4211_Project1

Billy Wencl
5544600
Wencl010@umn.edu

Extract & Build

Extract with

```
 tar -xzvf wencl010_4211_Project1.tar.gz
cd wencl010_4211_Project1
```

To compile the whole project at once use:

```
    make
```

To compile/recompile only the server files use:

```
    make server
```

To compile/recompile only the client files use:

```
    make client
```

Run

To start the server:

```
    make run_server
```

To start the client:

```
    make run_client
```

Summary

This project is an implementation of a message board server. Clients can publish messages to and subscribe to various topics. The server stores the current subscriptions and distributes published messages to all subscribers.

Current Limitations

- Topic paths must be less than 70 characters in length
- Messages must be less than 948 characters in length

Implementation

Basic Communication:

This project was implemented with C++ socket programming. A messaging format was defined in MsgProtocl.h as seen below.

```
struct MsgPacket {
    MsgType type;
    char topic[70];
    char msg[948];
    bool retain;
};
```

This struct is the sole data format passed between client and server. The MsgType enumerator is used to identify what type of message is being passed. For example: MT_Connect, MT_Conn_ACK, MT_Disconnect, MT_Subscribe... Once the server or client checks the MsgType it knows which fields of the struct to interact with.

Client Interface:

```
 ------------------------------------------------
Options:
W - Wait for messages from subscriptions
S - subscribe to messages
U - Unsubscribe from messages
L - List subscriptions
P - publish a message
Q - quit
------------------------------------------------
Enter choice:
```

The client code automatically handles the setup of the server socket and exchanges connect messages with the server. The UI allows access to send the server requests to subscribe to a topic, publish messages to topics, and disconnect. Upon selection of subscribe or publish, the user will be prompted for more information. It also allows users to listen to the server. Users must be in listen mode to receive any published messages.

```
Waiting for messages... Use ^C to return to menu
```

When in listening mode Ctrl + C will take users back to the options menu.

Server:

Once the server starts the main thread waits for new connections. For each connection it spawns a worker thread. The worker threads wait until they receive a message from a client then handle the message (create a subscription or publish a message) and reply before returning to listen mode. The workers all share one TopicManager instance and one SubscriptionManager instance to handle subscriptions. The access to these instances is controlled with two mutex locks, one for topics and one for subscriptions.

Retain Messages:

The RetainMSG.h file contains both the RetainMSG and RetainManager classes. The RetainMSG class holds the name and retainMsg of a topic. The RetainManager handles the vector of retainMSGs and provides an interface to simplify with working with them.

Subscriptions:

The subscription.h file both the Subscription and SubscriptionManager classes. Much like the Topic class, the Subscription class holds the information of a single subscription, namely the client's socket file descriptor and a pointer to the topic. SubscriptionManager keeps track of all the subscription and provides methods for accessing all of them.