Linsey Szabo
Computer Networks
Fall 2023

Project 3 Design Document

Summary of Overall System

I split the work into a transmitting and receiving thread like Project 2.

The transmitting thread sent protocol messages like SYN, SYN+ACK, and so on as well as chunks the file being downloaded.

The receiving thread processed these messages and advanced the state machine accordingly.

To achieve concurrent functionality, I ensure that each file download occurred in an isolated structure. I used a dictionary of a node's peers to hold information about peer-to-peer downloads as well as a state machine. Any access to this dictionary is protected by a lock. Lastly, I looped through all a node's peers continuously to keep progressing concurrent file downloads.

Final Protocol Design

I'll first discuss how I implemented the initial protocol messages. By setting a global Boolean when a file is entered in the command line, I get the requestor node to initiate contact with the owner node that contains the file to be downloaded. In this SYN message, I included the host, port, and filename of the requestor so that the owner node has access to this information in the future. For the SYN+ACK message, I included the length of the packet to be downloaded, so that this information can be used as a check for the final size of the file. Lastly, once the requestor sends ACK 0, the owner node is now able to start sending chunks of the file. Now, all subsequent ACK's will have sequence numbers greater than or equal to 1.

To get chunks of the file to send, I use .seek() and the current ACK to properly increment the file descriptor pointer. I maintain a variable corresponding to the number of messages sent and increment upon sending and decrement upon receiving to implement a sliding window.

Packet Format

For each packet, I started with "PUT <sequence number> " followed by the chunk of binary data itself. On the receiving end, I compared the bytes of PUT with incoming packets and parsed the sequence number by comparing to the binary representation of the space character.

State Diagram

Please see the attached picture. A state machine was kept for each peer-to-peer download. I used enumerations to make states. I created a .next() method in my state machine class that took in the strings on the transitions on the diagram to properly transition to the next state.

The receiving thread processed these messages and advanced the state machine accordingly.