

Application-Layer Protocols and HTTP

1

What's a protocol?

human protocol



network protocol:



2

What's a protocol?

Human Protocols:

- "thank you ... you're welcome"
- "hello ... hi ... my name is ... pleased to meet you"
- Price haggling

... specific msgs sent
... specific actions taken when msgs received
... may be context or culture sensitive

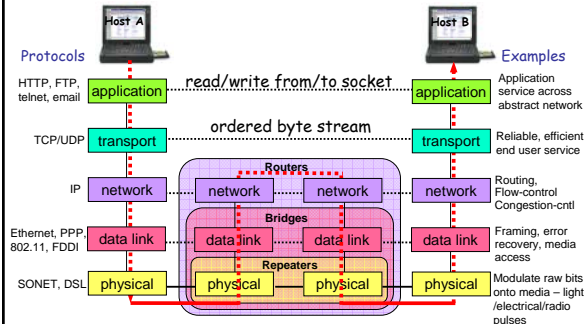
Network protocols:

- drive device, rather than human, interaction
- all communication activity in Internet is governed by protocols

protocols define format & order of messages sent and received among network entities, and actions taken on message transmission, receipt

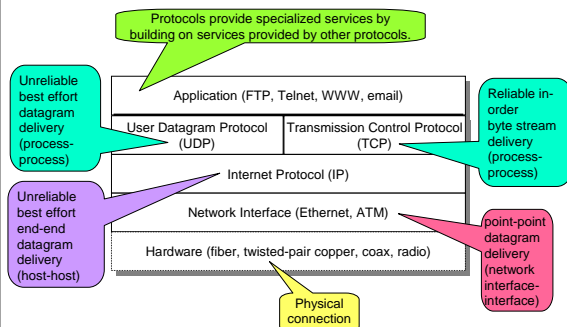
3

Protocol Stacks



4

Protocol layering



5

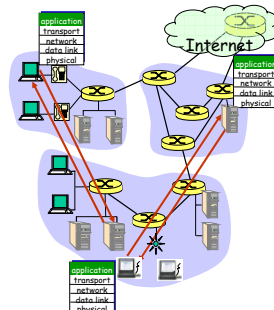
Applications and application-layer protocols

Application

- running in network hosts in "user space"
- exchange messages to implement app
- e.g., email, file transfer, the Web

Application-layer protocols

- one "piece" of an app
- define messages exchanged by apps and actions taken
- connectivity provided by lower layer protocols



6

Client-Server Paradigm

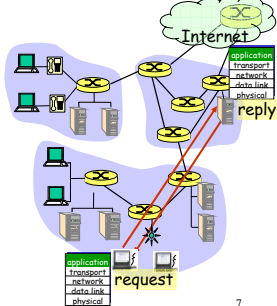
Client:

- initiates contact with server ("speaks first")
- typically requests service from server,
- for Web, client is implemented in browser; for e-mail, in mail reader

Server:

- provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail

Typical network app has two pieces: *client* and *server*



7

HTTP

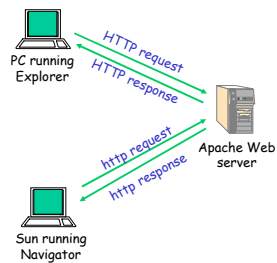
- HyperText Transfer Protocol
 - Created by Tim Berners-Lee at CERN
 - Defined 1989-1991
- Standardized and much expanded by the IETF
- Rides on top of TCP protocol
 - TCP provides: reliable, bi-directional, in-order byte stream
- Goal: transfer objects between systems
 - Do not confuse with other WWW concepts:
 - HTTP is not page layout language (that is HTML)
 - HTTP is not object naming scheme (that is URLs)
- Text-based protocol
 - Human readable

8

The Web: 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



9

HTTP in operation

Suppose user enters URL

www.toronto.edu/cs/index.html (containing text and references to 10 jpeg images)

- 1a. http client initiates TCP connection to http server (process) at www.toronto.edu. Port 80 is default for http server.
- 1b. http server at host www.toronto.edu waiting for TCP connection at port 80. "accepts" connection, notifying client
2. http client sends http *request message* (containing URL /cs/index.html) into TCP connection socket
3. http server receives request message, forms *response message* containing requested object (/cs/index.html), sends message into socket

time ↓

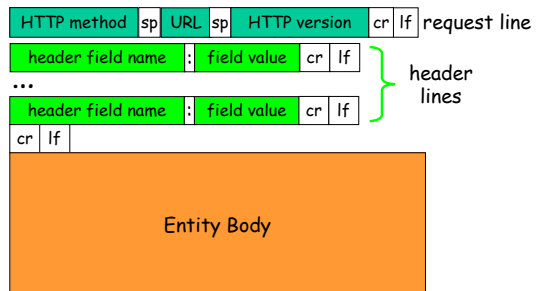
10

http in operation (cont.)

5. http client receives response message containing html file, displays html file, finds 10 referenced jpeg objects
6. Steps 1-5 repeated for each of 10 jpeg objects
4. http server closes TCP connection.

