Project 1: Group RPC Communication report

name: Shixiang Long

Student id: 4445538

1.Design of functional data structure

1) server

(1) Since there are several group, we need a map to store the <name, group>

```
name_group = MAP {name, group}
```

(2) When the server flooding the message to everyone in the group, we need to know the ip_adress of the users. Therefore, I used a map to store it.

```
name_ips = {name, ip}
```

(3) Besides, we need a data structure to store the message for each group, the key is group name, value is a list of message

```
group_message = MAP{group_name, list: massage}
message = 'name:' + 'message user send'
```

(4) From the request, "The server keeps track of the clients that has received the messages and only sends the unread messages", therefore, we need a data structure to track the message the user has readed. I use a map to store it, the key is username, value is the pointer point the `name_read` map's massage.

```
name_read = MAP{name, pointer}
```

(5) server class

```
type Server struct {
  mu           sync.Mutex
  nameGroup    map[string]string
  nameIps      map[string]string
  groupMessage map[string]string
  nameRead     map[string]int
}
```

2) client

client class

```
type Client struct {
  UserName     string
  UserGroup    string
  ServerIpPort string //serverIp:serverPort
  ClientPort   string // the port app listen, since multiple client may exist on the same machne
}
```

2. Design of RPC and RPC's data structure

1) client -> server

(1) When a client online, he need to receive all the message he hasn't read

```
client_getMessage RPC

Arguments:
  clien_ip_port
  user_name
  user's group

Return value:
  List of unreaded message
```

(2) When a client send a message to the group, the server should store it

```
client_sendMessage RPC

Arguments:
  user_name
  message

return value:
  success or not
```

2) server -> client

When a client send a message to the server, the server should send the message to the all clients who are in the same group.

```
server_sendMessage RPC

Argument:
  message

return value:
  success or not
```

3) Defaul RPC port

For server: 1234

For client：1220~1230

3. Pseudocode

1) server:

```
if receive client_getMessaga RPC:
    if client is not register:
      // register the user
      name_group[name] = group
      name_ips[name] = ip
      name_read[name] = -1 // there is no message the new user has read
    set the unreaded message to the latest idx
    return unreaded message to the user

if receive client_sendMessage:
    add the message into group_message[group]
    flooding the message to all the servers who are in the group by calling server_sendMessage except for sender
    if success, reset the unreaded pointer of the user
```

2) client

```
if the client from offline to online:
  call client_getMessaga RPC to register himself, get unreaded message, print the messages received
if receive server_sendMessage:
  print the messages received
if the client want to send the messages to group:
  call client_sendMessage RPC and print the message on the screen, if failed, print failed
```

4.Possible improvements of the system

1）The system can't tolerate faults. If the server down, all the information is over. Therefore, the server should create logs to record the history logs of users. When it online, the server should restore the users' messages from the logs.

2）The server should compact the messages which maintained in the physical memroy. The server needs to truncate some messages stored in physical memroy if all the users of one group have read the messages. Then store these part messages into disk. In this case, the free physical memroy is enough even the content of messages are very large.