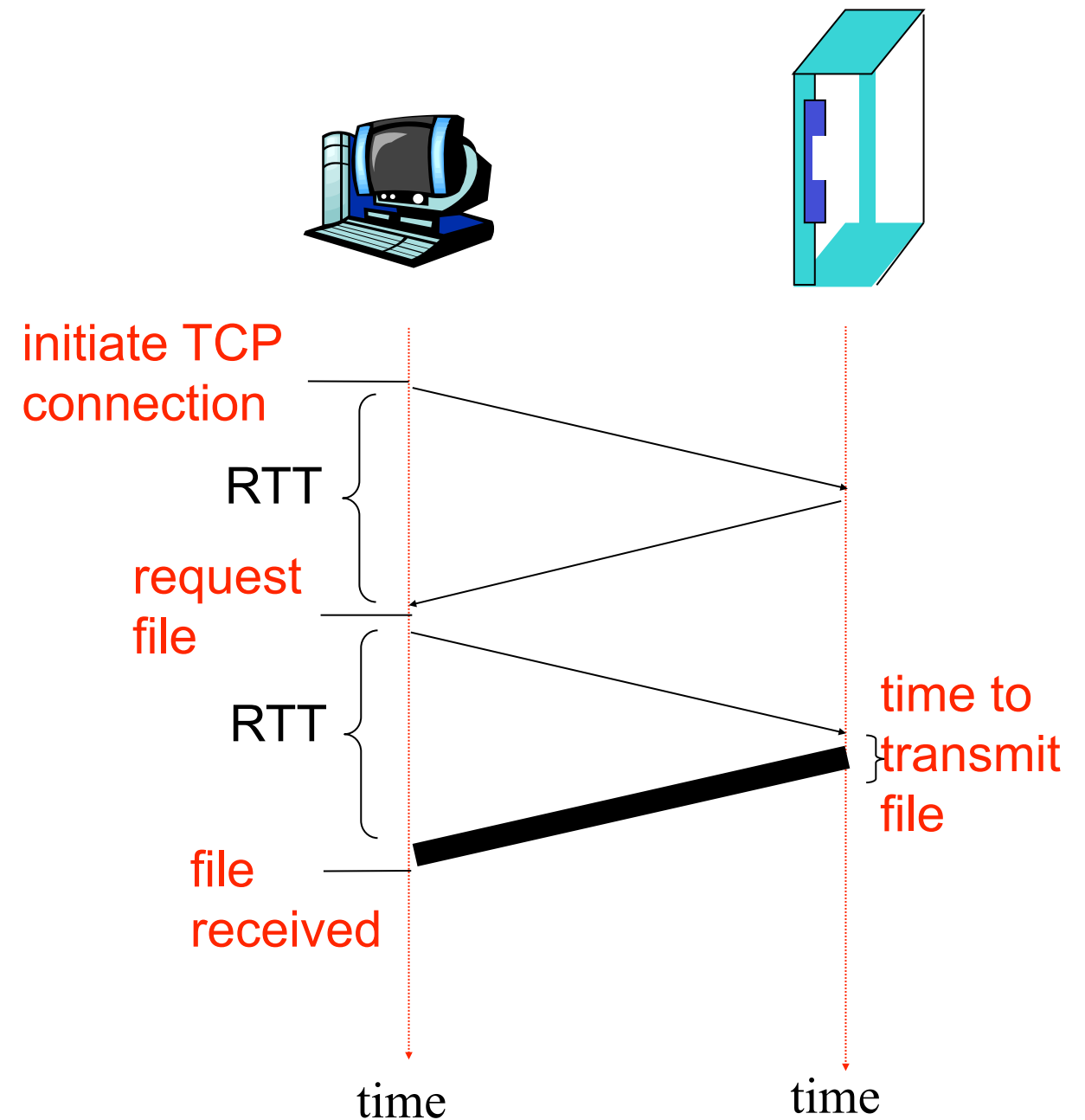
HTTP connections

- ## Nonpersistent HTTP

  - At most one object is sent over a TCP connection.

- ## Persistent HTTP

  - Multiple objects can be sent over single TCP connection between client and server.

