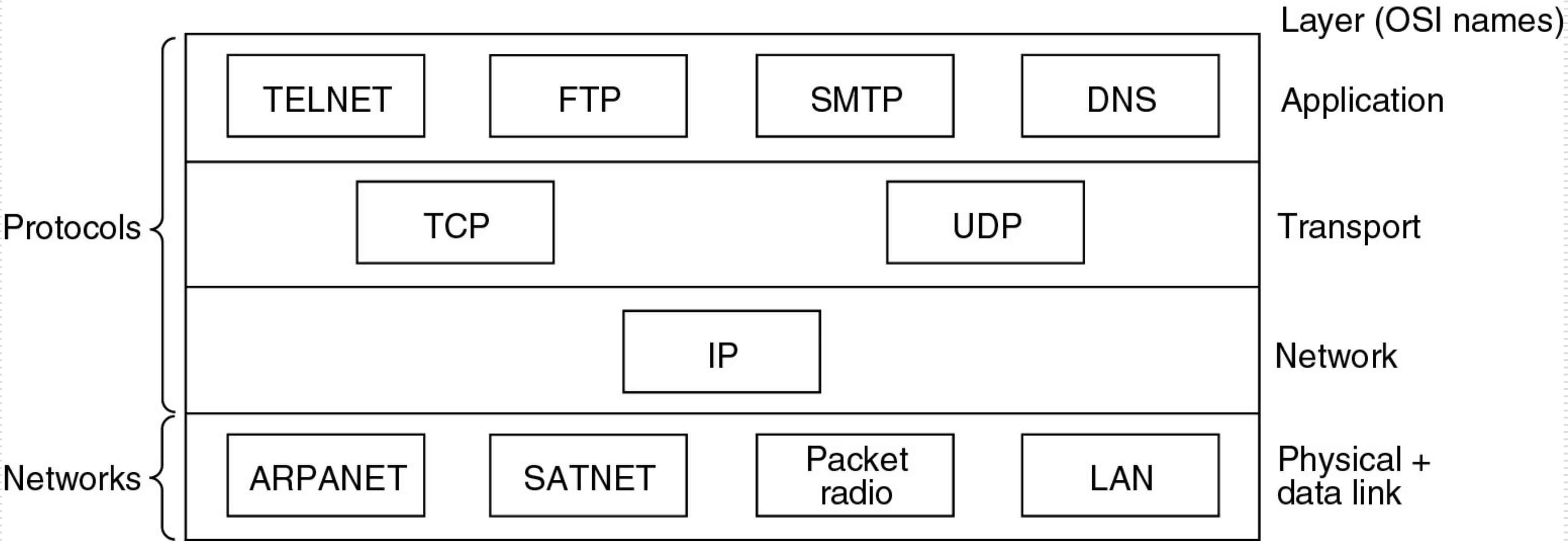
SCS 2105
Computer Networks I

Application Layer Protocols

Dr. Ajantha Atukorale
University of Colombo School of Computing (UCSC)

TCP/IP Suit



Layer (OSI names)

