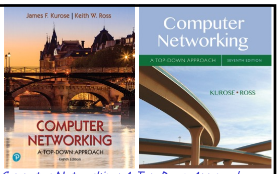
CS 655
Computer Networks

Abraham Matta
Computer Science
Boston University

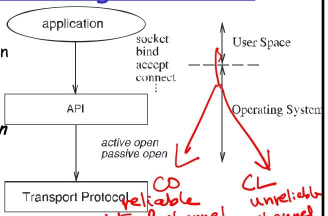
Chapter 2
Applications



Computer Networking: A Top-Down Approach, 8th edition. Jim Kurose, Keith Ross. Pearson.
7th edition is OK too!

Application Programming Interface

- Transport layer provides an ***interface*** to application programs to access the network. This interface is often called the ***application programming interface***, or API
- The API is usually defined by the operating system
- We focus on one specific API: ***sockets***
- Defined by BSD Unix, but ported to other systems



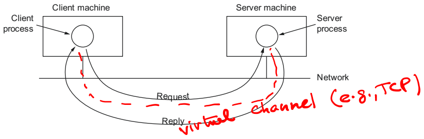
CO
reliable
virtual channel
(TCP)

CL
unreliable
channel
(UDP)

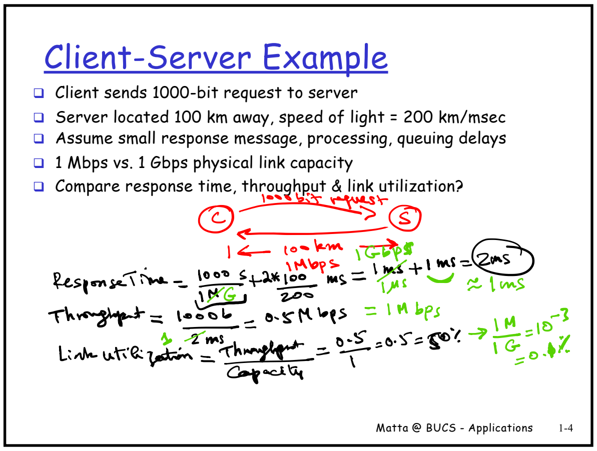
Socket $\triangleq$ end point of communication

socket = (IP address, port number)

Client-Server Communication



- Server performs a passive-open operation
  - runs on a ***well-known port number***
- Client performs an active-open operation
  - gets assigned an arbitrary ***unused*** port number
- Transfer data through ports (sockets)



Client-Server Example

- ❑ Client sends 1000-bit request to server
- ❑ Server located 100 km away, speed of light = 200 km/msec
- ❑ Assume small response message, processing, queuing delays
- ❑ 1 Mbps vs. 1 Gbps physical link capacity
- ❑ Compare response time, throughput & link utilization?

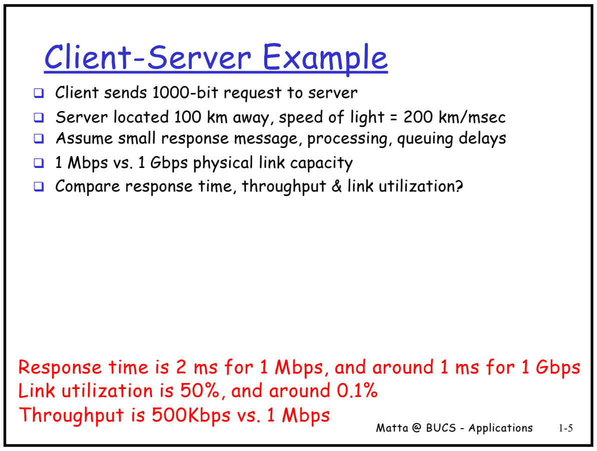
1000 bit request
C
S
1 ← 100 km →
1Mbps
1Gbps

$$\text{Response Time} = \frac{1000}{\cancel{1M}\,G}\,s + 2 * \frac{100}{200}\,ms = \cancel{1\,ms}\ 1\,\mu s + 1\,ms = \boxed{2ms} \approx 1\,ms$$

$$\text{Throughput} = \frac{1000b}{\cancel{2}\,1\ ms} = 0.5\,M\,bps = 1\,M\,bps$$

$$\text{Link utilization} = \frac{\text{Throughput}}{\text{Capacity}} = \frac{0.5}{1} = 0.5 = 50\% \rightarrow \frac{1M}{1G} = 10^{-3} = 0.1\%$$

