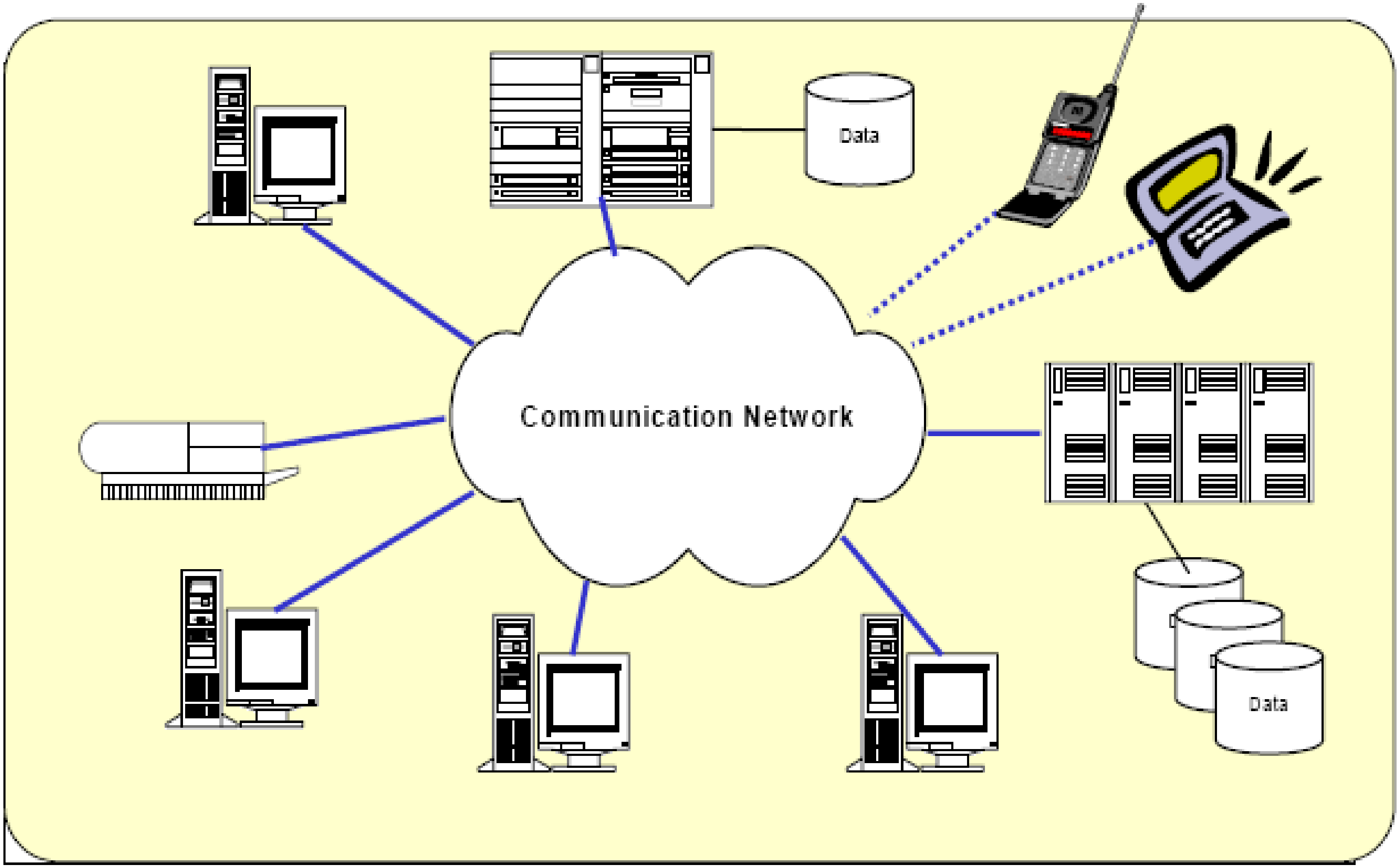
Distributed Systems

1. Examples and Motivation

Werner Nutt

A Distributed System …



Examples and Counterexamples

- Internet ?
- Uni Bozen intranet ?
- Mobile phone networks ?
- Bank account management,
  including automatic teller machines ?
- Instant messaging, e.g., Windows Live Messenger ?
- Collaborative editing tool ?
- Threads in an application ?
- Applications running on a PC ?

Definition

A *distributed system* consists of

- autonomous components (hardware, software)
- that are located at networked computers (hosts)
- and that communicate and coordinate their actions only be passing messages.

Motivation

- Sharing of resources

  (printers, disks, cameras, data, applications)
- …
- …

Implications

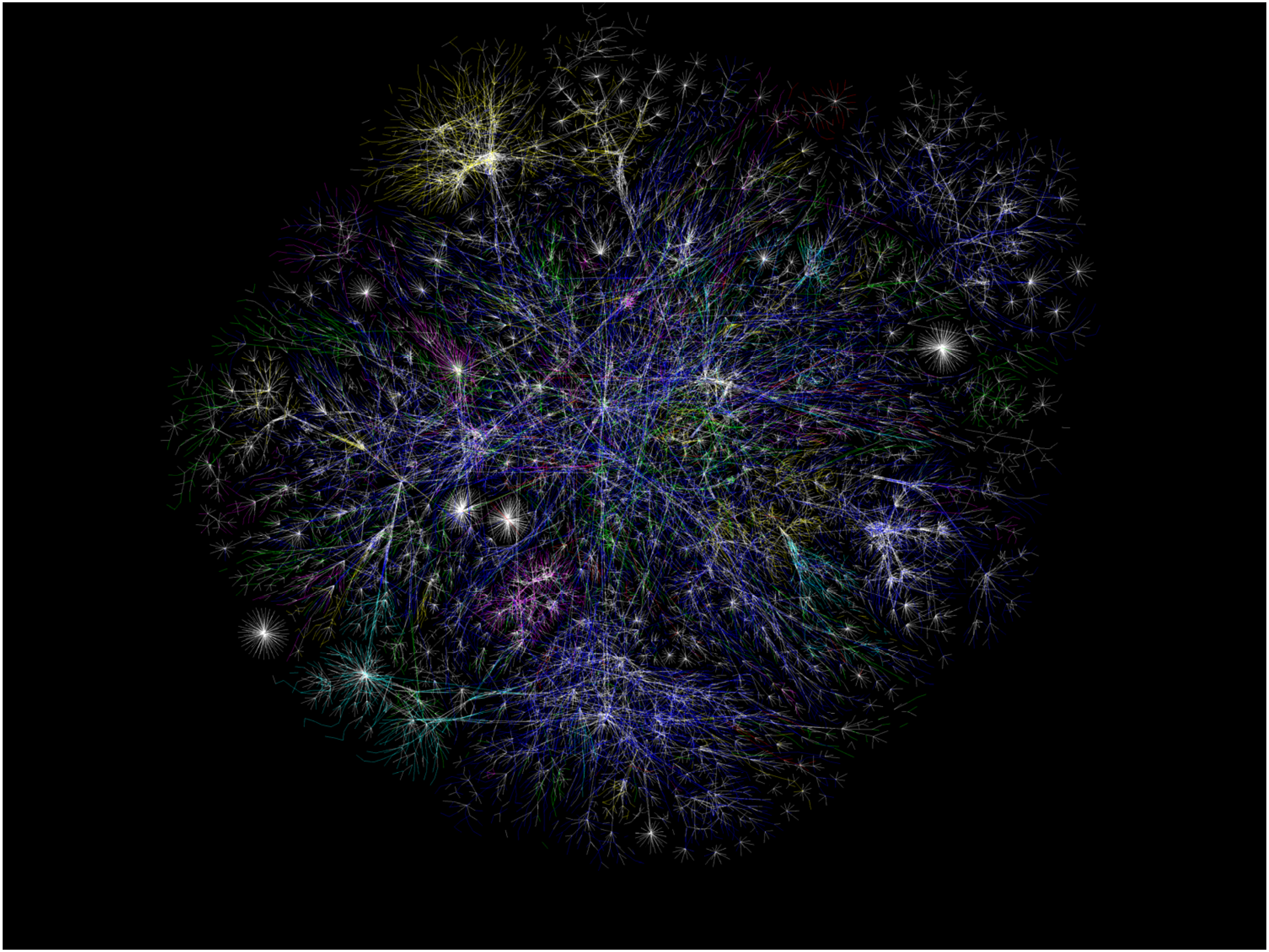
- Programs are executed concurrently

- There is no global time

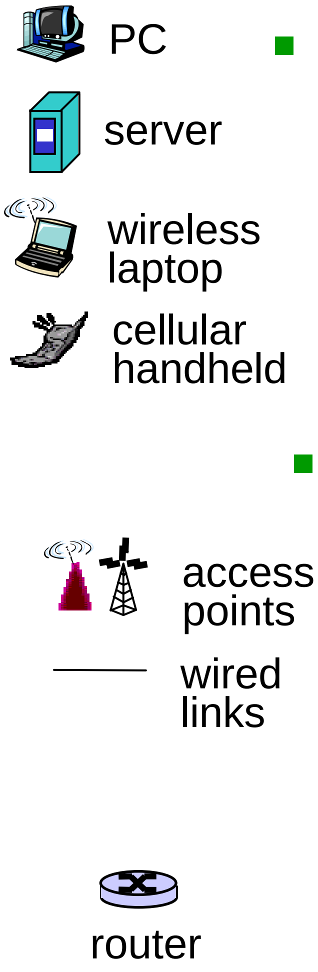
- Components can fail independently (isolation, crash)