Application: TELNET FTP SMTP DNS

Transport: TCP UDP

Network: IP

Physical + data link: ARPANET SATNET Packet radio LAN

Protocols

Networks

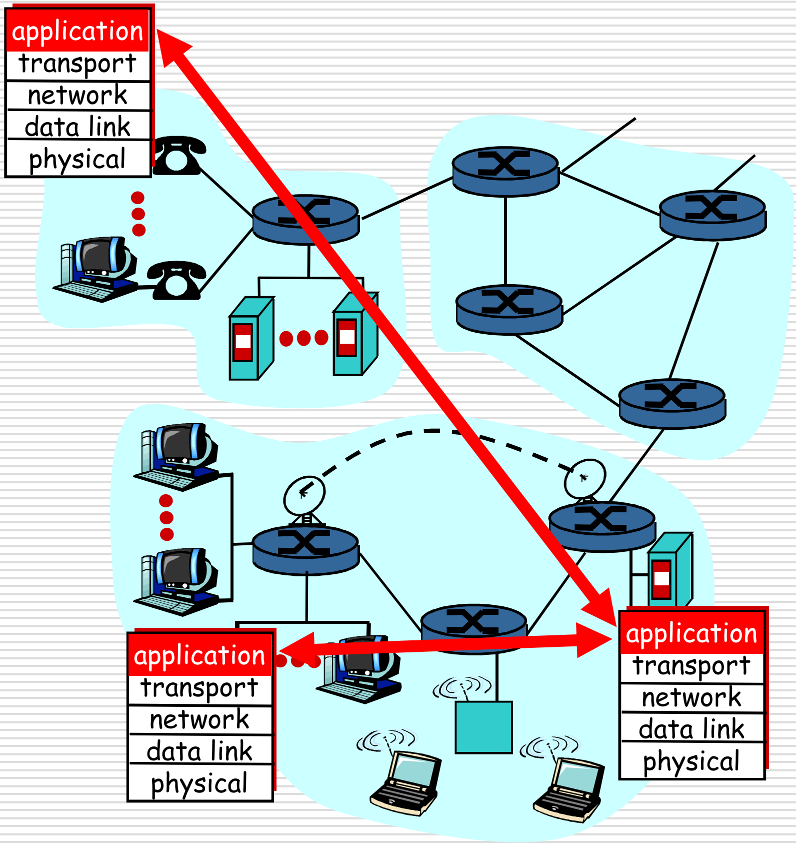
Applications and application-layer protocols

Application: communicating, distributed processes

- running in network hosts in "user space"
- exchange messages to implement application
- e.g., email, ftp, Web

Application-layer protocols

- one "piece" of an app
- define messages exchanged by apps and actions taken
- use communication services provided by lower layer protocols (TCP, UDP)
- SMTP, FTP, HTTP



* How to identify process?
IP address and Port number

Client-server paradigm

Client:

□ initiates contact with server ("speaks first")

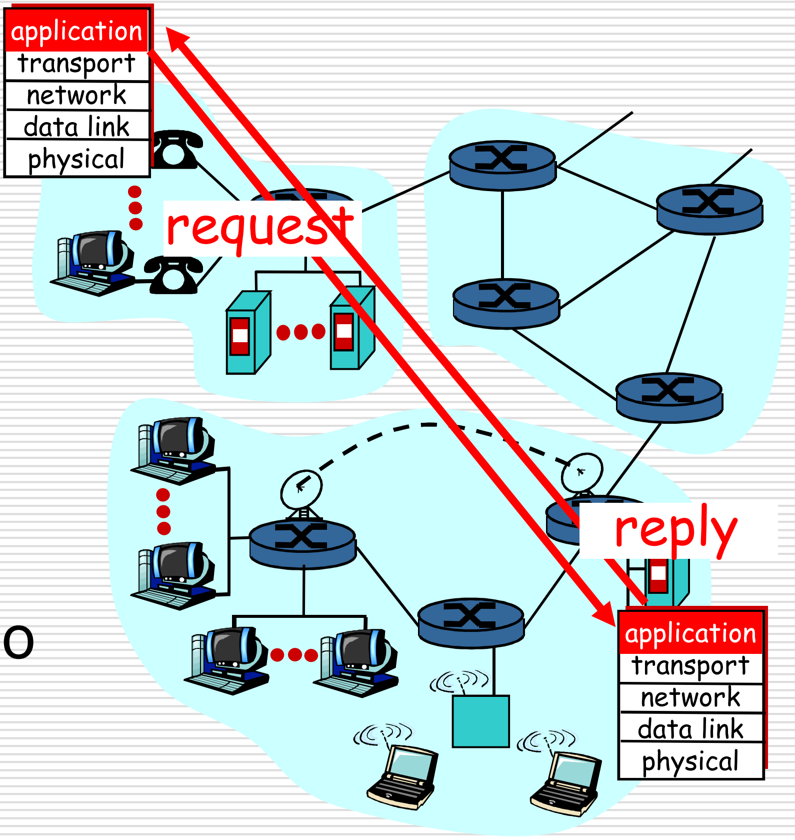
□ typically requests service from server

□ Web: client implemented in browser

Server:

□ provides requested service to client

□ e.g., Web server sends requested Web page



Peer-to-Peer ?

