Application Layer
Part 3

Mark Allman
Case / ICSI

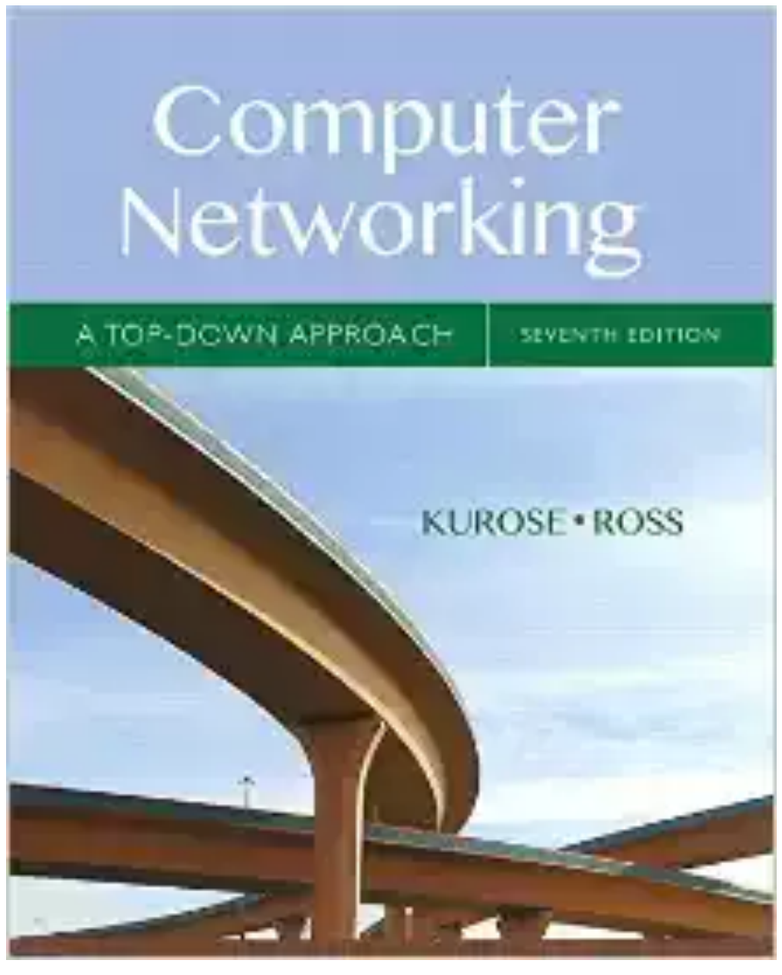
EECS 325/425
Fall 2018

"I get off on '57 Chevys, I get off on screamin' guitars,
I like the way it gets me, every time it hits me …"

Many of these slides are more-or-less directly from the slide set developed by Jim Kurose and Keith Ross for their book “Computer Networking: A Top Down Approach, 5th edition”.

The slides have been lightly adapted for Mark Allman’s EECS 325/425 Computer Networks class at Case Western Reserve University.

Reading Along ...



- 2.2: Web and HTTP

Web and HTTP

Web and HTTP

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL

Web and HTTP

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL
- example URL:

`http://www.someschool.edu/someDept/pic.gif`

Web and HTTP

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL
- example URL:

`http://www.someschool.edu/someDept/pic.gif`

protocol

Web and HTTP

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL
- example URL:

`http://www.someschool.edu/someDept/pic.gif`

protocol — host name (from DNS)

Web and HTTP

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL
- example URL:

`http://www.someschool.edu/someDept/pic.gif`

protocol | host name (from DNS) | path name (server local)

HTTP overview (continued)

HTTP overview (continued)

Client-server architectural model

HTTP overview (continued)

Client-server architectural model

Uses TCP:

HTTP overview (continued)

Client-server architectural model

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
  - recall `socket()` and `connect()`

HTTP overview (continued)

