WIFI Direct

για αποστολή μηνυμάτων και αρχείων

και συνδυασμός με δεδομένα 3G/4G για πιο γρήγορο κατέβασμα αρχείων

Λεονάρδος Ζήνων Χατζηγιαχτσίδης - 3140220
Θεόδωρος Ζερβούλιας - 3170050

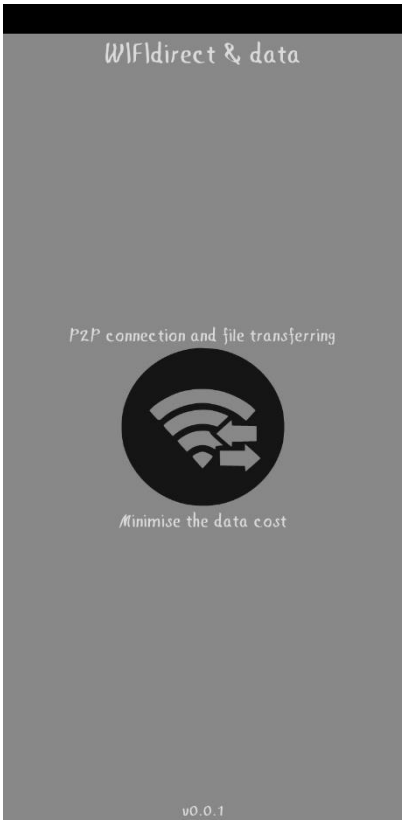
- **Σκοπός**
  Σκοπός της εργασίας μας είναι η ανάπτυξή μία εφαρμογής για κινητά android. Η εφαρμογή θα επιτρέπει σε κινητά να συνδέονται μεταξύ τους μέσω WIFI Direct ώστε να είναι δυνατή η επικοινωνία μέσω γραπτών μηνυμάτων και η αποστολή αρχείων. Επιπλέον οι συσκευές που συνδέονται να μπορούν να μοιράσουν το κόστος τον δεδομένων και να μειώσουν τον χρόνο που χρειάζεται για να κατέβει ένα αρχείο.

- **Η εφαρμογή μας**
  Η εφαρμογή μας επιτρέπει σε 2 κινητά να συνδέονται μέσω του WIFI Direct. Αφού συνδεθούν θα μπορούν να ανταλλάσσουν γραπτά μηνύματα και αρχεία. Επίσης, παρέχει τη δυνατότητα στις συσκευές, αφού διαλέξουν ένα αρχείο από το Διαδίκτυο, να κατεβάσει το μισό η κάθε μία και να το στείλει στην άλλη, ώστε να ενώνονται τα δυο μισά και στις δύο συσκευές, με αποτέλεσμα να έχουν και οι δύο συσκευές το αρχείο έχοντας κατεβάσει μόνο το μισό.
  Για περισσότερες λεπτομέρειες για τον τρόπο λειτουργείας της εφαρμογής μπορείτε να ανατρέξετε στης διαφάνειες της παρουσίασης της εργασίας και να παρακολουθήσετε το demo.

- **Υλοποίηση**
  Η εφαρμογή υλοποιήθηκε μέσω Android Studio σε γλώσσα Java. Παρακάτω θα εξηγήσουμε αναλυτικά πως υλοποιείται προγραμματιστικά η κάθε λειτουργεία που αναφέραμε προηγουμένως με περισσότερες λεπτομέρειες , αναφέροντας στις κλάσεις που δημιουργήθηκαν και σε κομμάτια του κώδικα. Ειδικότερα:
    - S**plashScreenActivity:**
      Κατά την εκκίνηση της εφαρμογής εμφανίζεται ένα αρχικό

