Group members:
Shreyans Jain (2018A7PS0253P)
Pranav Gupta (2018A7PS0190P)
Harshit Garg (2018A7PS0218P)

DESIGN DOCUMENT

P3 group communication client

Problem Statement - Design a peer application that can be part of multicast groups and share files with members of various groups. The peer must also be able to initiate polls in the group.

Command Line

The peer application is essentially a command line application that provides the following options:

Command	What it does
`join <grp_name>`	Join the multicast group called grp_name
`leave <grp_name>`	Leave the multicast group called grp_name
`create <grp_name, ip, port>`	Create a new multicast group with the name grp_name and this IP and port
`start-poll <grp_name>`	Start a new poll for this group
`search-grp <grp_name>`	Search for this group in the list of existing groups on LAN

At startup, the program asks the user to input a unicast IPv4 address, which will be used for all unicast communication (for example in downloading files). The list of available commands can be printed using `help`

Data Structures

Every group is represented as a `comm_grp` struct which contains the group's name, a boolean indicating whether a particular peer has joined that group and two sockets (one for send and one for receive) and their address structures. Because the storage of data has to be decentralized, every peer maintains its own copy of all the groups that exist on that LAN. This is represented in an array of `comm_grp` structs, which acts as the database for that peer. Hence the name, `grp_db`. Apart from this, every peer maintains some global data such as a broadcast socket and address, a unicast socket and address, a list of files that it has locally (and can share with other peers) and other fields.

A standard struct named `peer_msg` is used to represent any multicast messages exchanged between peers. This makes handling messages of different types significantly easier. The `peer_msg` struct contains 4 fields:

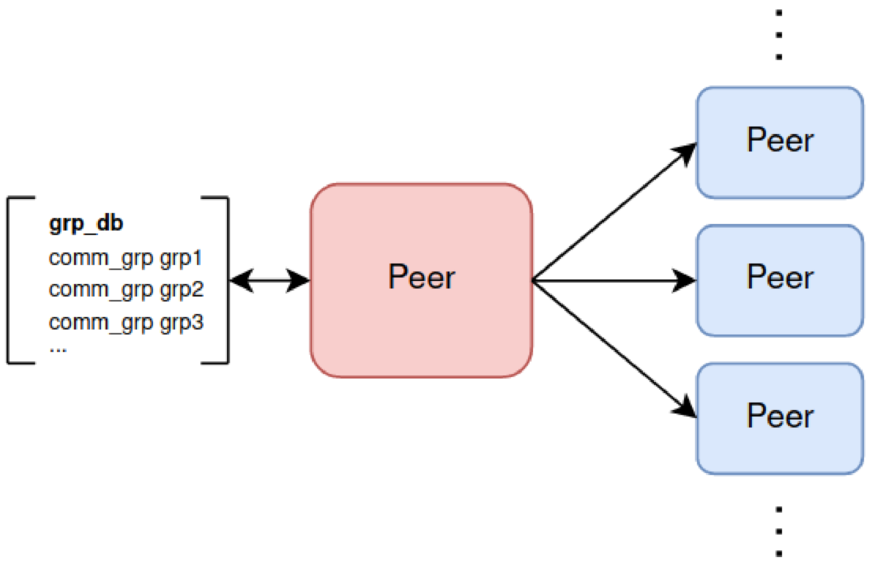
Field Name	Type	Details
`msg_type`	enum	Indicates the type of message like file request, file download, poll response etc.
`msg_buff`	string	Contains the payload of the message
`sender_ip`	string	The unicast IP of the sending peer - to support unicast communication between 2 peers
`flag`	int	Used for 2 purposes - first for indicating a poll response (yes/no) and second for indicating file download status

This single struct is a crucial part of the application as it enables standardised communication between entire groups, as well as between 2 individual pairs.

Application Flow

The first thing that a peer can do is create a new multicast group by supplying a name, an IP and a port. Once a group structure is created, *a broadcast message is sent out to all members on that LAN*, containing the new group's `comm_grp` struct as the payload. This broadcast tells every member that a new group has been created and that they can join this group in the future.

This also means that every peer will have groups in its `grp_db` that it may or may not have joined. All broadcast connections are managed over the port 8080.



The next set of operations that it can perform is joining a group and leaving a group, both of which involve `setsockopt` calls. Other than this, there are 2 categories of commands:

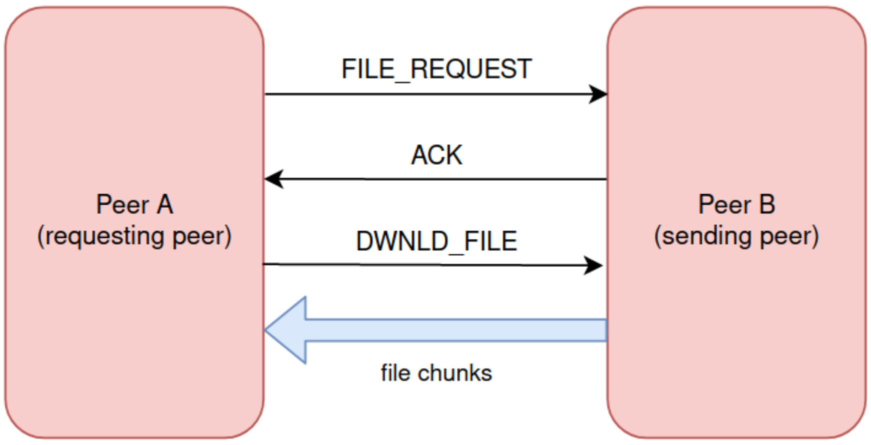
1. Sending/receiving multicast messages for file sharing. These include file list advertisements and file download request messages.
2. Creating a new poll and sending it to other members / receiving a poll's response.

All multicast messages are sent using a single function, `notify_grp`. This along with the `peer_msg` struct makes sending multicast messages very easy, and each message can be distinguished based on the type field.

File Sharing

Every 1 minute, a peer sends out a multicast message to a group. The message contains a list of files that are available locally with the peer, and can be downloaded by other peers. This message has the type FILE_ADVERTISE, as the message is in fact an advertisement of files available. Upon receiving a FILE_ADVERTISE message, the receiving peer forwards the same message to other groups it is a member of, under the type FILE_ADVERTISE_FWD, indicating that it is a forwarded message.

At the same time, the receiving peer verifies its own file list with the list of files it just received. If it finds any files that are missing, it generates a new message of the type FILE_REQUEST and sends it out to all its groups. If any of its peers have the file, they respond with an ACK (flag in `peer_msg` set to 1) containing that peer's unicast IP in the `sender_ip` field. This can be used to open a new unicast connection to allow the file to be downloaded by the requesting peer. All unicast connections are handled on the port 8081.



Note that any peer that receives a FILE_REQUEST message forwards it to all its own groups under the type FILE_REQUEST_FWD and the responses are accumulated and sent over to the original requester (with a timeout of 1 minute).

Creating Polls

Any peer can create polls for a particular group. This requires them to supply a simple yes/no question which will be sent to all group members as a `peer_msg`. The sender will then wait for 1 minute for any responses. Any responses within this time limit are recorded, and after the timeout, are printed to the console.

Monitoring Input

When the peer application is running, all the file descriptors are monitored, including STDIN, as that is where the user can enter a new command. This is done using the `select` call. Note that while a user enters a command, several other processes might be running in the background, such as file download, message forwarding etc.