

[Network] Term Project

2016310470 - Park Jae Hyeong (Computer Science)

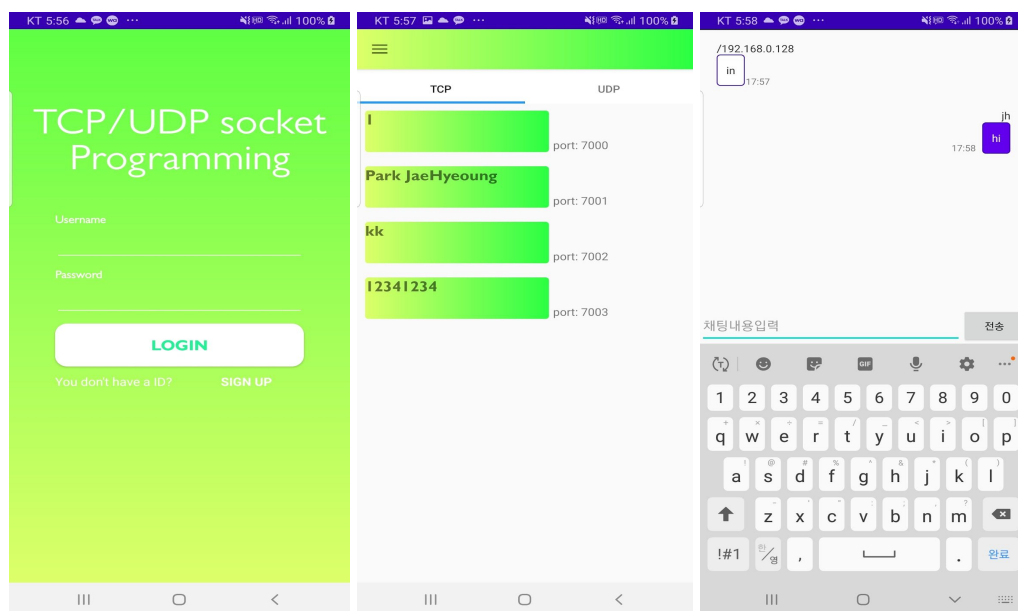
2018313395 - Kim Byoung Moo (Software)

1. Overview

- We made TCP/IP protocol socket programming and UDP protocol socket programming using android application. The Project's object is creating TCP and UDP protocol and analyzing the difference of those in chatting app.

2. User manual for installation and testing:

- This project is based on Android Studio, so please install Android Studio and open the project file to check the code.
- If you want to test this project, download the attached apk file on your smartphone. Otherwise, install Android Studio and connect your smartphone and run the project.
- You should login to use this application. You can login by Username: 1, Password: 1, or Username: k, Password: 1, or create your sign up your own account.
- If you want to test this application, follow below steps
 1. Login two different accounts at two different devices (device must be a real smartphone, not emulator. Also test this at the same wi-fi environment)
 2. Enter the same chatting room - TCP/IP chatting rooms are in FRIENDS tab, and UDP chatting rooms are in CHATTING tab.
 3. Check communication by typing and sending text.



Login screen, Main screen, Chatting room screen (left to right)

- Use recommended versions and more information: <https://developer.android.com/> will help you.

3. Table of content about Algorithm

1. Login System
2. Sign Up
3. Firebase Database and Storage
4. TCP fragment
5. UDP fragment
6. TCP protocol to chat
7. UDP protocol to chat
8. ViewPager2 to merge two fragment
9. Navigation toolbar using Drawer layout

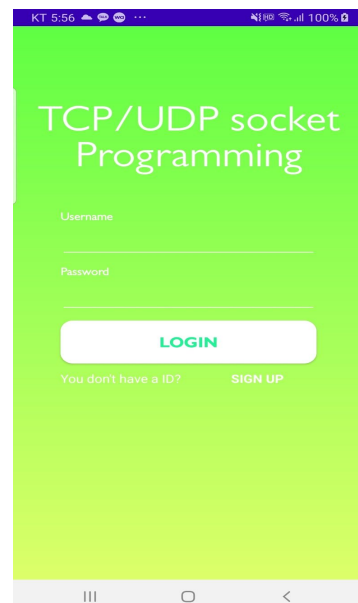
4. Details about Algorithm

4.1 Login System

- First, we use a login system to distinguish the user. If ID and password are not correct, it doesn't work.
- The Background color is used in drawable xml.
- login_background's color has a gradient using angle and start, end color.

