Report of CS 522 Assignment 5

Name: Yunfeng Wei
CWID: 10394963
E-mail: ywei14@stevens.edu
Video: In the ZIP Archive File
Single-Process Chat App: ChatAppWithSingleProcess.MOV
Chat App with Separated Service: ChatAppWithSeparatedProcess.MOV

Part 1: Single-Process Chat App

1. Create a sub package called “Service”, add an intent service called ChatReceiverService in the package as Figure 1.1 and Figure 1.2. Because the IntentService is worked on a WorkerThread, so the persist operation can work directly in the Worker Thread.

```
public class ChatReceiverService extends IntentService {
    private static final String TAG = ChatReceiverService.class.getCanonicalName();
    private static final String SEPARATE_CHAR = "|";
    private static final Pattern SEPARATOR = Pattern.compile(Character.toString(SEPARATE_CHAR.charAt(0)), Pattern.LITERAL);
    public static final int CHAT_SERVER_LOADER_ID = 1;
    private DatagramSocket serverSocket;
    private boolean socketOK = true;
    private MessageManager manager;

    public ChatReceiverService() { super("ChatReceiverService"); }

    @Override
    public void onCreate() {
        manager = new MessageManager(this, (cursor) -> {
                return new Message(cursor);
        }, CHAT_SERVER_LOADER_ID);
        super.onCreate();
    }
```

Figure 1.1

```
@Override
protected void onHandleIntent(Intent intent) {
    if (intent != null) {
        if (serverSocket == null) {
            try {
                int port = ChatAppActivity.clientPort;
                serverSocket = new DatagramSocket(port);
            } catch (Exception e) {
                Log.e(TAG, "Cannot open socket " + e.getMessage());
                return;
            }
        }
        while (serverSocket != null && !serverSocket.isClosed()) {
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            try {
                serverSocket.receive(receivePacket);
                Log.i(TAG, "Received a pakcet");
                InetAddress sourceIPAddress = receivePacket.getAddress();
                Log.i(TAG, "Source IP Address: " + sourceIPAddress);
                receiveData = receivePacket.getData();
                int serverPort = receivePacket.getPort();
                String temp = new String(receiveData, 0, receivePacket.getLength());
                if (!temp.isEmpty()) {
                    String[] nameAndContent = SEPARATOR.split(temp);
                    Peer peer = new Peer(nameAndContent[0], sourceIPAddress, serverPort);
                    Message message = new Message(nameAndContent[1], nameAndContent[0]);
                    // Since the IntentService is already run on a Worker Thread, just persist
                    // data directly
                    manager.persistSync(peer, message);
                    Intent broadcastIntent = new Intent(Intent.ACTION_PROVIDER_CHANGED);
                    sendBroadcast(broadcastIntent);
                }
            } catch (IOException e) {
                Log.e(TAG, e.getMessage());
                if (serverSocket != null && serverSocket.isClosed()) {
                    serverSocket = null;
                }
                socketOK = false;
            }
        }
    }
}
```

Figure 1.2

2. In the package “Service”, create another service called “ChatSenderService” which is used to send message to another terminal as Figure 2.1. In the service, define a class IChatSenderService which is used to implement send operation by the main activity as Figure 2.2. In the send method, because the service is still run on the UI thread, so define a new Thread to connect to the Internet and send the message.

```java
public class ChatSenderService extends Service {
    private static final String TAG = ChatSenderService.class.getCanonicalName();

    private DatagramSocket clientSocket = null;
    private boolean socketOK = true;

    private static final String SEPARATE_CHAR = "|";
    private static final int DEFAULT_SENDER_PORT = 6667;

    private final IBinder binder = new IChatSendService();

    @Override
    public IBinder onBind(Intent intent) { return binder; }

    @Override
    public void onCreate() {

        super.onCreate();
    }
```

Figure2.1

```java
    public class IChatSendService extends Binder {
        public ChatSenderService getService() { return ChatSenderService.this; }
    }

    public void send(final String dest, final int port, final String source, final String message) {
        Thread thread = new Thread((Runnable) () -> {
                if (clientSocket == null) {
                    try {
                        clientSocket = new DatagramSocket(DEFAULT_SENDER_PORT);
                    } catch (Exception e) {
                        Log.e(TAG, "Cannot open socket: " + e.getMessage());
                        socketOK = false;
                        return;
                    }
                }
                // TODO: Handle action send
                try {
                    InetAddress destAddr = InetAddress.getByName(dest);
                    String toSend = source + SEPARATE_CHAR + message;
                    byte[] sendData = toSend.getBytes(Charset.forName("UTF-8"));
                    DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, destAddr, port);
                    clientSocket.send(sendPacket);
                    Log.i(TAG, "Send packet: " + message);
                } catch (IOException e) {
                    Log.e(TAG, e.getMessage());
                }
        });
        thread.start();
    }


    @Override
    public void onDestroy() {
        if (socketOK && clientSocket != null) {
            clientSocket.close();
            clientSocket = null;
        }
        super.onDestroy();
    }
}
```

