Wi-Chat Application

CS201/CS263 Project

Course Instructors

Dr. Ashish Phophalia & Dr. Novarun Deb

201951018 Aman Kothari
201951024 Anirudh Mitra
201951049 Chirag Jain
201951177 Yash Sakle

Detailed Problem

Sometimes the network facilities are not that good, only some WiFi connection is available to us and we need to contact people available in the same LAN range but we don't have such a medium, like most of the hostels are connected with routers but the network facility is not always available. And sometimes we need to talk to a stranger or a specific person nearby but not in person so we need another medium. So, we are thinking of providing a medium in which two persons can contact each other whether they are strangers or known ones. They just require to be connected to the same WiFi network also this doesn't require any personal information about the user.

Approach/Solution

Basic Overview

We attempted to handle the issue by making an offline messaging application that will let the users chat with one another. The application needn't bother with any information association; it will work on WiFi associations (Wireless fidelity). By using this application we can chat with the individuals close by us. We can implement this thought in Student Hostels, as the network is generally associated with routers however the internet isn't generally accessible at all times. so all things considered, everybody in the range of those routers can message each other without any barrier. It is likewise ideal for fests, sports arenas, provincial networks, catastrophic events, travelling abroad, and significantly much more aspects. It doesn't need any personal data about the client so outsiders associated with a similar organization can have a conversation easily.

The idea of this app came when we used whatsapp and xender consecutively and thought why couldn't we merge the two and create an app that does both things without the cost of data(internet).

What we have done so far!

We have used android's Wi-Fi library to create server side and client-side connections and implemented it with help of OOPs concepts. We have used Access modifiers to keep the data of the user safe. Also, we have tried to complete the basic functionality of the application and from now we will try to improve UI and the efficiency of algorithms so that our application is as simple as possible.

We have created some Interfaces which we will embed in our XML for the application.

For now, we are not using any database service so one can see messages only on run time. And once the user closes the application all data will be erased.

TL;DR: we've basically created an application that simply sends messages over wifi without any other complex feature.

Programming approach:

1. Turn WiFi On/Off programmatically
2. Setting up broadcast receiver
3. Discovering list of peers
4. Connecting to peer
5. Creating Server and Client Thread
6. P2P Data Transfer over this network

APIs Frameworks and Softwares Used:

1. WiFi Direct connection - WiFi P2P API
2. For Socket Connection – Android sync adapter framework.
3. For Activities – Intent
4. Other listeners, handlers and context.
5. UI/UX Design – Adobe® XD and Photoshop™

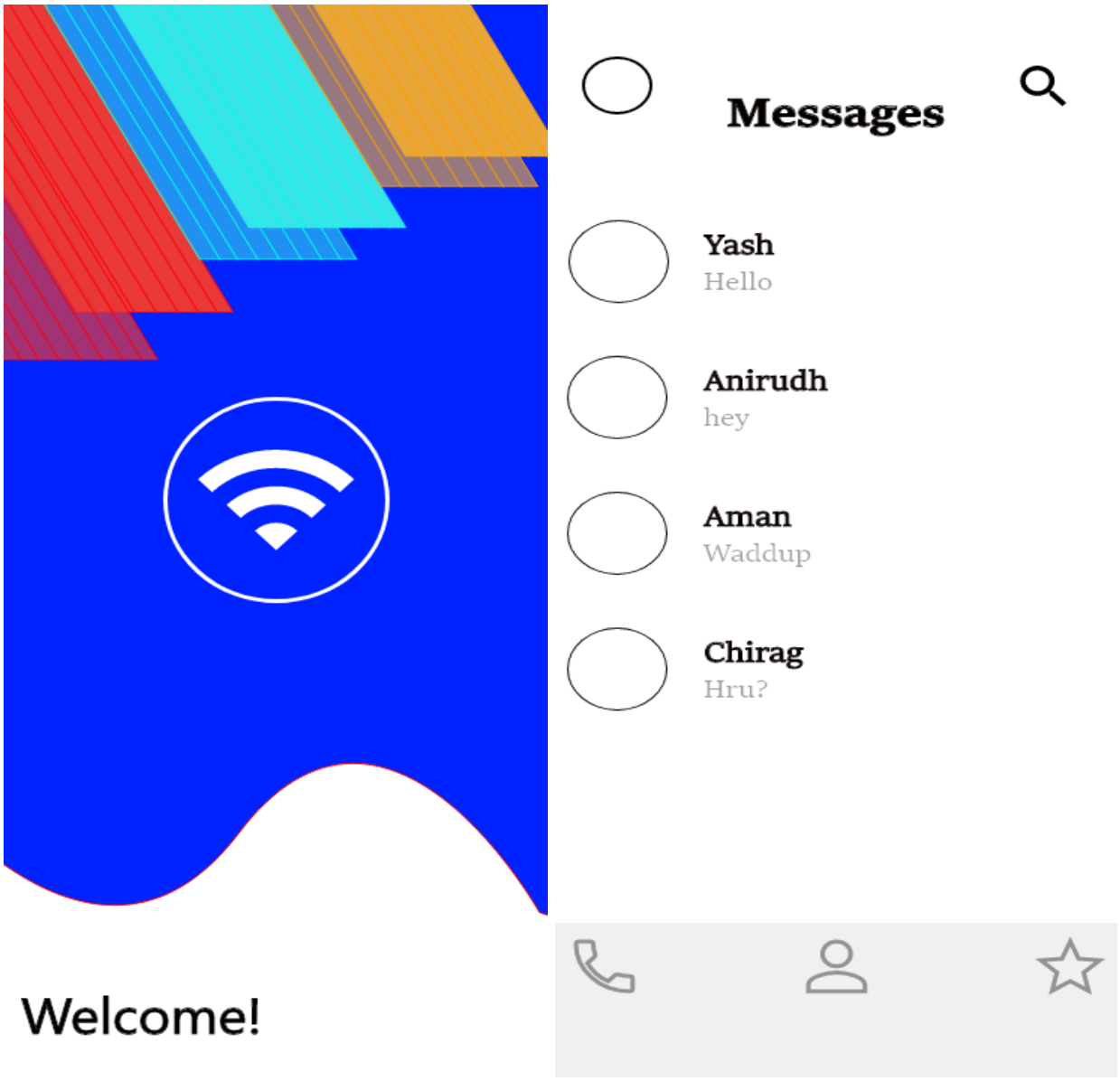
Main Features

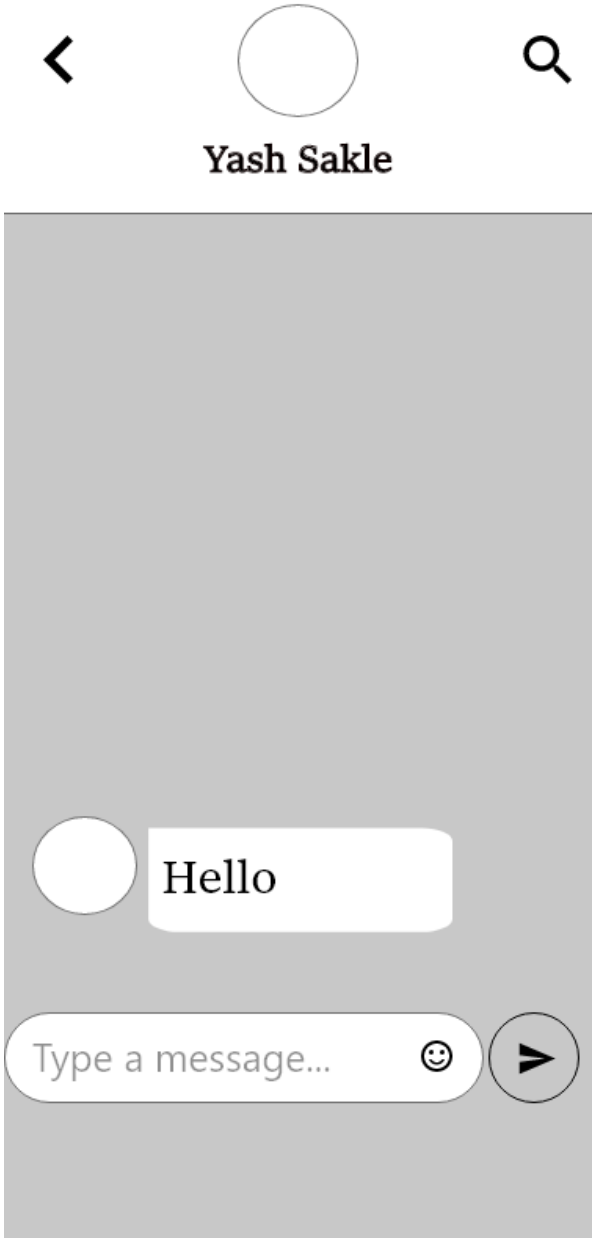
1. Simple Interface: all UI/UX designs are very simply designed without any complex options and using familiar windows.
2. Easy to Connect: the app lets users to connect very easily just as they would connect WiFi.
3. Fully Secure: the app doesn't ask for any permission or access any of the user's data. Also the messages sent are runtime and get deleted as soon as the app is shut.
4. Anonymity: Users will get to choose their chat name during the runtime making them fully anonymous if they want to. (to be added)

Future Work

1. Chat Name: users will be allowed to set their chat name every time they connect to a new peer.
2. Image support: this feature will allow users to send and receive images of all supported formats easily and quickly.
3. Voice Notes support: this will allow users to send recorded voice notes and also music files.
4. Better Structure: concepts like abstraction and encapsulation are yet to be implemented properly, after which the application will become much more secure and well structured.
5. Better Interface: better designs with subtle art visuals will be added to enhance the user experience.
6. Log system: the application will create log files to keep track of every event as they are registered by the broadcast receiver.

User Interfaces




Yash Sakle
Hello
Type a message...

Some code Snippets to show our Progress

I. MainActivity

```
public class MainActivity extends AppCompatActivity {


    private static final int
PERMISSIONS_REQUEST_CODE_ACCESS_FINE_LOCATION = 1001;


    // Button and other field objects


    Button OnOff;                       // WiFi On/Off button

    Button searchPeer, sendMsg;         // search button, send message
button

    TextView conStat, readMsg;          // connection status field,
message o/p field

    ListView showPeer;                  // available devices list

    EditText typeMsg;                   // message i/p field


    WifiManager wifiManager;

    WifiP2pManager mManager;

    WifiP2pManager.Channel mChannel;


    BroadcastReceiver mReceiver;

    IntentFilter mIntFilter;


    List<WifiP2pDevice> peers = new ArrayList<>();

    String[] deviceNameArray;           // used to show device name in
showPeer

    WifiP2pDevice[] deviceArray;        // this array is used to connect
a device


    static final int READ_MESSAGE = 1;


    serverClass serverClass;

    clientClass clientClass;

    SendReceive sendReceive;


    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initialWork();
        exqListener();
    }

    /* Handler initialisation using callback argument is deprecated
        TODO: update the deprecated constructor
                with some other alternative

                community say: use executor
                use: new Handler( looper.myLooper(), callback)
    */
    Handler handler = new Handler(new Handler.Callback() {
        @Override
        public boolean handleMessage(@NonNull Message message) {
            switch(message.what) {
                case READ_MESSAGE:
                    byte[] readBuff = (byte[]) message.obj;
                    String tempMessage = new String(readBuff, 0,
message.arg1);
                    readMsg.setText(tempMessage);
                    break;
            }
            return true;
        }
    });

    private void exqListener() {
        OnOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (wifiManager.isWifiEnabled()) {
                    wifiManager.setWifiEnabled(false);
                    OnOff.setText(R.string.wifi_on);
                } else {
                    wifiManager.setWifiEnabled(true);
                    OnOff.setText(R.string.wifi_off);
```

```
                }
            }
        });

        // !! the button won't work !!
    /* setWiFiEnabled(boolean) is deprecated in API level 29
        TODO: update the deprecated function
                with some other alternative

                community says: open settings
                                to let the user manually connect wifi
direct
                use: public void goToSettings(){
                            goToSettings.setOnClickListener(new
OnClickListener() {

                            @Override
                            public void onClick(View arg0) {

                                //Open Wifi settings
                                startActivityForResult(new
Intent(android.provider.Settings.ACTION_WIFI_SETTINGS), 0);
                             }
                        });
    */



        searchPeer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Discovery of available peers to connect
                // this only detects the devices in range, provide NO
other info
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M
                        &&
checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION)
                        != PackageManager.PERMISSION_GRANTED) {
                    requestPermissions(new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},

MainActivity.PERMISSIONS_REQUEST_CODE_ACCESS_FINE_LOCATION);
```

```
                    // After this point, wait for callback in
                    // onRequestPermissionsResult(int, String[], int[])
overridden method
                }


                mManager.discoverPeers(mChannel, new
WifiP2pManager.ActionListener() {
                    @Override
                    public void onSuccess() {
                        // search started
                        conStat.setText(R.string.search_pass);
                    }


                    @Override
                    public void onFailure(int i) {
                        // search start failed
                        conStat.setText(R.string.search_fail);
                    }
                });
            }
        });


        showPeer.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            // connecting to selected peer
            @SuppressLint("MissingPermission")
            @Override
            public void onItemClick(AdapterView<?> adapterView, View
view, int i, long l) {
                final WifiP2pDevice device = deviceArray[i];
                WifiP2pConfig config = new WifiP2pConfig();
                config.deviceAddress = device.deviceAddress;


                mManager.connect(mChannel, config, new
WifiP2pManager.ActionListener() {
                    @Override
                    public void onSuccess() {
                        Toast.makeText(getApplicationContext(),
"Connected to " + device.deviceName, Toast.LENGTH_SHORT).show();
                    }
```

```
                    @Override
                    public void onFailure(int i) {
                        Toast.makeText(getApplicationContext(),
"Connection Failed", Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });

        sendMsg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String message = typeMsg.getText().toString();
                sendReceive.write(message.getBytes());
            }
        });
    }
```

II. WiFiDirectBroadcastReceiver

```
public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {

    private WifiP2pManager mManager;

    private WifiP2pManager.Channel mChannel;

    private MainActivity mActivity;


    public WiFiDirectBroadcastReceiver(WifiP2pManager xManager,
WifiP2pManager.Channel xChannel, MainActivity xActivity) {

        this.mManager = xManager;

        this.mChannel = xChannel;

        this.mActivity = xActivity;

    }


    @SuppressLint("MissingPermission")

    @Override

    public void onReceive(Context context, Intent intent) {

        String action = intent.getAction();


        // CHECKING INTENTS


        // check if wifi p2p state enabled/disabled

        if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {

            int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE,
-1);

            // check if its enabled

            if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {

                // WiFi P2P is enabled

                Toast.makeText(mActivity, "WiFi P2p is supported",
Toast.LENGTH_SHORT).show();

            } else {

                // WiFi p2p is disabled

                Toast.makeText(mActivity, "WiFi P2p is not supported",
Toast.LENGTH_SHORT).show();

            }

        }


        // check if the peers in the range have changed
```

```
        else if
(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
                mManager.requestPeers(mChannel, mActivity.peerListListener);
            }

        // check if the connection with the peer has changed
        else
if(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {
            if (mManager == null) {
                return;
            }

            NetworkInfo networkInfo =
intent.getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);

            // check if the device is connected with peer
            if (networkInfo.isConnected()) {
                mManager.requestConnectionInfo(mChannel,
mActivity.connectionInfoListener);
            } else {
                mActivity.conStat.setText(R.string.discon);
            }
            /* NetworkInfo class is deprecated but works for some cases
                TODO: update the deprecated class/function
                        with some other alternative

                community says:
                use:
            */
        }
    }
}
```

Full Project at : [WiChat - Source Code on GitHub]