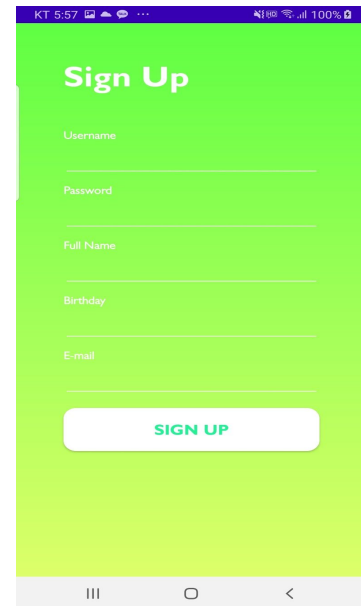
```
<gradient android:angle="270"  
  android:type="linear"  
  android:startColor="#5FFF43"  
  android:endColor="#DDFF69"/>
```

- If your input ID and password are correct, You can go to TCP/UDP page.



4.2 Sign Up

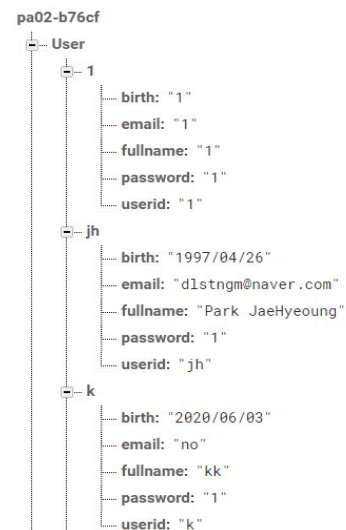
- You can sign up for our application.
- If you don't input any EditText, you can not sign up. It toasts messages to input text.
- This information is uploaded in Firebase Database.
- The data is saved by birth, email, fullname, password, userid.



A screenshot of a mobile application's 'Sign Up' screen. The screen has a light blue background. At the top, there's a status bar with 'KT 5:57' and battery level '100%'. Below the status bar, the title 'Sign Up' is displayed in a bold, dark blue font. Underneath the title, there are five input fields with labels: 'Username', 'Password', 'Full Name', 'Birthday', and 'E-mail'. Each field has a light blue border and a small blue icon on the right. At the bottom of the form, there is a large, rounded rectangular button with the text 'SIGN UP' in a bold, dark blue font. The bottom of the screen shows a standard Android navigation bar with three icons: a home button, a back button, and a recent apps button.

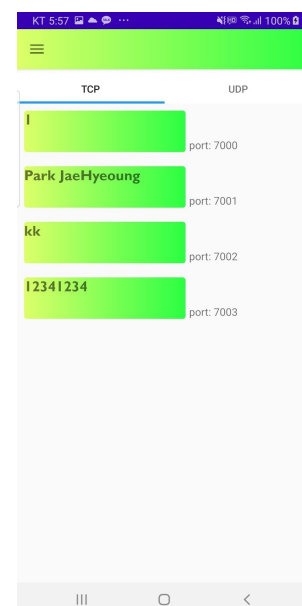
4.3 Firebase Database and Storage

- If you signed up, the information about the profile is saved in the Firebase Database.
- Right picture is Firebase Database's information for the test.
- You can login ID : 1 and Password : 1 to test our application.
- The birth and email and full name and password and userId is saved. At this, the key value is the user's ID. And User is the parent of those profiles.



4.4 TCP fragment

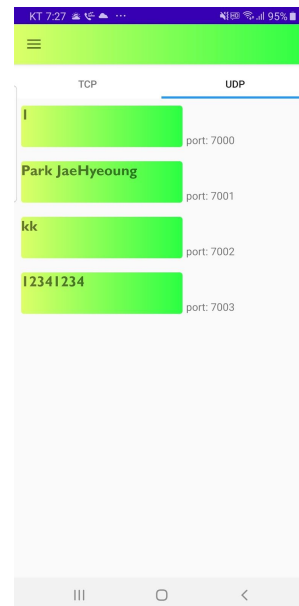
- If you login, you can see the TCP/UDP page.
- First fragment is TCP.
- In the TCP's fragment, you can see the user's name.
- We made the user's name as the server's name.
- So, If you click the name, you can go to the TCP socket chatting room.
- The port number of the server is different by name to connect each port.



- The fragment consists of a listView. And listView contains items.
- Items contain cardView and text.
- In the cardView, it represents the username. And We tried to use a profile image on the left of the username. But It doesn't work well. So we removed.

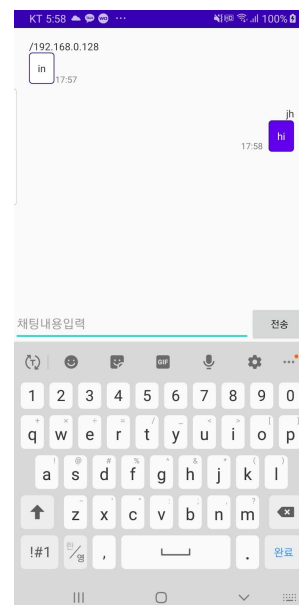
4.5 UDP fragment

- If you login, you can see the TCP/UDP page.
- Second fragment is UDP.
- In the UDP's fragment, you can see the user's name the same as TCP.
- We made the user's name as the server's name.
- So, If you click the name, you can go to the UDP socket chatting room.
- The port number of the server is different by name to connect each port.
- The fragment consists of a listView. And listView contains items.
- Items contain cardView and text.



4.6 TCP protocol to chat

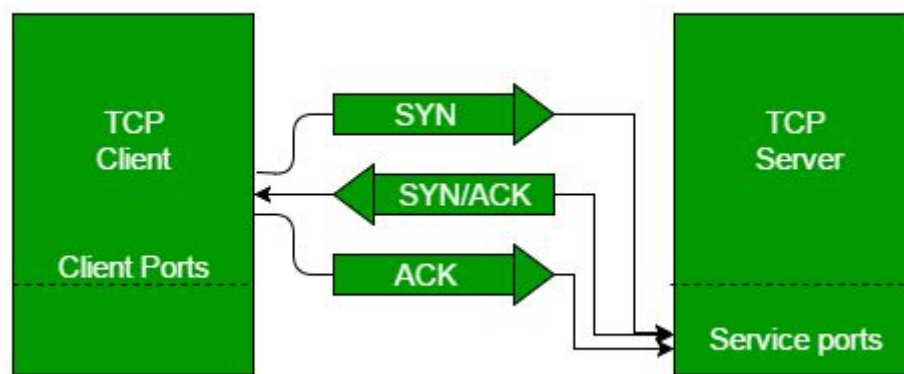
- If you touch one of the nicknames, you can go to the TCP chatting room.
- Our TCP socket communication app automatically gets the IP address of the host.
- If you are inside the chatting room, your IP address says "in".
- My chatting is on the right side, and the other people chatting are on the left side.
- The color is used "@color/colorPrimary"



- **TCP/IP protocol**

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network.

We use TCP/IP protocol in this project by creating a server, and clients connect with the server by TCP/IP's 3-way handshake protocol.



3-way handshake protocol

- **chat_room.java**

In this code, the program fetches the IP address from Firebase realtime database. However, if there is no IP address in the database, get a local address and store it at the Firebase. After this process, create the server by calling `serverCreate()` method, and join it as client by `joinServer()` method.

```
ref = FirebaseDatabase.getInstance().getReference().child("IPADDRESS"+username);

ref.addListenerForSingleValueEvent(new ValueEventListener() {

    @Override

    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        if(dataSnapshot.exists()) {

            address = dataSnapshot.getValue(String.class);

        }

        else{
```

```

        ref.setValue(getLocalIpAddress());

        address = getLocalIpAddress();

        isHost = true;

    }

    serverCreate();

    joinServer();

    is_called = false;

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

}

});

```

In this code, the program gets some messages from the server, and parses it with character ':'. After parsing, create a 'messageItem' class object by given values, and add it to the adapter to show it on the screen.

```

private Handler handler = new Handler() {

    @Override

    public void handleMessage(Message msg) {

        super.handleMessage(msg);

        if(serverSocket != null){

            switch (msg.what) {

                case SERVER_TEXT_UPDATE: {

                }

                break;

                case CLIENT_TEXT_UPDATE: {

                    String[] split_msg = clientMsg.split(":");

                    if(split_msg.length == 2){

```

```

        String name = split_msg[0];

        String content = split_msg[1];

        Calendar calendar= Calendar.getInstance(); //현재 시간을 가지고 있는
객체

        String
time=calendar.get(Calendar.HOUR_OF_DAY)+":"+calendar.get(Calendar.MINUTE);

        MessageItem messageItem = new MessageItem(name, content, time,
nickName);

        messageItems.add(messageItem);

        adapter.notifyDataSetChanged();

        listView.setSelection(messageItems.size()-1); //리스트뷰의 마지막
위치로 스크롤 위치 이동
    }

}

break;

}

}

}

};

```

- **JoinServer()**

JoinServer is a method that joins the corresponding server by the given address and port.

```

public void joinServer() {

    SharedPreferences registerInfo = getSharedPreferences("registerUserName",
Context.MODE_PRIVATE);
    nickName = registerInfo.getString("Username", "NULL");

    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                clientSocket = new Socket(address, port);
                clientOut = new DataOutputStream(clientSocket.getOutputStream());
                clientIn = new DataInputStream(clientSocket.getInputStream());

                clientOut.writeUTF(nickName);

                while (clientIn != null) {
                    try {
                        clientMsg = clientIn.readUTF();
                        Log.d("asdf", "client message received");
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    Log.d("asdf", "client message update");
                    handler.sendMessage(CLIENT_TEXT_UPDATE);
                }
            } catch (UnknownHostException e1) {
                e1.printStackTrace();
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
    }).start();
}

```

- **ServerCreate()**

serverCreate is a method that creates a server and sends messages to every connected client.

```

public void serverCreate() {

    Collections.synchronizedMap(clientsMap);

    try {
        if(serverSocket == null){

            serverSocket = new ServerSocket(port);

        }

        new Thread(new Runnable() {

            @Override

```



```

public void run() {

    while (serverSocket != null) {

        try {

            socket = serverSocket.accept();

        } catch (IOException e) {

            e.printStackTrace();

        }

        if(socket != null){

            msg = socket.getInetAddress() + "에서
접속했습니다.\n";

            String msg = socket.getInetAddress() + ":" +
"ip";

            try {

                if (clientOut != null)

                    clientOut.writeUTF(msg);

            } catch (IOException e){

                e.printStackTrace();

            }

        }

        handler.sendMessage(SERVER_TEXT_UPDATE);

        new Thread(new Runnable() {

            private String nick;

            @Override

            public void run() {

                try {

                    if(socket != null){

                        out = new
DataOutputStream(socket.getOutputStream());

                        in = new
DataInputStream(socket.getInputStream());

```

```
        nick = in.readUTF();

        addClient(nick, out);

        save_nick = nick;

    }

    } catch (IOException e) {

        e.printStackTrace();

    }

    try {

        while (in != null) {

            msg = in.readUTF();

            sendMessage(msg);

handler.sendMessage(SERVER_TEXT_UPDATE);

        }

    } catch (IOException e) {

        removeClient(nick);

    }

    }

    }).start();

    }

    }).start();

    } catch (IOException e) {

        e.printStackTrace();

    }

}
```

- **sendMessage(String msg)**

sendMessage is a method that sends messages to every connected client.

```
public void sendMessage(String msg) {  
    Iterator<String> it = clientsMap.keySet().iterator();  
    String key = "";  
    while (it.hasNext()) {  
        key = it.next();  
        try {  
            clientsMap.get(key).writeUTF(msg);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

- **onDestroy()**

onDestroy method is called when the server host exits the chatting room. The main role of this method is to close the server and remove IP data from the Firebase realtime database.

```
@Override  
protected void onDestroy() {  
    super.onDestroy();  
    String msg = nickName + ":" + "out";  
    try {  
        if(clientOut != null)  
            clientOut.writeUTF(msg);  
        clientOut = null;  
        clientIn = null;  
        if(!serverSocket.isClosed()){
```

```

serverSocket.close();

serverSocket = null;

if(isHost){
    ref.addListenerForSingleValueEvent(new ValueEventListener() {

        @Override

        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            if(dataSnapshot.exists()) {

                ref.removeValue();

            }

        }

        @Override

        public void onCancelled(@NonNull DatabaseError databaseError) {

        }

    });
}

}

} catch (IOException e) {

    e.printStackTrace();

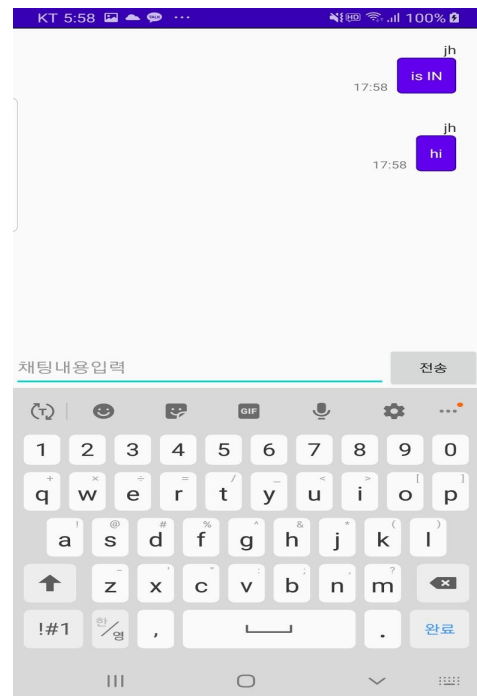
}

}

```

4.7 UDP protocol to chat

- We use the permission of INTERNET for the database of firebase.
- We use the permission of Network and WiFi for network and wifi.
- If you touch one of the nicknames, you can go to the UDP chatting room.
- Our UDP socket communication app automatically gets the IP address of the host.
- My chatting is on the right side, and the other people chatting are on the left side.
- By using listView and Adapter, we implement a chatting box.
- The chatting box's color is used "@color/colorPrimary"
- But our UDP socket chatting is just sending to the server and receiving it. So, if you are not the host, you can't see the whole message. You just can see the message that the server replied to.
- The host can see the whole message that was received.

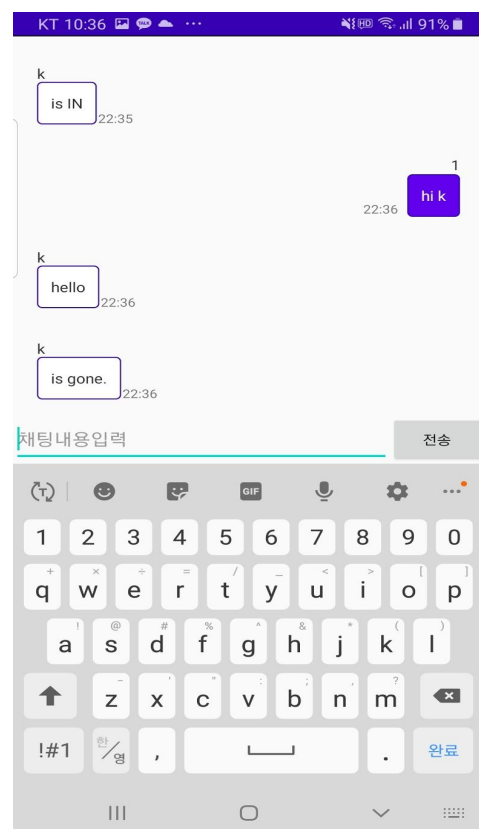


- If you are inside the chatting room, your name says "in".
- If you are out, your name says "is gone".
- This is the same as TCP chat.

- we make TerminateService.class to check the forced termination. If forced termination is occurred, this call onDestroy() function at udp_chat.

In udp_chat.java,

```
startService(new Intent(this, TerminateService.class));
```

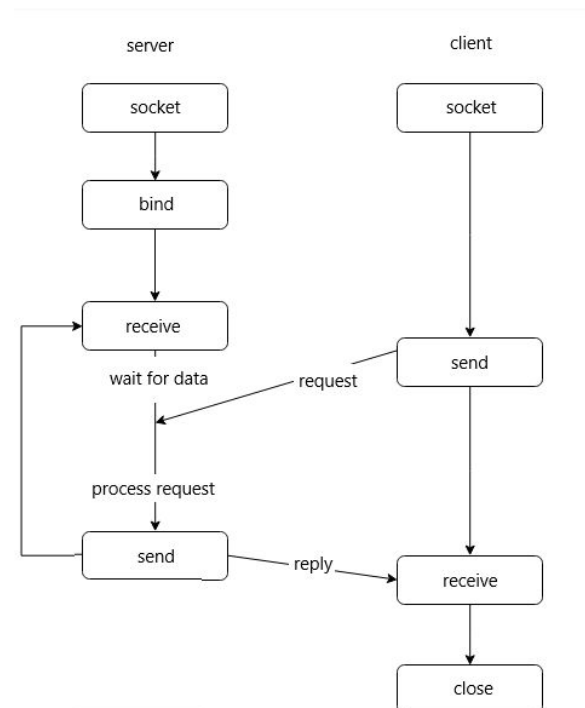


In TerminateService.class.

```
@Override
public void onTaskRemoved(Intent rootIntent) {
    Log.e( tag: "Error", msg: "onTaskRemoved - " + rootIntent);
    chat_room activity = (chat_room)chat_room.chatActivity;
    udp_chat activity2 = (udp_chat) udp_chat.udp_chatActivity;
    activity.finish();
    activity2.finish();
    stopSelf();
}
```

- **UDP protocol**

UDP can send messages without connection. So we can just send a packet to the server and receive it. This data can be lost. And we just think that data will be sent to the server. So to prevent this, the server sends a packet to the client to represent the message's data isn't lost. This is our UDP algorithm.



<Our UDP algorithm>

Server is open to receive the message and if the packet is sent to, the server is sent the packet to client and client receives the packet and we show the message in listview in android app.

Also, We use reliability to UDP. If the server receives the packet, we send the packet to the client. If we just send the message to the server, This message

may not reach the server. So, we only check the message if the server sends it to the client.

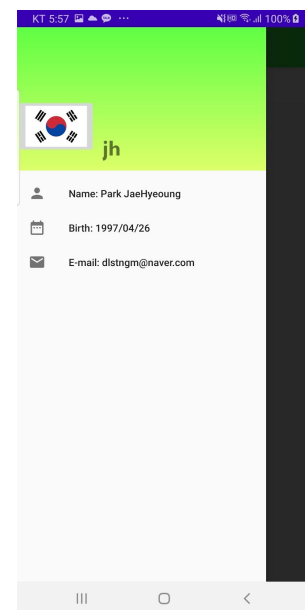
Details of code are in the source file.

4.8 ViewPager2 to merge two fragment

- We use Toolbar and Drawer and navigation view to show user profile
- Using viewPager2, we make two fragments moving by left and right.

4.9 Navigation toolbar using Drawer layout

- We use Navigation bar to show information of User profile (Name, Birth, Email)
- If profile image is input, storage of firebase saves the image.
- We tried to use this image to chat. But it doesn't work well.



References

1. TCP/IP - Internet protocols, <https://www.britannica.com/technology/TCP-IP>
2. TCP/IP concepts, https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.hala001/itctcpipcon.htm
3. UDP-protocol, https://en.wikipedia.org/wiki/User_Datagram_Protocol
4. UDP communication in java, <https://www.programmersought.com/article/8182364860/>