Client-server architectural model

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
  - recall `socket()` and `connect()`
- server accepts TCP connection from client

HTTP overview (continued)

Client-server architectural model

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
  - recall `socket()` and `connect()`
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)

HTTP overview (continued)

Client-server architectural model

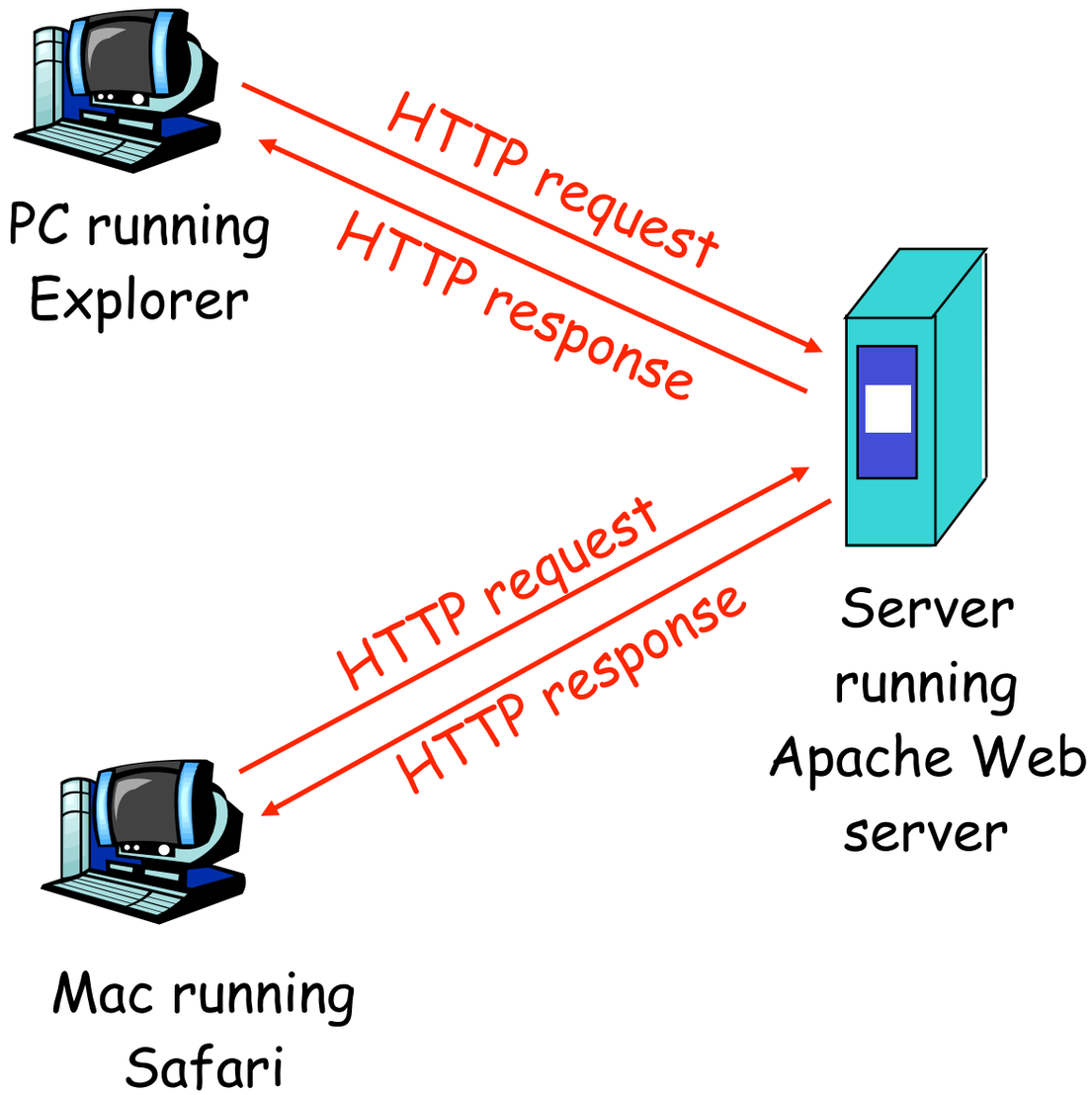
Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
  - recall `socket()` and `connect()`
- 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 overview