Nonpersistent HTTP

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

4. HTTP server closes TCP connection.

6. Steps 1-5 repeated for each of 10 jpeg objects

- ■ **Definition of RTT: time for a small packet to travel from client to server and back.**

- ■ **Response time:**
  - one RTT to initiate TCP connection
  - one RTT for HTTP request and first few bytes of HTTP response to return
  - file transmission time

- ■ **total = 2RTT+transmit time**

initiate TCP connection

RTT

request file

RTT

file received

time to transmit file

time   time

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross

Persistent HTTP

- **Nonpersistent HTTP issues:**
  - requires 2 RTTs per object
  - OS overhead for each TCP connection
  - browsers often open parallel TCP connections to fetch referenced objects

- **Persistent HTTP**
  - server leaves connection open after sending response
  - subsequent HTTP messages between same client/server sent over open connection
  - client sends requests as soon as it encounters a referenced object
  - as little as one RTT for all the referenced objects

- ■ two types of HTTP messages
  - request, response

- ■ HTTP-Request message
  - ASCII (human-readable format)

request line
(GET, POST,
HEAD commands)

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```
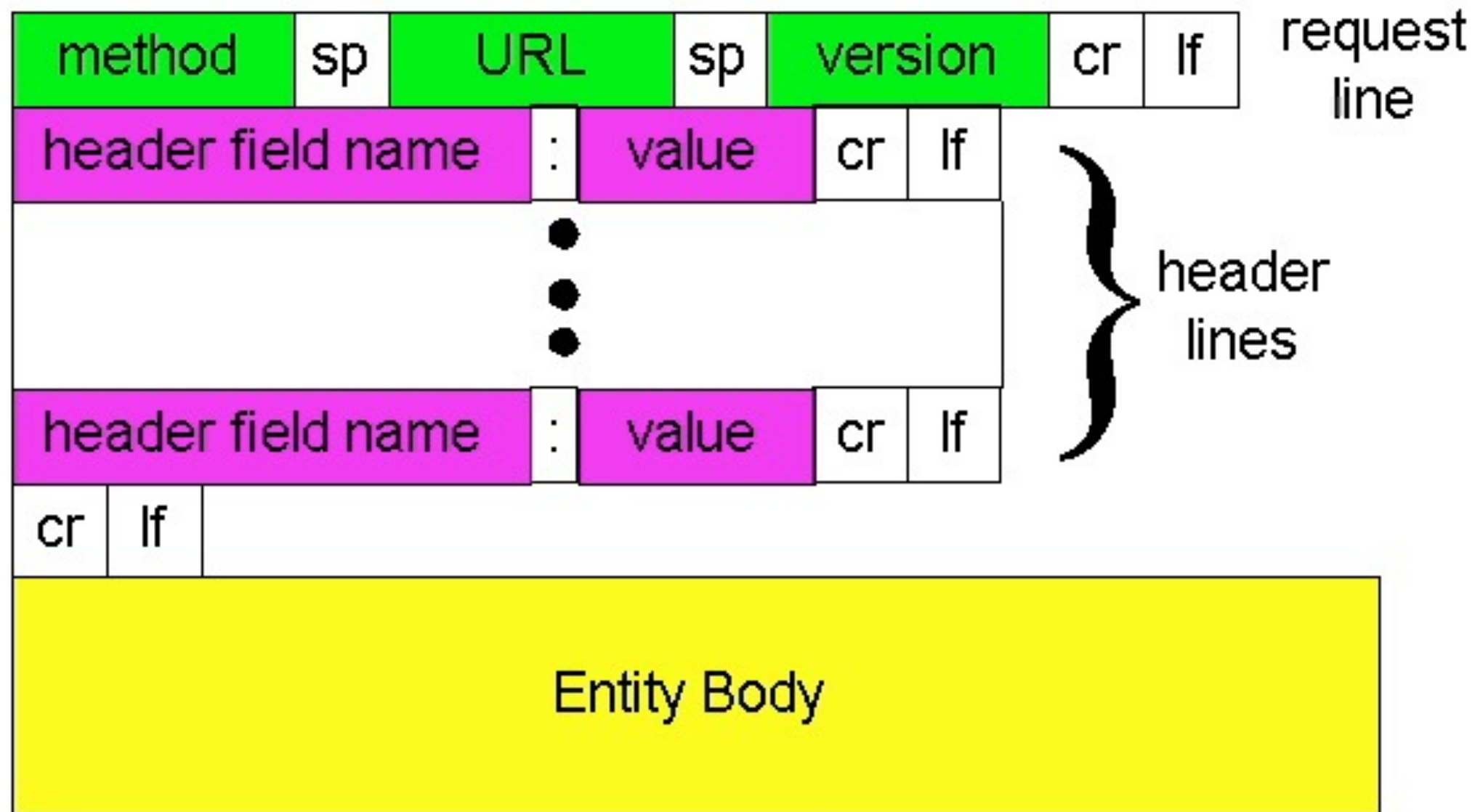
Carriage return,
line feed
indicates end
of message