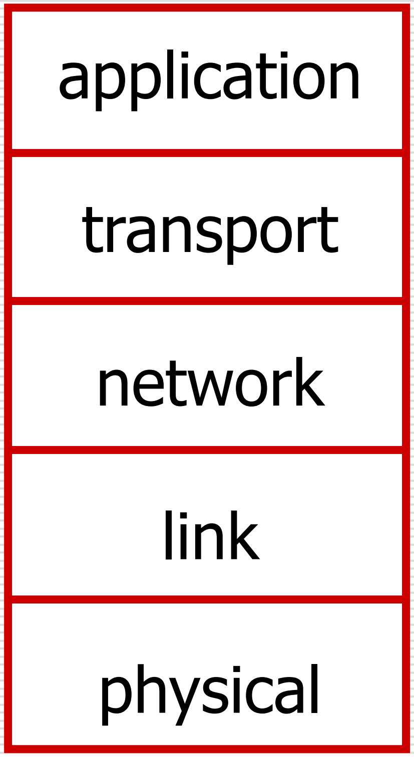
Application-layer protocols

HTTP, SMTP, FTP, Telnet, Proprietary Protocols, …

Application-layer protocols are implemented using Socket API which is provided by Operating System.

API: application programming interface

- □ defines interface between application and transport layers
- □ socket: Internet API
  - ■ two processes communicate by sending data into socket, reading data out of socket



application

transport

network

link

physical

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"

Bandwidth

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

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100's msec
financial apps	no loss	elastic	yes and no

Internet transport protocols services

TCP service:

- □ *connection-oriented:* setup required between client, server
- □ *reliable transport* between sending and receiving process
- □ *flow control:* sender won't overwhelm receiver
- □ *congestion control:* throttle sender when network overloaded
- □ *does not providing:* timing, minimum bandwidth guarantees

UDP service:

- □ unreliable data transfer between sending and receiving process
- □ does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NSF	TCP or UDP
Internet telephony	proprietary (e.g., Vocaltec)	typically UDP

Simple Mail Transfer Protocol

- Basic protocol for email exchange over the Internet
- Fundamental difference between SMTP and FTP/TELNET is that it is NOT an interactive protocol
  - Messages are queued and spooled by SMTP agent
- Users interact with email application
  - E.g. Microsoft Outlook Express!
- Application interfaces with Message Transfer Agent
  - *Sendmail* on UNIX
  - Setup and configured by admins.
- SMTP specifies how MTA's pass email across the Internet
  - Also uses NVT commands