11

HTTP request message: general format



12

HTTP request format

Request line

HTTP method sp URL sp HTTP version cr lf

- o HTTP method
 - GET - return content of specified document
 - HEAD - return headers only of GET response
 - POST - execute specified doc with enclosed data
- o URL (only domain portion)
 - /host-identifier/path
 - e.g. /www.toronto.edu/headlines/
- o HTTP version
 - e.g. HTTP/1.0

13

HTTP request format

Header fields

header field name : field value

Examples:

- o Accept: text/html
- o Accept: image/jpg
- o Accept-language: en; en-gr; fr
- o If-modified-since: 17 May 2001
- o Content-Length: 2540

14

HTTP request example

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

request line
(GET, POST,
HEAD
commands)

header
lines

(extra carriage return, line feed)

Carriage return,
line feed
indicates end
of message

15

HTTP response message: general format

HTTP version sp status code sp status phrase cr lf response line

header field name : field value cr lf

...

header field name : field value cr lf

cr lf

header
lines

Response Body

16

HTTP response format

Response line:

HTTP version sp status code sp status phrase cr lf

- o HTTP version
- o 3 digit response code
 - 1XX - informational
 - 2XX - success
 - 3XX - redirection
 - 4XX - client error
 - 5XX - server error
- o Brief text explanation of status code (e.g. OK)

17

Response status codes

A few commonly occurring sample codes:

200 OK

- o request succeeded, requested object later in this message

301 Moved Permanently

- o requested object moved, new location specified later in this message (Location:)

400 Bad Request

- o request message not understood by server

404 Not Found

- o requested document not found on this server

505 HTTP Version Not Supported

18

HTTP response format

Header fields

header field name: field value

Examples:

- o Content-Type: text/html
- o Content-Length: 4028
- o Language: en;
- o Last-modified: 17 May 2004

19

HTTP response example

```

HTTP/1.0 200 OK
Date: Thu, 25 Aug 2001 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Aug 2001 .....
Content-Length: 6821
Content-Type: text/html
data data data data data ...
data data data data data ...
data data data data data ...
    
```

status line: (protocol status code status phrase)

header lines

Carriage return, line feed indicates end of header

data, e.g., requested html file

20

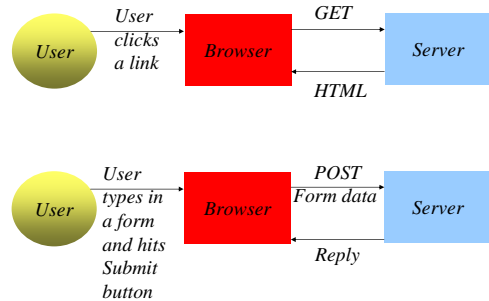
HTTP 1.0 other features

POST

- o Client can send information to server
- o Forms, annotations

21

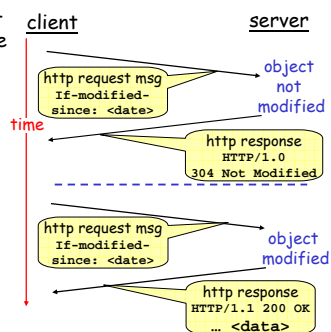
GET vs. POST - typical usage



22

User-server interaction: conditional GET

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



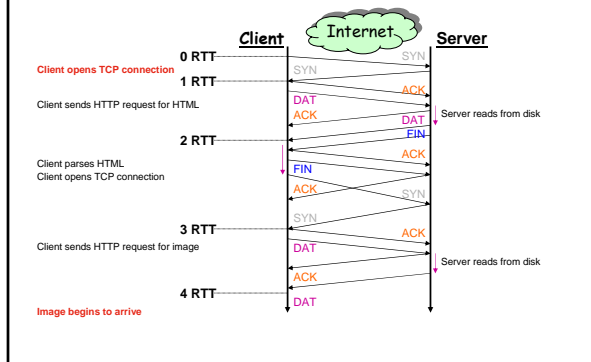
23

HTTP 1.0: Problems

- o Each request opens new connection
 - o Opening connection takes several packets (why?)
 - o Starting up is slow (why?)

24

Web Page with Single Image

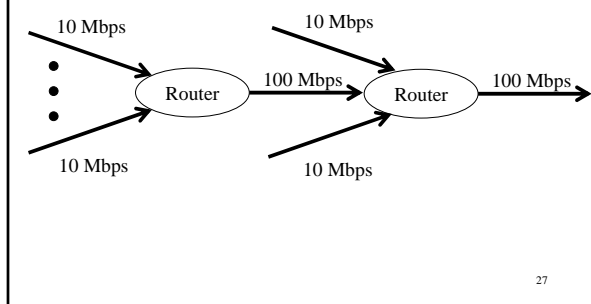


TCP

- HTTP rides on top of TCP transport service
- TCP provides: reliable, bi-directional, in-order byte stream
- Reliable
 - Prevent packet loss due to congestion
 - Overflow network queues
 - Send at rate at which network can forward packets
 - How to determine sending rate?
 - Dynamic
 - Depends on overall network condition

