Chapter 2, Section 2.1 of the book "Computer Networking: A Top-Down Approach" by Kurose and Ross, along with clarifications based on general networking knowledge.

R1. List five nonproprietary Internet applications and the application-layer protocols that they use.

1. **Web Browsing:** HTTP/HTTPS (HyperText Transfer Protocol / Secure)
2. **Email:** SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol), POP3 (Post Office Protocol 3)
3. **File Transfer:** FTP (File Transfer Protocol), SFTP (SSH File Transfer Protocol)
4. **Instant Messaging:** XMPP (Extensible Messaging and Presence Protocol)
5. **Domain Name Resolution:** DNS (Domain Name System)

Each of these applications uses a nonproprietary protocol at the application layer that enables communication across the Internet, with specific functionalities for their use cases. These protocols are standardized by bodies like IETF.

R2. What is the difference between network architecture and application architecture?

- **Network Architecture** refers to the design and structure of the underlying network, including layers such as physical, data link, network, transport, etc. It governs how data is transmitted and routed through a network.
- **Application Architecture** refers to how an application is designed in terms of communication. This could be client-server (e.g., web browsing) or peer-to-peer (P2P file sharing). It focuses on how the application interacts with the network and its own operational structure.

While network architecture is broader and includes the fundamental operations of the internet (e.g., routing, switching), application architecture is concerned with how specific applications are structured over the network.

R3. For a communication session between a pair of processes, which process is the client and which is the server?

In a communication session, the **client** is the process that initiates the communication, while the **server** is the process that waits for the client to make a request and then responds to it. For instance, in web browsing, the browser (client) requests web pages from a web server.

R4. For a P2P file-sharing application, do you agree with the statement, “There is no notion of client and server sides of a communication session”? Why or why not?

Yes, I agree. In a **P2P (Peer-to-Peer)** file-sharing application, every node (or peer) acts both as a client and a server. Each peer can request files (client behavior) and serve files to others (server behavior). This contrasts with traditional client-server models where roles are more clearly defined.

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 a process on another host:

- **IP Address** of the host
- **Port Number** assigned to the specific process on that host This combination of IP address and port number is called a **socket**.

R6. Suppose you wanted to do a transaction from a remote client to a server as fast as possible. Would you use UDP or TCP? Why?

You would use **UDP (User Datagram Protocol)** because it is a connectionless protocol that has minimal overhead compared to TCP (Transmission Control Protocol). UDP doesn't require connection establishment, error-checking, or acknowledgment of receipt, making it faster for applications that prioritize speed over reliability, such as gaming or real-time video streaming.

R7. Can you conceive of an application that requires no data loss and is also highly time-sensitive?

Yes, an example would be **real-time online trading systems**. In such applications, data loss can lead to financial losses, and timing is critical because prices and trading conditions can change in milliseconds. A hybrid approach that minimizes data loss while maintaining time constraints would be ideal.

R8. List the four broad classes of services that a transport protocol can provide. For each service class, indicate if either UDP or TCP (or both) provides such a service.

1. **Reliable Data Transfer:** TCP provides reliable data transfer through acknowledgment and retransmission. UDP does not.
2. **Throughput Guarantee:** Neither TCP nor UDP guarantees throughput, though TCP offers congestion control to optimize it.
3. **Timing Guarantee:** Neither TCP nor UDP provides timing guarantees. Applications that need real-time guarantees often use UDP and implement timing controls at the application layer.
4. **Security:** TCP can be enhanced with TLS to provide security services. UDP typically does not offer built-in security but can be used with DTLS (Datagram TLS).

R9. Does TLS operate at the transport layer or the application layer? If the application developer wants TCP to be enhanced with TLS, what does the developer have to do?

TLS (Transport Layer Security) operates above the transport layer but below the application layer, often viewed as part of the application layer. To enhance TCP with TLS, the developer must implement the TLS protocol in the application code or use libraries that handle the encryption and security handshake between the client and server.

sections 2.2-2.5, integrating the content from the textbook as well as external concepts to clarify them further:

R10. What is meant by a handshaking protocol?

A handshaking protocol refers to the process of two entities (usually a client and a server) establishing communication by sending and receiving signals before actual data transfer begins. This exchange ensures that both parties are ready and agree on the communication rules, such as using TCP for reliable delivery.

For example, **TCP uses a three-way handshake** to establish a connection. The client sends a SYN (synchronize) packet, the server responds with a SYN-ACK, and the client finally replies with an ACK.

R11. Why do HTTP, SMTP, and IMAP run on top of TCP rather than on UDP?

HTTP, SMTP, and IMAP run on TCP because these protocols require **reliable communication**, which includes error-checking, data integrity, and in-order delivery, which TCP ensures. TCP establishes a connection and provides acknowledgments for data sent, ensuring that no data is lost in transmission. UDP, on the other hand, does not guarantee reliable delivery or ordered packets, making it unsuitable for such applications.

R12. Consider an e-commerce site that wants to keep a purchase record for each of its customers. Describe how this can be done with cookies.

An e-commerce site can use **cookies** to store and track information about a customer's purchases. When a user visits the site, the server can store a unique identifier in the browser via a cookie. Each time the customer interacts with the website (e.g., making a purchase), this identifier is sent back to the server, allowing the server to access the corresponding purchase history and update it.

Cookies allow for **stateful interactions** on otherwise stateless HTTP sessions, enabling the server to "remember" a user across multiple page requests.

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 reduces delay by storing copies of frequently requested objects closer to the user, such as on a local server or within a browser cache. When a user requests an object, if it is already cached, it can be served quickly without the need to fetch it again from the original server, which may be located far away.

However, caching only reduces the delay for objects that have been previously requested and stored. For new objects, the client must retrieve them from the original server, experiencing the full delay.

R14. Telnet into a Web server and send a multiline request message. Include in the request message the If-modified-since: header line to force a response message with the 304 Not Modified status code.

You can simulate a request to a web server by using Telnet as follows:

bash
Copy code

```
telnet www.example.com 80
```

Once connected, send an HTTP request:

bash
Copy code

```
GET /index.html HTTP/1.1
Host: www.example.com
If-Modified-Since: Mon, 29 Jun 2020 07:28:00 GMT
```

If the file has not been modified since the specified date, the server will respond with a **304 Not Modified** status code, indicating the file hasn't changed.

R15. List several popular messaging apps. Do they use the same protocols as SMS?

Popular messaging apps include **WhatsApp, Facebook Messenger, Telegram, and Signal**. These apps do not use the same protocols as SMS, which relies on cellular networks' Short Message Service protocol. Instead, messaging apps use **Internet-based protocols** like **XMPP**

or proprietary messaging protocols over TCP/IP, allowing for richer features such as media sharing and encryption.

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.

1. **Alice composes the email on a web-based service** (like Gmail), which sends the message using **SMTP** (Simple Mail Transfer Protocol) to Gmail's outgoing mail server.
2. The outgoing server then forwards the message to **Bob's incoming mail server**, which supports **IMAP** (Internet Message Access Protocol).
3. **Bob's email client** uses **IMAP** to retrieve the message from the mail server.

Thus, the email flows through SMTP between servers, and Bob retrieves the message using IMAP.

R17. Print out the header of an e-mail message you have recently received. How many Received: header lines are there? Analyze each of the header lines in the message.

The **Received** header lines indicate the email's journey through different mail servers. Each line shows a server that handled the email, the time it was received, and the IP address of the server. There will be as many Received lines as the number of hops the email took between servers before reaching your inbox.

R18. What is the HOL blocking issue in HTTP/1.1? How does HTTP/2 attempt to solve it?

Head-of-Line (HOL) blocking in HTTP/1.1 occurs when a delay in receiving one object blocks subsequent objects from being processed, as multiple requests are handled sequentially over a single connection.

HTTP/2 solves this by introducing **multiplexing**, allowing multiple requests and responses to be sent simultaneously over a single connection. This eliminates the blocking that occurs when one object is delayed.

R19. Is it possible for an organization's Web server and mail server to have exactly the same alias for a hostname (for example, foo.com)? What would be the type for the RR that contains the hostname of the mail server?

Yes, an organization's web server and mail server can have the same hostname, like **foo.com**. The **MX (Mail Exchanger) record** type in the DNS specifies the mail server responsible for receiving email messages for that domain. In this case, the DNS configuration would include an **MX record** pointing to the mail server handling email for **foo.com**.

R20. Look over your received e-mails, and examine the header of a message sent from a user with a .edu e-mail address. Is it possible to determine from the header the IP address of the host from which the message was sent? Do the same for a message sent from a Gmail account.

The **Received** headers in the email will show the originating IP addresses for the message. For .edu addresses, the originating mail server's IP can usually be seen. However, Gmail often hides the sender's IP address for privacy reasons, so you may not find the originating IP address in Gmail-sent emails.

sections 2.5–2.7, incorporating both the text from the book and additional clarifications.

R21. In BitTorrent, suppose Alice provides chunks to Bob throughout a 30-second interval. Will Bob necessarily return the favor and provide chunks to Alice in this same interval? Why or why not?

Bob is not required to return the favor immediately in the same 30-second interval. In BitTorrent, peers decide whom to send chunks to based on the **tit-for-tat** mechanism, which incentivizes peers to upload data to those who reciprocate. If Bob is able to upload chunks to Alice, and Alice continues providing useful chunks, Bob may return the favor. However, this depends on factors such as **upload capacity** and **priority decisions** made by Bob at the moment.

R22. Consider a new peer Alice that joins BitTorrent without possessing any chunks. Without any chunks, she cannot become a top-four uploader for any of the other peers, since she has nothing to upload. How then will Alice get her first chunk?

When Alice joins BitTorrent as a new peer with no chunks, she starts by downloading chunks from other peers who allow free riders (initially). Some peers will send her chunks to help her get started, enabling her to begin participating in the tit-for-tat exchange system. Over time, Alice will collect enough chunks to start reciprocating uploads.

R23. What is an overlay network? Does it include routers? What are the edges in the overlay network?

An **overlay network** is a virtual network built on top of an existing network. It operates by creating logical connections between nodes (such as computers or servers) over the physical infrastructure of the internet. Overlay networks do not include traditional network-layer routers,

but instead rely on **application-layer connections**. The edges in an overlay network represent the **logical connections** between participating nodes (peers), rather than physical links as in the internet's underlying network layer.

R24. CDNs typically adopt one of two different server placement philosophies. Name and briefly describe them.

1. **Enter Deep**: In this strategy, CDN servers are placed deep inside access networks, close to end users. This minimizes delay by reducing the distance between users and the CDN servers. However, this requires managing many servers distributed across various networks.
2. **Bring Home**: In this approach, CDN servers are located in a few large data centers, usually at the edges of ISP networks. Although there is more distance between the end users and the servers, it simplifies the infrastructure and reduces management complexity.

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?

Other important factors include:

- **Load Balancing**: Distributing traffic evenly across CDN servers to prevent overload.
- **Geographical Proximity**: Choosing a server closer to the user to minimize latency.
- **Content Popularity**: Serving frequently requested content from servers closer to where it is most accessed.
- **Cost Considerations**: Reducing operational costs by choosing servers based on bandwidth costs and data transfer charges.
- **Legal and Regulatory Issues**: Compliance with data storage and transfer regulations in different regions.

R26. In Section 2.7, the UDP server described needed only one socket, whereas the TCP server needed two sockets. Why? If the TCP server were to support n simultaneous connections, each from a different client host, how many sockets would the TCP server need?

A **UDP server** needs only one socket because UDP is connectionless. It can receive and send data to multiple clients through the same socket. In contrast, a **TCP server** needs two sockets because TCP is connection-oriented. The server needs one **listening socket** to accept connection requests, and a **new socket** for each established connection. Therefore, if a TCP server supports **n simultaneous connections**, it would need **n+1 sockets** (one for listening, and one for each active connection).

R27. For the client-server application over TCP described in Section 2.7, why must the server program be executed before the client program? For the client-server application over UDP, why may the client program be executed before the server program?

In **TCP**, the server program must be executed first because it listens for incoming connection requests from clients. The client needs to initiate the connection, and this is only possible if the server is already running and listening for incoming connections.

In **UDP**, the client can send data to the server even if the server is not yet running. UDP is connectionless, so there is no need for the server to listen for a connection before the client starts sending data. The client can send a message, and when the server starts running, it will process any messages it receives.

P1. True or False?

- **(a)** True. A web page with some text and three images will require **one request** for the HTML document and **three requests** for the images. Therefore, four responses will be received.
- **(b)** True. **Persistent connections** in HTTP allow multiple requests and responses to be sent over the same connection, even if they refer to different web pages.
- **(c)** False. With **nonpersistent connections**, each TCP connection is closed after the server sends an object. Thus, only one request message and one response message are sent per TCP connection.
- **(d)** False. The **Date** header in the response message indicates when the message was sent by the server, not when the object was last modified.
- **(e)** False. HTTP response messages can have an empty body, especially in responses like **304 Not Modified**, where no content is returned.

P7. DNS and RTT Calculation

When a DNS lookup involves **n** DNS servers, each of which incurs an RTT of RTT1, ..., RTTn, and then the web object is retrieved with an RTT0:

- The total time elapses as: **RTT1 + RTT2 + ... + RTTn + RTT0**. This includes the time for DNS resolution and the time to fetch the object from the server.

P8. Elapsed Time for Retrieving Objects

- **(a) Non-persistent HTTP with no parallel connections**: Each of the eight objects is fetched in series, requiring one RTT per object. The total time is: RTT0+8×RTT0RTT0 + 8 \times RTT0RTT0+8×RTT0

- **(b) Non-persistent HTTP with 6 parallel connections**: Six objects are fetched in parallel, and two objects are fetched afterward. Total time is: RTT0+2×RTT0RTT0 + 2 \times RTT0RTT0+2×RTT0
- **(c) Persistent HTTP**: All objects are fetched using a single connection. The time elapses as: RTT0RTT0RTT0

P9. Total Average Response Time

Given that the average object size is 1,000,000 bits, the request rate is 16 requests per second, and the access link forwards requests to origin servers with a 3-second response time:

- **(a) Average access delay** is modeled as: d=LR/(1−Lb)d = \frac{L}{R} / (1 - Lb)d=RL/(1−Lb) where **L** is the object size and **R** is the link speed. Add the 3-second internet delay.
- **(b)** Installing a cache with a miss rate of 0.4 will reduce the total response time by serving cached objects more quickly, leading to a proportionate reduction in delay.

P10. Parallel Downloads with HTTP

For a **10-meter link** with a bandwidth of 150 bits/second:

- **Non-persistent HTTP** will suffer from overhead due to multiple connection establishments and closures for each object, so parallel downloads do not provide much gain.
- **Persistent HTTP** is more efficient because it avoids the overhead of establishing a new connection for each object, resulting in significant gains.

P11. Bob's Parallel Connections

- **(a)** Bob’s parallel connections do help him get web pages more quickly because parallelism can reduce waiting time when multiple objects are retrieved.
- **(b)** If all users open parallel instances, the available bandwidth is divided, reducing the effectiveness of parallel connections for everyone, including Bob.

P12. Simple TCP Program

A simple TCP server program can be written to accept and print lines of input from a client. You can modify the provided **TCPServer.py** to test GET request messages and check if conditional GET messages are generated for cached objects.

P13 & P14. HTTP/2 Frame Transmission and Prioritization

- **P13 (a)** If all video frames are sent first, it will take **2000 frame times** before any image frames are sent.

- **P13 (b)** If frames are interleaved, it will take much fewer frame times as frames for the video and images are transmitted simultaneously.
- **P14** With HTTP/2 prioritization, since images are prioritized, the second image will be sent soon after the first, and before the video clip starts transmitting its later frames.

P15. Difference between MAIL FROM in SMTP and From: in Mail

- **MAIL FROM:** is part of the **SMTP envelope** and indicates the sender's email address at the protocol level.
- **From:** is part of the actual **email header** and is seen by the recipient, representing the sender of the email.

P16. End of Message in SMTP vs HTTP

- **SMTP** marks the end of the message body with a sequence of **"<CRLF>.<CRLF>"**.
- **HTTP** uses the **Content-Length header** to indicate the end of the message body. HTTP cannot use the same method as SMTP because the content can be structured differently and contain multiple line breaks.

P17. What does MTA stand for?

MTA stands for **Mail Transfer Agent**, a software application responsible for sending, receiving, and routing emails between mail servers. MTAs handle the transport of messages across networks using protocols such as **SMTP (Simple Mail Transfer Protocol)** .

Identifying the Malicious Host: In the spam email provided, multiple `Received` headers help trace the path of the email. The malicious host can be identified by its IP address, typically the first non-legitimate server in the chain. The spam email originated from IP address **58.88.21.177**, which appears to be a suspicious source. Therefore, **asusus-4b96 ([58.88.21.177])** is likely the malicious host that generated this email .

P18. Whois Database and DNS Reconnaissance:

- **(a)** A **whois** database is a publicly available directory service that provides information about registered domain names and the contact details of domain owners, as well as the DNS servers associated with them.
- **(f)** An attacker can use whois databases and the **nslookup** tool to gather reconnaissance data about an organization’s network. By identifying the domain’s DNS servers and IP ranges, an attacker can probe for vulnerabilities and map the infrastructure, which helps in planning a more targeted attack .

P19. Exploring the DNS Hierarchy with dig:

- **(a)** You can use the `dig` tool to trace DNS query delegation through the hierarchy, starting from a root server. For example, querying the DNS for **google.com** would first reach a root server, then a TLD server for **.com**, and finally an authoritative server for **google.com**. The list of DNS servers forms the delegation chain .

P20. DNS Cache and Popularity:

To estimate the most popular web servers, you can analyze the cached entries in your department's local DNS server. Frequent DNS queries for specific external sites will lead to cached records, which indicates that those web servers are likely popular within the department .

P21. Detecting Recent External Access:

As an ordinary user, you cannot directly access DNS server logs. However, if you observe DNS cache responses or TTL values from a previous query, you might infer whether a site was accessed recently. This requires administrator-level access to DNS logs for precise data .

P22. Minimum File Distribution Time:

In both client-server and peer-to-peer (P2P) architectures, the minimum distribution time is influenced by factors such as the server's upload speed and the peers' download/upload rates. In a client-server model, the server's upload rate is the bottleneck, while in P2P, the distribution load is shared among peers, reducing the distribution time significantly for larger networks .

P23. Client-Server Distribution Schemes:

- **(a)** When the server's bandwidth is split across all peers (`us/N ≤ dmin`), the distribution time can be calculated as `NF/us`.
- **(b)** When the per-peer download rate is the limiting factor (`us/N ≥ dmin`), the distribution time is `F/dmin`. Thus, the overall minimum distribution time is determined by the maximum of these two formulas .

P24. P2P File Distribution:

- **(a)** When the server's upload speed dominates (`us ≤ (us + Σui)/N`), the distribution time is limited by `F/us`.
- **(b)** If the collective peer upload bandwidth is sufficient, the distribution time improves, reducing to `NF/(us + Σui)` .

P25. Overlay Network Nodes and Edges:

An overlay network consists of **N active peers**, where each peer has a direct connection to every other peer. If **M routers** are involved, the number of edges in the overlay network is proportional to the number of direct TCP connections between peers, which is determined by the network topology and peer connections .

P26. BitTorrent Free-Riding:

- **(a)** Bob's claim that he can receive a complete copy of the file without uploading is **not feasible**. BitTorrent uses a tit-for-tat system, meaning peers prioritize uploading to others who also upload. Without contributing, Bob’s download speed would be significantly limited.
- **(b)** By using multiple computers with distinct IP addresses, Bob could potentially evade the tit-for-tat mechanism and free-ride more efficiently. Each computer would appear as a separate peer, allowing Bob to download more chunks without uploading .