CSE2029: Data Communication & Computer Networks

Lecture-5: Web and HTTP

Faculty: Dr. Sandeep Kumar

Outline

- *World Wide Web*
- *Overview of HTTP*
- *HTTP over Non-Persistent and Persistent Connections*
- *Request-Response Time Calculations in HTTP*

World Wide Web

- Until the early 1990s, the Internet was used primarily by researchers, academics, and university students to log in to remote hosts, to transfer files from local hosts to remote hosts and vice versa, to receive and send news, and to receive and send electronic mail.
- Although these applications were extremely useful, the Internet was essentially unknown outside of the academic and research communities.
- Then, in the early 1990s, a major new application arrived on the scene—the **World Wide Web** [Berners-Lee 1994]. The Web was the first Internet application that caught the general public's eye. It dramatically changed how people interact inside and outside their work environments. It elevated the Internet from just one of many data networks to essentially the one and only data network.

World Wide Web

- Perhaps what appeals the most to users is that the **Web operates on demand**. Users receive what they want, when they want it.
- This is unlike traditional broadcast radio and television, which force users to tune in when the content provider makes the content available.
- In addition to being available on demand, the Web has many other wonderful features that people like the most. It is enormously easy for any individual to make information available over the Web—everyone can become a publisher at extremely low cost.
- Hyperlinks and search engines help us navigate through an ocean of information. Forms, JavaScript, video, and many other devices enable us to interact with pages and sites. And the Web and its protocols serve as a platform for YouTube, Web-based e-mail (such as Gmail), and most mobile Internet applications, including Instagram and Google Maps.

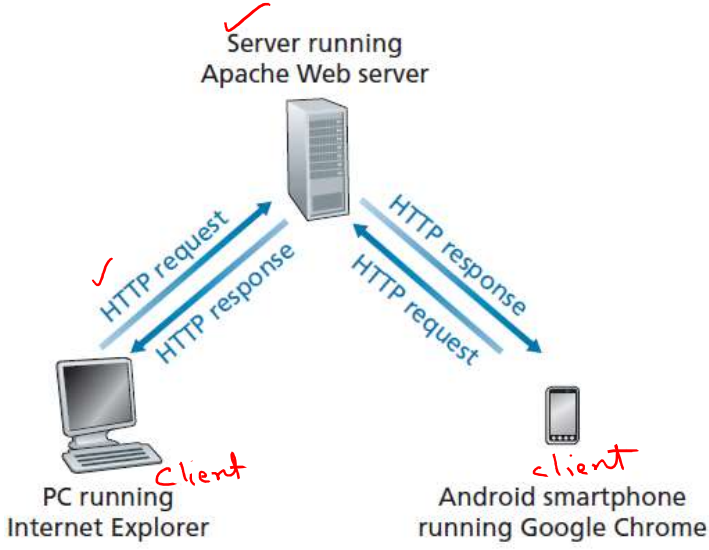
Overview of HTTP

- The HyperText Transfer Protocol (HTTP), the Web's application-layer protocol, is at the heart of the Web.
- HTTP is implemented in two programs: a client program and a server program. The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and how the client and server exchange the messages.
- HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients. The general idea is illustrated in Figure (on next slide)

Overview of HTTP

- When a user requests a Web page (for example, clicks on a hyperlink), the browser sends HTTP request messages for the objects in the page to the server.
- The server receives the requests and responds with HTTP response messages that contain the objects.

- HTTP uses TCP as its underlying transport protocol (rather than running on top of UDP). The HTTP client first initiates a TCP connection with the server. Once the connection is established, the browser and the server processes access TCP through their socket interfaces.