HTTP: hypertext transfer protocol

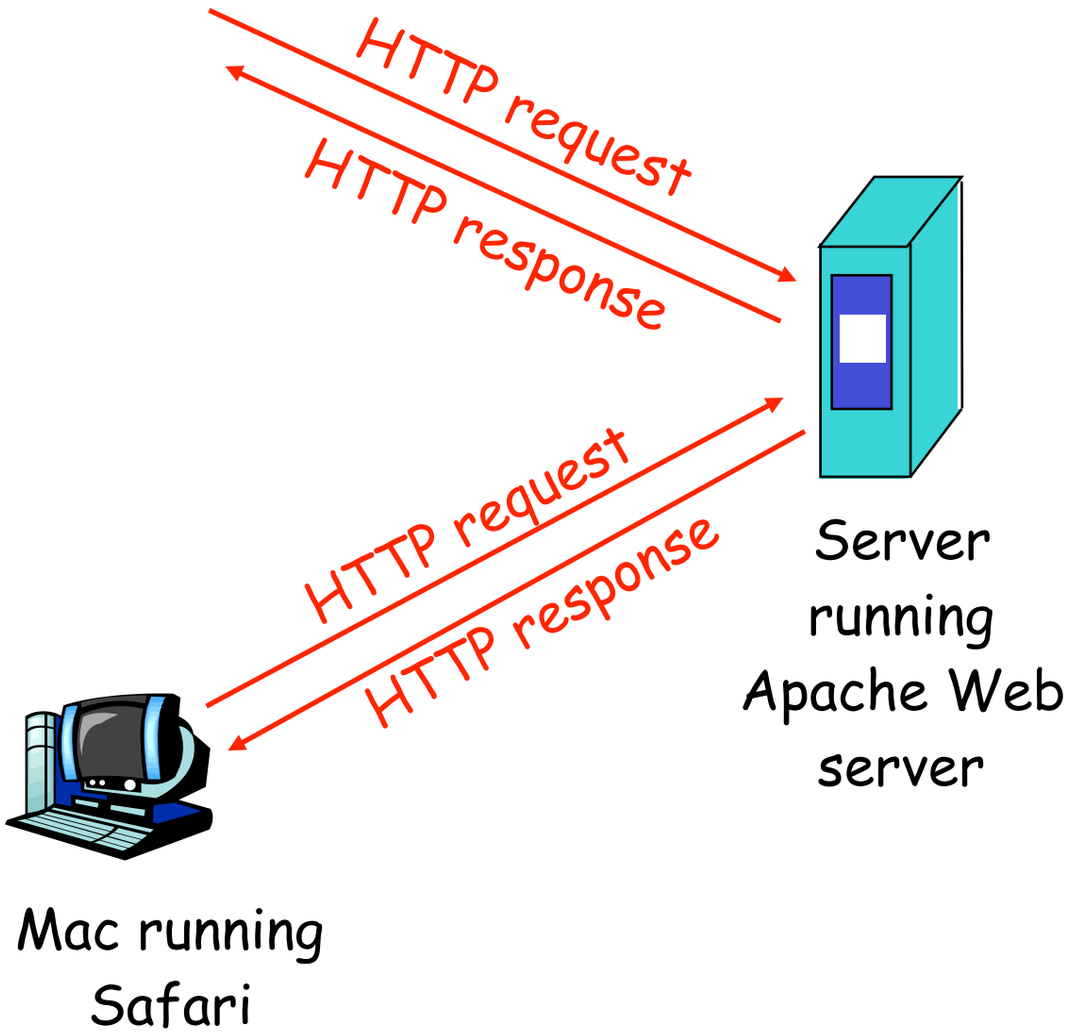
- Web's application layer protocol
- client/server model
  - client: browser that requests, receives, "displays" Web objects
  - server: Web server sends objects in response to requests