Examples

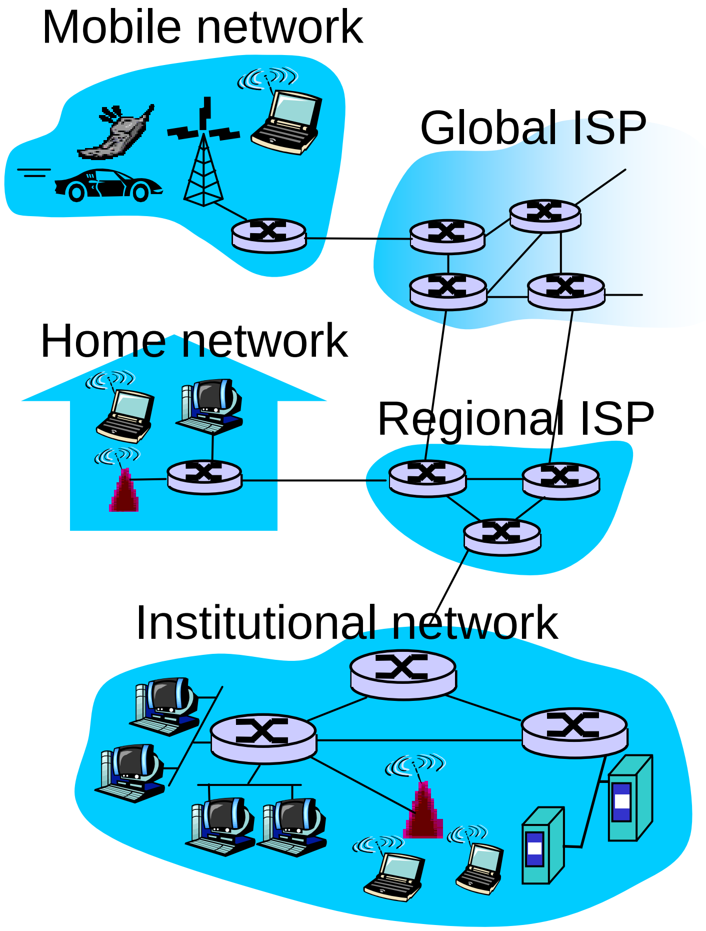
1. The Internet
2. The Word Wide Web



The Internet: the Nuts and Bolts

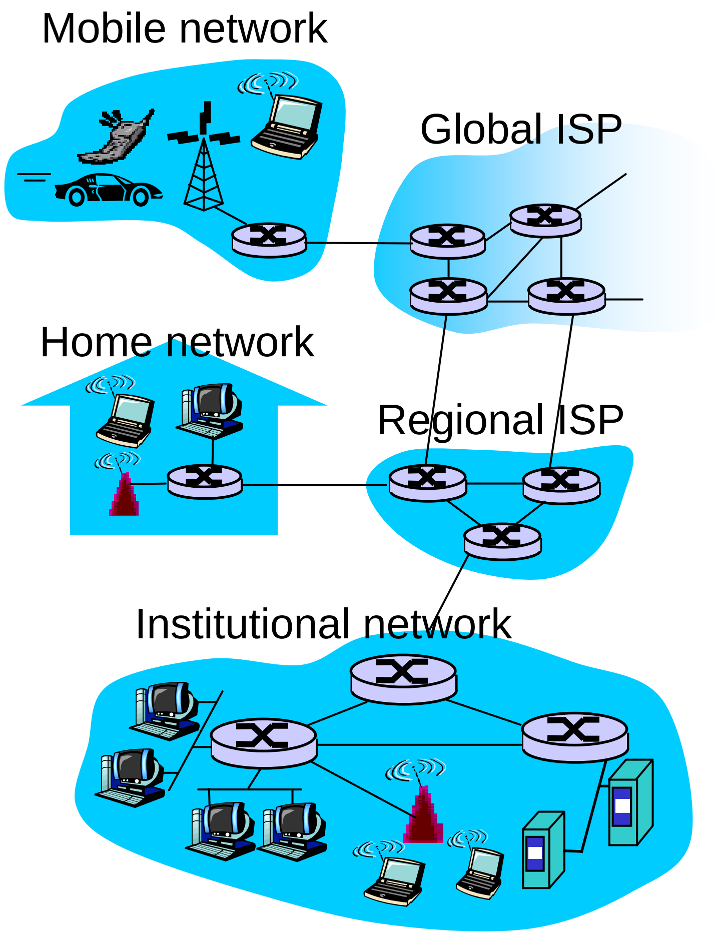


- Millions of connected computing devices: *hosts = end systems*
  - running *network applications*
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = *bandwidth*
- *routers:* forward packets (chunks of data)



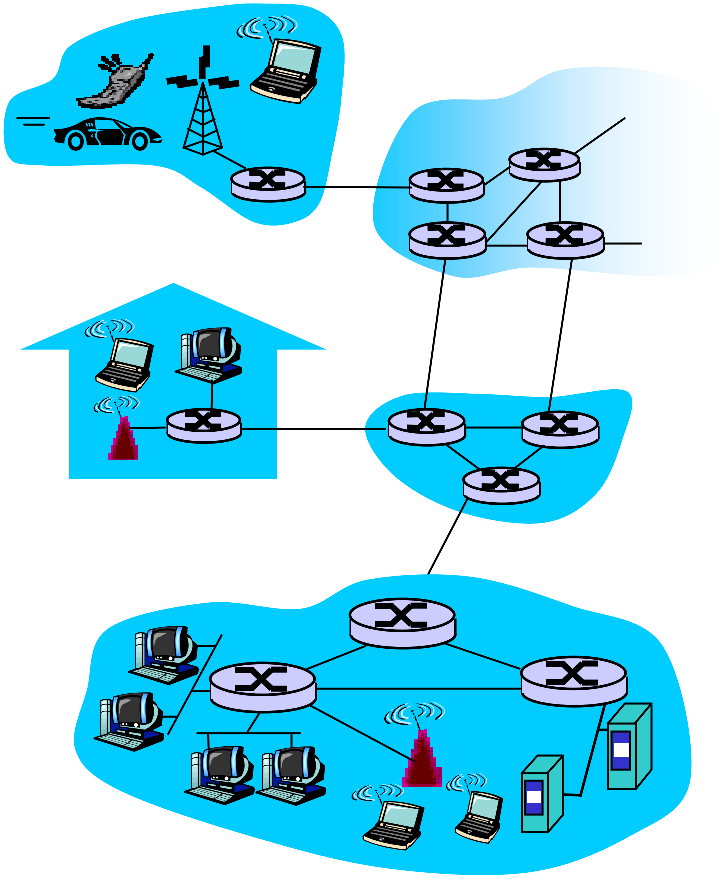
The Internet: the Nuts and Bolts

- *Protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, Ethernet
- *Internet:* “network of networks”
  - loosely hierarchical
  - public Internet versus private intranet
- Internet standards
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



The Internet: a Service View

- Communication *infrastructure* enables distributed applications:
  - Web, VoIP, email, games, e-commerce, file sharing
- Communication services provided to applications:
  - reliable data delivery from source to destination
  - “best effort” (unreliable) data delivery



