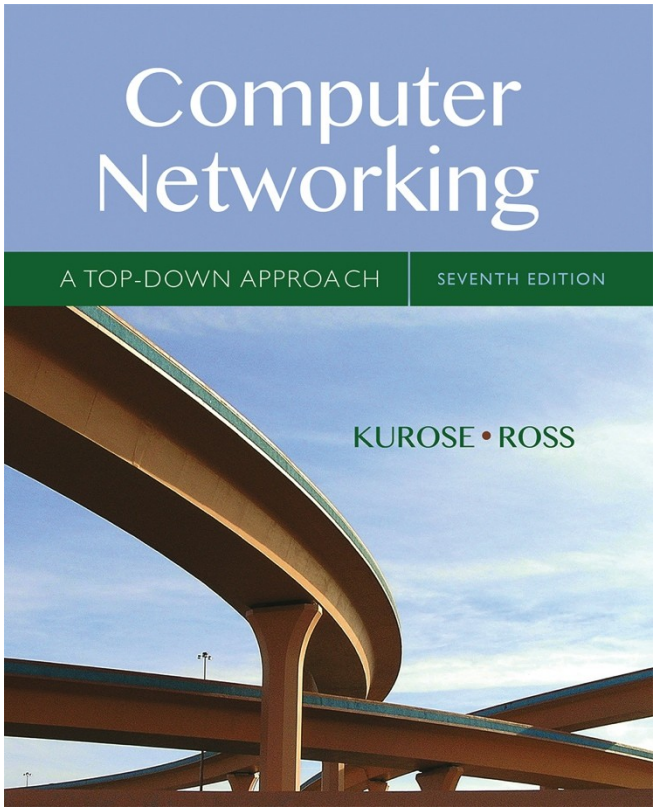
Lecture 02 Application Layer



Computer Networking: A Top Down Approach

7th edition

Jim Kurose, Keith Ross

Pearson/Addison Wesley

April 2016

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.5 P2P applications

Web and HTTP

First, a review…

- *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*, e.g.,

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

host name | path name

HTTP overview

HTTP: hypertext transfer protocol

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

HTTP overview (continued)

uses TCP:

- 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 connections

non-persistent HTTP

- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections

persistent HTTP

- multiple objects can be sent over single TCP connection between client, server

Non-persistent HTTP

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

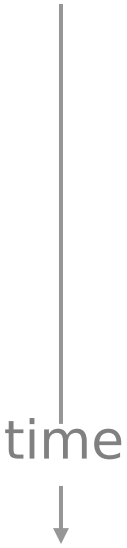
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