HTTP overview

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



HTTP Example

suppose user enters URL:

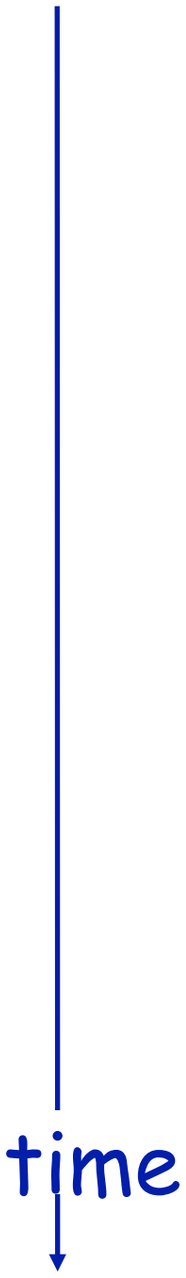
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

HTTP Example

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



HTTP Example

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

time

HTTP Example

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

time

HTTP Example

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/ home.index

time

HTTP Example

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/ home.index

3. HTTP server receives request message, forms response message containing requested object, and sends message into its socket

time

HTTP Example (cont.)

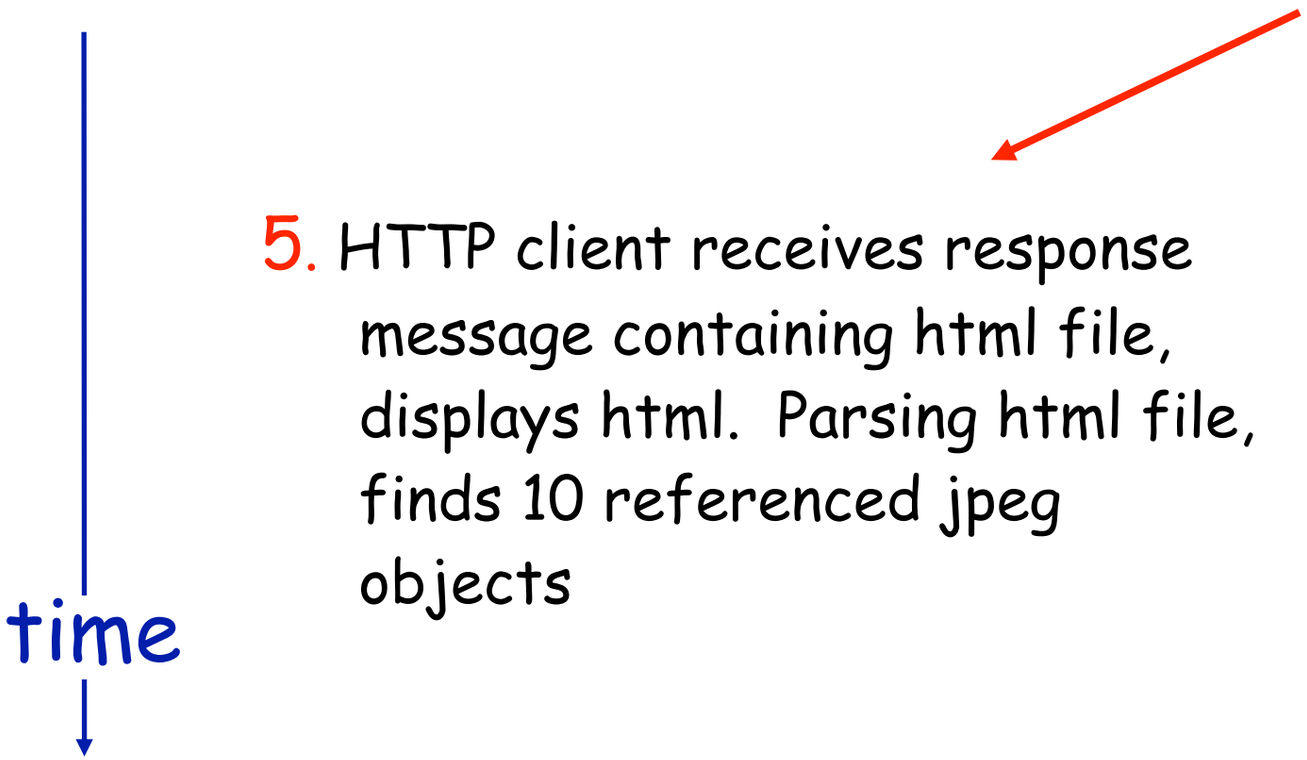
time

HTTP Example (cont.)

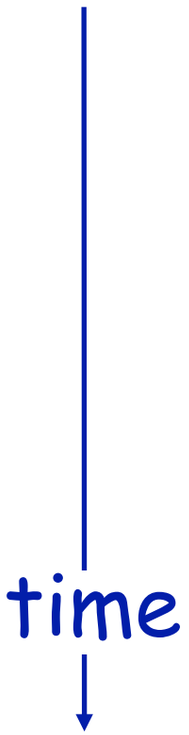
4. HTTP server closes TCP connection.

time

HTTP Example (cont.)



