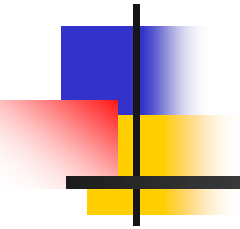
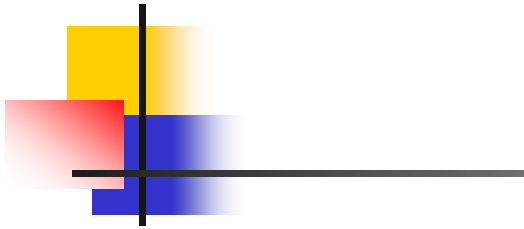
Application Layer



Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 06

puc@marshall.edu



Complete Review Quiz 2 on Blackboard

Non-Persistent and Persistent Connections

- In many Internet applications, the client and server communicate for an extended period of time
  - the client making a series of requests
  - the server responding to each of the requests
  - depending on the application
    - back-to-back, periodically at regular intervals, or intermittently
- Important decision:
  - should each request/response pair be sent over a ***separate*** TCP connection?
    - **non-persistent connections**
  - should all of the requests and their corresponding responses be sent over the ***same*** TCP connection?
    - **persistent connections**

HTTP Connections

To gain a deep understanding of this design issue, let's examine the advantages and disadvantages of persistent connections in HTTP, which can use both non-persistent connections and persistent connections.

(although HTTP uses persistent connections in default mode, but it can be configured to use non-persistent connections.)

Non-persistent HTTP

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

Persistent HTTP

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

Non-Persistent and Persistent Connections (cont.)

- transferring a Web page from server to client using *non-persistent connections*
  - the page consists of a base HTML file and 10 JPEG images
  - all 11 of these objects reside on the same server
  - url for the base HTML

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

HTTP Connections: 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 via 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 Connections: 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

time

6. steps 1-5 repeated for each of 10 jpeg objects

Non-Persistent and Persistent Connections (cont.)

- Here is what happens:
  1. The HTTP client process initiates a TCP connection to the server www.someSchool.edu on port number 80, which is the default port number for HTTP.
     - Associated with the TCP connection, there will be a socket at the client and a socket at the server.
  2. The HTTP client sends an HTTP request message to the server via its socket. The request message includes the path name /someDepartment/home.index.
  3. The HTTP server process receives the request message via its socket, retrieves the object /someDepartment/home.index from its storage (RAM or disk), encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.

Non-Persistent and Persistent Connections (cont.)

- Here is what happens:
  4. The HTTP server process tells TCP to close the TCP connection.
  5. The HTTP client receives the response message. The TCP connection terminates. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds references to the 10 JPEG objects.
  6. The first five steps are then repeated for each of the referenced JPEG objects.

Non-Persistent and Persistent Connections (cont.)

- As the browser receives the Web Page, it displays the page to the user
- Different browsers may interpret a Web page in different ways
  - HTTP has nothing to do with Web page interpretation
- HTTP specifications define only the communication protocol
  - between the client HTTP program and the server HTTP program

Non-Persistent and Persistent Connections (cont.)

- non-persistent connections
  - each TCP connection is closed after the server sends the object
    - the connection does not persist for other objects
  - each TCP connection transports exactly one request msg. and one response msg.
  - in previous example, 11 TCP connections are generated

Non-Persistent and Persistent Connections (cont.)

- Question: whether the client obtains the 10 JPEGs over 10 TCP connections, or whether some of JPEGs are obtained over parallel TCP connections?
  - user configuration on browsers
    - most browsers open 5 to 10 parallel TCP connections
      - each of these connections handles one request-response transaction
    - user can set the max number of parallel connections to one
      - 10 connections are established serially
  - the use of parallel connections shortens the response time

Non-persistent HTTP Connections: Response Time

Q: the amount of time that elapses from when a client requests the base HTML file until the entire file is received by the client?