Protocols

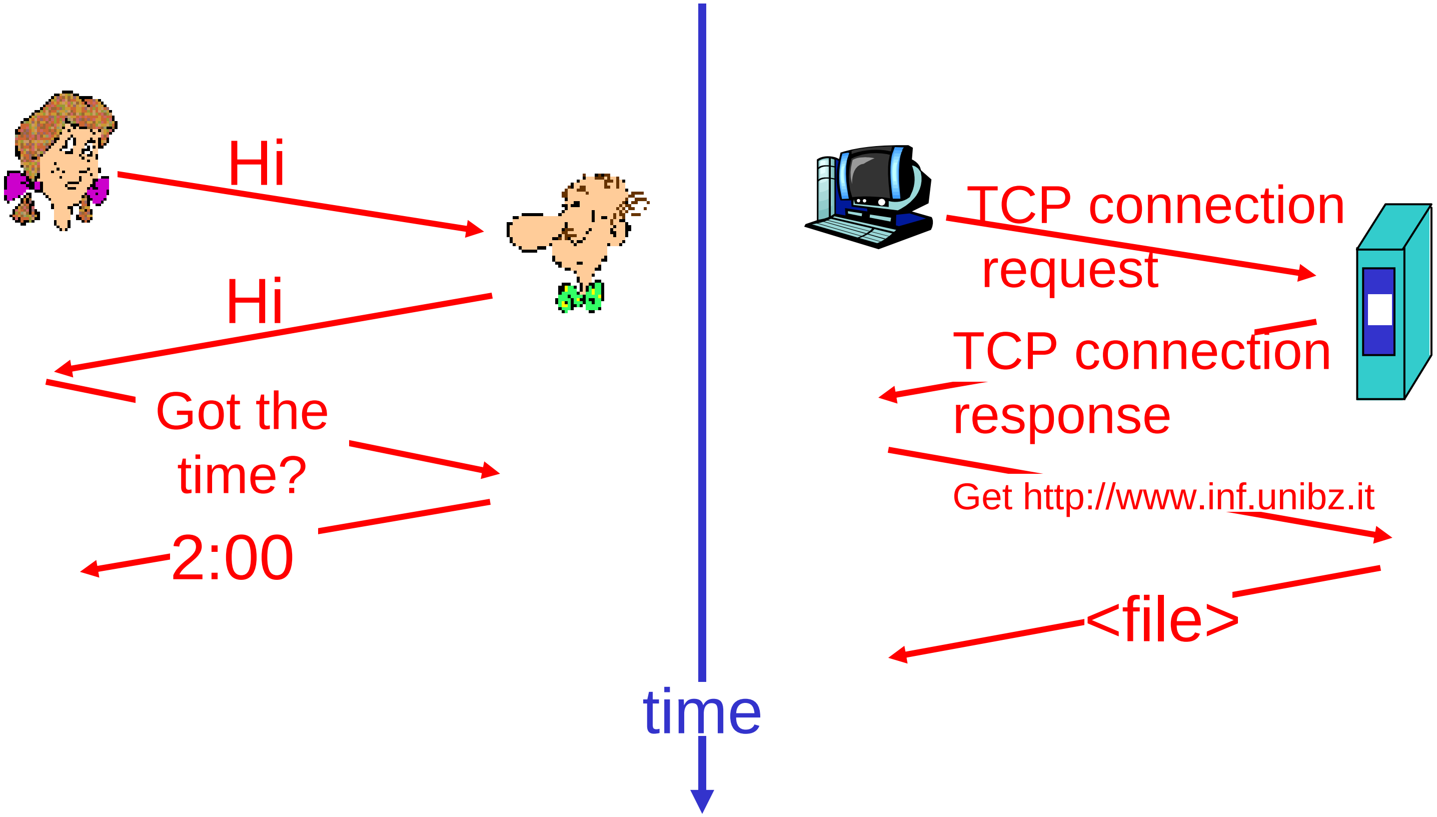
- Set of conventions specifying
  - types and order of messages to be exchanged
  - data formats in messages

- Implemented by pairs of modules for
  - sending
  - receiving messages

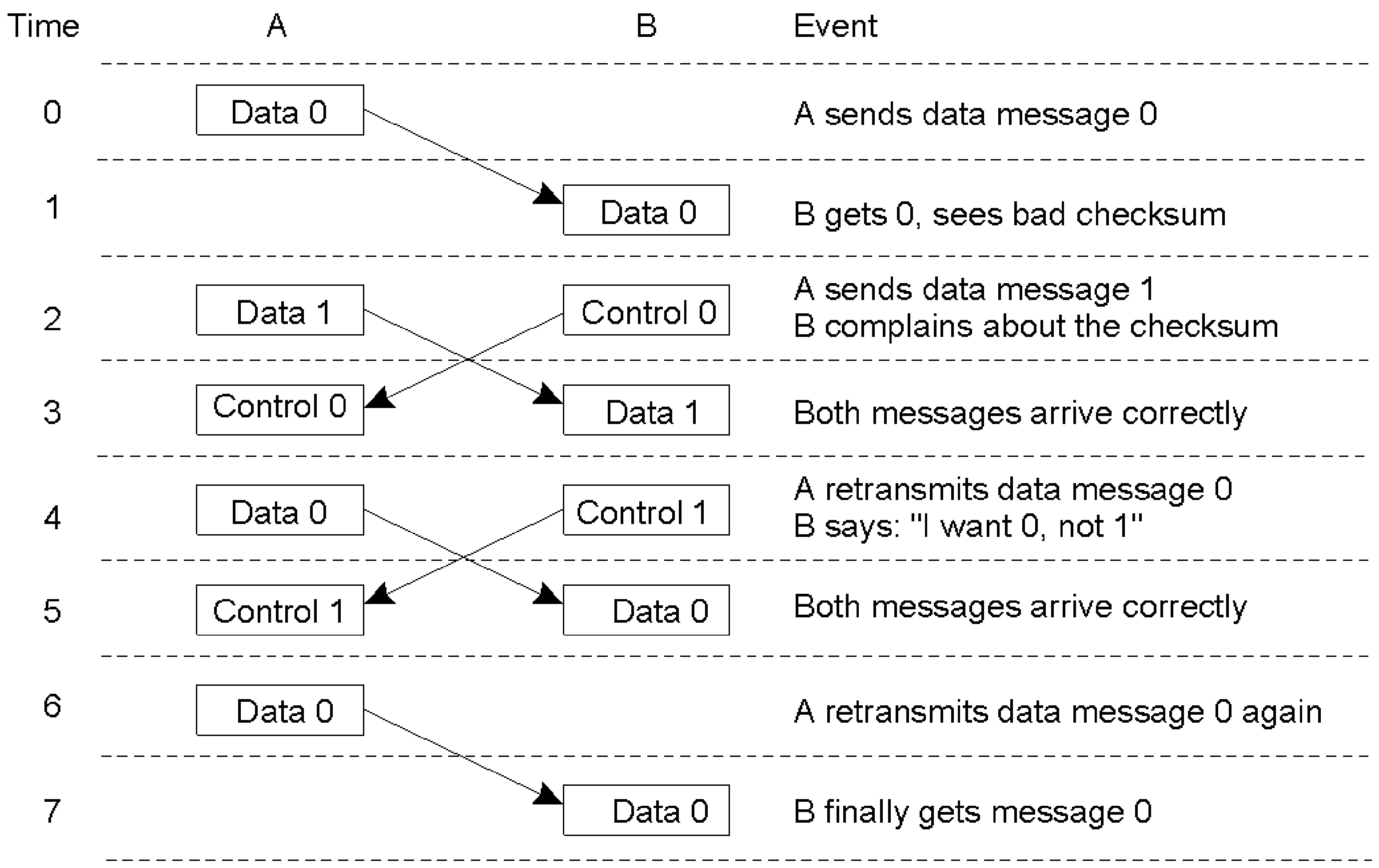
- Arranged in layers
  - protocol suite/stack (= complete set of layers)

What is a Protocol?

A human protocol and a computer network protocol:

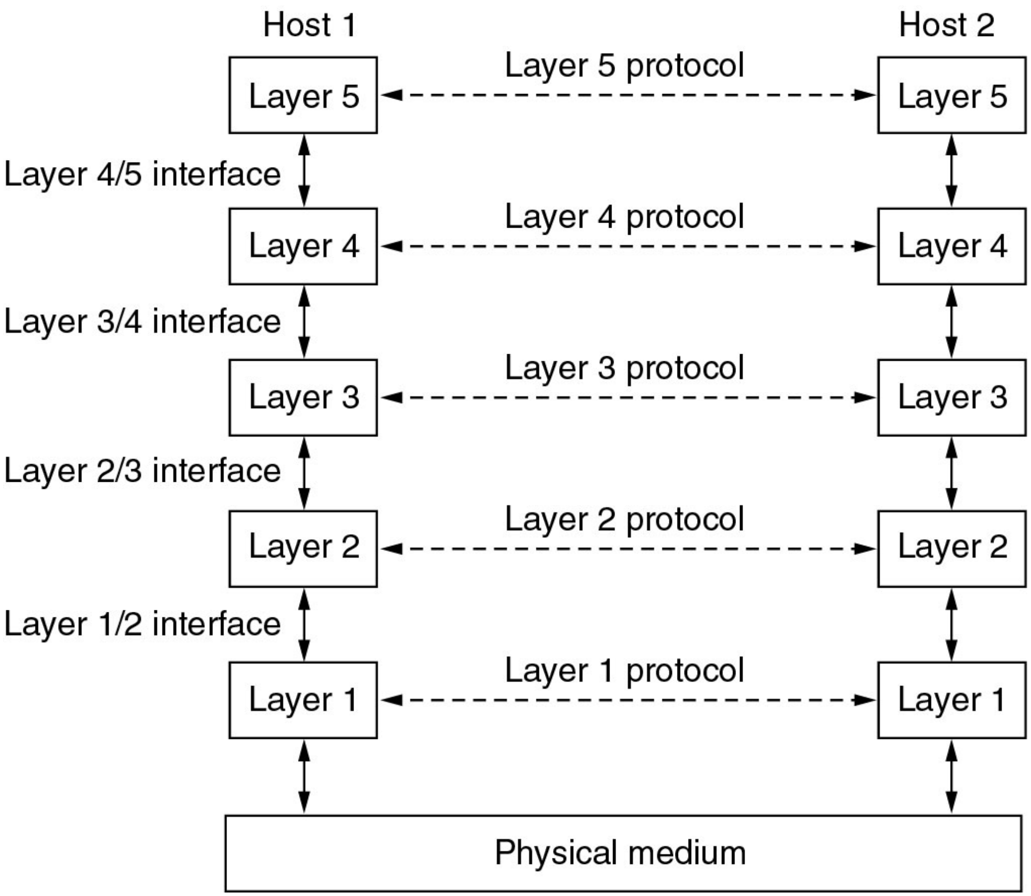


Fictitious Interaction Between Sender and Receiver



Time	A	B	Event
0	Data 0		A sends data message 0
1		Data 0	B gets 0, sees bad checksum
2	Data 1	Control 0	A sends data message 1 B complains about the checksum
3	Control 0	Data 1	Both messages arrive correctly
4	Data 0	Control 1	A retransmits data message 0 B says: "I want 0, not 1"
5	Control 1	Data 0	Both messages arrive correctly
6	Data 0		A retransmits data message 0 again
7		Data 0	B finally gets message 0

Protocols, Layers, Interfaces

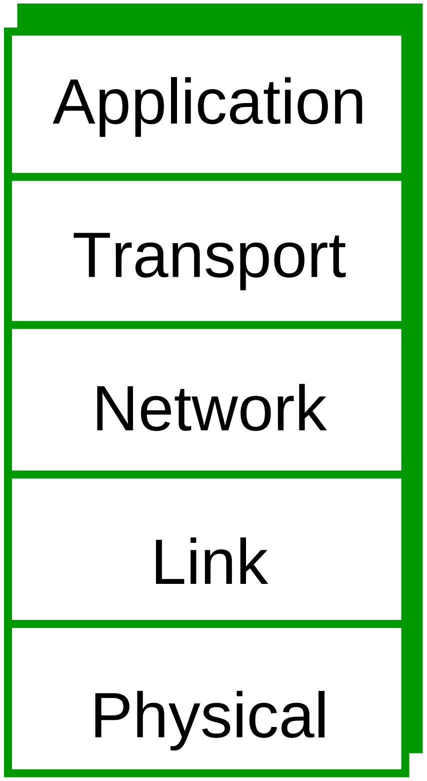


Protocol Stack

- Actual communication takes place only through the physical medium
- Each protocol layer offers virtual communication
- Interfaces define operations and services offered to higher layer

Internet Protocol Stack

- Application: supporting network applications
  - FTP, SMTP, HTTP
- Transport: process-process data transfer
  - TCP, UDP
- Network: routing of datagrams from source to destination
  - IP, routing protocols
- Link: data transfer between neighboring network elements
  - PPP, Ethernet
- Physical: bits “on the wire”



Application

Transport

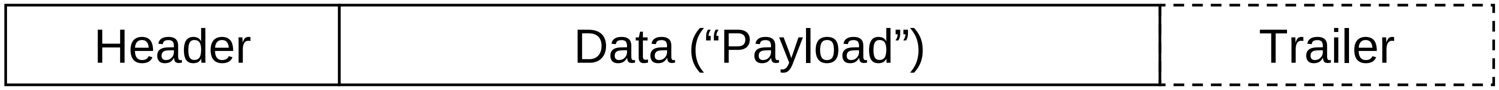
Network

Link

Physical

Information Flow Through Protocol Stack

- Each protocol specifies the structure of its packets

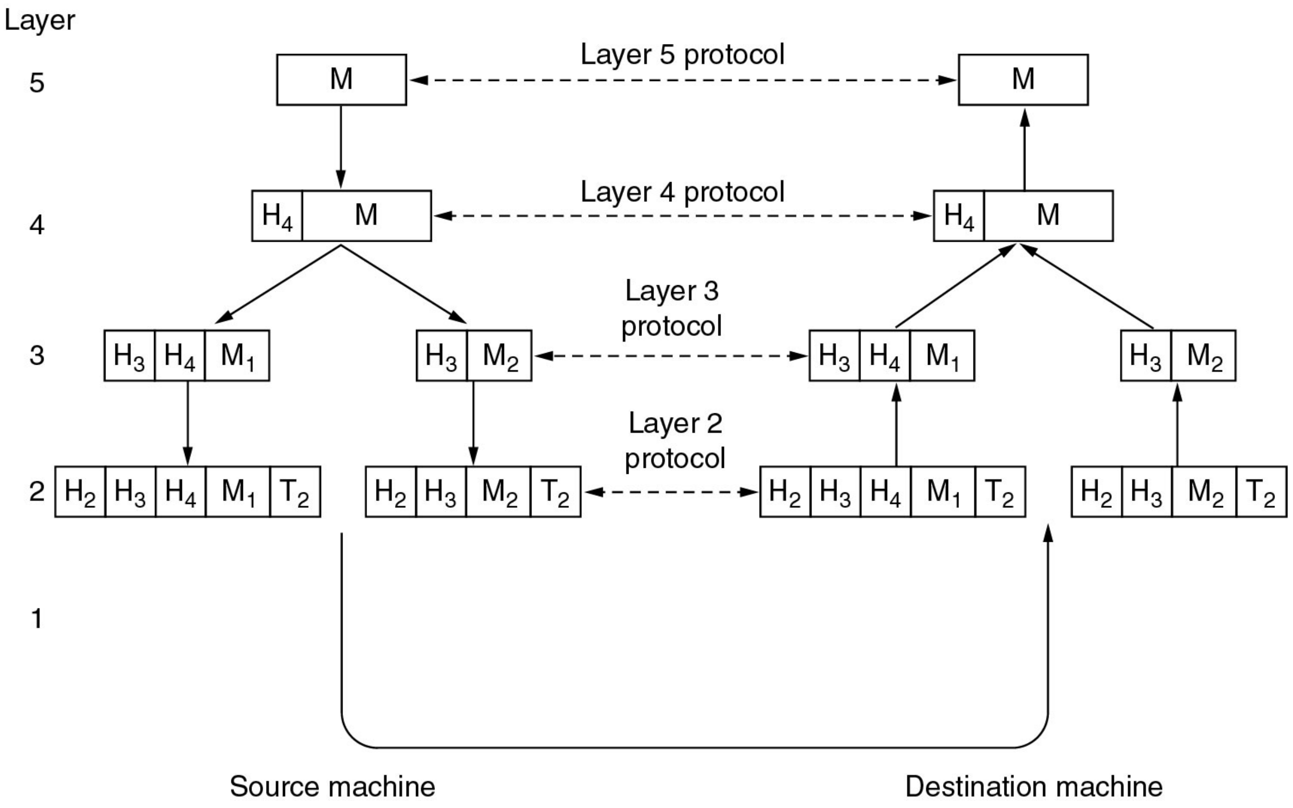


Header

Data (“Payload”)