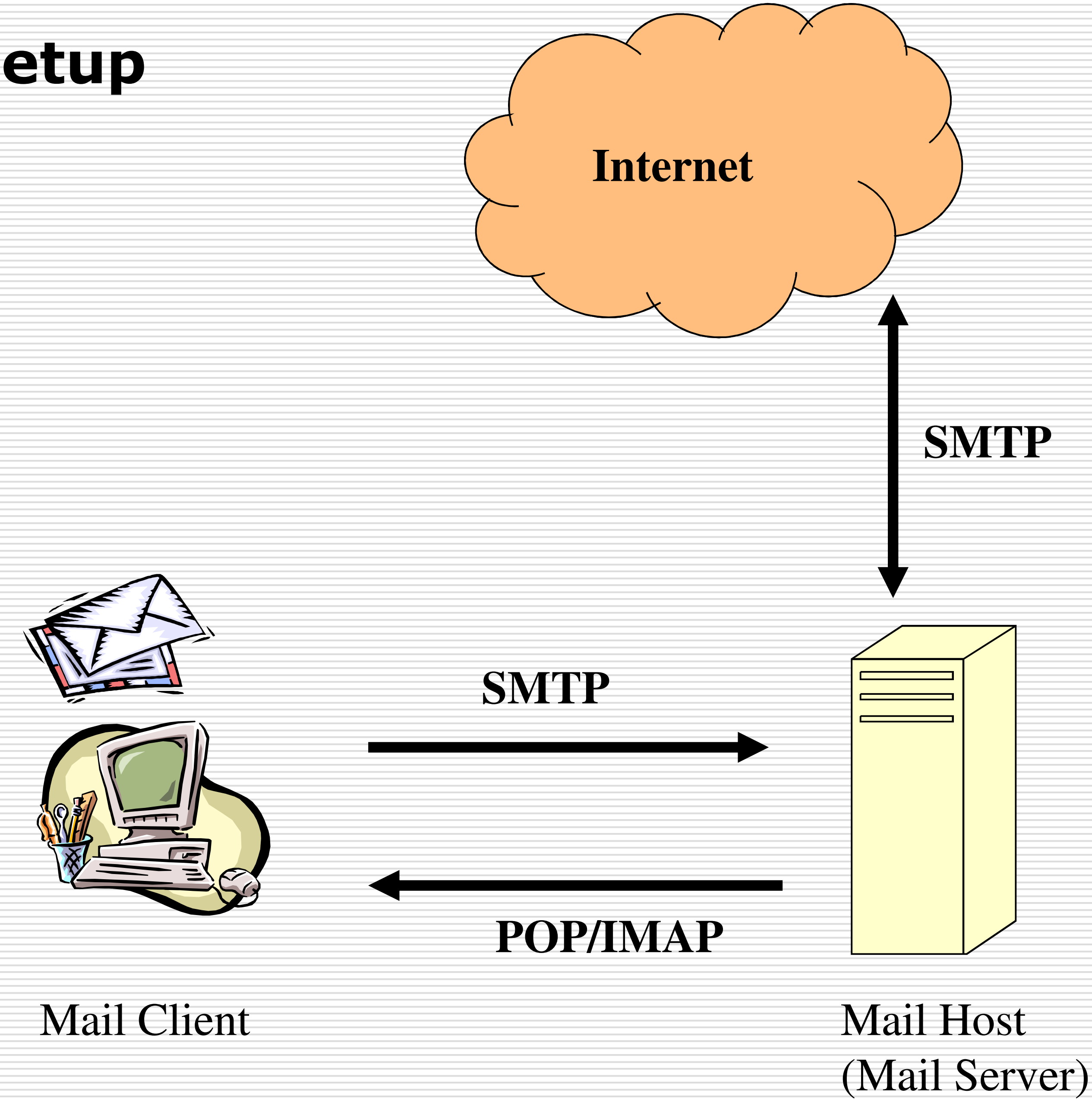
Simple Mail Transfer Protocol Contd.

- Client uses email application to construct and send messages
- Message is passed to mail spooler which is part of MTA
  - Application communicates with MTA via email transfer protocol
    - Post Office Protocol (POP3) is common, but not very secure
    - Our department uses IMAP
- MTA's on remote systems listen for incoming mail on well known port (25)
- Messages are delivered in two parts – header and body
  - Header format has exact specification (RFC 822)
  - Body content types are specified by MIME

SMTP Commands

- **helo** (Hello) Identify the SMTP sender to the SMTP receiver.
- **mail** (Mail) Start an e-mail transaction to deliver the e-mail to one or more recipients.
- **rcpt** (Recipient) Identify an individual recipient of e-mail.
- **data** (Data) Consider the lines following the command to be e-mail from the sender.
- **send** (Send) Deliver e-mail to one or more work stations.
- **soml** (Send or mail) Deliver e-mail to one or more work stations or recipients if the
- user is not active.
- **saml** (Send and mail) Deliver e-mail to one or more work stations and recipients if the
- user is not active.
- **rset** (Reset) End the current e-mail transaction.
- **vrfy** (Verify) Ask the receiver to confirm that a user has been identified.
- **expn** (Expand) Ask the receiver to confirm that a mailing list has been identified.
- **help** (Help) Ask the receiver to send helpful information to the sender.
- **noop** (Noop) Ask the receiver to send a valid reply (but specify no other action).
- **quit** (Quit) Ask the receiver to send a valid reply, and then close the transmission channel.
- **turn** (Turn) Ask the receiver to send a valid reply and then become the SMTP sender, or else ask the receiver to send a refusal reply and remain the SMTP receiver.