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

HTTP request message

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

HTTP request message

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

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP request message

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

request line (GET, POST, HEAD commands)

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP request message

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

request line (GET, POST, HEAD commands)

carriage return character

line-feed character

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

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

carriage return character

line-feed character

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

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

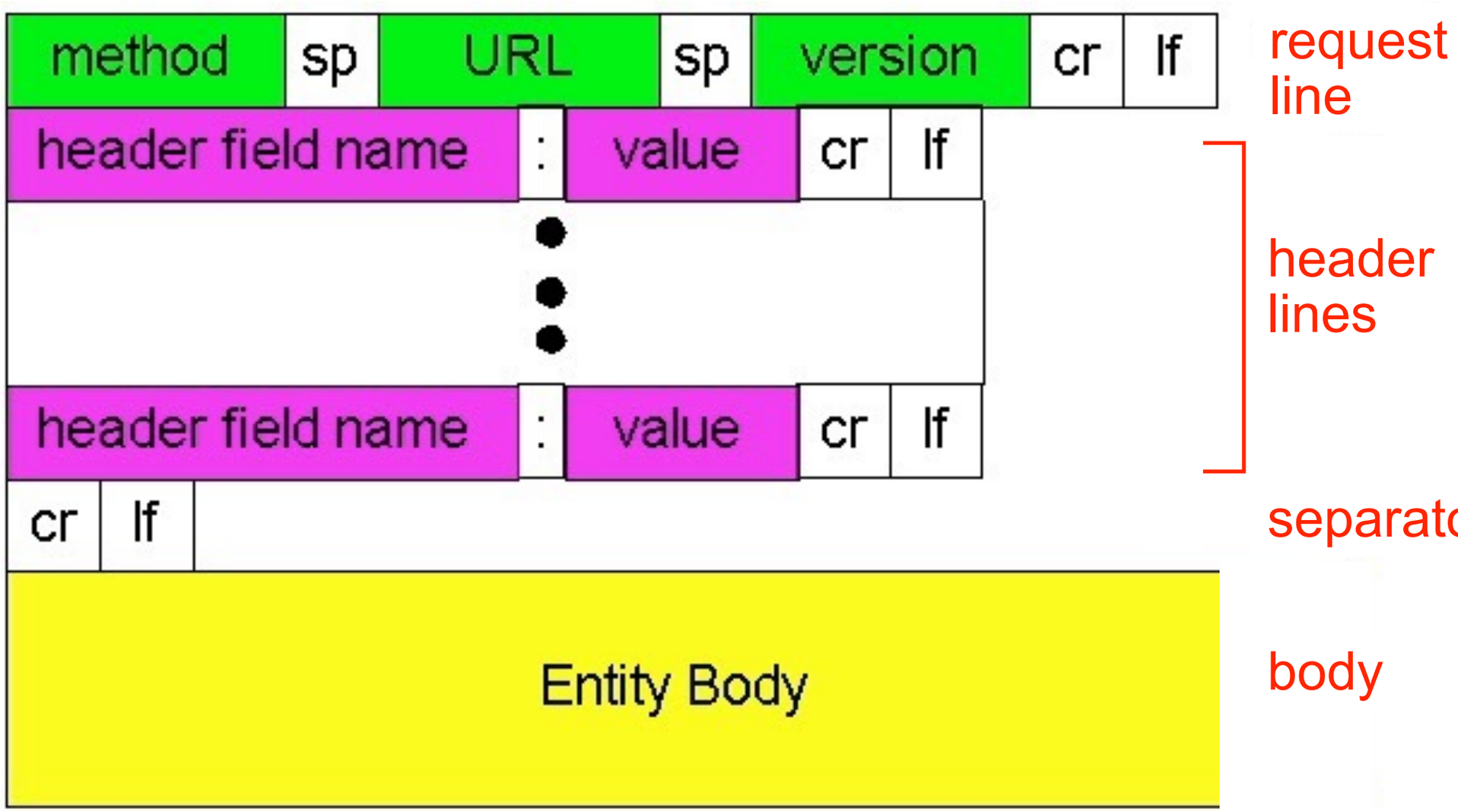
carriage return, line feed at start of line indicates end of header lines