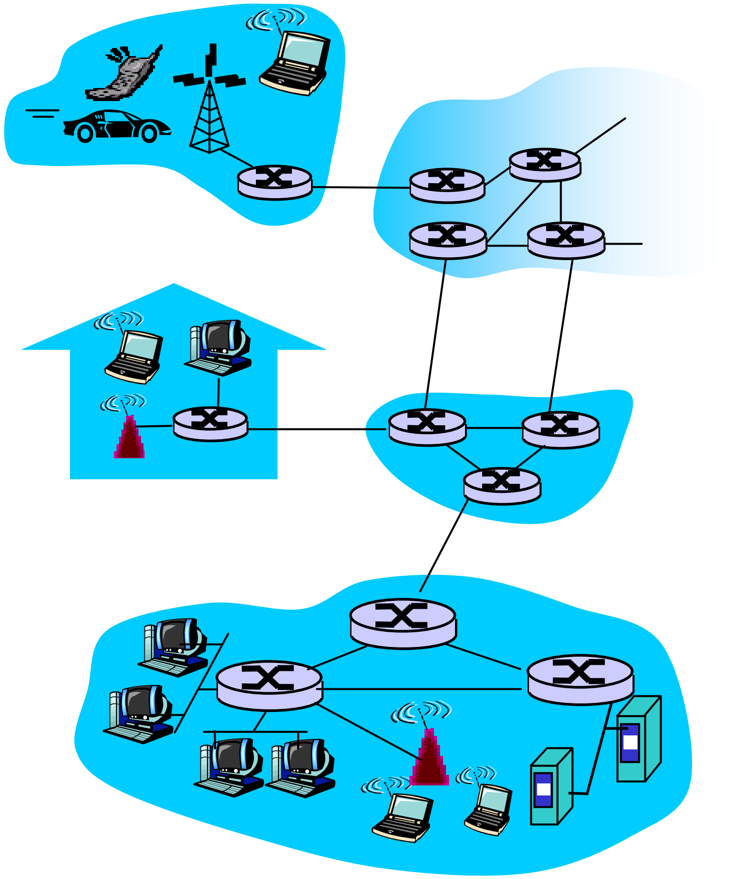
Trailer

- Lower level packets carry higher level packets as their “payload”



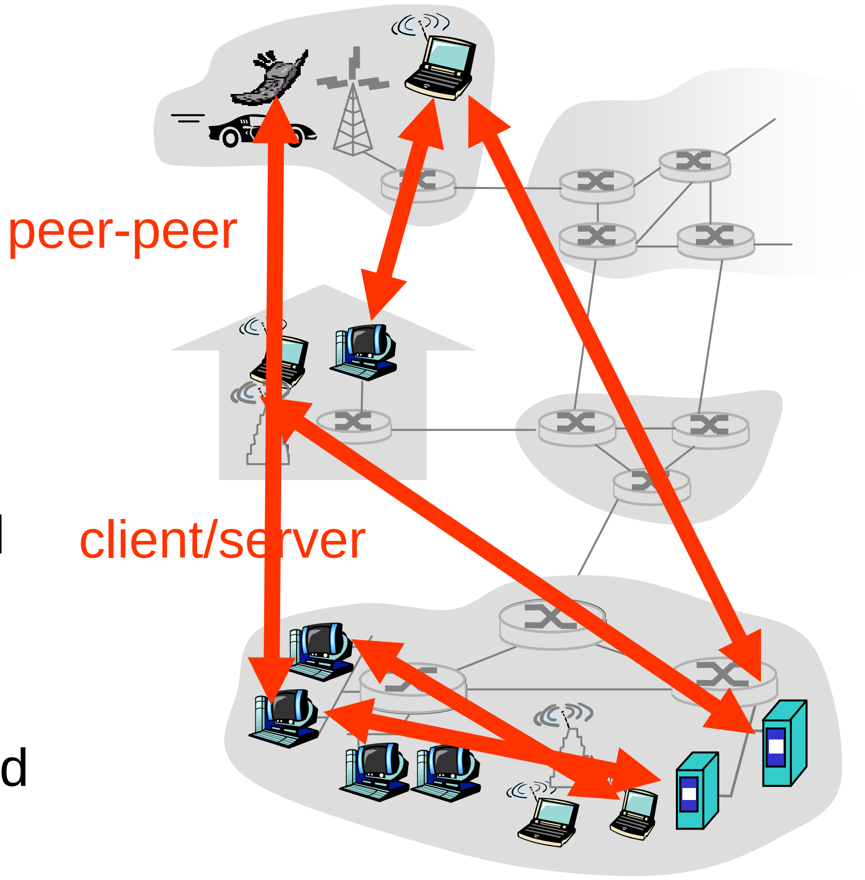
A Closer Look at Network Structure:

- Network edge: applications and hosts
- Access networks, physical media: wired, wireless communication links
- Network core:
  - interconnected routers
  - network of networks



The Network Edge

- End systems (hosts):
  - run application programs
  - e.g. Web, email
  - at “edge of network”
- Client/server model
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
- Peer-peer model:
  - minimal (or no) use of dedicated servers
  - e.g. Skype, Gnutella



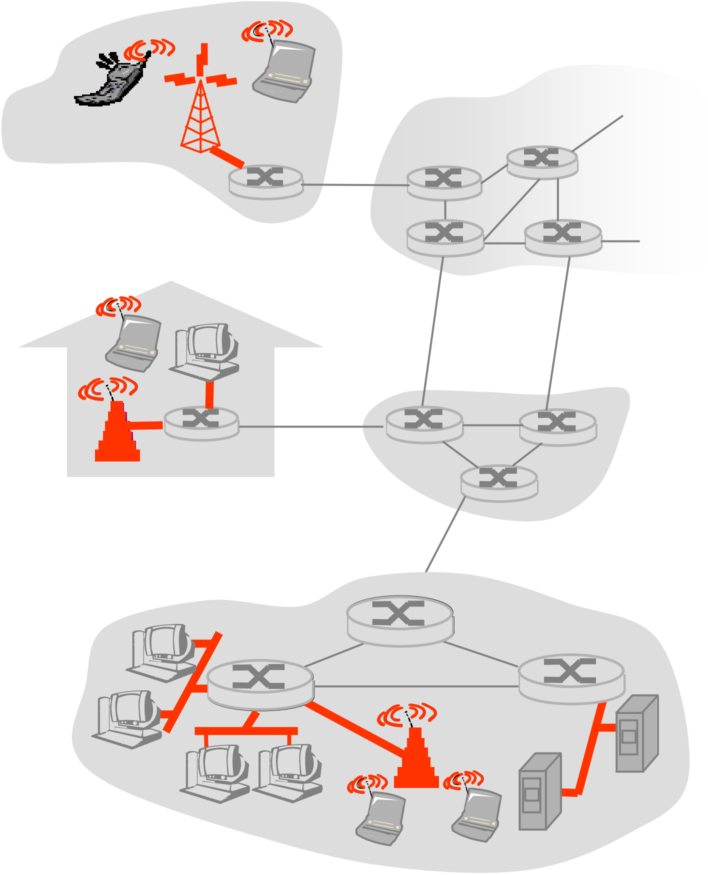
peer-peer

client/server

Access Networks and Physical Media

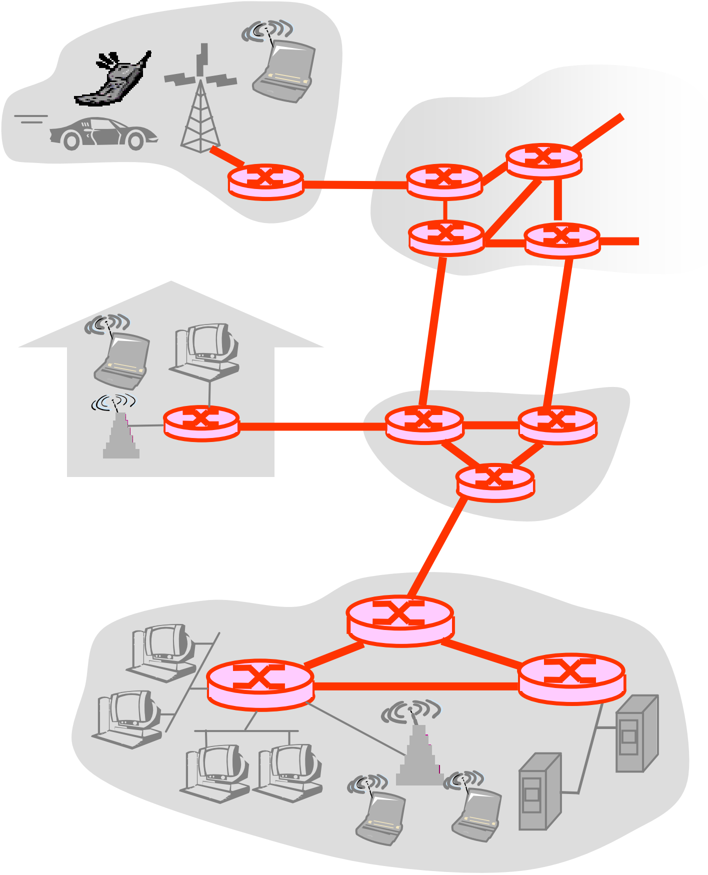
How to connect end systems to edge router?

