Group communication On LAN using Multicast

CS F422: Parallel Computing

Guntaas Singh (2018A7PS0269P)
Bir Anmol Singh (2018A7PS0261P)
K Vignesh (2018A7PS0183P)

April 28, 2021

Basic Data structures

We have used two data structures mainly in this project which are given as follows:

Groups or group_info

This Data Structure is used to provide us the information about the groups which have been formed on the network. It comprises of three elements, Group ID, IP and Group Name.

Explanation for including each element are given below:-
Group ID is used to uniquely identify every group over the network and is same across all users across the network for the same Multi-cast group. We could use the ip for this unique identification but we needed a type of variable which could provide a range of utility functions (comparison, equal, hashing etc.) at a low cost, and string (the type of IP address) didn't provide them at effective cost.

IP stores the IP address of the Multi-cast group which a user can use to join or send messages to the group. This was needed because when we shared the information of IP address to different users also so that they can join this Multi-cast group if they desire

Group Name stores the name of the group which the creator of the group decides. This is purely for user ease of access and provides no direct benefit on the backend of the group communication. It helps us see which message was sent in which group as identifying the group is easier through names for humans.

Packet

This DS is used to store the messages to be sent across the network and all messages are in this format if they are for the multicast groups and in some cases also the unicast. It comprises of 5 elements, the message itself, the time at which it was sent, the group for which it is intended, the original sender of the message and lastly the type of the packet itself.

Explanation for including each element is given below: -
The message itself is stored in an array of character and is the main component of the packet. This is where the message to be sent in a group or the topic of a poll or the name of the file to be searched is stored depending upon the type field of packet.

The time at which the packet was created is needed to discard the message which come at a time when they are no longer needed or when certain waiting times need to be timed out on the receiving end. This can

be in the following cases, when a vote arrives after the voting phase has timed out, when the reply message of a file search comes up in the next file search call.

The group which it was intended is used to access the group for which this message was intended towards and is used by voting response functionality and speedy access to other information about groups at the receiving end like name and ip address for displaying.

Original sender of the message is a field which is used by the receiving in case of identifying the user who sent a group message and it also is used in the search file functionality to store the Unicast address of the downloader which the uploader will use for creating a unicast connection for replying and uploading/downloading the file.

The type field of a packet determines what purpose the packet fulfills, it has two context and different meaning in each case. For Multi-cast context-

- Type – 0 is a group message/the result of a vote to be displayed on the group
- Type – 1 is a search request message where someone is requesting to search a file.
- Type – 2 is a vote initiating message which asks the user to vote on the given poll
- Type – 3 is a response message to voting to count the favor/against votes.
- Type – 4 is a request message to request file data to all common group members
- Type – 5 is a update message to update the network file list in the system

For Uni-cast context

- Type – 0 is a negative response to the requested file, implying that the file was not found.
- Type – 1 is a positive response to the requested file, implying that the file was found.

We also use a membership array to see if the user is a member of the $group\_id^{th}$ indexed group in O(1) time(like a hash table).

We also use a id_used array to check which group id have already been used in O(1) time(like a hash table).

Basic Utility functions

The following custom-made utility function for ease of use in the application code.

1. **get_min_id()** - This function is used to get the minimum Group ID which is available for use. This is an O(N) where N is the total number of IDs.
2. **find_group(int group_id)** – This function is used to find the index in the group_info array of the group identified by the Group ID equal to the group_id argument. This is an O(N) where N is the number of Group info available.
3. **Check_collision(char* ip)** - This function is used to find if the said IP is already in use by some other group of which we have records.This is an O(N) where N is the number of group info available.
4. **File_exists(char* filename, int options)** – This function is used to check if a file with the name identified by string of filename is present in my system files records(for option = 1) or my network file records(for option = 0).This is an O(N) operation where N is the number of file indexed in the respective system or network name array.
5. **Refresh_file_info()** – This function is used to refresh the file records for the new files added in current directory. This is an O(N) operation where N is the number of files in the concerned directory which is shared over the network.
6. **Refresh_group_info()** – This function is used to receive any and all messages that are received on the special broadcast socket. This is an O(N) operation where N is the number of update messages received. Also this special socket for broadcast message has a receive timeout to prevent a block here.
7. **Request_group_info()** – This function is used to send a broadcast message to all the users on the network to send their group_info data structure for this user to get information about all the Multicast groups already existing in the network. After this we wait for some time to consolidate replies and then call the **Refresh_group_info()** function.
8. **display_file_data()** - This function is used to display all the names of the files known to be on the system and on the known network.
9. **populate_packet(packet* pkt, char* msg, int init_time, int group_id, struct in_addr og_sender, int type)** - This function takes in all the members of the packet struct and use it to populate the packet which is also given in the arguments.

Implementation of Main Features

Creating A Group

- We first update our group_info array with any new groups formed using *refresh_group_info* function call.
- This feature is implemented by asking for the IP address for Multicast group from the user and supplying a Port number by the program on it's own. The availability of this IP address and any Group ID is checked using *check_collisions* and *get_min_id()* function calls. If available, this information along with the group name is stored in a struct of group_info and is then added to the group_info array. All Multicast groups have the same port number(8888) and can be differentiated only on the basis of their IP. This streamlines the further processes very easily.
- the id_used and membership array are updated accordingly.

- Lastly this new instance of struct of group_info is then broadcast on the system_fd or the socket meant for sending and receiving broadcast messages for updating the new group_info to all systems across the network.