4

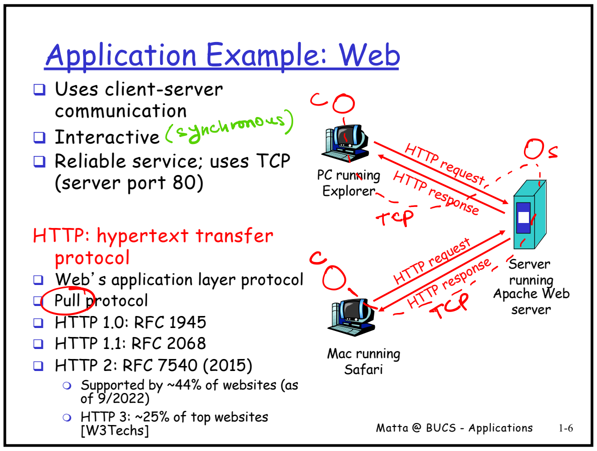


Client-Server Example

- ❑ Client sends 1000-bit request to server
- ❑ Server located 100 km away, speed of light = 200 km/msec
- ❑ Assume small response message, processing, queuing delays
- ❑ 1 Mbps vs. 1 Gbps physical link capacity
- ❑ Compare response time, throughput & link utilization?

Response time is 2 ms for 1 Mbps, and around 1 ms for 1 Gbps
Link utilization is 50%, and around 0.1%
Throughput is 500Kbps vs. 1 Mbps

5



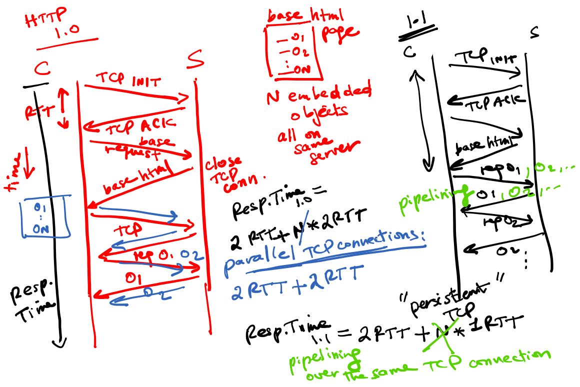
Application Example: Web

- ❑ Uses client-server communication
- ❑ Interactive (synchronous)
- ❑ Reliable service; uses TCP (server port 80)

HTTP: hypertext transfer protocol

- ❑ Web's application layer protocol
- ❑ Pull protocol
- ❑ HTTP 1.0: RFC 1945
- ❑ HTTP 1.1: RFC 2068
- ❑ HTTP 2: RFC 7540 (2015)
  - ○ Supported by ~44% of websites (as of 9/2022)
  - ○ HTTP 3: ~25% of top websites [W3Techs]

6



$\text{Resp.Time}_{1.0} =$

$2\,RTT + \cancel{N} * 2RTT$

parallel TCP connections:

$2RTT + 2RTT$

"persistent TCP"

$\text{Resp.Time}_{1.1} = 2RTT + \cancel{N} * 1RTT$

pipelining over the same TCP connection

7

HTTP connections

Nonpersistent HTTP

- ❑ At most one object (html file, jpeg image, audio clip file, …) is sent over a TCP connection
- ❑ HTTP/1.0 uses nonpersistent connections

Persistent HTTP

- ❑ Multiple objects can be sent over single TCP connection between client and server
- ❑ HTTP/1.1 uses persistent connections in default mode

8

Persistent HTTP

Nonpersistent HTTP issues:

- ❑ requires 2 RTTs per object
- ❑ OS must work and allocate host resources for each TCP connection
- ❑ but 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 are sent over connection

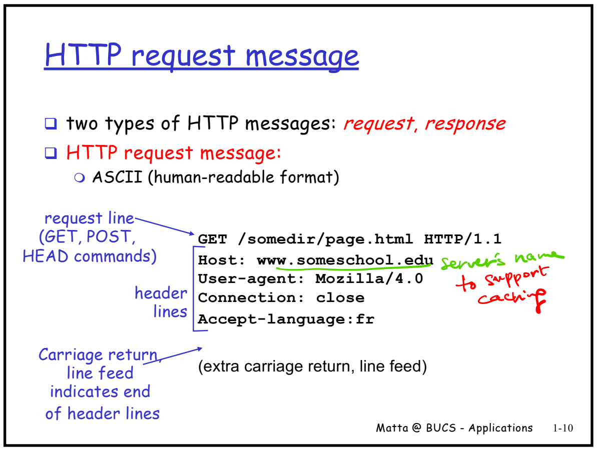
Persistent without pipelining:

- ❑ client issues new request only when previous response has been received
- ❑ one RTT for each referenced object

Persistent with pipelining:

- ❑ default in HTTP/1.1
- ❑ client sends requests as soon as it encounters a referenced object
- ❑ as little as one RTT for all the referenced objects

9



HTTP request message

- ❑ 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
```

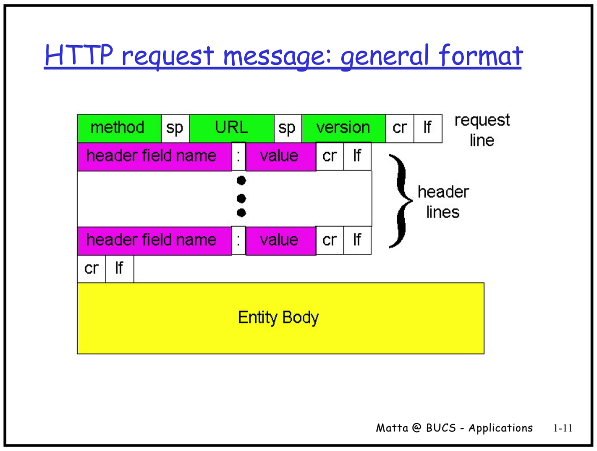
server's name to support caching

header lines

(extra carriage return, line feed)

Carriage return, line feed indicates end of header lines

10



HTTP request message: general format

method | sp | URL | sp | version | cr | lf — request line

header field name | : | value | cr | lf

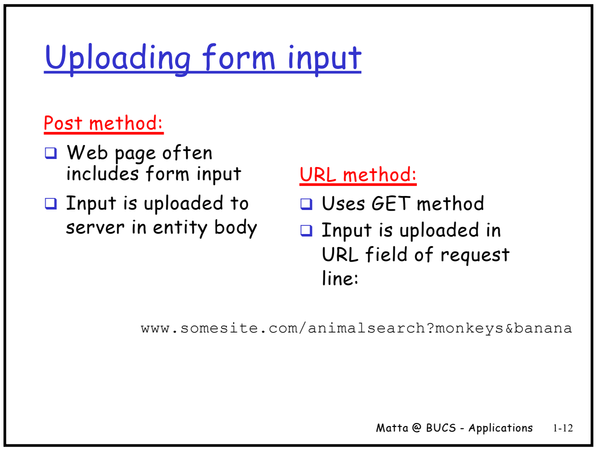
⋮

header field name | : | value | cr | lf — header lines

cr | lf

Entity Body

11



Uploading form input

Post method:

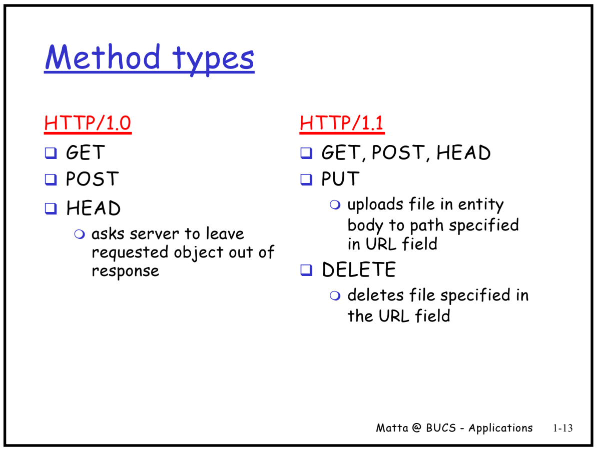
- ❑ Web page often includes form input
- ❑ Input is uploaded to server in entity body

URL method:

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

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

12

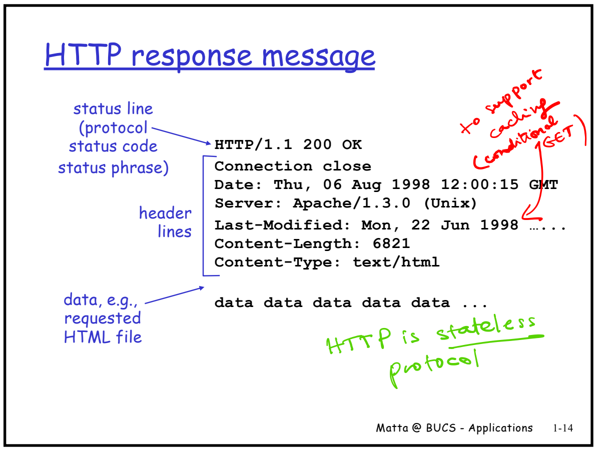


Method types

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

13



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 ...
```