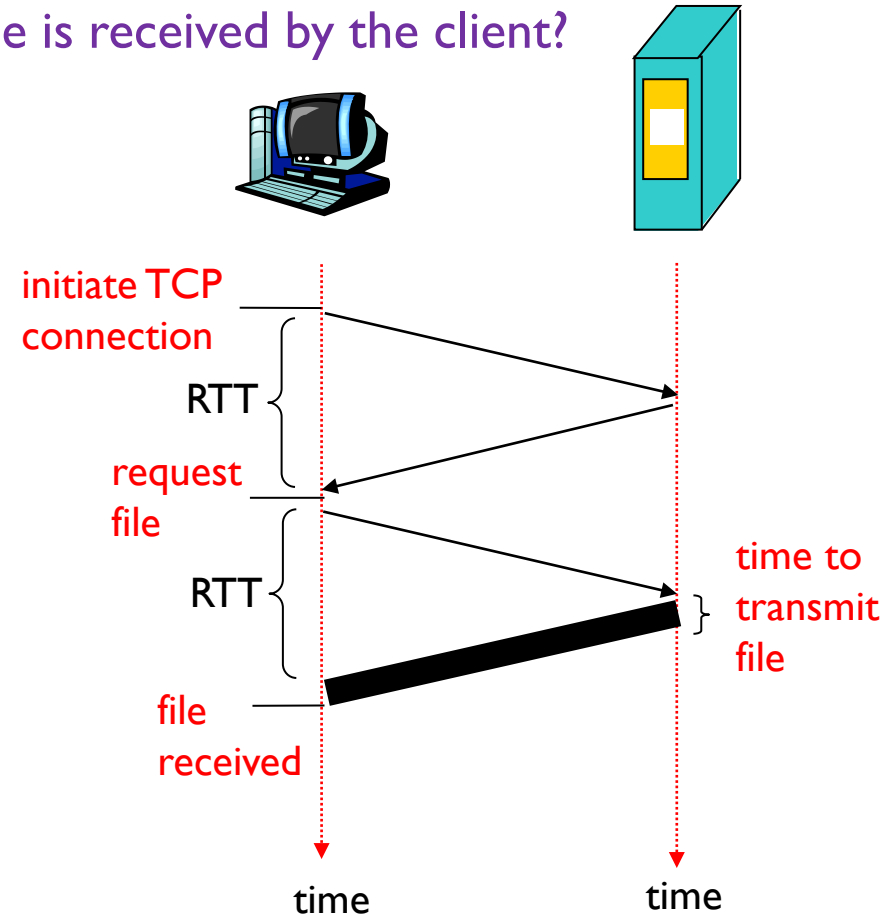
Definition of Round-Trip Time (RTT): time for a small packet to travel from client to server and back to the client.

- packet-propagation delays
- packet queuing delays
- packet-processing delays

HTTP Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and third part of three-way handshake
- file transmission time

total = 2RTT + transmit time



Persistent HTTP Connections

Non-persistent HTTP issues:

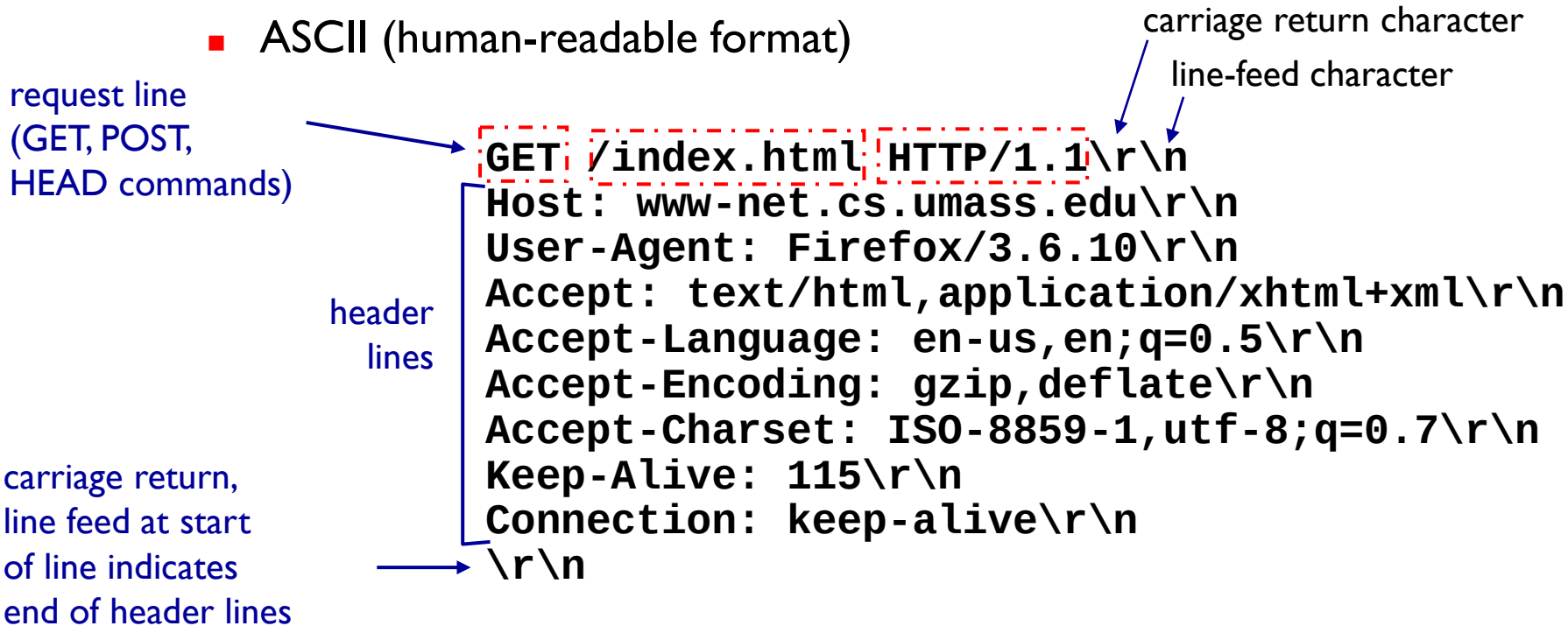
- a brand-new connection established and maintained for each requested object
  - OS overhead for *each* TCP connection
- require 2 RTTs per object
  - one RTT to establish the TCP connection
  - one RTT to request and receive object

Persistent HTTP

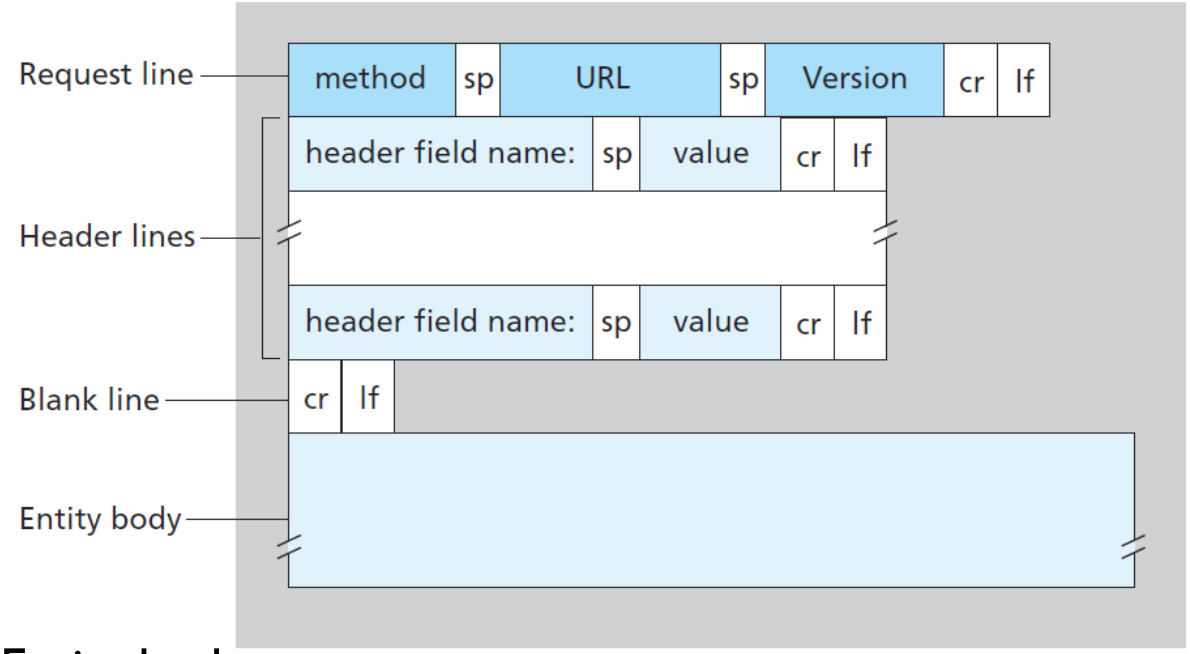
- server **leaves connection open** after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP Request Message