Figure 2.2

3. Modify the main activity, in the onCreate method, define an Intent called intentReceiver to start the ChatReceiverService as Figure 3.1. Override the onStart method, use the sendIntent to bind to the ChatSenderService as Figure 3.2. Use the ServiceConnection to get the service and when the send button is clicked, execute the send method as Figure 3.3 and Figure 3.4. Override the onStop and onDestroy method to stop the ChatReceiverService and unbind the ChatSenderService as Figure 3.5. Finally, define a BroadcastReceiver called Receiver to make a toast when a message is received by the ChatReceiverService as Figure 3.6.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_chat_app);
    Intent callingIntent = getIntent();
    if (callingIntent != null && callingIntent.getExtras() != null) {
        clientName = callingIntent.getExtras().getString(CLIENT_NAME_KEY);
        clientPort = callingIntent.getExtras().getInt(CLIENT_PORT_KEY);
    } else {
        clientName = DEFAULT_CLIENT_NAME;
        clientPort = DEFAULT_CLIENT_PORT;
    }
    destinationHost = (EditText)findViewById(R.id.destination_host);
    destinationPort = (EditText)findViewById(R.id.destination_port);
    messageText = (EditText)findViewById(R.id.message_text);
    messageList = (ListView)findViewById(R.id.message_list);
    manager = new MessageManager(this, (cursor) -> {
            return new Message(cursor);
    }, CHAT_SERVER_LOADER_ID);
    String[] from = new String[] {PeerContract.NAME, MessageContract.MESSAGE_TEXT};
    int[] to = new int[] {R.id.peer_row, R.id.message_row};
    cursorAdapter = new SimpleCursorAdapter(this, R.layout.peer_row, null, from, to, 0);
    messageList.setAdapter(cursorAdapter);

    manager.QueryAsync(MessageContract.CONTENT_URI, new IQueryListener<Message>() {
        @Override
        public void handleResults(TypedCursor<Message> cursor) {
            cursorAdapter.swapCursor(cursor.getCursor());
            //cursor.getCursor().setNotificationUri(getContentResolver(), MessageContract.CONTENT_URI);
        }

        @Override
        public void closeResults() { cursorAdapter.swapCursor(null); }
    });
    // Start receiver service
    intentReceiver = new Intent(this, ChatReceiverService.class);
    startService(intentReceiver);
}
```

Figure 3.1

```
@Override
protected void onStart() {
    // Bind sender service
    sendIntent = new Intent(ChatAppActivity.this, ChatSenderService.class);
    bindService(sendIntent, connection, Context.BIND_AUTO_CREATE);
    super.onStart();
}
```

Figure 3.2

```
private ChatSenderService service;
private boolean serviceBound;

private ServiceConnection connection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder binder) {
        service = ((ChatSenderService.IChatSendService)binder).getService();
        serviceBound = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName name) { serviceBound = false; }
};
```

Figure 3.3

```
public void onClick(View view) {
    String destination = destinationHost.getText().toString();
    int port = Integer.parseInt(destinationPort.getText().toString());
    String message = messageText.getText().toString();
    service.send(destination, port, clientName, message);
    messageText.setText("");
}
```

Figure 3.4

```
@Override
protected void onStop() {
    if (serviceBound) {
        unbindService(connection);
    }
    super.onStop();
}

@Override
protected void onDestroy() {
    if (intentReceiver != null) {
        stopService(intentReceiver);
    }
    super.onDestroy();
}
```

Figure 3.5

```
public static class Receiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "A message has been received", Toast.LENGTH_LONG).show();
        context.getContentResolver().notifyChange(MessageContract.CONTENT_URI, null);
    }
}
```

Figure 3.6

4. Run the App as the video shows.

Part 2: Chat App with Separated Service

1. Follow the steps described in the Part 1.

2. Modify the ChatAppActivity. Create a Messenger called messenger, in the onServiceConnected method, initialize the messenger by the IBinder as Figure 2.1. Define a class called AckReceiverWrapper which extends ResultReceiver which is used to acknowledge that the ChatSenderService has received the message to be sent as Figure 2.2 and Figure 2.3. In the onStart method, initialize the wrapper and bind to the ChatSenderService as Figure 2.4. Define the sendRequest method which packs the message data into the messenger and send to the Service to handle the data as Figure 2.5. The sendRequest method will be executed when the send button is clicked.

