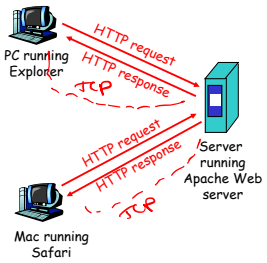
Application Example: Web

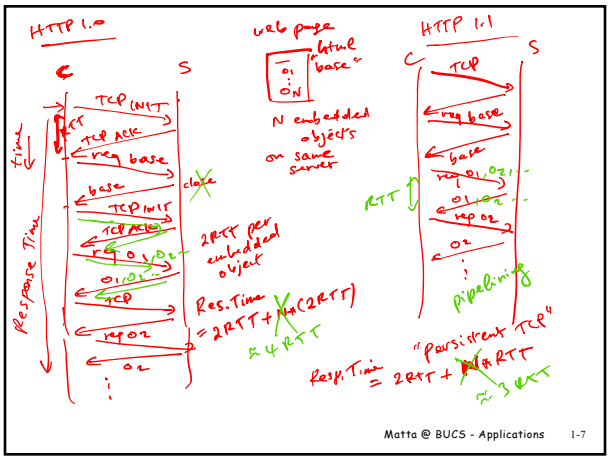
- Uses client-server communication
- Interactive (synchronous)
- Reliable service; uses TCP (server port 80)



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 ~48% of websites (as of 9/2020)
  - HTTP 3: ~7% of websites

6



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)

header lines

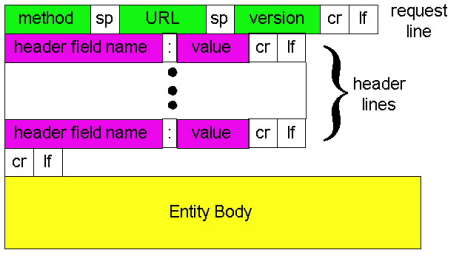
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Carriage return, line feed indicates end of header lines

(extra carriage return, line feed)

10

HTTP request message: general format



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

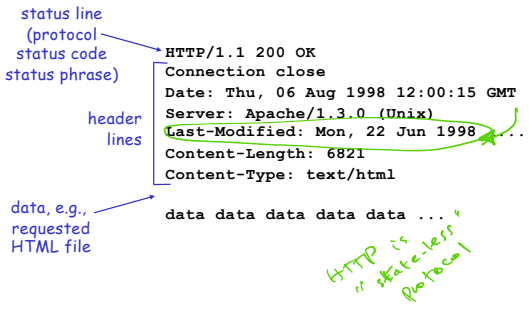
- GET
- POST
- HEAD
  - asks server to leave requested object out of response

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

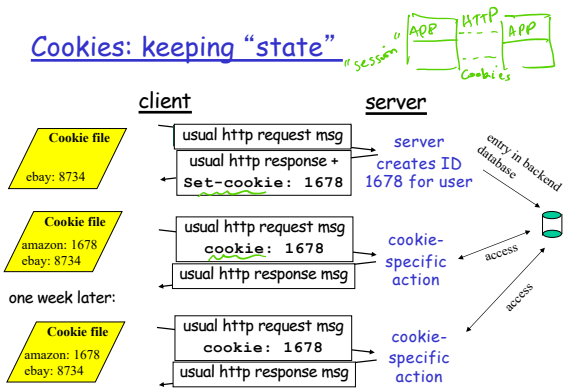
13

HTTP response message



14

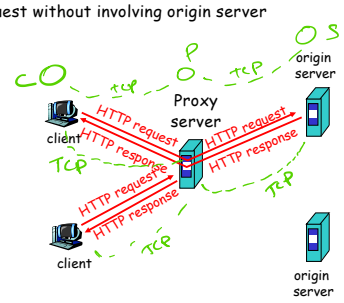
Cookies: keeping "state"



15

Web caches (proxy server)

Goal: satisfy client request without involving origin server

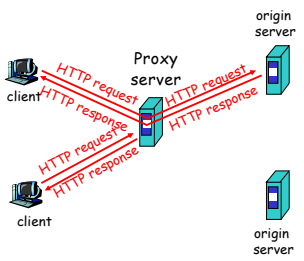


16

Web caches (proxy server)

Goal: satisfy client request without involving origin server

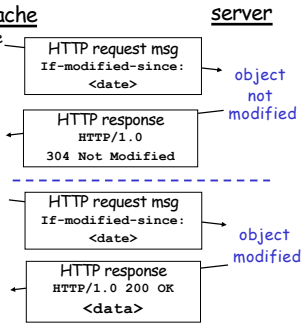
- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client
- Reduce response time for client request
- Reduce traffic on an institution's access link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

Socket Interface

- Interface between application programs and TCP/IP software (introduced in Berkeley UNIX Operating System)
- Centers around *socket abstraction*
- Follows open-read-write-close paradigm

- socket (endpoint) = <IP address, port number>

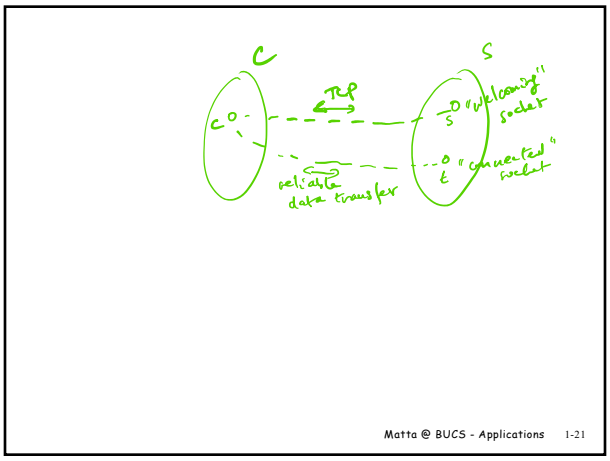
19

Connection-oriented (TCP) Based Application

- Server Program
  - Create a socket
  - Bind it to a well-known port on local machine
  - Wait for clients

- Client Program
  - Create a socket
  - Connect it to a server on a remote machine
  - Use it to send/receive data to/from remote machine
  - When done, close socket

20


C
S
TCP
"welcoming" socket
"connected" socket
reliable data transfer
Matta @ BUCS - Applications 1-21

21