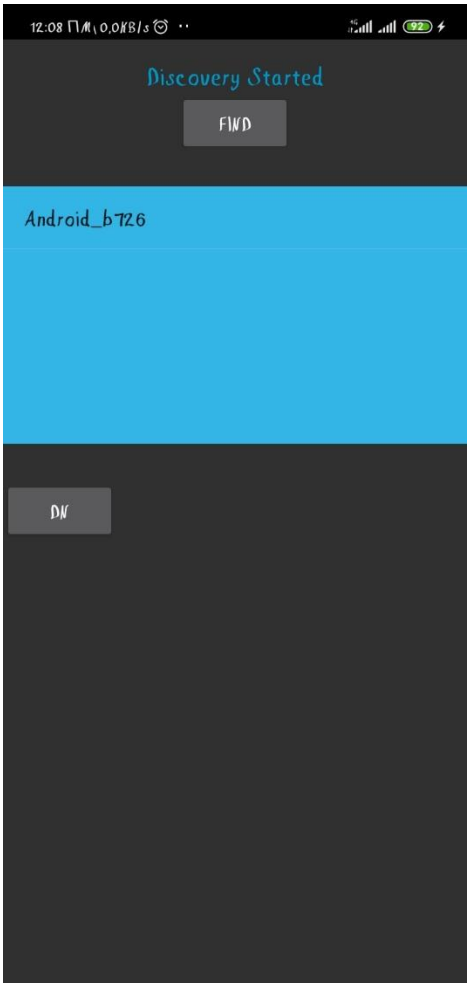
παράθυρο με το logo, τον τίτλο και την έκδοση της εφαρμογής. Μετά από λίγα δευτερόλεπτα η εφαρμογή προχωράει στο επόμενο παράθυρο.
Για την υλοποίηση χρησιμοποιήθηκε η βιβλιοθήκη EasySplashScreen (https://android-arsenal.com/details/1/3514, Owner: Leonidas Maroulis).

EasySplashScreen splashscreen = new EasySplashScreen(SplashScreenActivity.this);
View easySplashScreen = splashscreen.create();
setContentView(easySplashScreen);



- **MainActivity:**
  Το επόμενο παράθυρο μετά το SplashScreen είναι το MainActivity. Εκεί γίνεται η διαδικασία της εύρεσης άλλων συσκευών και σύνδεσης με αυτές μέσω του WIFI Direct. Στο παράθυρο αυτό υπάρχει το κουμπί Find και μια λίστα με τις διαθέσιμες συσκευές. Το κουμπί Find αρχίζει να αναζητεί συσκευές που μπορεί να συνδεθεί και κάνει την ίδια ορατή σε άλλες συσκευές. Στη λίστα με τις διαθέσιμες συσκευές, όταν πατηθεί το στοιχείο τις λίστας με το όνομα της συσκευής που θέλουμε να συνδεθούμε, θα γίνει η σύνδεση.

btnFindDevices = (Button) findViewById(R.id.findDevices);
peerListView = (ListView) findViewById(R.id.peerListView);



Για την υλοποίηση της σύνδεσης χρησιμοποιούμε αντικείμενα WifiP2pManager, WifiP2pManager.Channel, BroadcastReciever.

wifiP2pManager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
wifiChannel = wifiP2pManager.initialize(this, getMainLooper(), null);
broadcastReceiver = new WiFiDirectBroadcastReceiver(wifiP2pManager, wifiChannel, this);

Με την εντολή discoverPeers του WifiP2pManager βρίσκουμε συσκευές, και με την εντολή connect συνδεόμαστε στη συσκευή που επιλέγουμε. Όταν συνδεθούν δύο συσκευές, η μία θα λειτουργήσει ως host και η άλλη ως client.

```
wifiP2pManager.discoverPeers(wifiChannel, new WifiP2pManager.ActionListener() {…});
```

```
final WifiP2pDevice device = devices[position];
WifiP2pConfig config = new WifiP2pConfig();
config.deviceAddress = device.deviceAddress;
wifiP2pManager.connect(wifiChannel, config, new WifiP2pManager.ActionListener() {…});
```

Για το BroadcastReciever υλοποιούμε μια δικιά μας κλάση **WiFiDirectBroadcastReciever** η οποία κληρονομεί από την BroadcastReciever. Αυτή η κλάση ακούει μηνύματα για αλλαγές στην κατάσταση του WIFI (συνδέσεις, αποσυνδέσεις κτλ.) και εκτελεί τις κατάλληλες εντολές. Αυτή η κλάση όταν καταλάβει ότι έχουν συνδεθεί οι δύο συσκευές θα ειδοποιήσει την MainActivity ώστε αυτή να προχωρήσει στο επόμενο παράθυρο.

```
if(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {
        ….
if(networkInfo.isConnected()) {
        wifiP2pManager.requestGroupInfo(wifiChannel, mainActivity.groupInfoListener);
        wifiP2pManager.requestConnectionInfo(wifiChannel,
                                        mainActivity.connectionInfoListener);
        }
}
```

```
WifiP2pManager.ConnectionInfoListener connectionInfoListener = new
WifiP2pManager.ConnectionInfoListener() {
        …..

if(info.groupFormed) {
        Intent i = new Intent(getApplicationContext(), MessageActivity.class);
        …..
        startActivity(i);
        } }
```