A Mail Setup


Internet
SMTP
SMTP
POP/IMAP
Mail Client
Mail Host
(Mail Server)

Email Exchange

Major parts involved in an email exchange

1. The server daemon (MTA)
2. A daemon for users to read mail from mailhost (MUA)
3. DNS

Mail server daemons: **sendmail**, **qmail, postfix**, **exim**, **mmdf, smail**, **zmailer** etc.

The server daemon usually has 2 function:

- looks after receiving incoming mail
- delivers outgoing mail

The server daemon does not allow you to read your mail. For this you need an additional daemon (**POP**, **IMAP**, etc).

The DNS and its daemon “**named**” play a large role in the delivery of email.

Hyper Text Transfer Protocol

- Client can make requests
  - GET for requesting a file from the server
  - POST for submitting information to the server
  - When it makes a request, the client also passes some client side descriptors to the server
- Server responds
  - HTTP headers
  - HTML document
    - or JPEG, or GIF, or…
- Browser implements client side of this service
- Web server implements server side of this service

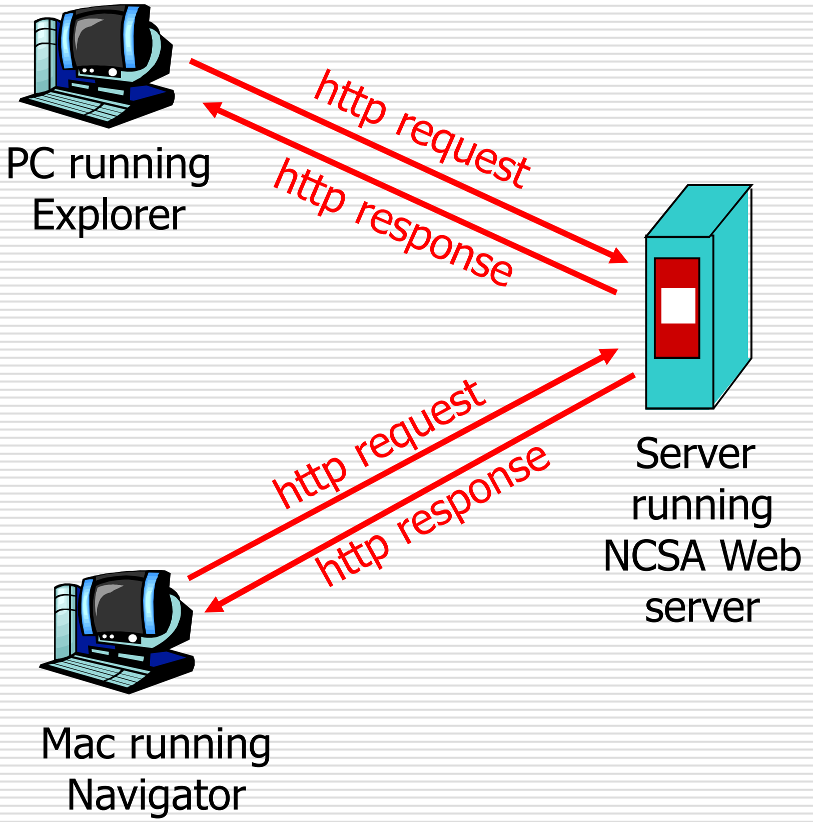
HTTP Request Methods

METHOD	DESCRIPTION
GET	Request to read a web page
HEAD	Request to read a web page’s header
PUT	Request to store web page
POST	Append to a named resource
DELETE	Remove the web page
TRACE	Echo the incoming request
CONNECT	Reserved for future forecast
OPTIONS	Query certain options

The Web: the HTTP protocol

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
- □ http1.0: RFC 1945
- □ http1.1: RFC 2068



The http protocol: more

http: TCP transport service:

- □ 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

http is “stateless“

- □ server maintains no information about past client requests

aside

Protocols that maintain “state“ are complex!