♦ HTTP request-response behavior

Overview of HTTP

- TCP provides a reliable data transfer service to HTTP. This implies that each HTTP request message sent by a client process eventually arrives intact at the server; similarly, each HTTP response message sent by the server process eventually arrives intact at the client.
- Here we see one of the great advantages of a layered architecture—HTTP need not worry about lost data or the details of how TCP recovers from loss or reordering of data within the network. That is the job of TCP and the protocols in the lower layers of the protocol stack.
- It is important to note that the server sends requested files to clients without storing any state information about the client. If a particular client asks for the same object twice in a period of a few seconds, the server does not respond by saying that it just served the object to the client; instead, the server resends the object, as it has completely forgotten what it did earlier. Because an HTTP server maintains no information about the clients, HTTP is said to be a **stateless protocol**.

HTTP: Non-Persistent and Persistent Connections

- In many Internet applications, the client and server communicate for an extended period of time, with the client making a series of requests and the server responding to each of the requests.
- Depending on the application and on how the application is being used, the series of requests may be made back-to-back, periodically at regular intervals, or intermittently.
- When this client-server interaction is taking place over TCP, the application developer needs to make an important decision—should each request/response pair be sent over a separate TCP connection, or should all of the requests and their corresponding responses be sent over the same TCP connection? In the former approach, the application is said to use **non-persistent connections**; and in the latter approach, **persistent connections**.
- To gain a deep understanding of this design issue, we have to examine the advantages and disadvantages of persistent connections in the context of a specific application, namely, HTTP, which can use both non-persistent connections and persistent connections. Although HTTP (version1.1) uses persistent connections in its default mode, HTTP clients and servers can be configured to use non-persistent connections as well.

HTTP with Non-Persistent Connections

- Let's walk through the steps of transferring a Web page from server to client for the case of non-persistent connections. Let's suppose the page consists of a *base HTML file and 10 JPEG images*, and all these 11 objects reside on the same server. Further suppose the URL for the base HTML file is:

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

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.

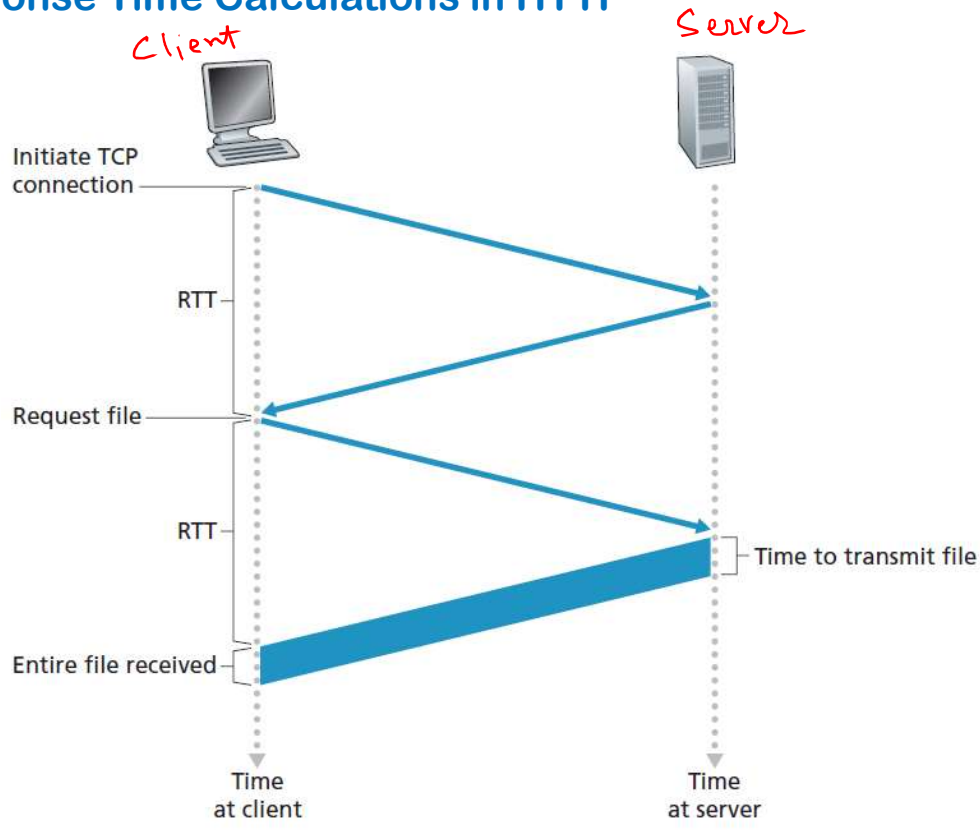
HTTP with Non-Persistent Connections

- (4) The HTTP server process tells TCP to close the TCP connection. (But TCP does not actually terminate the connection until it knows for sure that the client has received the response message intact.)
- (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 four steps are then repeated for each of the referenced 10 JPEG objects.

- The steps above illustrate the use of non-persistent connections, where each TCP connection is closed after the server sends the object—the connection does not persist for other objects. **HTTP/1.0** employs non-persistent TCP connections. Note that each non-persistent TCP connection transports exactly one request message and one response message. Thus, in this example, when a user requests the Web page, 11 TCP connections are generated.

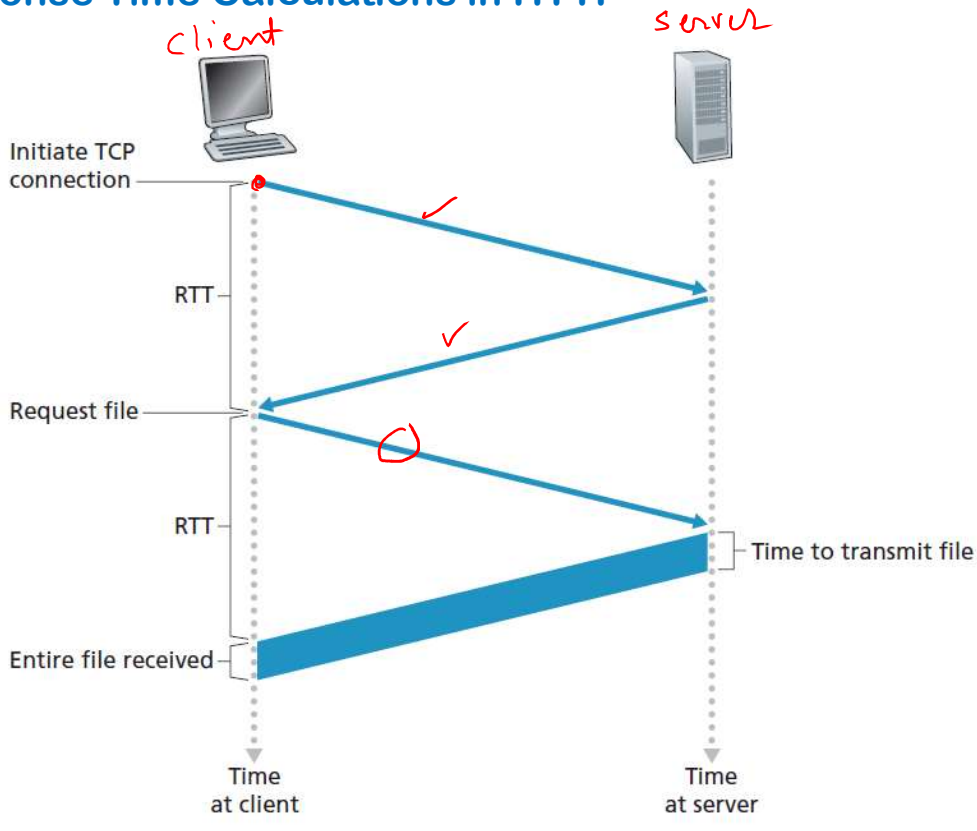
Request-Response Time Calculations in HTTP

- Estimate the amount of time that elapses from when a client requests a base HTML file until the entire file is received by the client.
- To this end, we define the **round-trip time (RTT),** which is the time it takes for a small packet to travel from client to server and then back to the client.
- The RTT includes packet-propagation delays, packet queuing delays in intermediate routers and switches, and packet-processing delays.