(extra carriage return, line feed)
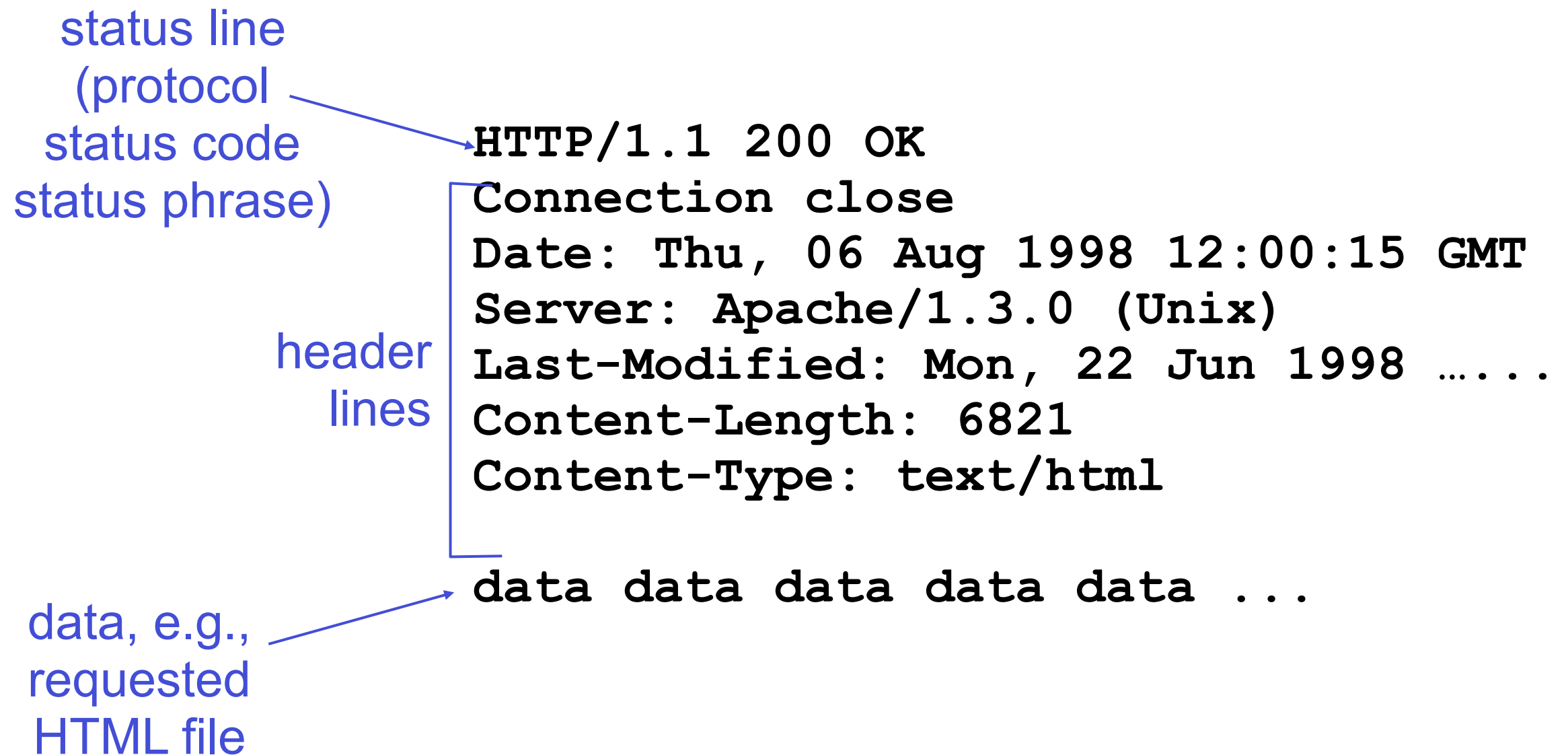
Upload