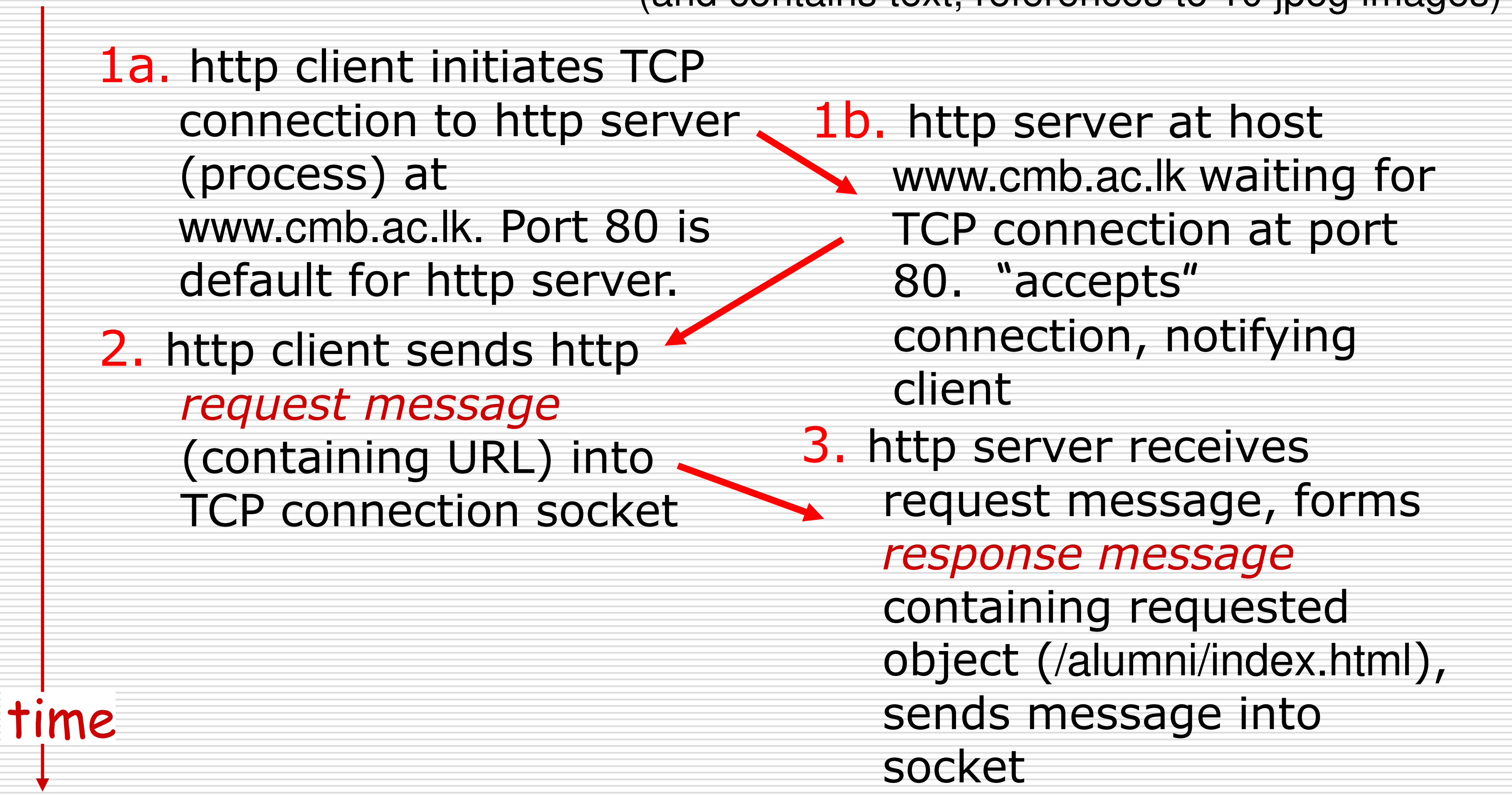
- □ past history (state) must be maintained
- □ if server/client crashes, their views of “state“ may be inconsistent, must be reconciled

http example

Suppose user enters URL

http://www.cmb.ac.lk/alumni/index.html

(and contains text, references to 10 jpeg images)



1a. http client initiates TCP connection to http server (process) at www.cmb.ac.lk. Port 80 is default for http server.

1b. http server at host www.cmb.ac.lk waiting for TCP connection at port 80. "accepts" connection, notifying client

2. http client sends http *request message* (containing URL) into TCP connection socket

3. http server receives request message, forms *response message* containing requested object (/alumni/index.html), sends message into socket

http example (cont.)