Joining a Group

- We first update our group_info array with any new groups formed using *refresh_group_info* function call.
- It then asks the user to enter the Group ID of the group it wants to join and adds the membership if not added already using IP_ADD_MEMBERSHIP flag in the Multicast socket with this group's IP.
- The membership array is changed accordingly

Search Group

- We first update our group_info array with any new groups formed using *refresh_group_info* function call.
- The user is asked to type an identifier (Group ID) of the desired group and we return all the group information including IP Address and Port of the Multicast group identified by this identifier from the global group_info array. If -1 is given as the group ID then information about all known group is printed for the user from the same array.

Sending a message

- The user is asked the Group ID of the Multicast group and the message which needs to be sent.
- Using *populate_packet* function call we populate a packet with the current time,original sender being filled automatically and the type field equal to 0. This packet is then sent to the Multicast socket with the IP address of the Multicast group retrieved from the group info array using group ID.

Sharing File data into Network

- There is a timer which is set for every 60 seconds, whenever this timer expires it created a sig_event which starts a SIGALRM signal. This signal's handler simply sets a variable sharing=1 and reset the timer.
- This flag is used to initiate the sharing process in which the user sends its file data to every multicast group the user is a member of using the multicast socket.
- Whenever a user receives such data, it updates its network file records and wait for its own timer to expire to send the updated records to all of its own groups.

NOTE - This signal may be blocked by masks if the signal may interrupt the waiting period of voting or receiving replies of search request or periods where the application is taking input from stdin. Thus the time may not be exactly 60 seconds.

Request File Data

- Every user can choose to request data from all the groups he is a member of and all members of these groups would then give their network file records to the user who requested the data. This does not transitively propagate and only reaches the common neighbors.

- The Requestor sends a type – 4 packet to every group he is a member of and then waits for type-5 packets for updating the network file records.
- All those who receive the type-4 packet ,send a type-5 message containing the file records data back into the Multicast group from which they received the type-4 packet. The Group is known through the Group ID stored in the packet.

NOTE - Though the request doesn't transitively propagate, but the requested data can transitively propagate by the per-minute timer sharing of data

Search File Locally

- It uses two utility functions of *refresh_file_info()* and *file_exists()*.
- First a call of *refresh_file_info()* is done to update file records by listing all the files which are available in the current directory. Then we call *file_exists()* to check If the desired file is present in the updated records.
- if we find the file, then suitable message is shown otherwise we try finding it on the network

Search File on Network

- When the earlier search fails to find the file locally , we send a type - 1 packet to all groups(of whose we are member) with file name in the message field and our unicast address in the original sender field.
- All those who receive this type-1 packet search for this file locally. The unicast address, stored in the original sender field is used to create a unicast address for the confirmation message of file existence on different systems, if less than 60 seconds have been passed from the initial time of search request(initial time store in the packet too) then this message is sent to the file seeker. This sent to the unicast IP address and the port 8890
- A type 0 packet here means the file was not found and type 1 means that the file was found.
- The positive confirmation is followed by 1000 byte chunks of data to the unicast address and download of same chunk proceeds on the file seeker machine.

Voting Process

- Any user can initiate a poll in a group and it sends a type-2 message to the multicast group where such a poll starts.
- Every member which receives a type-2 message votes on the given matter by pressing one of two keys (y for favor and n for against.). These users send a type 3 message with their vote to the vote initiator.
- The vote initiator receives such messages and increments the suitable counters. After 30 seconds of vote initiation(stored in the initial time of sent message), no votes are sent and if received then they are discarded.
- The result of the vote is sent to all members of the group using a normal multicast group message as explained earlier.

Receiving Messages

- To receive messages from multiple groups simultaneously we have used a threaded approach in which the main thread does the main work of whatever the user desires, and another receiving thread runs in parallel to receive messages from the multicast socket.
- This thread does a *recvfrom()* call for the Multicast socket and as each Multicast group uses the same socket so we need to be reading from one socket only for the retrieval of messages.
- a switch case is used to decide what needs to be done with what type of packet and this has been already explained earlier.
- A type 0 packet means the message is simply printed on the screen
- A type 1 packet means a search request is to fulfilled and then upload file if need be.
- A type -2 packet means this user needs to vote on a matter displayed
- A type-3 packet means the suitable (favor/against) counter needs to be incremented.
- A type 4 message means that data was requested and we suitably share the data with the concerned multicast group(s).
- A type 5 message means an update message has arrive and the network file records needs to be updated.

General Flow of Program In Main

Here is the flow of main program in terms of the functions and implementation given above.

- We first do a *request_group_info* function call to know all the groups which are present on the network on start-up and *refresh_file_info* for updating files at startup
- We first create three sockets, one for the multicast communication, one broadcast socket for group info updates and a last for unicast communication. All of these have a Receive timeout of 2 seconds for the messages to arrive. While the broadcast socket has the appropriate bit mask applied for broadcasting messages
- We then start a timer for each 60 seconds to share data periodically into the network.
- After this we create a separate thread for receiving messages which happens concurrently with our main thread.
- We are then allowed to choose from the following 8 options
    1. Create a Group
    2. Join a Group
    3. Search a Group
    4. Send a Message
    5. Search A file
    6. Start a Poll
    7. Request Data
    8. Display File Data
- The files which are stored in the same directory as the running file are only the ones which are shared with the network.
- After this we also check for any broadcast message which seeks group info and then share the whole groups array if need be for the newly joined system on the LAN.