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



HTTP Request Message: General Format



- Entity body:
  - It is empty with the GET method
  - It is used with the POST method
    - Using POST method when filling out a form
    - The entity body contains what user entered into the form

HTTP Request Message: Method Types

HTTP/1.0

- **GET**
  - requests the specified resource. retrieve data only
- **POST**
  - submits an entity to the specified resource
- **HEAD**
  - asks server to leave requested object out of response
  - just receive HTTP header info

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 Response Message

status line (protocol status code status phrase)

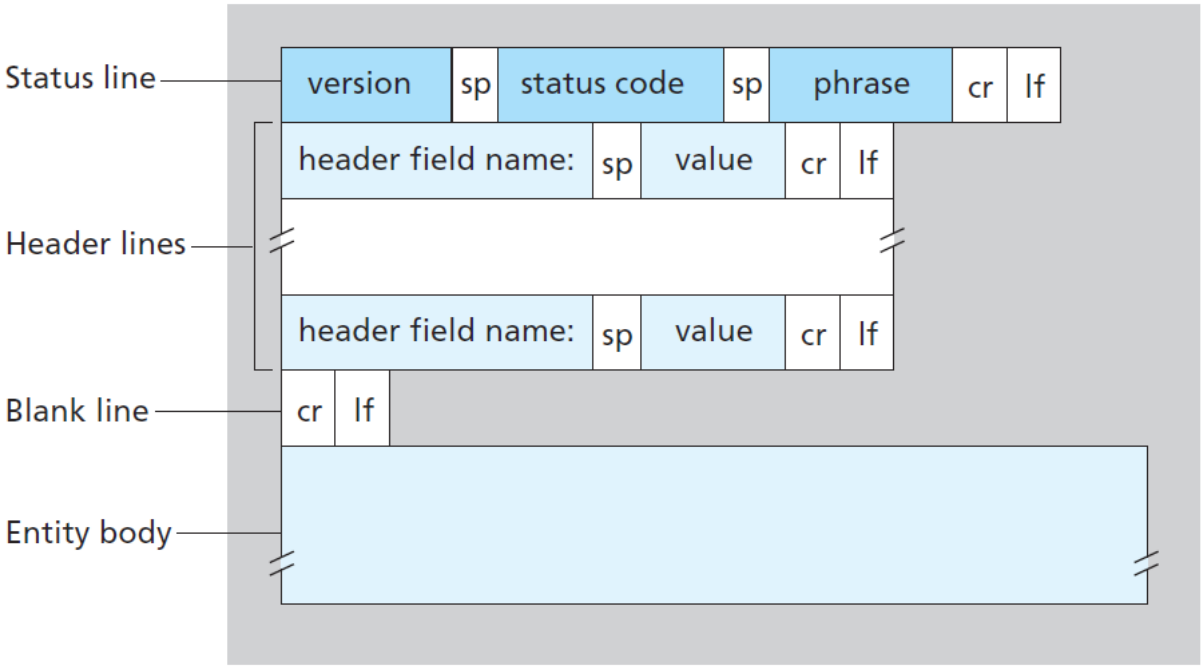
header lines

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 ...
```

ETag: an identifier for a specific version of a resource

HTTP Response Message: General Format



HTTP Response Message: Status Code

- status code appears in first line in server-to-client response message
  - e.g., a few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

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

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported