Assignment 2

[037831852]

Review Questions

R4)

Why are the terms client and server still used in peer-to-peer applications?

In peer-to-peer (P2P) applications, the terms *client* and *server* are still used because they describe the temporary roles nodes. A node acts as a *client* when it requests data or services and as a *server* when it provides them. Additionally, many P2P protocols are derived from traditional client-server models, making the use of these terms convenient for describing network interactions.

R5)

What information is used by a process running on one host to identify a process running on another host?

A process running on one host uses the following information to identify and communicate with a process on another host:

- IP Address: The Internet Protocol (IP) address uniquely identifies the destination host on the network.
- Port Number: The port number identifies the specific process or application running on the destination host. Each process is associated with a unique port number on the host.
- Transport Protocol: The transport layer protocol, such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), defines the rules for communication between processes.

Together, the combination of the destination host's IP address, port number, and transport protocol is used to establish communication between processes on different hosts.

R12)

How can websites keep track of users? Do they always need to use cookies?

Websites employ several methods to keep track of users, and while cookies are a common tool, they are not the only option. Here are some key methods:

- **Cookies:** Small pieces of data stored on the user's device by the web browser.
  Usage: Cookies can store user preferences, session identifiers, and tracking information. They help maintain user sessions and personalize the browsing experience.

- **Web Cache/Proxy Servers**:

i. **Caching:** Proxy servers cache web content to improve performance and reduce server load. Cached content can include user-specific data, which helps in quicker retrieval of frequently accessed resources.
ii. **Conditional GET:** This technique allows proxy servers to check if the cached content is still valid. By sending a conditional GET request to the original server, the proxy can determine if the cached resource has been modified. If not, the cached version is served, thus minimizing the need for fresh data retrieval from the origin server.

- **Other Methods**:
  i. **Session Storage:** Temporarily stores data within the browser session, which is cleared when the browser is closed.
  ii. **Local Storage:** Allows for persistent data storage on the user's device, which remains until explicitly cleared.
  iii. **Fingerprinting:** Collects device-specific information to create a unique user profile without relying on cookies.
  iv. **Web Beacons:** Invisible graphics embedded in web pages or emails that track user activity.

R13)

Describe how Web caching can reduce the delay in receiving a requested object. Will Web caching reduce the delay for all objects requested by a user or for only some of the objects? Why?

Web Caching Overview: Web caching, implemented through proxy servers, helps reduce delays in delivering requested objects by storing copies of frequently accessed content closer to the user. Here's how it works and its impact on delay:

Working:

A user configures their browser to send HTTP requests to a web cache. The cache checks if the requested object is stored:

- **Cache Hit:** If the object is in the cache, it is returned directly to the client, reducing the need for further communication with the origin server.
- **Cache Miss:** If the object is not in the cache, the cache requests it from the origin server, stores it, and then returns it to the client.

Reduction in Delay:

- **Closer Access:** The cache is typically located closer to the client than an ISP, reducing the round-trip time (RTT) compared to retrieving the object directly from a remote origin server.
- **Reduced Traffic:** By serving content from the cache, the amount of traffic over the access link to the origin server is reduced, which decreases network congestion and delay.

Will Web caching reduce the delay for all objects requested by a user or for only some of the objects? Why?

Web caching does not reduce the delay for all objects uniformly. Caching effectiveness depends on the cache hit rate and the nature of the objects. Web caching does not reduce the delay for all objects uniformly. Caching effectiveness depends on the cache hit rate and the nature of the objects:

- **Frequently Accessed Objects:** Objects that are frequently accessed are more likely to be cached, leading to reduced delay when these objects are requested.
- **Less Frequently Accessed Objects:** Objects that are infrequently accessed or not cached will still experience delays due to the need to retrieve them from the origin server.

Thus,

Without caching, a request for a web object involves the access link, the institutional network, and the RTT to the origin server, resulting in significant delays, especially with high utilization. With caching, if the cache hit rate is 40%, 40% of requests are served directly from the cache with minimal delay, while only 60% of requests incur the longer delay of contacting the origin server. This leads to a lower average end-to-end delay compared to scenarios without caching.

So, Web caching does not reduce the delay for all objects uniformly. web caching can significantly reduce delays by serving content closer to the client and reducing traffic on the access link. However, it only reduces delays for objects that are cached, meaning that less frequently accessed objects may not benefit as much.

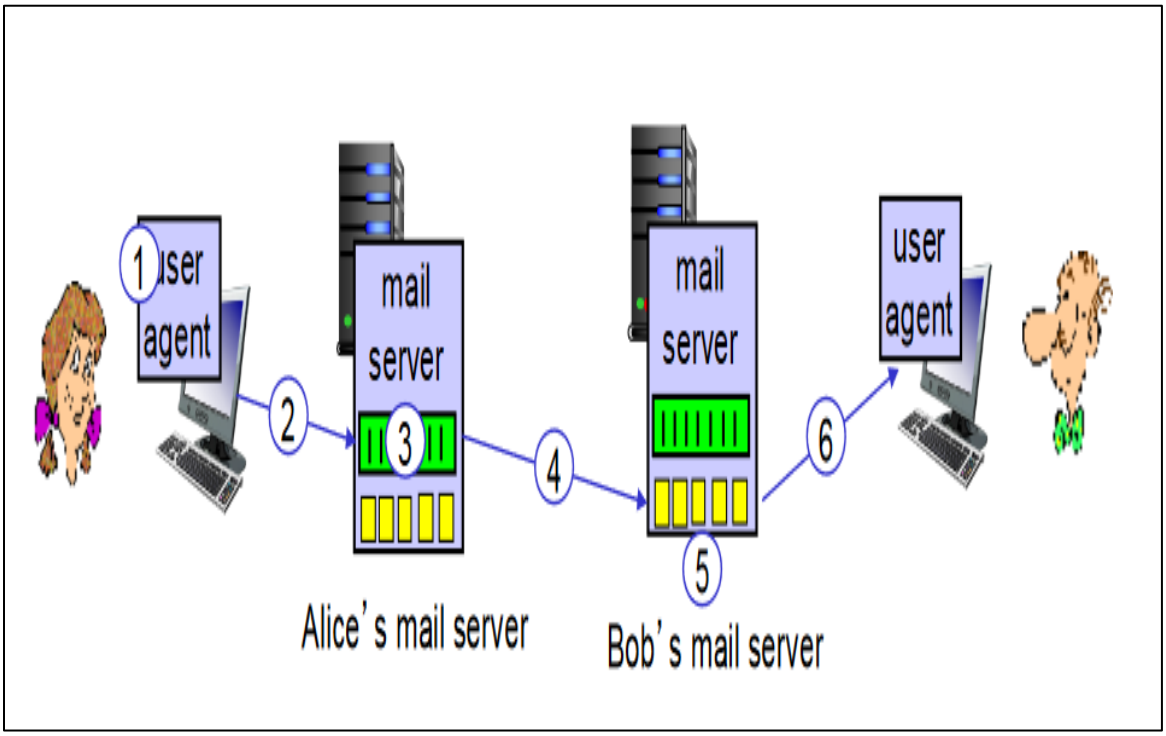
R16)

Suppose Alice, with a Web-based e-mail account (such as Hotmail or Gmail), sends a message to Bob, who accesses his mail from his mail server using IMAP. Discuss how the message gets from Alice's host to Bob's host. Be sure to list the series of application-layer protocols that are used to move the message between the two hosts.

When Alice sends an email to Bob, using a web-based email service (like Hotmail or Gmail) and Bob accessing his mail through IMAP, the following series of application-layer protocols and 6 steps are involved:

- **Alice Composes the Email:**
  Alice drafts her email message and specifies Bob's address (e.g., bob@someschool.edu) using her web-based email interface. Protocol Used: HTTP (Hypertext Transfer Protocol) is used to send the email from Alice's web browser to her email service server. This protocol enables the interaction between Alice's user agent (UA) and her email provider's server.
- **Email Submission to Alice's Mail Server:**
  Alice's web-based email client submits the email to her mail server. The email is placed in the server's message queue. Protocol Used: SMTP (Simple Mail Transfer Protocol) is employed to transfer the email from Alice's user agent to her mail server, preparing it for further transmission.
- **SMTP Client Opens TCP Connection:**
  The client side of SMTP on Alice's mail server opens a TCP connection with Bob's mail server to facilitate the transfer of the email. Protocol Used: SMTP is responsible for opening and managing the TCP connection to ensure reliable email delivery.

- **Email Transfer to Bob's Mail Server:**
  SMTP client on Alice's mail server sends the email over the established TCP connection to Bob's mail server. Protocol Used: SMTP continues to handle the email transmission until it reaches Bob's mail server.
- **Email Delivery to Bob's Mailbox:**
  Bob's mail server receives the email and places it into Bob's mailbox. Protocol Used: SMTP ensures that the email is properly delivered and stored on Bob's mail server.
- **Bob Retrieves and Reads the Email:**
  Bob uses his email client (user agent) to connect to his mail server and retrieve the new email message. Protocol Used: IMAP (Internet Message Access Protocol) is used by Bob's email client to access, organize, and manage his emails stored on the mail server. IMAP allows Bob to view and synchronize the new email with his client.



Summary of Protocols Used:

HTTP: Facilitates communication between Alice's web-based email client and the email service server.
SMTP: Handles the transmission of the email from Alice's server to Bob's server.
IMAP: Manages the retrieval and organization of emails by Bob's email client from his mail server.

R23)

Assume a BitTorrent tracker suddenly becomes unavailable. What are its consequences? Can files still be downloaded?

If a BitTorrent tracker becomes unavailable, but files may still be downloaded in some cases.

Consequence:

- **Slower Discovery of Peers:** A tracker helps peers (clients) find each other. If the tracker goes offline, new peers won't easily discover other peers.
- **No Peer List Updates:** Peers regularly contact the tracker to update their peer list. Without this, they may not know about the full set of peers, limiting potential download speeds.

Can Files Still Be Downloaded?

Yes, files can still be downloaded under certain conditions:

- **DHT (Distributed Hash Table):** If the torrent client supports DHT, it allows peer discovery in a decentralized way, independent of the tracker. This enables downloading even without a functioning tracker.
- **Peer Exchange (PEX):** Some clients can share information about other peers with the ones they are connected to, which helps maintain connections even without a tracker.
- **Previously Connected Peers:** If your client is already connected to some peers before the tracker went down, it can continue downloading from them.

R25)

Besides network-related considerations such as delay, loss, and bandwidth performance, there are other important factors that go into designing a CDN server selection strategy. What are they?

When designing a CDN (Content Delivery Network) server selection strategy, several factors beyond network-related considerations like delay, loss, and bandwidth performance need to be considered. Key additional factors include:

- Geographic Location of Users
- Server Load and Capacity
- Content Type and Size
- Server Health and Availability
- User Preferences and Behavior
- Redundancy and Fault Tolerance
- Regulatory and Compliance Considerations
- Caching Policies
- Cost Efficiency
- Security Features
- CDN Edge Capabilities

Problem Questions

P1.

True or false?
a. A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.
b. Two distinct Web pages (for example, www.mit.edu/research.html and www.mit.edu/students.html) can be sent over the same persistent connection.
c. With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.
d. The Date: header in the HTTP response message indicates when the object in the response was last modified.
e. HTTP response messages never have an empty message body.

Answer:

a. **FALSE**, the client sends one request for the Web page and receives separate responses for the page and each image.
b. **TRUE**, Persistent connections allow multiple requests and responses over the same connection.
c. **FALSE**, Nonpersistent connections require a new TCP connection for each HTTP request/response pair.
d. **FALSE**, The `Date:` header indicates when the response was generated, not when the object was last modified. The `Last-Modified:` header provides the last modification time.
e. **FALSE**, HTTP responses can have an empty body, such as in the case of a 204 No Content response.

P7.

Suppose within your Web browser, you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT1, ..., RTTn. Further suppose that the Web page associated with the link contains exactly one object, consisting of a large amount of HTML text. Let RTT0 denote the RTT between the local host and the server containing the object. Assuming transmission duration of 0.002 * RTT0 of the object, how much time elapses from when the client clicks on the link until the client receives the object?

The steps involved in the process is summarized as follows:

1. **Click on the Link:** You initiate the request for a web page by clicking a link in your browser.
2. **DNS Lookup:**
    - Check Cache: The browser checks if the IP address for the URL is cached locally.

- DNS Query: If not cached, the browser sends a DNS query to resolve the URL to an IP address.
- Visit DNS Servers: The DNS query is passed through n DNS servers to resolve the IP address. Each visit incurs a Round Trip Time (RTT).

3. **Calculate DNS Lookup Time:**
   - Total DNS Lookup Time = RTT1 + RTT2 + ... + RTTn
4. **Send Request for Web Page:**
   - Establish Connection: The browser establishes a connection to the web server using the resolved IP address.
   - Send Request: The browser sends an HTTP request to the server for the web page.
5. **Receive Response:**
   - Round Trip Time (RTT0): The time for the request to travel to the server and the response to come back is RTT0.
   - Transmission Duration: Time required to transmit the HTML object, given as 0.002 * RTT0.
6. **Calculate Time to Fetch the Object:**
   - Total Time to Fetch the Object = RTT0 + Transmission Duration
   - Transmission Duration = 0.002 * RTT0
   - Total Time to Fetch the Object = 1.002 * RTT0
7. **Calculate Total Elapsed Time:**
   - Total Time Elapsed = Total DNS Lookup Time + Time to Fetch the Object
   - Total Time Elapsed = (RTT1 + RTT2 + ... + RTTn) + 1.002 * RTT0

P8.

Consider Problem P7 again and assume RTT0 = RTT1 = RTT2 = ... = RTTn = RTT. Furthermore, assume a new HTML file, small enough to have negligible transmission time, which references nine equally small objects on the same server. How much time elapses with:
a. Non-persistent HTTP with no parallel TCP connections?
b. Non-persistent HTTP with the browser configured for 6 parallel connections?
c. Persistent HTTP?

a. Non-persistent HTTP with no parallel TCP connections

Process:

- DNS Lookup Time:
  Time to resolve the IP address through n DNS servers.
  Total DNS Lookup Time: **n × RTT**
- Fetching the HTML File:
  Establish TCP Connection: 1 RTT
  Send HTTP Request and Receive Response: 1 RTT
  Total Time for HTML File: **2 RTT**

- Fetching the Objects:
  Each of the 9 objects requires a new TCP connection.
  Time per Object: 2 RTT (1 RTT for connection setup + 1 RTT for request/response)
  Total Time for 9 Objects: 9 objects × 2 RTT = **18 RTT**
- Total Time Elapsed:
  Total Time=DNS Lookup Time + Time for HTML File + Time for Objects
  Total Time=(n×RTT) + 2 RTT + 18 RTT
  Total Time=(n+20) RTT

b. Non-persistent HTTP with the browser configured for 6 parallel connections

Process:

- DNS Lookup Time:
  Time to resolve the IP address through n DNS servers.
  Total DNS Lookup Time: **n × RTT**
- Fetching the HTML File:
  Establish TCP Connection: 1 RTT
  Send HTTP Request and Receive Response: 1 RTT
  Total Time for HTML File: **2 RTT**
- Fetching the Objects:
  Parallel Connections: Up to 6 objects can be fetched simultaneously.
  Batch 1: Fetch 6 objects in parallel.
  Time for Batch 1: 2 RTT (1 RTT for establishing connections + 1 RTT for request/response)
  Batch 2: Fetch remaining 3 objects.
  Time for Batch 2: 2 RTT
- Total Time for Objects:
  Total Time: 2 RTT (Batch 1) + 2 RTT (Batch 2) = **4 RTT**
- Total Time Elapsed:
  Total Time=DNS Lookup Time + Time for HTML File + Time for Objects
  Total Time=(n×RTT) + 2 RTT + 4 RTT
  Total Time=(n+6) RTT

c. Persistent HTTP

Process:

- DNS Lookup Time:
  Time to resolve the IP address through n DNS servers.
  Total DNS Lookup Time: **n × RTT**
- Fetching the HTML File:
  Establish TCP Connection: 1 RTT
  Send HTTP Request and Receive Response: 1 RTT
  Total Time for HTML File: **2 RTT**
- Fetching the Objects:
  Persistent Connection: Reuses the same TCP connection for all requests.

Time per Object: 1 RTT (1 RTT for request/response)
Total Time for 9 Objects: 9 objects × 1 RTT = **9 RTT**

- Total Time Elapsed:
  Total Time=DNS Lookup Time + Time for HTML File + Time for Objects
  Total Time=(n×RTT)+2 RTT+9 RTT
  Total Time=(n+11) RTT

SUMMARY:

Scenario	DNS Lookup Time	Time for HTML File	Time for Objects	Total Time Elapsed
a. Non-persistent HTTP, no parallel	n×RTT	2 RTT	18 RTT (9 × 2 RTT)	n+20 RTT
b. Non-persistent HTTP, 6 parallel	n×RTT	2 RTT	4 RTT (2 RTT for 6 objects + 2 RTT for 3 objects)	n+6 RTT
c. Persistent HTTP	n×RTT	2 RTT	9 RTT (9 × 1 RTT)	n+11 RTT

Note: For all scenarios, since the first object is sent along with the HTML file request, 2 RTT is saved for the first object. This adjustment is reflected in the updated table below.

Scenario	DNS Lookup Time	Time for HTML File	Time for Objects	Total Time Elapsed
a. Non-persistent HTTP, no parallel	n×RTT	2 RTT	16 RTT (8 × 2 RTT)	n+18 RTT
b. Non-persistent HTTP, 6 parallel	n×RTT	2 RTT	2 RTT (6 parallel objects + 1 RTT)	n+4 RTT
c. Persistent HTTP	n×RTT	2 RTT	9 RTT (9 × 1 RTT)	n+10 RTT

P9.

Consider Figure 2.12, for which there is an institutional network connected to the Internet. Moreover, assume the access link has been upgraded to 54 Mbps, and the institutional LAN is upgraded to 10 Gbps. Suppose that the average object size is 1,600,000 bits and that the average request rate from the institution's browsers to the origin servers is 24 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is 3 seconds on average (see Section 2.2.5). Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use Δ/(1 - Δb), where Δ is the average time required to send an object over the access link and b is the arrival rate of objects to the access link.
a. Find the total average response time.
b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.3. Find the total response time.

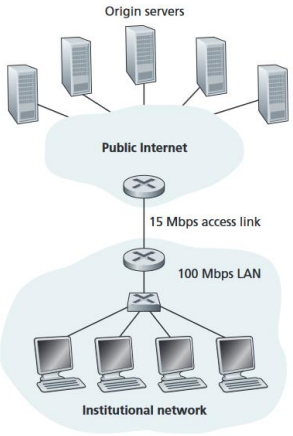


Figure 2.12 ♦ Bottleneck between an institutional network and the Internet

Given Data:

- Access link capacity = 54 Mbps
- LAN capacity = 10 Gbps
- Average object size = 1,600,000 bits
- Average request rate = 24 requests per second
- Internet delay (time from sending request to receiving response) = 3 seconds
- Cache miss rate: 0.3

Where:

- O = Average object size = 1,600,000 bits
- C = Access link capacity = 54 Mbps
- R = Request rate = 24 requests/second
- D_Internet = Average Internet delay = 3 seconds
- Δ = O/C = Time to send one object over the access link
- b = R = 24 requests/second

Access Delay Formula:

- The average access delay, D_access, is given by:

$$D_{access} = \frac{\Delta}{1-\Delta b}$$

- Total Average Response Time Formula: The total average response time, D_total, is the sum of the access delay and the Internet delay:

$$D\_total = D\_access + D\_Internet$$

a. Find the total average response time.

$$\Delta = \frac{O}{C} = \frac{1600000\ bits}{54\ X\ 10^6\ bps} = 0.0296\ seconds$$

$$D_{access} = \frac{\Delta}{1-\Delta b} = \frac{0.0296\ seconds}{1-(0.0296\ X\ 24)} = 0.102209$$

$$D_{total} = D_{access} + D_{Internet} = 0.102209 + 3 = 3.1022\ seconds$$

Thus, the **total average response time** is approximately **3.10 seconds**.

b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.3. Find the total response time.

The total response time with caching is a weighted average of the cache hit and miss times:

$$D\_cache = (0.7 \times LAN\ time) + (0.3 \times D\_total)$$

Since LAN time is approximately 0 seconds:

$$D\_cache = 0.7 \times 0 + 0.3 \times 3.10256 = 0.93077\ seconds$$

Thus, the **total response time with caching** is approximately **0.93 seconds**.

P25.

Consider an overlay network with N active peers, with each pair of peers having an active TCP connection. Additionally, suppose that the TCP connections pass through a total of M routers. How many nodes and edges are there in the corresponding overlay network?

In an overlay network where each pair of N active peers has an active TCP connection, the network can be represented as a complete graph. And the total number of routers MMM does not affect the count of nodes or edges in the overlay network.

Nodes: In the overlay network, each active peer is a node. Therefore, the number of nodes is **N.**

Edges: Each pair of peers is connected by an edge. In a complete graph with NNN nodes, the number of edges is given by the combination formula ${}^{N}C_2$, which is:

$$\textbf{Number of edges} = \frac{N\ (N-1)}{2}$$

P27.

Consider a DASH system for which there are N video versions (at N different rates and qualities) and N audio versions (at N different rates and qualities). Suppose we want to allow the player to choose at any time any of the N video versions and any of the N audio versions.
a. If we create files so that the audio is mixed in with the video, so the server sends only one media stream at a given time, how many files will the server need to store (each a different URL)?
b. If the server instead sends the audio and video streams separately and has the client synchronize the streams, how many files will the server need to store?

Given:

Number of video versions: N
Number of audio versions: N

a. Audio Mixed with Video

In this scenario, the server sends a combined media stream (audio mixed with video), meaning each file contains both audio and video. Each video version is mixed with each audio version to create a unique file.

Each combination of video and audio requires a separate file. Thus, the number of files needed is:

Number of files=N×N=**N^2**

b. Audio and Video Streams Separately

In this scenario, the server sends audio and video streams separately, and the client is responsible for synchronizing them.

The server needs to store: N separate audio files & N separate video files.

Therefore, the total number of files the server needs to store is:

Number of video files + Number of audio files=N + N = **2N**