- residential access nets
- institutional access networks (school, company)
- mobile access networks



The Network Core

- Mesh of interconnected routers
- *Fundamental question*: how is data transferred through net?
  - circuit switching: dedicated circuit per call: telephone net
  - packet-switching: data sent thru net in discrete “chunks”



The World Wide Web

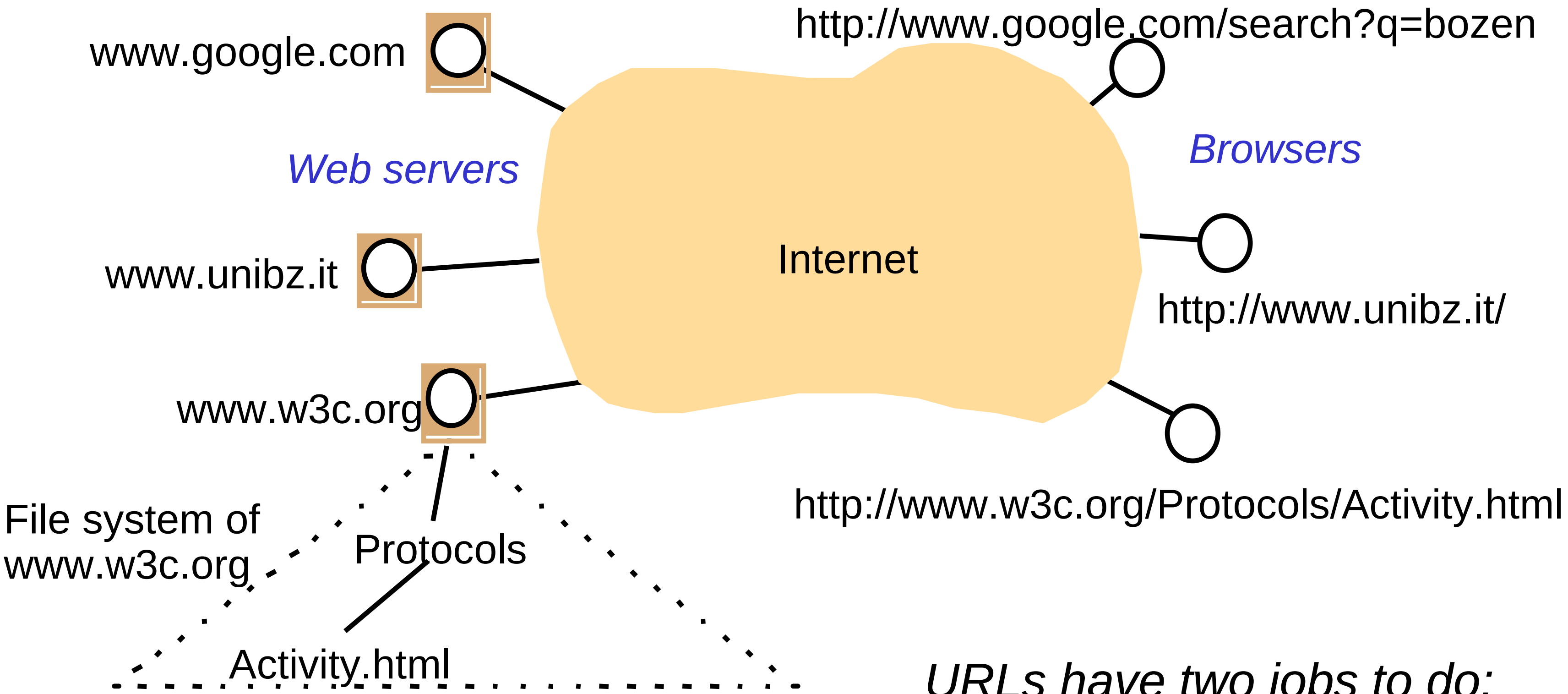
System for publishing and accessing resources across the internet

Open system

- The WWW can be reimplemented and extended in many ways
  - many browsers, web servers with increasing capabilities
  - on all kinds of platforms
- Increasing number of resource types can be published
  - data
  - services

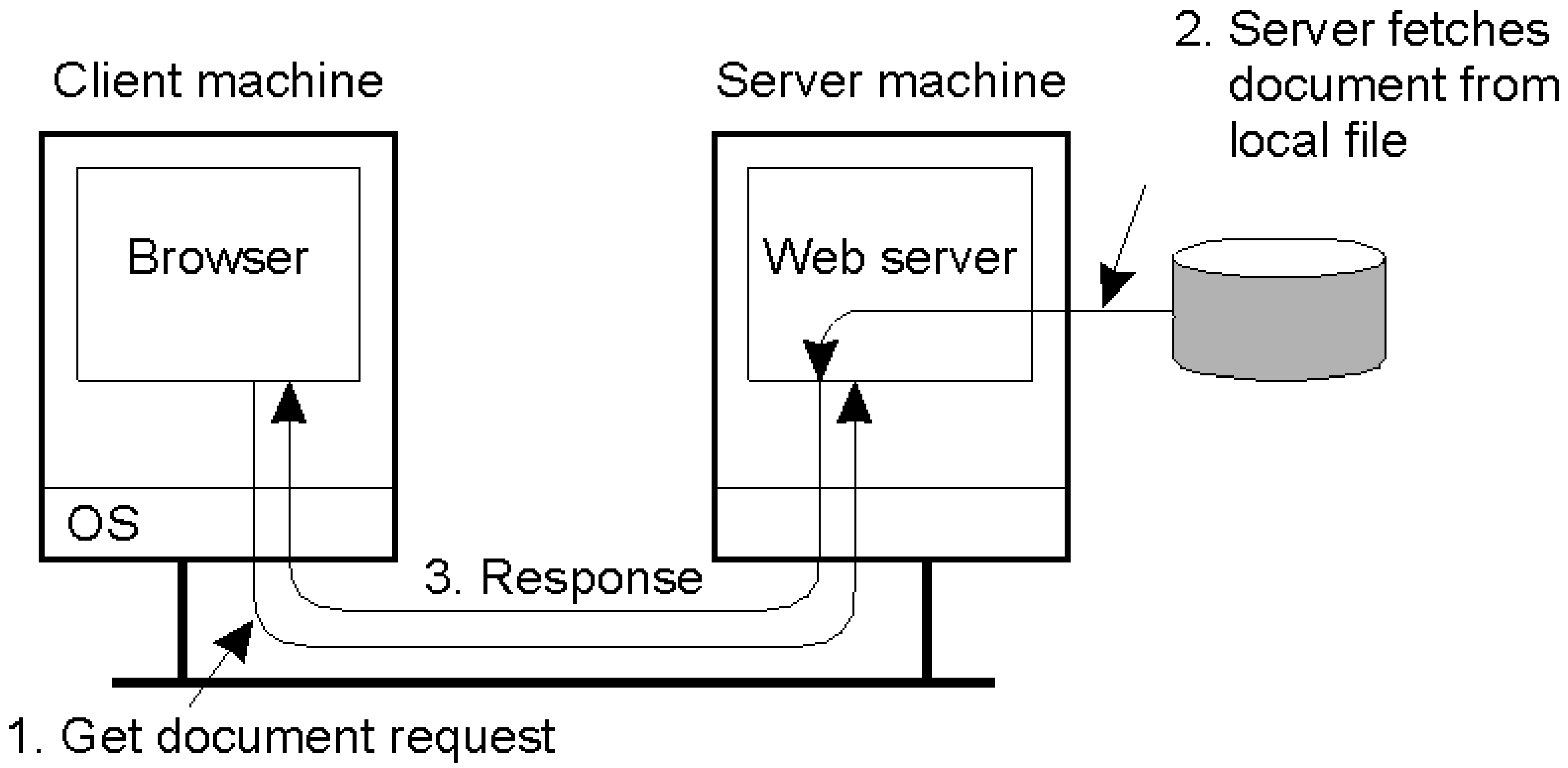
- Basis: Document Model: HTML + XML + Internet Media Types (MIME)
  Resource Location: URLs
  Interaction: HTTP

Web Servers and Web Browsers



URLs have two jobs to do:
- *identify web server*
- *identify resource*

The Basic Web Mechanism



Refinements exist both at the client and the server side

HTML Document with Head

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=iso-8859-1">
   <meta NAME="GENERATOR" CONTENT="Mozilla/4.7 (Macintosh; I; PPC) [Netscape]">
   <title>Werner Nutt</title>
</head>

<body TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#3333FF"
      VLINK="#3366FF"ALINK="#CC3232">
<center><a href="index.html"><img SRC="Pictures/WernerNutt.jpg"
      ALT="WN" BORDER=0 HEIGHT=100 WIDTH=270></a>
<br>&nbsp;
<br>&nbsp;
<br>&nbsp;
<a href="http://www.provincia.bz.it/wetter/suedtirol.htm">
<img SRC="http://www.provincia.bz.it/wetter/images/m_suedtirol_deutsch.jpg"
     ALT="Weather" BORDER=0></a>
</center>