- ■ Post method:
  - Web page often includes form input
  - Input is uploaded to server in entity body

- ■ URL method:
  - Uses GET method
  - Input is uploaded in URL field of request line:

```
www.somesite.com/animalsearch?monkeys&banana
```

Methods

- **HTTP/1.0**
  - GET
  - POST
  - HEAD
    - asks server to leave requested object out of response
- **HTTP/1.1**
  - GET, POST, HEAD
  - PUT
    - uploads file in entity body to path specified in URL field
  - DELETE
  - deletes file specified in the URL field

HTTP response message

status line
(protocol
status code
status phrase)

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

header
lines

data, e.g.,
requested
HTML file

HTTP response status codes

- In first line in server->client response message.

- A few sample codes
  - 200 OK
    - request succeeded, requested object later in this message
  - 301 Moved Permanently
    - requested object moved, new location specified later in this message (Location:)
  - 400 Bad Request
    - request message not understood by server
  - 404 Not Found
    - requested document not found on this server
  - 505 HTTP Version Not Supported

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

    telnet cis.poly.edu 80 Opens TCP connection to port 80
    (default HTTP server port) at cis.poly.edu.
    Anything typed in sent
    to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

    GET /~ross/ HTTP/1.1 By typing this in (hit carriage
    Host: cis.poly.edu return twice), you send
    this minimal (but complete)
    GET request to HTTP server

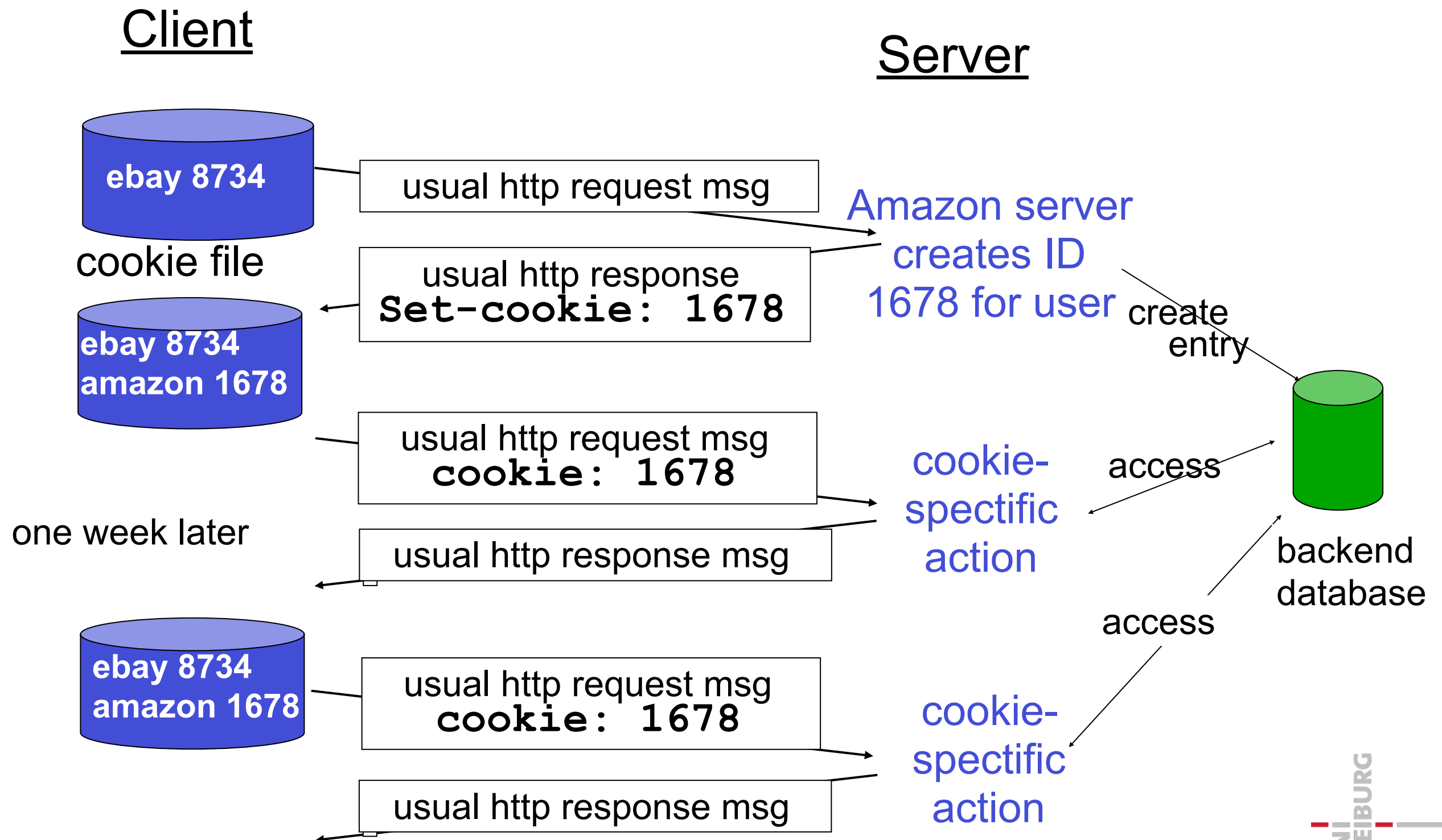3. Look at response message sent by HTTP server!