- **MessageActivity:**
  Κατά τη δημιουργία αυτού του παραθύρου τα κινητά θα συνδεθούν μέσω sockets. Η συσκευή που καθιερώθηκε ως host από την διαδικασία της σύνδεσης με WIFI Direct θα δημιουργήσει server socket στο οποίο θα συνδεθεί η άλλη. Μετά την σύνδεση θα

αρχίσουν δύο threads στην κάθε συσκευή, ένα για να διαβάζει δεδομένα μέσω του socket.getInputStream() και ένα για να στέλνει μέσω του socket.getOutputStream();

Η δημιουργία των sockets γίνεται στις κλάσεις **ServerClass** και **ClientClass** για το server και το client socket αντίστοιχα. Η αποστολή και ανάγνωση μηνυμάτων από τα ouput και input streams των sockets γίνεται στις κλάσεις **Send** και **Receive** αντίστοιχα.

```
if(type.equals("host")) {
        serverClass = new ServerClass(this);
         serverClass.start();
    } else if(type.equals("client")) {
        clientClass = new ClientClass(groupOwnerAddress, this);
         clientClass.start();
     }
```

Server Class:

```
serverSocket = new ServerSocket(8888);
socket = serverSocket.accept();
receive = new Receive(socket, messageActivity);
receive.start();
send = new Send(socket, messageActivity);
send.start();
```

Client Class:

```
socket.connect(new InetSocketAddress(hostAddress, 8888));
receive = new Receive(socket, messageActivity);
receive.start();

send = new Send(socket, messageActivity);
send.start();
```

Send:

```
outputStream = socket.getOutputStream();
….
outputStream.write(message);
```

Receive:

```
inputStream = this.socket.getInputStream();
….
inputStream.read(buffer);
```

Ό,τι μήνυμα στέλνεται από τη μία συσκευή στην άλλη στέλνεται σε μορφή πίνακα από bytes. Για κάθε διαφορετικό είδος μηνύματος στην αρχή του πίνακα προσθέτουμε ένα αναγνωριστικό byte ώστε όταν το διαβάσει η άλλη συσκευή να ξέρει τι είδος μήνυμα είναι. Παράδειγμα τέτοιου αναγνωριστικού είναι το TEXT_MESSAGE που ισούται με 1. Για την προσθήκη τέτοιου αναγνωριστικού byte στην αρχή του πίνακα, καθώς και για άλλες πράξεις μεταξύ πινάκων έχουμε φτιάξει μια κλάση **Byte**. Όταν μία συσκευή διαβάσει κάτι από το stream εισόδου στέλνεται το μήνυμα από το thread Receive στην MessageActivity μέσω ενός αντικειμένου handler.

Παράδειγμα αποστολή γραπτού μηνύματος:

```
String msg = writeMsgText.getText().toString();
byte[] message = Bytes.addIdentifier(TEXT_MESSAGE, msg.getBytes());
if(type.equals("host")) {
        serverClass.send.setMessage(message);
    } else if(type.equals("client")) {
        clientClass.send.setMessage(message);
    }
```

Και η ανάγνωση:

```
Receive:

byte id = buffer[0];
messageActivity.handler.obtainMessage(id, count, -1, buffer).sendToTarget();

MessageActivity:

Handler handler = new Handler(new Handler.Callback() {
    @Override
    public boolean handleMessage(@NonNull Message msg) {
        switch (msg.what) {
                ……
            case TEXT_MESSAGE:
                byte[] readBuff = Bytes.removeIdentifier((byte[]) msg.obj);
                // εμφάνιση μηνύματος στην οθόνη
```

Για την αποστολή ενός αρχείου στέλνεται πρώτα ένα μήνυμα για να ειδοποιήσουμε την άλλη συσκευή ότι πρόκειται να στείλουμε ένα αρχείο λέγοντάς της το μέγεθος και το όνομα του αρχείου. Όταν διαβαστεί ένα τέτοιο μήνυμα δεσμεύουμε τον κατάλληλο χώρο με βάση το μήνυμα και όταν τα bytes που διαβάζουμε ύστερα

αποθηκεύονται σε αυτόν μέχρι να γεμίσει.