4. http server closes TCP connection.

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

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

Non-persistent, persistent connections

Non-persistent

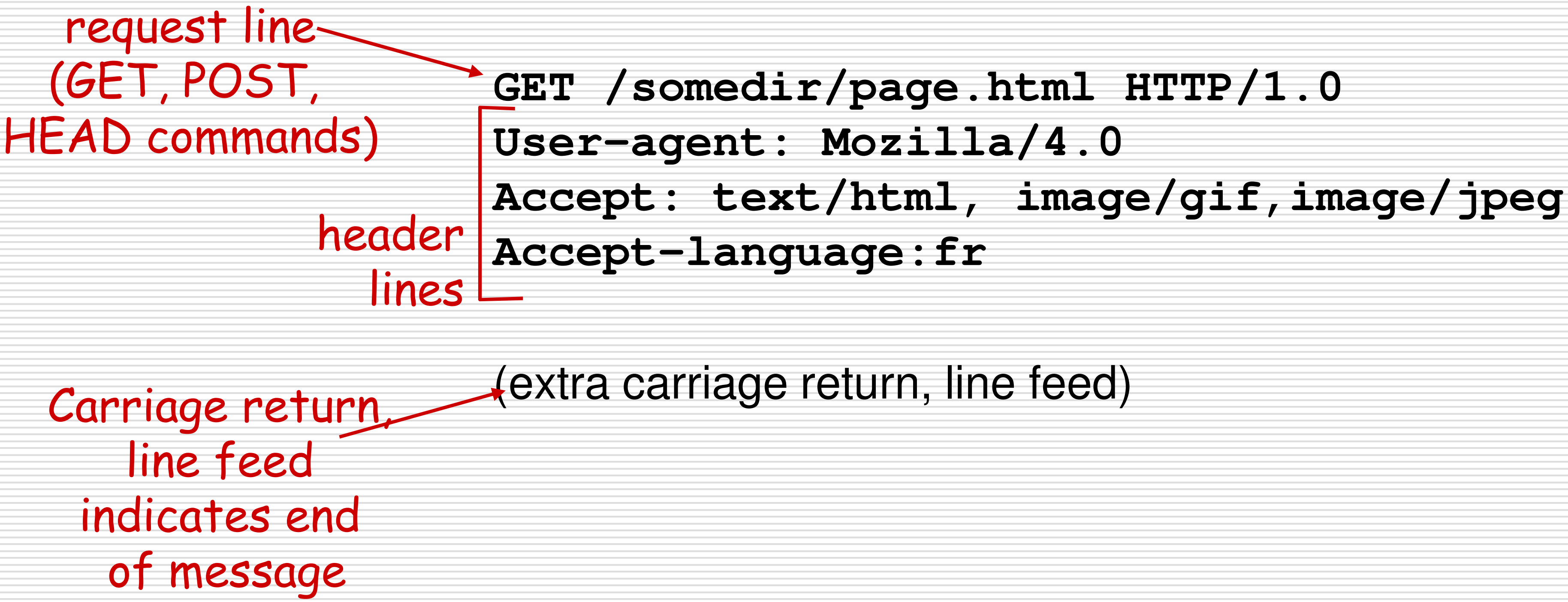
- http/1.0: server parses request, responds, closes TCP connection
- (2 + x) RTTs to fetch object
  - TCP connection
  - object request/transfer
- each transfer suffers from TCP's initially slow sending rate
- many browsers open multiple parallel connections

Persistent

- default for http/1.1
- on same TCP connection: server, parses request, responds, parses new request,..
- client sends requests for all referenced objects as soon as it receives base HTML.
- fewer RTTs, less slow start.
- With pipelining and without pipelining.