```
private Messenger messenger;
AckReceiverWrapper.IReceiver receiver = (resultCode, resultData) -> {
        Toast.makeText(ChatAppActivity.this, "The message has been sent out", Toast.LENGTH_LONG).show();
};
AckReceiverWrapper wrapper;

private ServiceConnection connection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder binder) {
        messenger = new Messenger(binder);
        serviceBound = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName name) { serviceBound = false; }
};
```

Figure 2.1

```
public static class AckReceiverWrapper extends ResultReceiver {
    private IReceiver receiver;
    public AckReceiverWrapper(Handler handler) { super(handler); }

    public void setReceiver(IReceiver receiver) { this.receiver = receiver; }

    @Override
    protected void onReceiveResult(int resultCode, Bundle resultData) {
        if (receiver != null) {
            receiver.onReceiveResult(resultCode, resultData);
        }
    }

    public interface IReceiver {
        public void onReceiveResult(int resultCode, Bundle resultData);
    }
}
```

Figure 2.2

```
AckReceiverWrapper.IReceiver receiver = (resultCode, resultData) -> {
        Toast.makeText(ChatAppActivity.this, "The message has been sent out", Toast.LENGTH_LONG).show();
};
AckReceiverWrapper wrapper;
```

Figure 2.3

```
@Override
protected void onStart() {
    // Bind sender service
    sendIntent = new Intent(ChatAppActivity.this, ChatSenderService.class);
    wrapper = new AckReceiverWrapper(new Handler());
    wrapper.setReceiver(receiver);
    sendIntent.putExtra(ACK, wrapper);
    bindService(sendIntent, connection, Context.BIND_AUTO_CREATE);
    super.onStart();
}
```

Figure 2.4

```
public void onClick(View view) {
    sendRequest();
    messageText.setText("");
}

private void sendRequest() {
    String destination = destinationHost.getText().toString();
    int port = Integer.parseInt(destinationPort.getText().toString());
    String message_text = messageText.getText().toString();
    android.os.Message message = android.os.Message.obtain(null, ChatSenderService.MESSAGE_OBTAIN_KEY, 0, 0);
    Bundle args = new Bundle();
    args.putString(ChatSenderService.DESTINATION_KEY, destination);
    args.putInt(ChatSenderService.PORT_KEY, port);
    args.putString(ChatSenderService.MESSAGE_KEY, message_text);
    args.putString(ChatSenderService.SOURCE_KEY, clientName);
    message.setData(args);
    try {
        messenger.send(message);
    } catch (RemoteException e) {
        Log.e(TAG, e.getMessage());
    }
}
```

Figure 2.5

3. Modify the ChatSenderService. In the onBind method, get the result receiver from the intent sent by the main activity as Figure 3.1. Define a new Handler called MessageHandler which will connect to the Internet, send the message and send the acknowledgement to the result receiver as Figure 3.2. In the onCreate method, define a new HandlerThread which will run as a background thread. Make the messenger to be handled in the HandlerThread as Figure 3.3.

```java
public class ChatSenderService extends Service {
    private static final String TAG = ChatSenderService.class.getCanonicalName();

    public static final int MESSAGE_OBTAIN_KEY = 1;
    public static final String DESTINATION_KEY = "edu.stevens.cs522.chatapp.destination";
    public static final String PORT_KEY = "edu.stevens.cs522.chatapp.port";
    public static final String MESSAGE_KEY = "edu.stevens.cs522.chatapp.message";
    public static final String SOURCE_KEY = "edu.stevens.cs522.chatapp.source";

    private Messenger messenger;
    private Handler handler;
    private DatagramSocket clientSocket = null;
    private boolean socketOK = true;

    private static final String SEPARATE_CHAR = "|";
    private static final int DEFAULT_SENDER_PORT = 6667;

    private ResultReceiver resultReceiver;

    @Override
    public IBinder onBind(Intent intent) {
        resultReceiver = intent.getParcelableExtra(ChatAppActivity.ACK);
        return messenger.getBinder();
    }
```

Figure 3.1

```java
private class MessageHandler extends Handler {
    public MessageHandler(Looper looper) { super(looper); }

    @Override
    public void handleMessage(Message msg) {
        Bundle data = msg.getData();
        String dest = data.getString(DESTINATION_KEY);
        String source = data.getString(SOURCE_KEY);
        String message = data.getString(MESSAGE_KEY);
        int port = data.getInt(PORT_KEY);
        if (clientSocket == null) {
            try {
                clientSocket = new DatagramSocket(DEFAULT_SENDER_PORT);
            } catch (Exception e) {
                Log.e(TAG, "Cannot open socket: " + e.getMessage());
                socketOK = false;
                return;
            }
        }
        // TODO: Handle action send
        try {
            InetAddress destAddr = InetAddress.getByName(dest);
            String toSend = source + SEPARATE_CHAR + message;
            byte[] sendData = toSend.getBytes(Charset.forName("UTF-8"));
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, destAddr, port);
            clientSocket.send(sendPacket);
            Log.i(TAG, "Send packet: " + message);
            resultReceiver.send(ChatAppActivity.RESULT_ACK_OK, null);
        } catch (IOException e) {
            Log.e(TAG, e.getMessage());
        }
    }
}
```

Figure 3.2

```java
@Override
public void onCreate() {
    super.onCreate();
    HandlerThread messengerThread = new HandlerThread(TAG, android.os.Process.THREAD_PRIORITY_BACKGROUND);
    messengerThread.start();
    Looper messengerLooper = messengerThread.getLooper();
    handler = new MessageHandler(messengerLooper);
    messenger = new Messenger(handler);
}
```

Figure 3.3

4. Run the App as the Video shows.