McAlvain Colin

Dr. Batyr Charyyev

CPE 400 Computer Communication Networks

2023/05/26

Class Project Abstraction

- Team Members:
  - Colin McAlvain
- Tentative Project Title:
  - Study and Implementation of a Reliable Transmission Control Protocol in Python
- Overview:
  - This project is going to create a TCP protocol layer. This TCP will be able to segment data files of various types and transfer them across a network connection. Several features will be implemented including headers, decoders, ports numbers, sequence numbers, acknowledgment numbers, window size, check sum, and control flags. These features will help create reliable data transfer for files. This TCP should be able to transmit multiple data files to different devices simultaneously using a multiplexer and a demultiplexer. If data is received out of order it should be able to recognize the and sort the data back into the original composition and recognize when data has been corrupted or lost during transfer. This software will be based on a client server approach.
- Transmission Control Protocol (TCP) description, parts, and purpose.
  - Description:
    - Transmission Control Protocol is a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.
    - TCP works with IP to ensure data is delivered to its intended destination in a network.
    - TCP and IP differ where IP is an identification that is assigned to a specific device, and the TCP is responsible for transporting and routing data through the network.
    - TCP uses packets of data that can travel though different paths in a network to get to the correct IP destination.
  - Parts:
    - Segment:
      - TCP breaks the data into smaller units called segments for transmission over the network. Each segment contains a header and payload.
    - Header:
      - The TCP header contains control information necessary for the transmission and delivery of data. It includes fields such as source and

destination port numbers, sequence number, acknowledgment number, window size, checksum, and control flags.

- Port Numbers:
  - TCP uses port numbers to identify specific applications or services running on a device. The source and destination port numbers in the TCP header help direct the data to the correct application or service.
- Sequence Number:
  - TCP uses sequence numbers to ensure that data is delivered to the receiving end in the correct order. The sequence number field in the TCP header allows the receiver to reorder out-of-order segments.
- Acknowledgment Number:
  - TCP uses acknowledgments to ensure reliable data delivery. The acknowledgment number field in the TCP header indicates the sequence number of the next expected data segment from the sender.
- Window Size:
  - The window size field in the TCP header specifies the amount of data, in bytes, that the receiver is willing to accept at a given time. It helps regulate the flow of data and supports flow control.
- Checksum:
  - TCP uses a checksum field in the header to detect errors in the transmitted data. The sender calculates the checksum based on the segment's content, and the receiver verifies it to ensure data integrity.
- Control Flags:
  - TCP uses control flags, also known as TCP flags or control bits, to indicate specific control functions and states. Some common control flags include SYN (synchronize), ACK (acknowledge), FIN (finish), RST (reset), and URG (urgent).

- Purpose:
  - Reliable Data Delivery:
    - TCP ensures that data sent from a source device reaches the destination device reliably and without errors. It achieves this reliability through mechanisms such as acknowledgments, retransmissions, and error detection using checksums.
  - Ordered Delivery:
    - TCP guarantees that data segments arrive at the destination in the same order they were sent. It uses sequence numbers to order and reorder out-of-order segments, ensuring the correct assembly of the transmitted data.
  - Flow Control:
    - TCP incorporates flow control mechanisms to prevent the receiver from being overwhelmed by data sent by the sender. The receiver can dynamically control the rate at which it accepts data, and the sender adjusts its transmission rate accordingly, preventing data loss or congestion.

        - Congestion Control:
            - TCP employs congestion control algorithms to detect and respond to network congestion. When congestion is detected, TCP slows down the rate of data transmission to alleviate congestion, ensuring fair sharing of network resources and maintaining overall network stability.
        - Connection Establishment and Termination:
            - TCP establishes a connection between a source and a destination device before data transmission begins. It uses a three-way handshake process (SYN, SYN-ACK, ACK) to establish the connection and a similar process to terminate it gracefully.
        - Full-Duplex Communication:
            - TCP supports full-duplex communication, allowing simultaneous bi-directional data transfer. Both the source and destination devices can send and receive data at the same time over a single TCP connection.
        - Multiplexing and Demultiplexing:
            - TCP uses port numbers to enable multiple applications or services to run concurrently on a device. The combination of source and destination port numbers helps identify the specific application or service to which incoming data should be delivered.
- TCP Uses:
    - Web Browsing:
        - When you visit a website and load web pages, TCP is used to establish a connection between your browser and the web server. It ensures that the web page data is transmitted reliably and in the correct order.
    - Email Communication:
        - TCP is used for sending and receiving emails via protocols like Simple Mail Transfer Protocol (SMTP) and Internet Message Access Protocol (IMAP). TCP guarantees that email data is delivered without errors and in the intended order.
    - File Transfer:
        - When you download files from File Transfer Protocol (FTP) servers or transfer files over protocols like SSH File Transfer Protocol (SFTP) or Secure Copy Protocol (SCP), TCP ensures the reliable and ordered delivery of file data.
    - Remote Access and Terminal Services:
        - TCP is used in protocols such as Secure Shell (SSH) and Telnet, which enable remote access to servers or network devices. TCP ensures that the interactive session data (commands, responses, etc.) is transmitted reliably between the client and server.
    - Database Access:
        - TCP is commonly used in database management systems, where client applications communicate with database servers. TCP guarantees the reliable transfer of SQL queries and database responses, ensuring the integrity and order of the data.
    - VoIP (Voice over IP):