http message format: request

- two types of http messages: *request, response*
- http request message:
  - ASCII (human-readable format)



```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

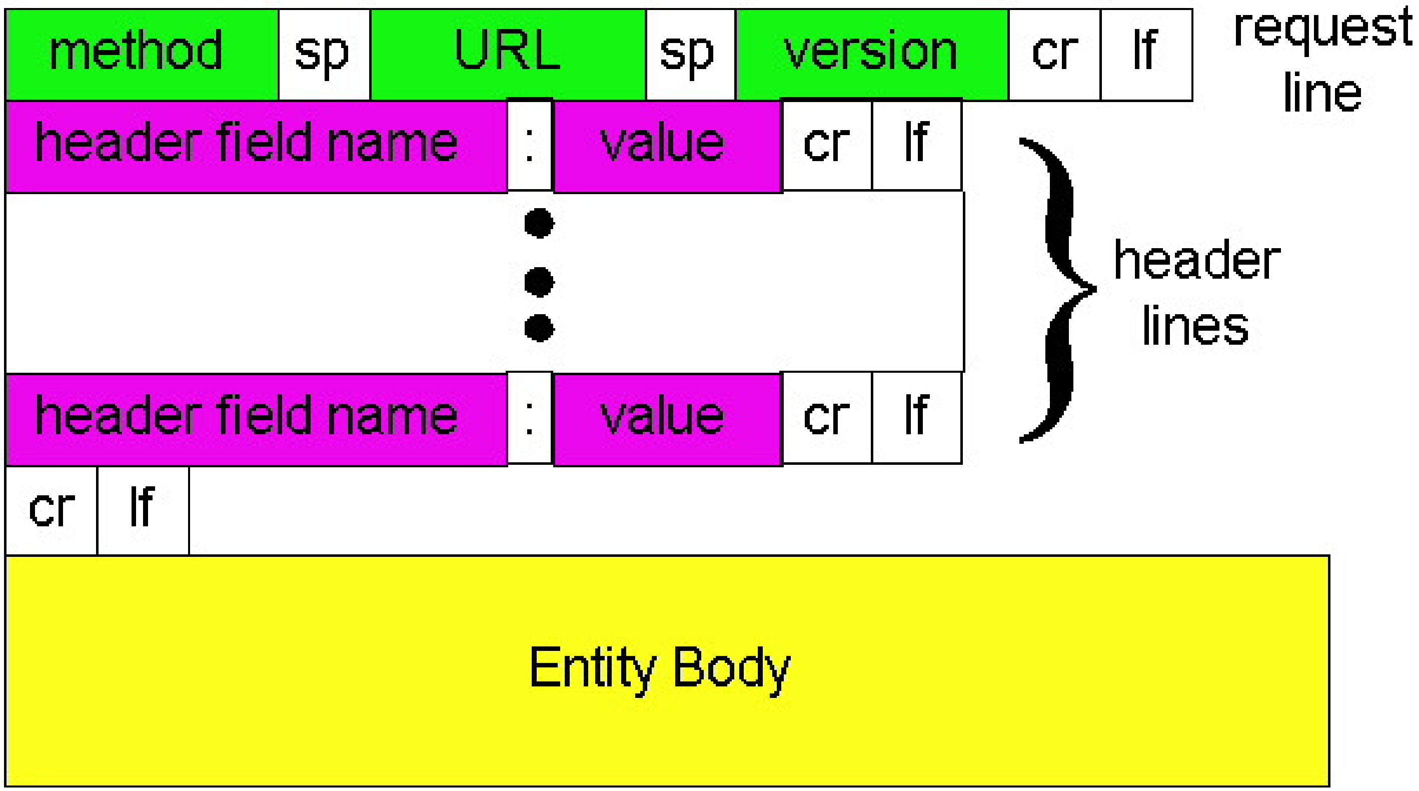
(extra carriage return, line feed)

request line (GET, POST, HEAD commands)

header lines

Carriage return, line feed indicates end of message

http request message: general format



http message format: response

status line (protocol status code status phrase)

header lines

```
HTTP/1.0 200 OK
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 ...
```

data, e.g., requested html file

http response status 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 www.cmb.ac.lk 80
```

Opens TCP connection to port 80 (default http server port) at www.cmb.ac.lk. Anything typed in sent

to port 80 at www.cmb.ac.lk

2. Type in a GET http request:

```
GET index.html HTTP/1.0
```

By typing this in (hit carriage return twice), you send this minimal (but complete)

GET request to http server

3. Look at response message sent by http server!

Any Questions?