Application Layer

Josh Wilcox (jw14g24@soton.ac.uk)

February 28, 2025

Contents

1 Overview

- The appplication is the top most layer of the TCP/IP Stack
- It acts as an **interface** between the user and the network
- Protocols define how data is *formatted*, *transmitted*, and *interpreted* so that applications can understand and process it
- The application layer is **encapsulated** from the other layers

2 SMTP - Simple Mail Transfer Protocol

- Used for sending and routing emails between email servers and clients
- Responsible for *sending emails* - **not responsible for receiving emails**

2.1 Process

a) Client to Server
   - Senders client connects to server and sends message

b) Server Relay
   - Sever checks recipients domains and uses **DNS** to find the IP address of the relevant email server

c) SMTP Transfer
   - Senders server makes a connection to the recipients SMTP sever and transfers the email

d) Final Delivery
   - Recipients SMTP sever stores the email until received via **IMAP** or **POP3**

2.2 Characteristics

- Text based protocol
- Uses specific commands:
  - TCP - To establish connnection
  - EHLO - Used by the client to introduce itself to the main server
  - 250 (Extension list)- Replies to ELHO with list of SMTP extensions
  - QUIT - Close the TCP transmission channel

3 IMAP - Internet Message Access Protocol

- Used for the **access** and **retreival** of emails stored on an email server

3.1 Process

a) Connection to Mail Server
   - Uses port 143 for standard connections
   - Uses 993 for encrypted connections

b) Email Sync
   - Client keeps a TCP connection open to send requests or receive notifications in real time

- Unlike **POP3**, IMAP *keeps emails* on the server

4 HTTP - Hypertext Transfer Protocol

- Used for **web browsing** and communication between servers and clients

4.1 Process

a) Browser connects to web server using TCP
   - Port 80 for HTTP
   - Port 443 for HTTPS

b) Browser sends a GET request to the server

c) Server responds with an **HTTP** status code
   - E.g `200 OK` or `404 Not Found`

d) Browser processes the response and displays the webpage
   - Text response for classical websites is usually HTML, CSS etc
   - API requests also can use HTTP and usually respond in JSON or HTML

4.2 HTTP Requests

- GET
  - Requests data from a specified resource
  - Should not have any side effects (i.e., it should be safe)
  - Can be cached and bookmarked
- HEAD
  - Similar to GET but only requests the headers
  - Used to check if a resource is available or to get metadata
- POST
  - Submits data to be processed to a specified resource
  - Can result in a change of state or side effects on the server
  - Not idempotent (repeating the request may have different effects)
- PUT
  - Updates a current resource with new data
  - Idempotent (repeating the request will have the same effect)
- DELETE
  - Deletes the specified resource
  - Idempotent (repeating the request will have the same effect)

4.3 Status Codes

- `1xx` - Informational
  - `100 Continue` - The server has received the request headers, and the client should proceed to send the request body.
  - `101 Switching Protocols` - The requester has asked the server to switch protocols, and the server has agreed to do so.
- `2xx` - Success

  - `200 OK` - The request has succeeded.
  - `201 Created` - The request has been fulfilled, resulting in the creation of a new resource.
  - `204 No Content` - The server successfully processed the request, but is not returning any content.
- `3xx` - Redirection
  - `301 Moved Permanently` - The requested resource has been assigned a new permanent URI.
  - `302 Found` - The requested resource resides temporarily under a different URI.
  - `304 Not Modified` - The resource has not been modified since the version specified by the request headers.
- `4xx` - Client Error
  - `400 Bad Request` - The server could not understand the request due to invalid syntax.
  - `401 Unauthorized` - The client must authenticate itself to get the requested response.
  - `403 Forbidden` - The client does not have access rights to the content.
  - `404 Not Found` - The server can not find the requested resource.
- `5xx` - Server Error
  - `500 Internal Server Error` - The server has encountered a situation it doesn't know how to handle.
  - `502 Bad Gateway` - The server, while acting as a gateway or proxy, received an invalid response from the upstream server.
  - `503 Service Unavailable` - The server is not ready to handle the request, usually due to maintenance or overload.

4.4 HTTPS

4.4.1 Drawbacks of HTTP

- **No Security Guarantees**: HTTP provides no encryption, making data transmitted over HTTP vulnerable to interception and eavesdropping.
- **Data Integrity**: There is no mechanism to ensure that the data has not been altered during transmission.
- **Authentication**: HTTP does not provide a built-in way to verify the identity of the communicating parties, making it susceptible to man-in-the-middle attacks.
- **Privacy Concerns**: Sensitive information such as passwords, credit card numbers, and personal data can be easily intercepted.

4.4.2 What HTTPS Does

- **Encryption**: HTTPS wraps HTTP in a **TLS** (Transport Layer Security) session, encrypting the data transmitted between the client and server to ensure confidentiality.
- **Data Integrity**: TLS provides mechanisms to detect any tampering or alteration of data during transmission, ensuring that the data received is the same as the data sent.
- **Authentication**: HTTPS uses digital certificates to verify the identity of the server, helping to prevent man-in-the-middle attacks and ensuring that the client is communicating with the intended server.
- **Privacy Protection**: By encrypting the data, HTTPS protects sensitive information from being intercepted by unauthorized parties.
- **Trust and Credibility**: Websites using HTTPS are often perceived as more trustworthy and credible by users, which can improve user confidence and engagement.

4.5 HTTP/2 and HTTP/3

4.5.1 Version 2

- First major revision to HTTP
- Implements **Header Compression**
  - Reduces amount of data sent with each request
- Multiplexes multiple request over a single TCP connection
  - Loads multiple requests to be sent at the same time over one connection
- Performance is increased significantly compared to version 1

4.5.2 Version 3

- Instead of using TCP connections it runs over QUIC which is faster and reliable

5 CoAP - Contained Application Protocol

- Provides Lightweight HTTP-esque protocol for simpler devices
  - Has a **minimal overhead**
  - Useful for IoT which sends lots of data and doesn't need useless stuff
- Uses **UDP** instead of TCP for faster communication
- Uses Status Codes like HTTP (Get, Post, Put, Delete)

6 QUIC - Quick UDP Internet Connections

- Designed to replace **TCP**
- A new transport protocol
- Built on top of UDP
- Developed by Google and now standardized by the IETF

6.1 Key Features

- **Multiplexing**: Allows multiple streams of data to be sent over a single connection without head-of-line blocking, improving performance.
- **Connection Migration**: Supports seamless connection migration when a client changes IP addresses, such as moving from Wi-Fi to cellular data.
- **Reduced Latency**: Establishes connections faster than TCP by combining the handshake and encryption setup, reducing latency.
- **Built-in Security**: Integrates TLS 1.3 for encryption, providing security and privacy for data transmission.
- **Congestion Control**: Implements advanced congestion control algorithms to optimize network throughput and reduce congestion.
- **Error Correction**: Uses Forward Error Correction (FEC) to recover lost packets without retransmission, improving reliability.

6.2 Advantages over TCP

- **Faster Connection Establishment**: QUIC combines the transport and cryptographic handshakes, reducing the time needed to establish a secure connection.
- **Improved Performance**: By avoiding head-of-line blocking and supporting multiplexing, QUIC can deliver data more efficiently, especially in high-latency networks.
- **Better Handling of Network Changes**: QUIC's connection migration feature allows it to maintain connections even when the client's IP address changes, providing a more resilient connection.
- **Enhanced Security**: QUIC's integration with TLS 1.3 ensures that all connections are encrypted by default, enhancing security and privacy.

6.3 Use Cases

- **Web Browsing**: QUIC is used by HTTP/3, the latest version of the HTTP protocol, to improve web page load times and overall browsing experience.
- **Video Streaming**: QUIC's low latency and error correction features make it ideal for streaming high-quality video content with minimal buffering.
- **Online Gaming**: The reduced latency and improved performance of QUIC provide a smoother and more responsive gaming experience.
- **Mobile Applications**: QUIC's connection migration capability ensures stable connections for mobile users who frequently switch between different networks.

7 RTSP - Real Time Streaming Protocool

- Designed for **streaming media** over the internet
- Good for security camera streams for example
- Controls playback with the following commands
  - DESCRIBE
    - Requests a description of the media
    - Typically returns an SDP (Session Description Protocol) message
  - SETUP
    - Establishes a session for the media stream
    - Specifies transport parameters
  - PLAY
    - Starts or resumes the media stream
    - Can specify a starting point in the media
  - PAUSE
    - Temporarily halts the media stream
    - The session remains active
  - TEARDOWN
    - Terminates the media session
    - Frees up resources associated with the stream

8 RTP - Real Time Transport Protocol

- Handles the **actual transmission** of real-time media
- Runs over UDP for low latency (faster than TCP)
- Uses sequence numbers to detect lost packets
- Include timestamps to sync audio and video