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

Non-persistent HTTP (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

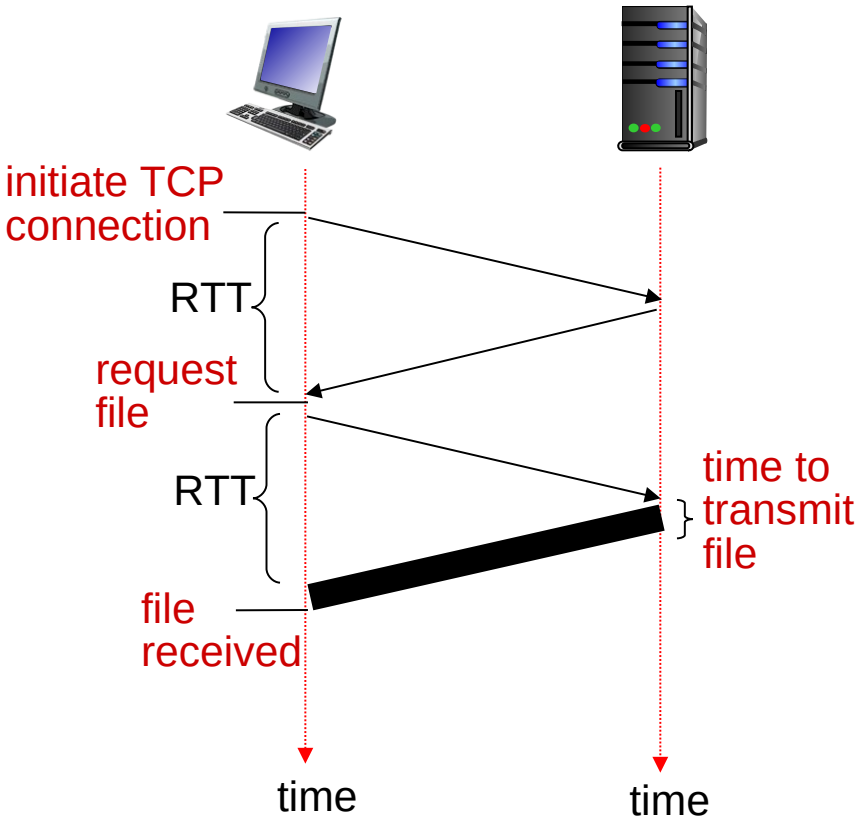
Non-persistent HTTP: response time

Round-Trip Time RRT (definition): time for a small packet to travel from client to server and back

HTTP response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time =

  2RTT+ file transmission time



Persistent HTTP

non-persistent HTTP issues:

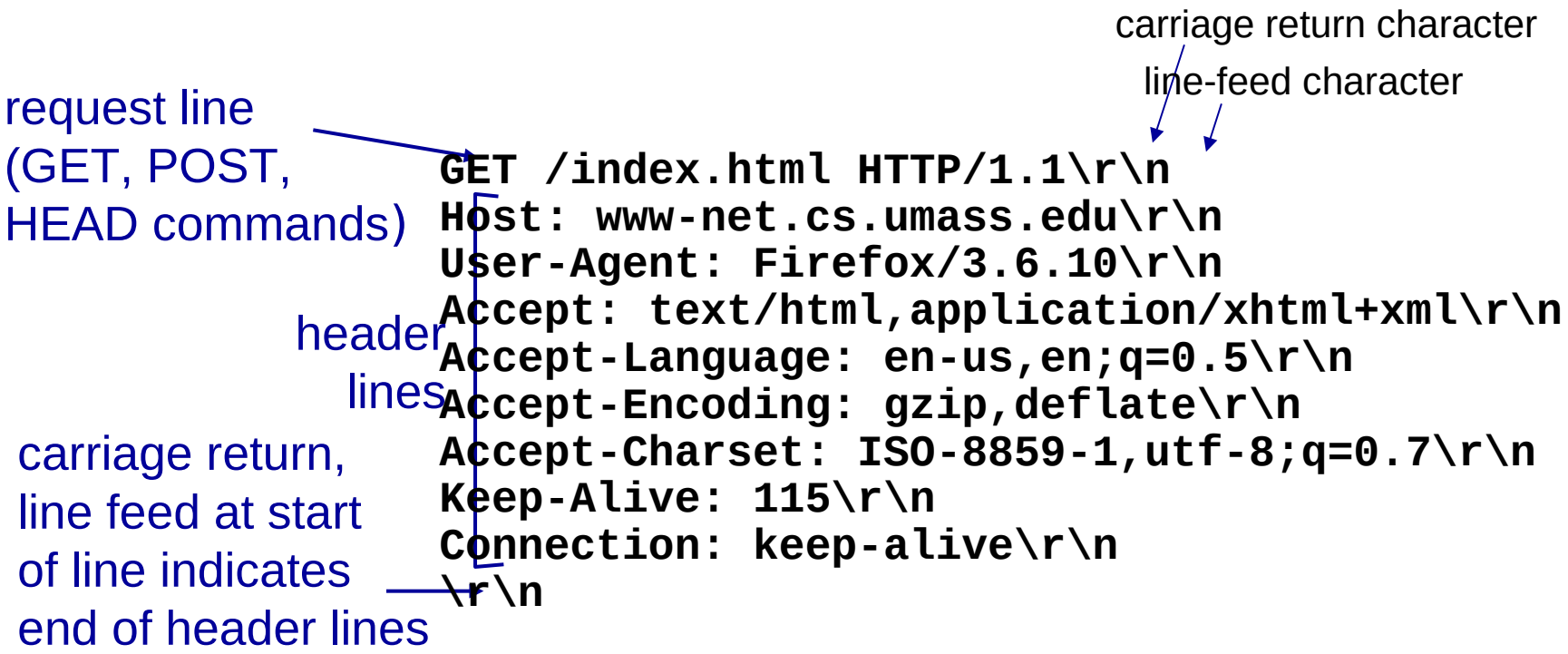
- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