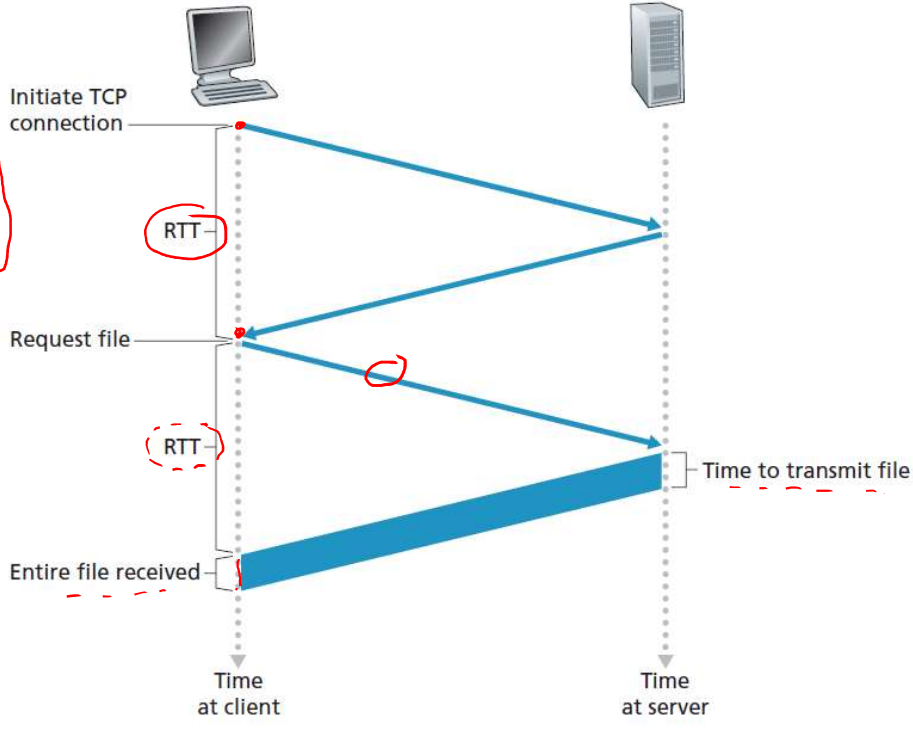
Request-Response Time Calculations in HTTP

- Consider what happens when a user clicks on a hyperlink. As shown in Figure, this causes the browser to initiate a TCP connection between the browser and the Web server.
- This involves a "three-way handshake"—the client sends a small TCP segment to the server, the server acknowledges and responds with a small TCP segment, and, finally, the client acknowledges back to the server.



Request-Response Time Calculations in HTTP

- The first two parts of the three-way handshake take one RTT. After completing the first two parts of the handshake, the client sends the HTTP request (for file) message combined with the third part of the three-way handshake (the acknowledgment) into the TCP connection.
- Once the request message arrives at the server, the server sends the HTML file into the TCP connection. This HTTP request/response eats up another RTT.
- Thus, roughly, the total response time is two RTTs plus the transmission time at the server of the HTML file.



HTTP with Persistent Connections

- Non-persistent connections have some shortcomings. First, a brand-new connection must be established and maintained for each requested object. For each of these connections, TCP buffers must be allocated and TCP variables must be kept in both the client and server. This can place a significant burden on the Web server, which may be serving requests from hundreds of different clients simultaneously.
- Second, as we just described, each object suffers a delivery delay of two RTTs—one RTT to establish the TCP connection and one RTT to request and receive an object.
- With **HTTP/1.1 persistent connections**, the server leaves the TCP connection open after sending a response. Subsequent requests and responses between the same client and server can be sent over the same connection.
- In particular, an entire Web page (in the previous example, the base HTML file and the 10 images) can be sent over a single persistent TCP connection.

Thank you.