User-server state: cookies

- **Many major Web sites use cookies**

- **Four components:**
  - 1) cookie header line of HTTP response message
  - 2) cookie header line in HTTP request message
  - 3) cookie file kept on user's host, managed by user's browser
  - 4) back-end database at Web site

User-server state: cookies

- Example:
  - Susan always access Internet always from PC

- visits specific e-commerce site for first time

- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

Cookies: keeping "state"

Client

Server

ebay 8734

cookie file

ebay 8734
amazon 1678

one week later

ebay 8734
amazon 1678

usual http request msg

usual http response
Set-cookie: 1678

usual http request msg
cookie: 1678

usual http response msg

usual http request msg
cookie: 1678

usual http response msg

Amazon server
creates ID
1678 for user

create
entry

cookie-
spectific
action

access

access

cookie-
spectific
action

backend
database

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross

60

UNI FREIBURG

Cookies

- **What cookies can bring:**
  - authorization
  - shopping carts
  - recommendations
  - user session state (Web e-mail)

- **How to keep "state":**
  - protocol endpoints: maintain state at sender/receiver over multiple transactions
  - cookies: http messages carry state

- **Cookies and privacy:**
  - cookies permit sites to learn a lot about you
  - you may supply name and e-mail to sites

Server-Farm

- Um die Leistungsfähigkeit auf der Server-Seite zu erhöhen
  - wird eine Reihe von Web-Servern eingesetzt
- Front end
  - nimmt Anfragen an
  - reicht sie an separaten Host zur Weiterbearbeitung weiter
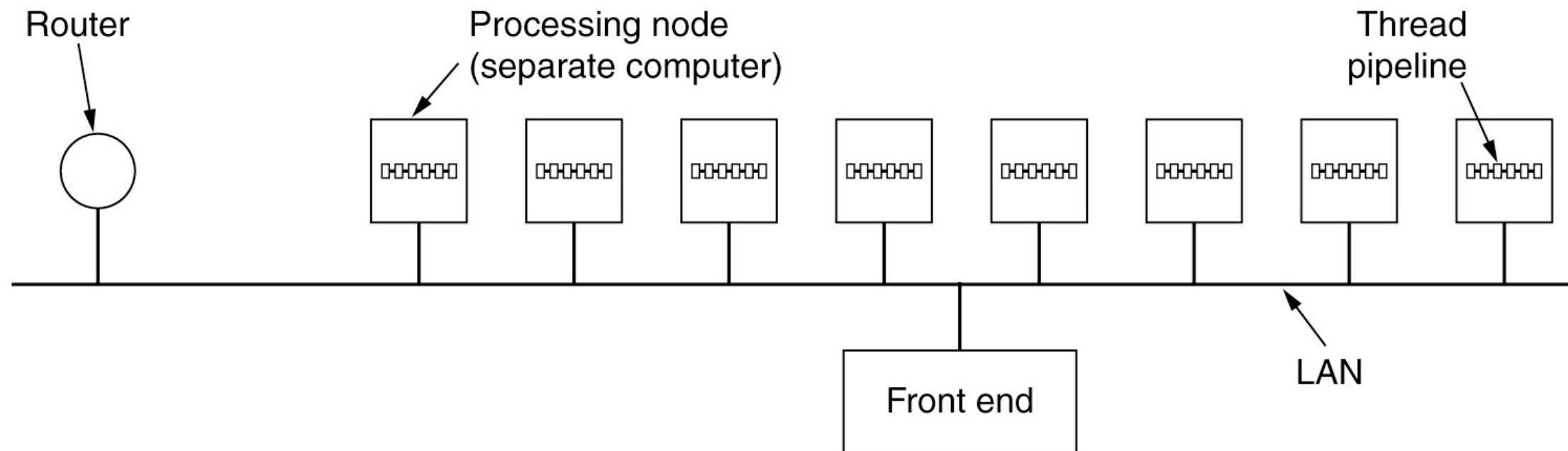
Web-Servers and Databases