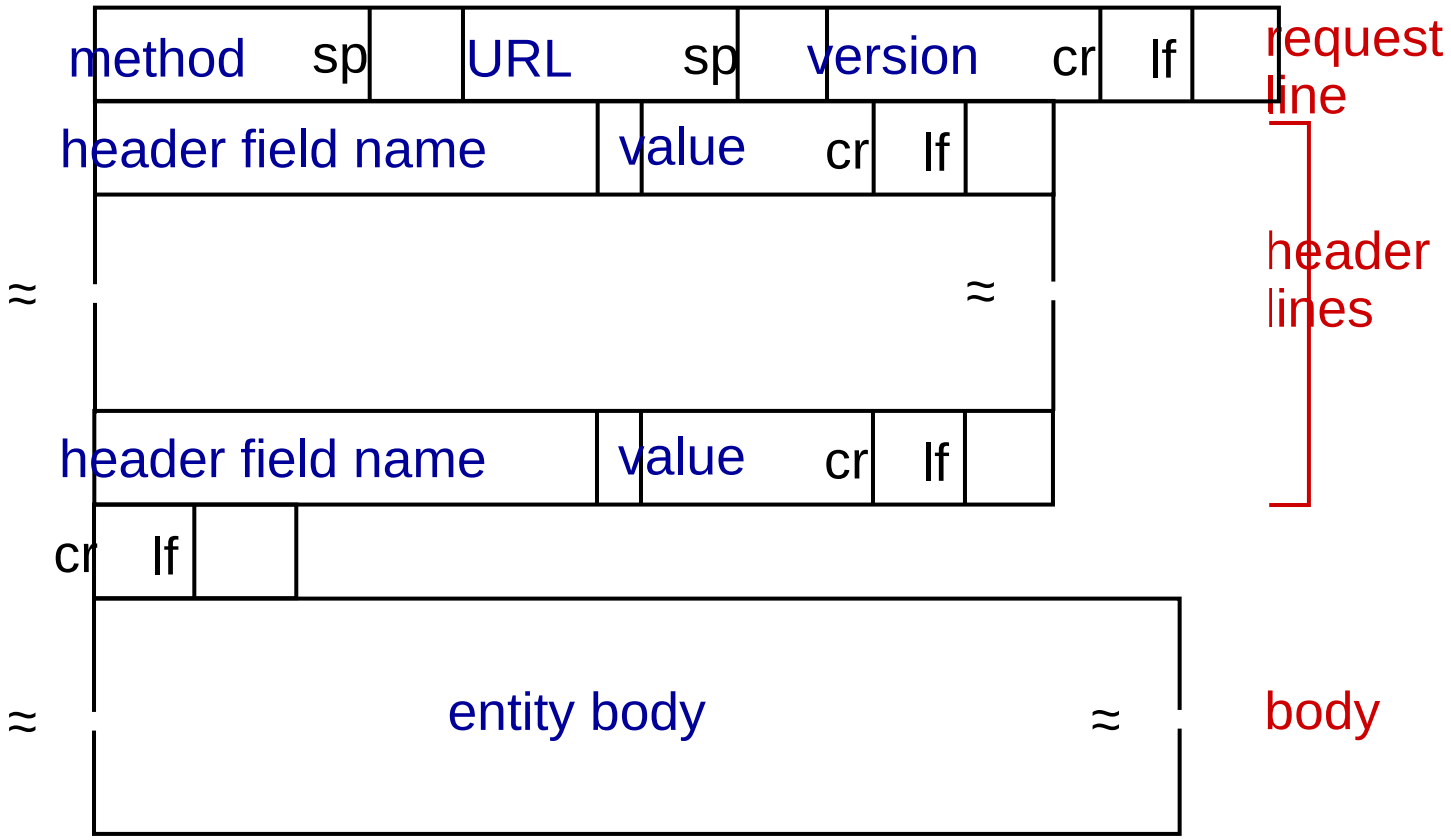
HTTP request message

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



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

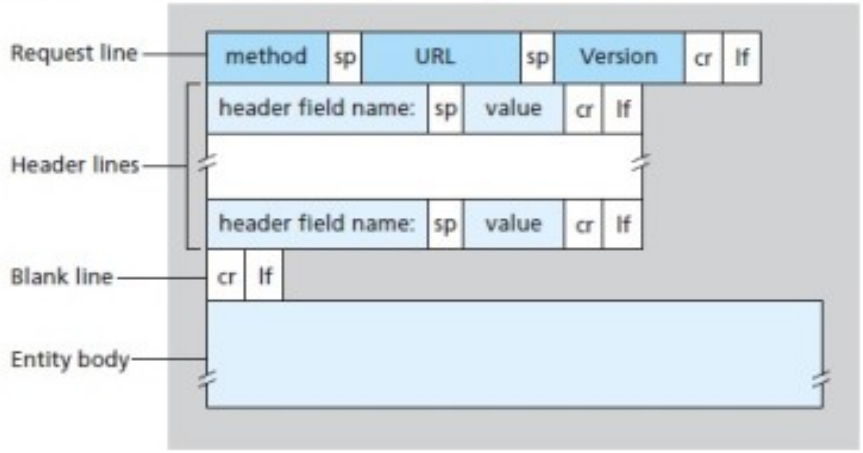
HTTP request message: general format



HTTP request message: general format

HTTP Request Message

- The format of an HTTP request message is as follow:

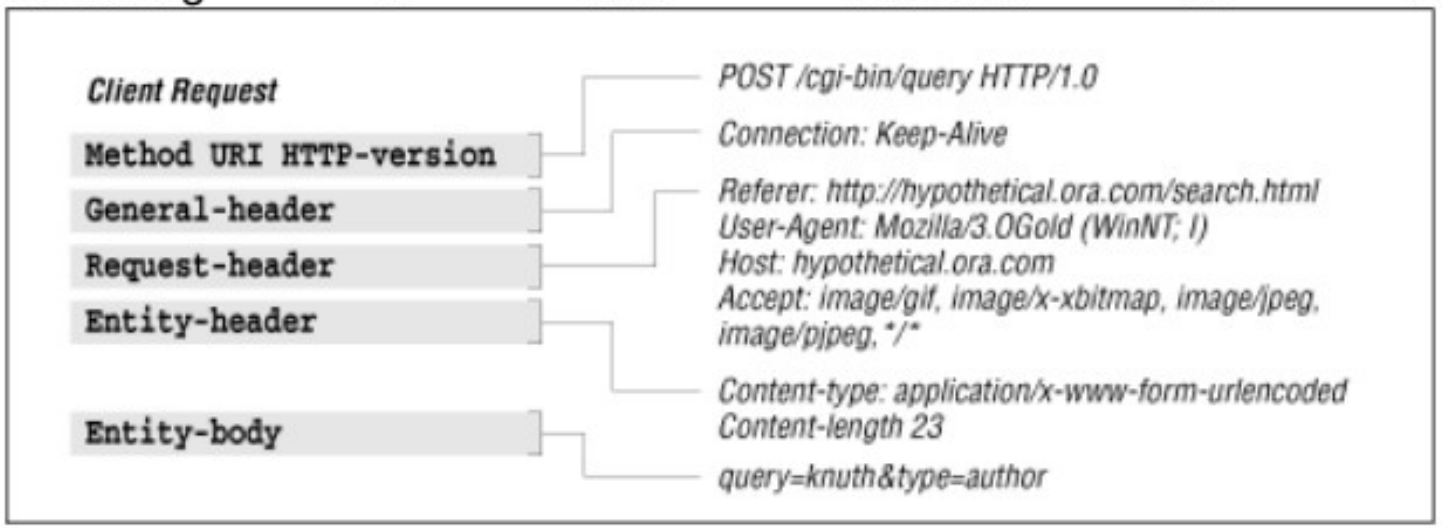


- The first line of the header is called the request line, followed by optional request headers.
- The request line has the syntax:request-method-name request-URI HTTP-version, e.g. GET /test.html HTTP/1.1 or HEAD /query.html HTTP/1.0.
- HTTP has three common versions, HTTP/1.0, HTTP/1.1 and HTTP/2.
- Request Headers (optional to the request message) have the syntax: request-header-name: request-header-value1, request-header-value2, ... For example, Connection: Keep-Alive or Accept-Language: us-en, fr.

HTTP request message: general format

HTTP Request headers

- HTTP Request headers can be grouped according to their contexts as follows:



- General header: Headers applying to both requests and responses but with no relation to the data eventually transmitted in the body. Examples, Date (represents the date and time at which the message was originated); Connection(allows the sender to specify options that are desired for that particular connection), . . . etc.

HTTP request message: general format

- Request header: Headers containing more information about the resource to be fetched or about the client itself. Examples, Host (specifies the Internet host and port number of the resource being requested), User-Agent (contains information about the user agent originating the request), Accept (specify certain media types which are acceptable for the response), . . . etc.
- Entity header: Headers containing more information about the body of the entity. Examples, Content-Length (indicates the size of the entity-body in decimal number of bytes), Last-Modified (indicates the date and time at which the origin server believes the variant was last modified), . . . etc.

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
```

Method types