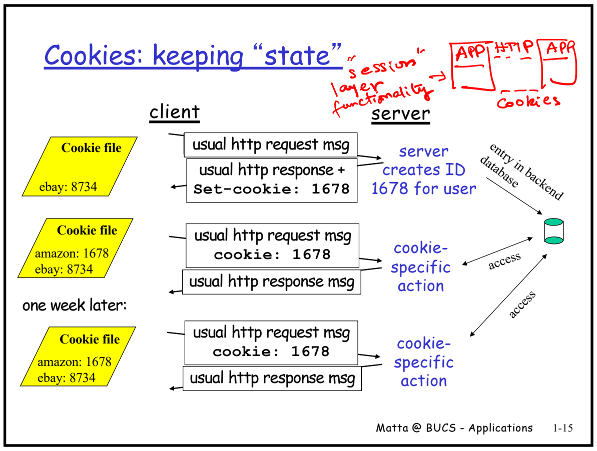
header lines

data, e.g., requested HTML file

to support caching (conditional GET)

HTTP is stateless protocol

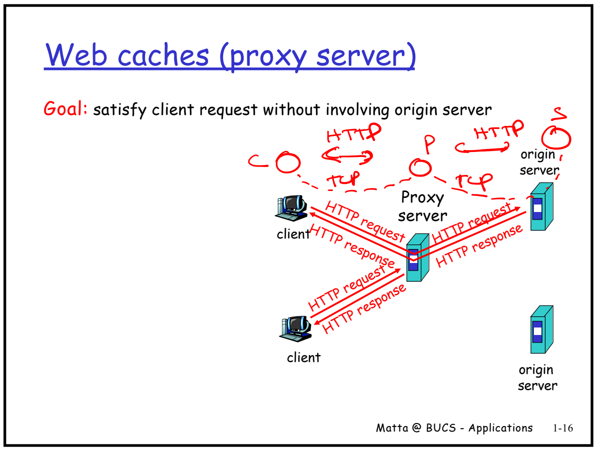
14



Cookies: keeping "state"

"session" layer functionality →

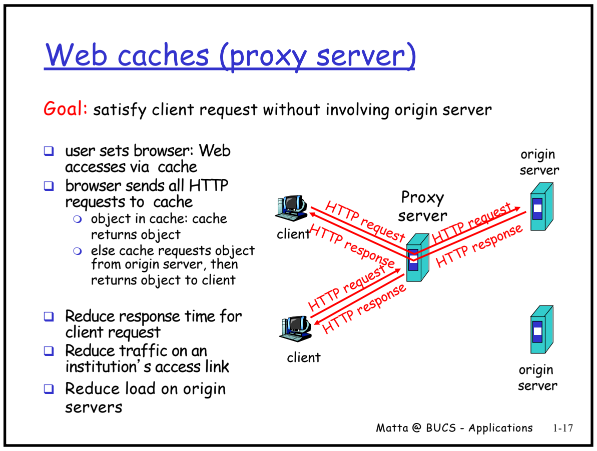
15



Web caches (proxy server)

Goal: satisfy client request without involving origin server

16

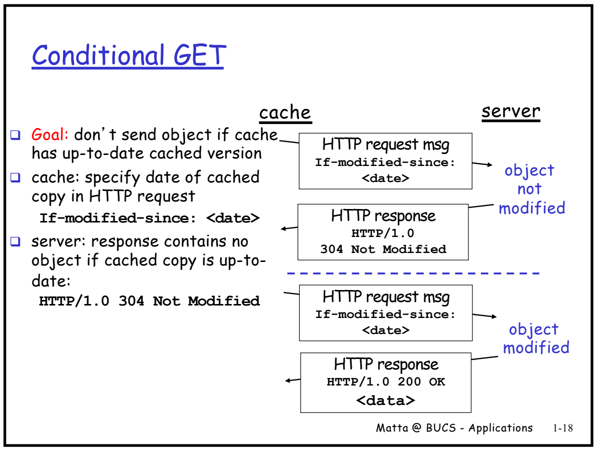


Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client
- Reduce response time for client request
- Reduce traffic on an institution's access link
- Reduce load on origin servers

17



Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
  `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:
  `HTTP/1.0 304 Not Modified`

18