        - TCP can be used in some VoIP applications to provide reliable voice communication. User Datagram Protocol (UDP) is more commonly used in VoIP for its lower latency, but TCP can be utilized when reliability is prioritized over real-time performance.
    - VPN (Virtual Private Network):
        - TCP is often employed in VPN protocols such as OpenVPN and SSTP to establish secure and encrypted tunnels between the client and VPN server. TCP ensures that VPN traffic is transmitted reliably across the public internet.
    - HTTP (Hypertext Transfer Protocol):
        - HTTP can operate over both TCP and UDP. HTTP/HTTPS protocols primarily rely on TCP for their connection-oriented and reliable nature. TCP guarantees that web page data and other HTTP requests/responses are delivered correctly.
- Code Design:
    - Client-Server Architecture:
        - The TCP implementation can follow a client-server architecture, where one party acts as the client initiating the connection, and the other party acts as the server listening for incoming connections.

    - Modules and Components:
        - The implementation will be divided into several modules/components to handle different aspects of the TCP functionality:
            - Socket Module:
                - This module handles the creation of TCP sockets, binding to a specific address and port, and listening for incoming connections.
                - It provides functions for accepting client connections, creating new sockets for each client, and managing the connection lifecycle.
            - Data Handling Module:
                - This module is responsible for managing data transmission and reception.
                - It handles the segmentation of data into TCP segments, assembling segments into complete messages, managing sequence numbers and acknowledgments, retransmissions for lost segments, and flow control.
            - Connection Management Module:
                - This module manages the establishment, maintenance, and termination of TCP connections.
                - It implements the three-way handshake for connection establishment, maintains connection state information, handles connection timeouts, and manages the closure of connections.
            - Error Handling Module:
                - This module provides mechanisms for error detection and handling.
                - It will include functions for checksum calculation and verification, handling out-of-order segments, detecting, and

responding to network congestion, and handling various error conditions that may occur during TCP operation.

- API and Interfaces:
  - Create clear and well-documented APIs and interfaces for each module/component to enable interaction and integration between different parts of the TCP implementation.
- Testing and Validation:
  - Implement comprehensive unit tests, integration tests, and validation mechanisms to ensure the correctness and robustness of the TCP implementation.
  - Test various scenarios, including normal operation, edge cases, error conditions, and stress testing under high load.
- Performance Optimization:
  - Consider performance optimizations such as buffering mechanisms, efficient data handling, and congestion control algorithms to enhance the overall throughput and responsiveness of the TCP implementation.
- Security Considerations:
  - Address security aspects such as encryption, authentication, and secure key exchange if required.
  - Consider integrating SSL/TLS protocols for secure communication over TCP.
- Documentation:
  - Detailed documentation and comments throughout the codebase to improve readability and maintainability.
  - Include explanations of design choices, algorithms used, and any assumptions made.

- Implementation:
  - Source Code Control
    - Git will be used for version control.
    - GitHub will be used for source code centralization, review, and control.
      - Master Branch for running implementation.
      - Dev Branch for work.
  - Testing
    - The code will be tested in python's unittest.
      - Unit test:
      - Integration Tests:
    - GitHub will be used for automated testing implementing YAML files.
  - Deployment
    - Docker will be used to hold images for updates to code.
    - DockerHub will be used to control version updates and deployment.
- Approach:
  - A simple TCP will be created with a server and client code and run in two terminals for concept understanding. The code used is a commonly used example copied from a CS 446 project and rewritten in python from C++.
  - Using resources from the internet, class text, and class sides, an outline will be created for the proposed code architecture.
  - This architecture will be used to create unit tests and integration tests.

        - Test Driven Development includes building tests, running them to make sure they fail, writing code, testing code, refactoring code to pass all tests.
    - Code will be written in python.
    - Code will be tested automatically via push to GitHub.
    - Coding will be refactored continuously until standards are met.
    - Software testing from end to end will be done manually with data transfer in one of the several areas defined above. The focus of this TCP will be determined during software architecture design.

Citations

Fortinet. (2023b). *What is TCP/IP in networking?*. Fortinet. https://www.fortinet.com/resources/cyberglossary/tcp-ip

Kurose, J. F., & Ross, K. W. (2017). *Computer networking: A top-down approach*. Pearson.

Unittest - unit testing framework. Python documentation. (n.d.). https://docs.python.org/3/library/unittest.html

Quickstart for github actions. GitHub Docs. (n.d.). https://docs.github.com/en/actions/quickstart

Building and testing python. GitHub Docs. (n.d.-a). https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python