- There are several request methods each serves specific purposes, such as,

Method	Description
GET	A client can use the GET request to get a web resource from the server.
POST	Used to post data up to the web server.
HEAD	A client can use the HEAD request to get the header that a GET request would have obtained. Since the header contains the last-modified date of the data, this can be used to check against the local cache copy.
PUT	Ask the server to store the data.
DELETE	Ask the server to delete the data.
CONNECT	Converts the request connection to a transparent TCP/IP tunnel (e.g. connection through a proxy server).
TRACE	Used to tell a proxy to make a connection to another host and simply reply the content, without attempting to parse or cache it. This is often used to secure the connection through a proxy server.
OPTIONS	Ask the server to return the list of HTTP request methods it supports.

- The most commonly used two are GET and POST.

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

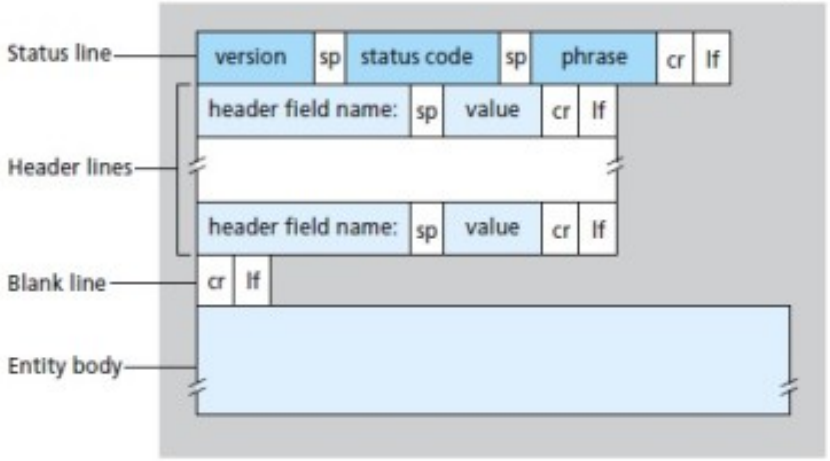
```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: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

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP response message

HTTP Response Message

- The format of an HTTP respond message is as follow:



- The first line is called the status line, followed by optional response header(s).
- The status line has the following syntax:HTTP-version status-code reason-phrase, e.g. HTTP/1.1 200 OK or HTTP/1.0 404 Not Found.
- Response Headers (optional) have the syntax: response-header-name: response-header-value1, request-header-value2, ... For example, Content-Type: text/html or Keep-Alive: timeout=15, max=100.

HTTP response status codes

- status code appears in 1st line in server-to-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