Snowcast README

Author: Shan Lu (slu5)

For better reading experience on this document, please click here to oepn a rich-formatted PDF file.

Data Structures

```
typedef struct
{
    int tcpport;    // listening port no
    int listen_fd;  // socket file descriptor
    int numStations;
    fd_set master;
    int fdmax;
    StationInfo *stations;
    int fdCapacity;
    int fdCount;
    int *clients_control_fd;
    pthread_mutex_t fdlock;
} ServerInfo;
```

```
typedef struct
{
    int stationID;
    FILE *fp; // mp3 file that station is playing
    char *filename;
    int numClients;
    ClientInfo *head; // linked-list
    pthread_mutex_t lock;
} StationInfo;
```

```
typedef struct ClientInfo ClientInfo;
struct ClientInfo{
    ClientInfo *next;
    char hostname[INET6_ADDRSTRLEN];
    int udpport;
    struct sockaddr dest_addr;
    socklen_t dest_addrlen;
    int control_fd;
    int listener_fd;
};
```

In this project, I defined three structures to store and maintain data from server and clients in a systematic manner.

- `ServerInfo` stores all the data that can be accessed globally. Concretely, `StationInfo` contains runtime allocated array of music stations of size `numStations`. Variables `master,fdmax,fdCapacity,fdCount` are used to bookkeep clients' control program's sockets data(which seems redundant at first glance as we will save them into `ClientInfo`), which will be used for system call function `select(int, fd_set *,fd_set *,fd_set *,struct timeval *)`, while `clients_control_fd` is a dynamically sized array storing those file descriptors.
- `StationInfo` saves all the information regarding a given music channel. Most of the variable names are self-explanatory such as `stationID` or `numClients`. Variable `head` serves as the head pointer of a linked-list of clients that are listening to current station. The occurrence of `pthread_mutex_t lock` will be discussed in next section.
- `ClientInfo` includes all data that a server requires to communicate with a specific client. The `ClientInfo *next`, as you may have figured out, points to next `ClientInfo` structure. Though both controller and listener's file descriptor have been created, we still need the `dest_addr` and `dest_addrlen` as arguments of function `sendto()`, which is called when sending data under UDP.

Design

According to the project writeup, we need to build a radio station rather than a mp3 player. In other words, clients connecting to same channel must hear the same part of a song. This demand immediately gets my mind into shape that a multi-threaded model is required. In this implementation, one music station occupies one single thread and takes fully charge of all clients that connect to it. Since the overall number of clients is uncertain and may change frequently due to clients' switches, a linked list works better than a dynamic array or anything else. In this way, each thread could iteratre through all its clients and send the same UDP packet almost the same time.

Thread, thread, thread!

At first I thought one-thread-per-station design is pretty straightforward and mutexes are required only if stations add a client or one client switches to another music station. However, this assumption was broken when more than one client connecting to the server with one client has not set its station. Under this circumstance, the server's main thread will wait until the client completed setting the station.

So we must split threads handling those clients that have sent hello but not set their preferred stations yet. Once `stationNumber` has been picked, the `ClientInfo` could fill out its attributes and attach itself to one of the stations on server. And client's right of control is transferred to that specific station thread automatically.

Here is the formula to compute the total number of threads active in a server:

```
totalThreads = 1(main) + numPendingClients(have not set station) + numStations
```

Bugs

Yes there are bugs. Most of them occur when one of the stations has sent hello but not set its `stationNumber` . Sometimes it will interfere the switch station process of newcoming clients.

Reference

- Beej's Network Programming Guide