carriage return character

line-feed character

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP request message: general format



Uploading form input

POST method:

- web page often includes form input
- input is uploaded to server in entity body

Uploading form input

POST method:

- web page often includes form input
- input is uploaded to server in entity body

GET method:

- input is uploaded in URL field of request line:

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

Method types

HTTP/1.0

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

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

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

HTTP/2

- exists
- much added complexity

HTTP response message

HTTP response message

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response message

status line
(protocol
status code
status phrase)

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response message

status line (protocol status code status phrase)

header lines

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response status codes

- status code appears in 1st line in server->client response message.
- some sample codes:

200 OK

  - request succeeded, requested object later in this msg

301 Moved Permanently

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

400 Bad Request

  - request msg 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 cis.poly.edu 80
```

opens TCP connection to port 80 (default HTTP server port) at cis.poly.edu. anything typed in sent to port 80 at cis.poly.edu

2. type in a GET HTTP request:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

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!

Simple HTTP Transaction

Simple HTTP Transaction

```
% telnet www.icir.org 80
```

Simple HTTP Transaction

```
% telnet www.icir.org 80
Trying 192.150.187.12...
Connected to www.icir.org.
Escape character is '^]'.
```

Simple HTTP Transaction

```
% telnet www.icir.org 80
Trying 192.150.187.12...
Connected to www.icir.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.icir.org
```

Simple HTTP Transaction

```
% telnet www.icir.org 80
Trying 192.150.187.12...
Connected to www.icir.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.icir.org

HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 16:01:08 GMT
Server: Apache/2.4.25 (Fedora)
Accept-Ranges: bytes
Content-Length: 7812
Content-Type: text/html; charset=UTF-8
```

Simple HTTP Transaction

```
% telnet www.icir.org 80
Trying 192.150.187.12...
Connected to www.icir.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.icir.org

HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 16:01:08 GMT
Server: Apache/2.4.25 (Fedora)
Accept-Ranges: bytes
Content-Length: 7812
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional […]
<!--Do not edit this autogenerated HTML file; edit src/xdocs files[…]
[…]
```