</body>
</html>
```

Document Model: HTML

HTML is a markup language for hypertext documents

Mobile code can be embedded as JavaScript or Java Applet

```
<HTML>                                  <!- Start of HTML document    -->
<BODY>                                  <!- Start of the main body    -->
<H1>Hello World/H1>                     <!- Basic text to be displayed   -->
<P>                                     <!- Start of a new paragraph   -->
<SCRIPT type = "text/javascript">       // Identify scripting language
alert ('Hello World');                  // Create Alert Window
</SCRIPT>                               <!- End of scripting section     -->
</P>                                    <!- End of paragraph section -->
</BODY>                                 <!- End of main body             -->
</HTML>                                 <!- End of HTML section          -->
```

Document Model: XML (1)

An XML document using the tags defined in the DTD

```
<?xml = version "1.0">
<!DOCTYPE article SYSTEM "article.dtd">
<article>
    <title> Prudent Engineering Practice for Cryptographic Protocols</title>
    <author><name>M. Abadi</name></author>
    <author><name>R. Needham</name></author>
    <journal>
            <jname>IEEE Transactions on Software Engineering</jname>
            <volume>22</volume>
            <number>12</number>
            <month>January</month>
            <pages>6 – 15</pages>
            <year>1996</year>
    </journal>
</article>
```

Document Model: XML (2)

In XML, new tags can be defined by document type definitions (DTDs) or XML schema declarations

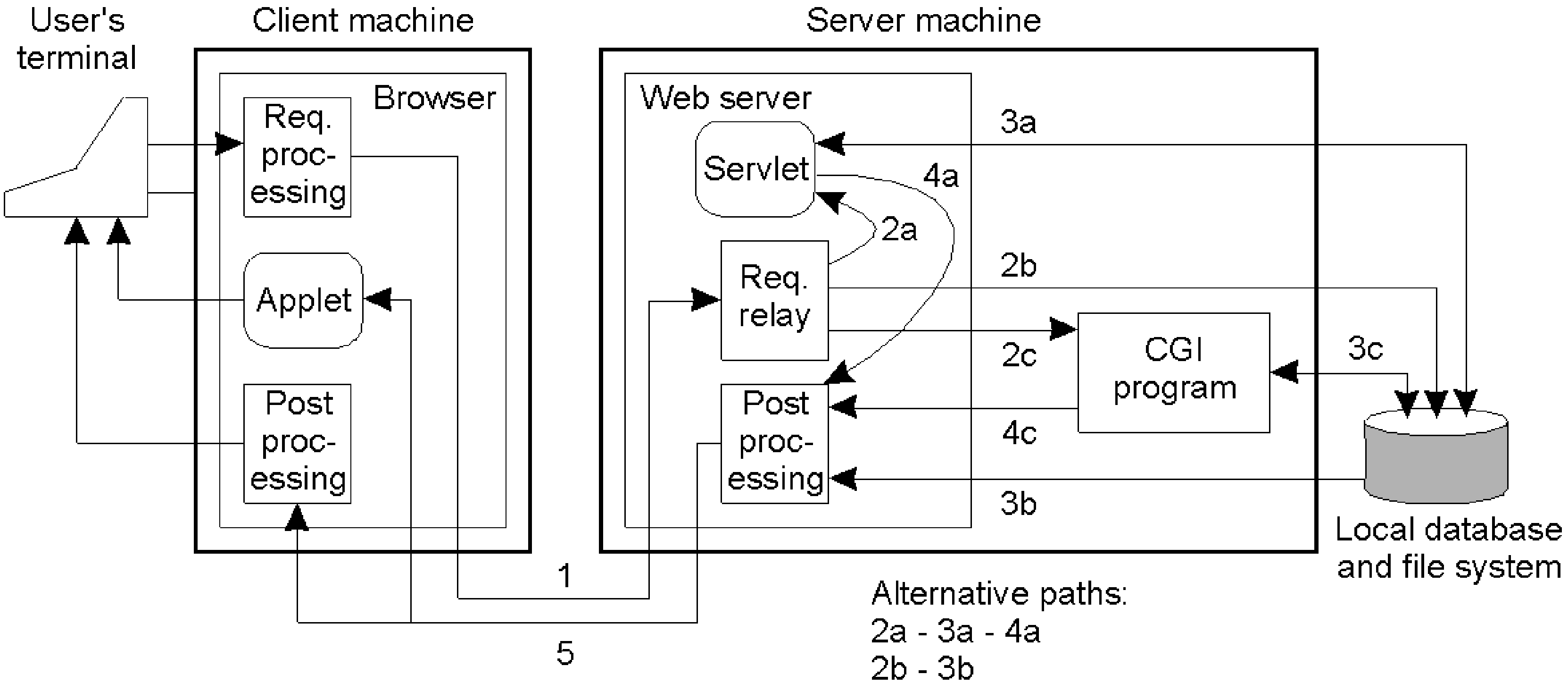
```
(1)     <!ELEMENT article (title, author+,journal)>
(2)     <!ELEMENT title (#PCDATA)>
(3)     <!ELEMENT author (name, affiliation?)>
(4)     <!ELEMENT name (#PCDATA)>
(5)     <!ELEMENT affiliation (#PCDATA)>
(6)     <!ELEMENT journal (jname, volume, number?, month? pages, year)>
(7)     <!ELEMENT jname (#PCDATA)>
(8)     <!ELEMENT volume (#PCDATA)>
(9)     <!ELEMENT number (#PCDATA)>
(10)    <!ELEMENT month (#PCDATA)>
(11)    <!ELEMENT pages (#PCDATA)>
(12)    <!ELEMENT year (#PCDATA)>
```

Document Model: MIME Types

Type	**Subtype**	**Description**
Text	Plain	Unformatted text
	HTML	Text including HTML markup commands
	XML	Text including XML markup commands
Image	GIF	Still image in GIF format
	JPEG	Still image in JPEG format
Audio	Basic	Audio, 8-bit PCM sampled at 8000 Hz
	Tone	A specific audible tone
Video	MPEG	Movie in MPEG format
	Pointer	Representation of a pointer device for presentations
Application	Octet-stream	An uninterrupted byte sequence
	Postscript	A printable document in Postscript
	PDF	A printable document in PDF
Multipart	Mixed	Independent parts in the specified order
	Parallel	Parts must be viewed simultaneously

Six top-level MIME types (= internet media types)
and some subtypes

The Web Mechanism: Details



Alternative paths:
2a - 3a - 4a
2b - 3b
2c - 3c - 4c

URLs

- URL = *Uniform Resource Locator*

  standard mechanism to identify resources

  exists for several access schemes (protocols)

- Syntax:

  <scheme> : <scheme dependent info>

- Examples:

  http://www.isoc.org/internet/history/

  mailto:rossi@inf.unibz.it

  ftp://nutt:blabla@inf.unibz.it

  ftp://ftp.phys.ufl.edu/

URLs

- Hyper Text Transfer Protocol (HTTP) for Web

  http:// *host*[:*port*][/*path*][?*arguments*]

- File Transfer Protocol (FTP)

  ftp://[*user*[:*password@*]]*host*[:*port*][*path*]

- OS accessible files (file)

  file://*host/path*

URLs

- Email (mailto)

  mailto:*account@site*

- Telnet (telnet)

  telnet://*user@host*

- Others:

  news and NNTP (newsgroups),

  jar (Java jar files), imap (mail servers)

Browser

– generates hypertext view

– or calls helper application

HTTP

- Request-reply protocol for transferring documents
- Documents
  - may be of different types (MIME types)
  - may have embedded documents
- Several types of request messages
- One resource per request

  one web page may imply *many requests*,

  e.g., one for the html and one for every image

APIs in various programming languages, e.g.,
class **`HttpURLConnection`** in **`java.net`**

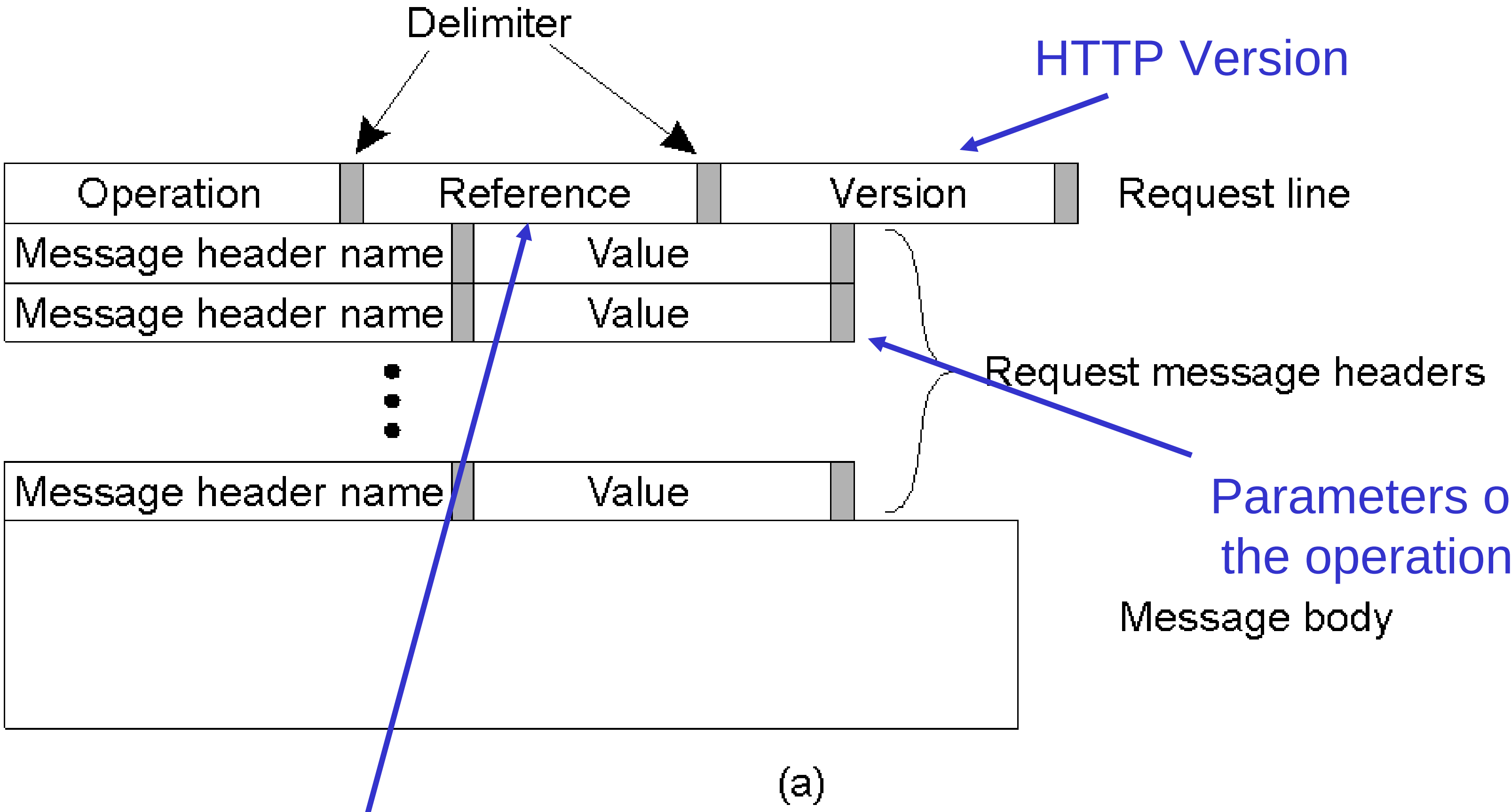
HTTP Operations

Commonly used HTTP requests

Operation	**Description**
Head	Request to return the header of a document
Get	Request to return a document to the client
Put	Request to store a document
Post	Provide data that is to be added to a document (collection)
Delete	Request to delete a document

Requests are part of messages

HTTP Request Message



HTTP Version

Parameters of the operation

Reference to document (on server)

HTTP Conversation: Request

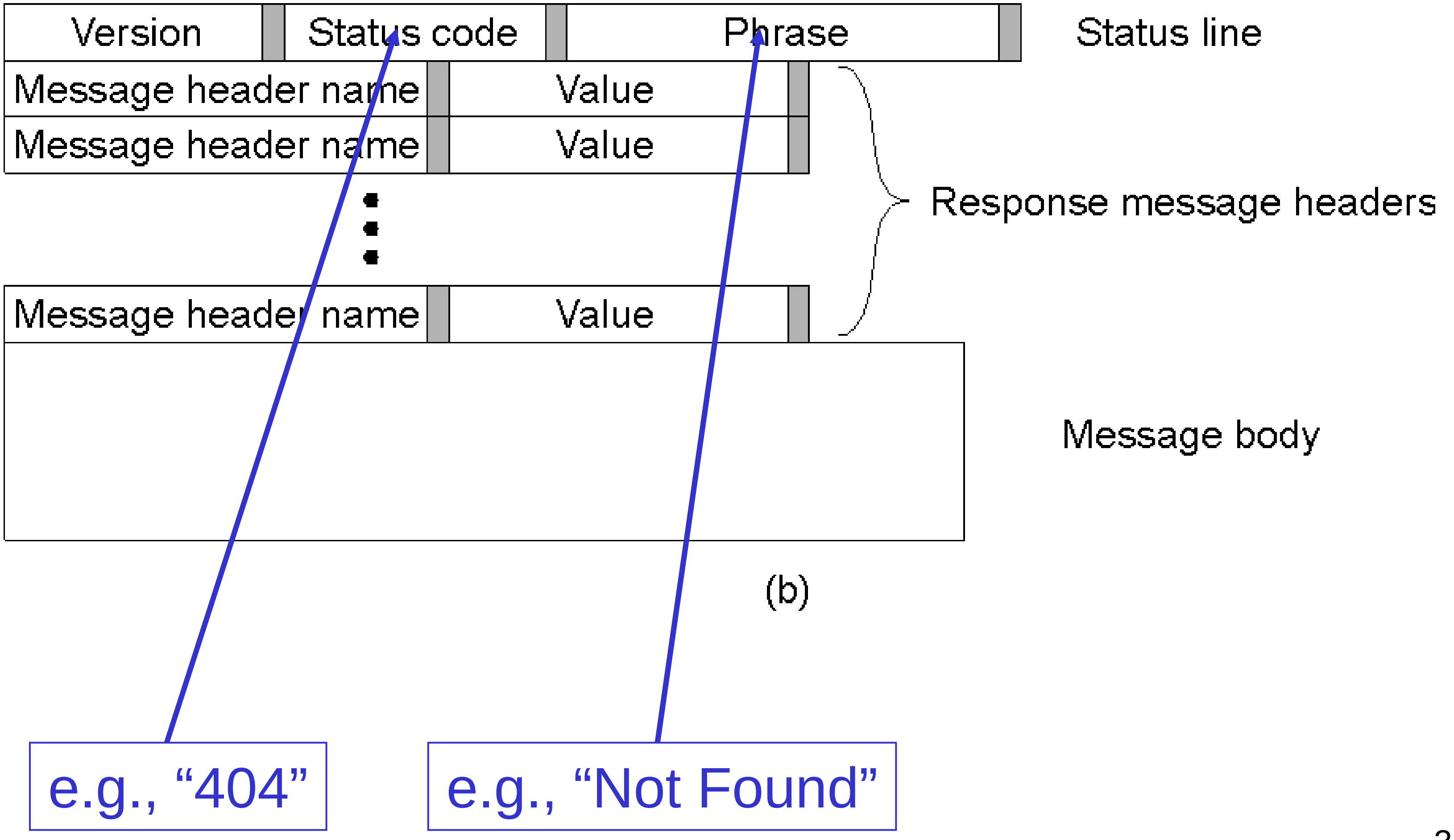
Client:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Some HTTP Message Headers

Header	**Source**	**Contents**
Accept	Client	The type of documents the client can handle
Accept-Language	Client	The natural language the client can handle
WWW-Authenticate	Server	Security challenge the client should respond to
Date	Both	Date and time the message was sent
Expires	Server	The time how long the response remains valid
Host	Client	The TCP address of the document's server
If-Modified-Since	Client	Tells the server to return a document only if it has been modified since the specified time
Last-Modified	Server	The time the returned document was last modified
Location	Server	A document reference to which the client should redirect its request
Referer	Client	Refers to client's most recently requested document
Upgrade	Both	The application protocol the sender wants to switch to
Warning	Both	Information about the status of the data in the message

HTTP Response Message



HTTP Conversation: Reply

Server:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2009 22:38:34 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2007 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

[HTML Document]

References

In preparing the lectures I have used several sources.

The main ones are the following:

Books:

- Coulouris, Dollimore, Kindberg. Distributed Systems – Concepts and Design (CDK)
- Tannenbaum, van Steen. Distributed Systems – Principles and Paradigms
- Kurose/Ross. Computer Networking: A Top-Down Approach

Slides:

- Kurose/Ross, Material for lecturers
- Andrew Tanenbaum, Slides from his website
- CDK Website