- Web servers are not only static Web pages
    - Web pages are also created automatically
    - For this purpose, use a database
    - non static and can be altered through interactions
- Problem:
    - consistency
- solution
    - Web services and data-bank in a 3-tier architecture
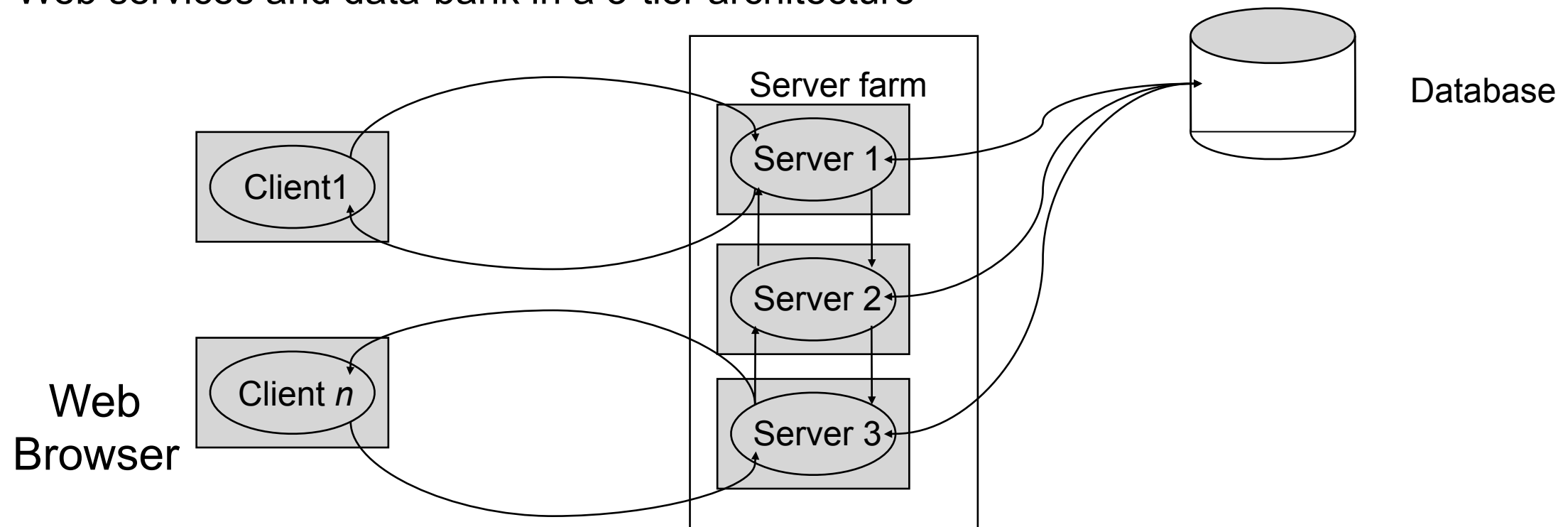
Beispiel: Google Data Centers

CoNe
Freiburg

- Estimated cost of data center: $600M
- Google spent $2.4B in 2007 on new data centers
- Each data center uses 50-100 megawatts of power

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross

Energy Informatics

02 Internet Protocols

Christian Schindelhauer

Technical Faculty

Computer-Networks and Telematics

University of Freiburg