26

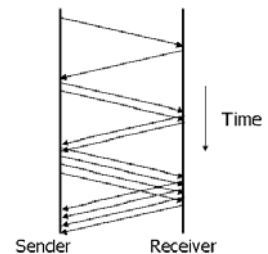
Network Congestion



27

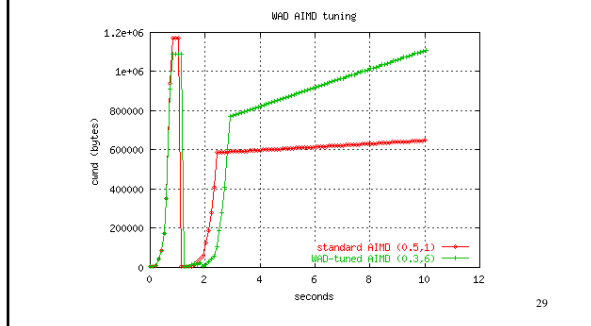
TCP Slow Start

- Determine sending rate by probing network
- Increase sending rate until a packet is dropped
- Double the number of unacknowledged packets (window size) for every new acknowledgement
- After a drop,
 - Reset window size to 1 packet
 - Cut maximum window size in half
 - Grow window with additive increase



28

TCP Slow Start



29

More Problems

- Short transfers are hard on TCP
 - Stuck in "slow start" phase of TCP connection
 - Loss recovery is poor when windows are small
- Lots of extra connections
 - Increases server state/processing

30

Netscape Solution

- Use multiple concurrent connections to improve response time
 - Different parts of Web page arrive independently
 - Can grab more of the network bandwidth than other users
- Doesn't necessarily improve response time
 - TCP loss recovery ends up being timeout dominated because windows are small

31

HTTP 1.1: Persistent Connections

- Keeps connection open for a time after server response so that multiple requests can ride on single connection -> reduced connection setup overhead.

GET index.html

Connection: keep-alive

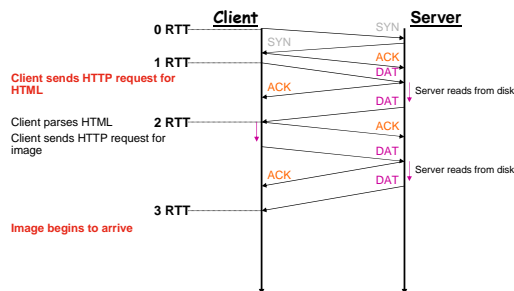
... multiple HTTP requests ...

Get banner.gif

Connection: close

32

Persistent Connections



Connection length

- When does the data end?
 - Without persistent connections, when connection closes.
 - With persistent connections, reply header includes content length.

34

Non-persistent and persistent connections

Non-persistent

- HTTP/1.0
- server parses request, responds, and closes TCP connection
- 2 RTTs to fetch each object
- Each object transfer suffers from slow start

But most 1.0 browsers use parallel TCP 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 and less slow start.

35

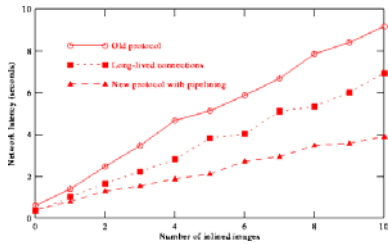
HTTP1.0 vs. HTTP1.1

- Venkata N. Padmanabhan and Jeffrey C. Mogul, "Improving HTTP Latency," in Proceedings of the The 2nd International WWW Conference, Chicago, IL, USA, Oct 1994
- Compared download latency for HTML documents with varying number of embedded images

36

HTTP1.0 vs. HTTP1.1

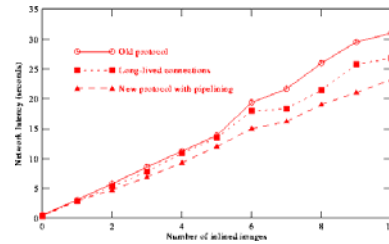
- Image size 2544 bytes



37

HTTP1.0 vs. HTTP1.1

- Image size 45566 bytes



38

Persistent Connection Performance

- Benefits greatest for small objects.
- Serialized requests do not improve response time.
- Pipelining requests can result in large win.
- Server resource utilization reduced due to fewer connection establishments and fewer active connections.
- TCP behavior improved.
 - Longer connections help adaptation to available bandwidth.
 - Larger congestion window improves loss recovery.

39

HTTP is Stateless

- Server does not maintain status information across client requests
 - No way to correlate multiple request from same user

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

40

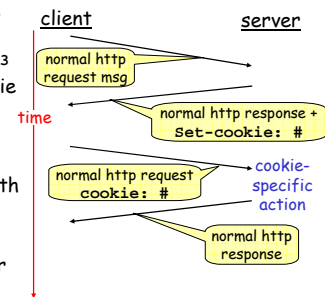
Cookies

- Store *cookie* on client side.
- Small amount of information (typically server-generated user id)
- Sent by client with each request
- Updated by server with response

41

User-server interaction: cookies

- server sends "cookie" to client in response msg
set-cookie: 1678453
- client presents cookie in later requests
cookie: 1678453
- server matches presented-cookie with server-stored info
 - authentication
 - remembering user preferences, previous choices



42