Αποστολή:

```
byte[] message = Bytes.addIdentifier(FILE_INFO,
Bytes.concatenateArrays(Bytes.intToByteArray(latestFile.length), latestFileName.getBytes()));
sendMessage(message);

sendMessage(latestFile);
```

Ανάγνωση:

```
if(id == messageActivity.FILE_INFO || id == messageActivity.HALF_FILE_INFO) {
        sizeSoFar = 0;
        latestFile = new byte[Bytes.byteArrayToInt(Arrays.copyOfRange(buffer, 1, 5))];
        while(sizeSoFar < latestFile.length) {
            //διάβασμα από το Input stream
        }
```

Για την ανάγνωση και αποθήκευση αρχείων στον χώρο μνήμης έχουμε φτιάξει την κλάση **Files** η οποία υλοποιεί τις μεθόδους loadFile και saveFile. Οι μέθοδοι αυτοί χρησιμοποιούν FileInputStream και FileOutputStream αντίστοιχα.

Για το κατέβασμα ενός αρχείου έχουμε φτιάξει την κλάση Downloader η οποία τρέχει σε καινούριο thread και κατεβάζει το πρώτο ή το δεύτερο μισό του αρχείου ανάλογα με τις παραμέτρους κατά την κατασκευή του αντικειμένου αυτού του τύπου. Η συσκευή που κατεβάζει το πρώτο κομμάτι στέλνει το link στην άλλη για να κατεβάσει το δεύτερο.

```
public void run() {
    if(firstHalf) {
      downloadFirstHalf();
    } else {
      downloadSecondHalf();
    }
}
```

Για το κατέβασμα, δημιουργούμε ένα αντικείμενο HttpsURLConnection με βάση το URL και παίρνουμε το μέγεθος του αρχείου με την εντολή connection.getContentLength(). Υστερα χωρίζουμε το αρχείο σε chunks του 1KB και υπολογίζουμε πόσα θα

πρέπει να κατεβάσουμε. Στο κατέβασμα του πρώτου μισού του αρχείου κατεβάζουμε το αρχείο από την αρχή μέχρι να γεμίσουν τα chunk που πρέπει να κατεβάσουμε (περίπου να μισά, ίσως ένα ή δύο περισσότερα από το δεύτερο μισό). Για το δεύτερο μισό κατεβάζουμε τα chunks που πρέπει αρχίζοντας από τη μέση μέχρι το τέλος του αρχείου. Για να κατεβάζουμε το αρχείο από τη μέση χρησημοποιούμε την εντολή connection.setRequestProperty("Range", "bytes=" startPosition + "-" + fileLength). Ο χωρισμός του αρχείου σε chunks έγινε ώστε να μπορούμε να διαβάζουμε το αρχείο ανάποδα σε chunks. Αλγοριθμικά βρήκαμε την λύση χρησιμοποιώντας την εντολή setRequestProperty που είδαμε προηγουμένως πολλές φορές για διάφορες τιμές, άλλα δεν προλάβαμε να την υλοποιήσουμε σωστά άρα το αφήσαμε ως έχει. Επίσης δεν προλάβαμε να υλοποιήσουμε κατέβασμα αρχείου όταν είναι http και όχι https αν και γενικά γίνεται με παρόμοιο τρόπο. Για περσότερες λεπτομέρειες ανατρέξτε στον κώδικά μας, στην κλάση Downloader.java.

Η αποστολή του μισού αρχείου γίνεται με παρόμοιο τρόπο με την αποστολή ολόκληρου αρχείου που είδαμε προηγουμένως.

Μπορείτε να δείτε αναλυτικά τον κώδικά μας στο GitHub στον παρακάτω σύνδεσμο: https://github.com/TeoZMS/Project_WIFIdirect1

Σε αυτή την έκδοση της εφαρμογής μας έχουμε αφήσει και διάφορες άλλες λειτουργίες που μας βοηθούν για debugging (το κουμπί DN στην mainActivity το οποίο κατεβάζει ένα αρχείο μισό μισό και το ενώνει) ή λειτουργείες που δεν προλάβαμε να ολοκληρώσουμε (loadImage κουμπί και ImageView στην message Activity για εμφάνιση φωτογραφιών όταν αποστέλνονται από το ένα κινητό στο άλλο).