CSCI 4211
TCP Server Project Writeup
Shane Stodolka | stodo050@umn.edu
11-28-22

INTRO:

This project implements an inter-network subscription service based on the client-server model. The client works very simply as a sender and receiver of messages while the server handles heavier tasks such as keeping track of client subscriptions.

I chose to write this program in C to gain more experience working with threads and implementing my own data structures. The code is split into several different header and source files which are used in the tcp_server. Each data structure is implemented in its own file for sake of simplicity.

STRUCTURE:

The Server is built on a foundation of threads, creating one thread for each client. Each thread handles its own client messaging and connections. When a client connects, a new thread is created. The thread then is closed upon client disconnection.

Subscriptions to *topics* are structured hierarchically in the server. Every *topic* is loaded into memory from a .txt file upon initialization. Topics in the .txt file are stored in the form of a *path* which dictates how they will sit in the topic hierarchy. Topics can have the same name but not under the same parents. Examples of two valid paths are:

news/sports/football
and
shopping/sports/football

These paths are valid, but will create topics with the empty string:

news/sports/football/
and
/shopping//sports

Upon starting the server each topic will be printed out in a hierarchy so that the user can easily visualize their subscriptions.

For the sake of simplicity clients are not allowed to create new topics. This feature could be easily implemented using existing functions with a new command.

INSTRUCTIONS:

To run this program, unzip the downloaded contents and open a terminal window in the directory of the unzipped folder.

To compile, type ‘make’ in the console. This will create two executables ‘client’ and ‘server’. To execute, type ‘./server’ or ‘./client’

Commands:
All commands are handled as strings to increase user accessibility.

Subscriptions: ***(un)subscribe*** **<path>**
Subscriptions are handled by two simple commands, *subscribe,* and *unsubscribe.* Each command required a second argument of a valid *path* to function correctly. If the path is not valid the server will return an error. To subscribe to several topics at once the client may use ‘#’ and ‘+’ as wildcard separators.

‘#’ functions as a multi-level wildcard. Using it will subscribe a client to every topic underneath the specified path in the hierarchy.

‘+’ functions as a single level wildcard. Its use case is to subscribe to grandchildren of topics.

Wildcard functionality is not available for unsubscribing

Publish: *publish* <path> <message> <retain>
A client can publish a message to other clients who are subscribed to a topic. To do this, use the publish command. This command takes in two additional arguments, path, and message. A fourth optional retain flag can be set with the command -r. This saves a message to a topic to be send whenever a new client subscribes to the topic.

List: *list*
The command ‘list’ will list all currently subscribed topics.

IMPLEMENTATION:

This server/client is written in C. The server opens a new thread for every client and handles requests separately.

For ease-of-use port numbers and topics are made static. Topics can be modified in the text ‘topics.txt’ The server will load topics into a tree and validate them using the Topic structure. To make things easier, a Path structure was implemented to parse the incoming/outgoing paths.