گزارش پروژه ی نهایی

اعضای گروه:

مبینا کاشانیان 96522321

فاطمه صفاری 95522103

درس شبکه های تلفن همراه

دکتر دیانت

تابستان 1399

فهرست

موضوع پروژه

نحوه ی اندازه گیری پارامتر های QoE

بدست آوردن اطلاعات سلول

نحوه ی ذخیره ی اطلاعات اندازه گیری شده در پایگاه داده

نمایش location و کیفیت سلول بر روی نقشه

1. افزودن نقشه
2. تنظیم permission ها برای دسترسی به location
3. نمایش location روی نقشه
4. بروزرسانی location و اطلاعات

موضوع پروژه

توسعه ی یک اپلیکیشن در گوشی های تلفن همراه با سیستم عامل Android که بتوان توسط آن پارامترهای QoE یک شبکه تلفن همراه را مورد ارزیابی قرار داد.

نحوه ی اندازه گیری پارامتر های QoE

بدست آوردن اطلاعات سلول

نحوه ی ذخیره ی اطلاعات اندازه گیری شده در پایگاه داده

نمایش location و کیفیت سلول بر روی نقشه

1.اندازه گیری پارامتر ها:

-مقدار و کیفیت سیگنال:

یکی از مهم ترین قسمت های این پروژه محاسبه ی پارامتر های مورد نیاز برای نمایش آنها روی نقشه است.

در ابتدا به دست آوردن پارامتر های مربوط به گوشی میپردازیم برای اینکار از کتاب خانه ی Telephony managerاستفاده میکنیم کتاب خانه ی Telephony manager یکی از کتاب خانه های پر استفاده در این پروژه است با استفاده از این کلاس میتوانیم اطلاعاتی را در مورد خدمات تلفنی در دستگاه فراهم کنیم از جمله اطلاعاتی که میتوانیم به دست آوریم اطلاعاتی در خصوص subscriber information است.

برای گرفتن پارامتر ها یک تابعی را به نام getMobileInfo() پیاده سازی میکنیم.

لازم به ذکر است که این پارامتر ها را در دیتابیس ذخیره میکنیم که توضیحات مربوط به دیتابیس را در بخش 3 مشاهده میکنید.

با استفاده از کلاس PhoneStateListener میتوانیم تغییرات پارامتر های مربوط به سیگنال انواع شبکه های تلفن همراه را دریافت کنیم در واقع این کلاس یک ناظر بر تغییرات حالت های تلفن می باشد مانند تغییرات در وضعیت سرویس،قدرت سیگنال و پست صوتی و...

لازم به ذکر است که هنگامی که می خواهیم از این کلاس استفاده کنیم نیازمند دادن یک سری اجازات به برنامه هستیم.برای تغییر وضعیت سیگنال گوشی از signal listener استفاده میکنیم.

برای گرفتن دو پارامتر PLMN و PLMN name از کلاس telephony managerاستفاده میکنیم و کد زیر را برای گرفتن این دو پارامتر مینویسیم:

```
plmn = telephonyManager.getSimOperator();
plmnname = telephonyManager.getSimOperatorName();
```

حال به سراغ اندازه گیری پارامتر های مربوط به کیفیت و توان سیگنال در هر نسل شبکه ی تلفن همراه میپردازیم برای اینکار ابتدا با استفاده از کتابخانه ی telephony manager اطلاعات تمام سلول را دریافت میکنیم از آنجایی که برنامه های حاضر باید تمام نسل های تلفن همراه را پوشش دهد برای هر اطلاعات دریافتی که در لیست قرار دارد ابتدا نسل مورد نظر را پیدا میکنیم و سپس با توجه به نسل شروع به گرفتن پارامتر ها میکنیم برای مثال نسل دوم تلفن همراه که شبکه ی GSM نام دارد را شرح میدهیم و بقیه ی نسل ها همانند همین نسل قابل محاسبه هست،لطفا توجه بفرمایید در برخی از نسل ها مفاهیم همان مفاهیمی که در درس به آنها پرداختیم میباشد و فقط نام های آن تغییر کرده است:

```
for (CellInfo cellInfo : cellInfoList) {
    if (cellInfo instanceof CellInfoGsm) {

        gsm_int_dbm = ((CellInfoGsm) cellInfo).getCellSignalStrength().getDbm();
        gsm_int_rxlev = getDbmLevel(gsm_int_dbm);
        color = gsm_int_rxlev;
    }
```

ابتدا سیگنال را دریافت میکنیم و در یک متغییر آن را ذخیره میکنیم سپس با توجه به تابعی به نام getDbmLevel مقداری که از پارامتر دریافت کردیم را در چهارچوب چهار کیفیت نمایش میدهیم

```
public static int getDbmLevel(int dbm) {
    if (dbm < -100) return 0;     // to describe clearly 0 is very poor
    else if (dbm < -95) return 1; // poor
    else if (dbm < -85) return 2; // fair
    else if (dbm < -75) return 3; // good
    else if (dbm != 0) return 4;  // excellent
    else return -1;               // null
}
```

همان طور که ملاحظه میکنید 5 کیفیت را برای تمام پارامتر ها در نظر گرفتیم که آنها را بتوانیم روی نقشه نشان دهیم و انتخاب این بازه ها به صورت تحقیقی در اینترنت صورت گرفته است.

طبق داکیومنت نوشته شده در سایت https://developer.android.com/ با استفاده از asusleve() پارامتر rssi قابل محاسبه میباشد و همچنین از طریق getlevel میتوان کیفیت سیگنال را اندازه گیری کرد برای اندازه گیری LACهم میتوان از کد زیر استفاده نمود:

```
TelephonyManager m = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
CellLocation location = m.getCellLocation();
GsmCellLocation gsmLocation = (GsmCellLocation) location;

int cellId = gsmLocation.getCid();
int lac = gsmLocation.getLac();
String ret = "LAC In Hex: " + Integer.toHexString(gsmLocation.getLac()) + " CID In Hex: " + Integer.toHexString(gsmLocation.getCid());
```

برای بقیه ی نسل ها هم میتوان طبق همین روش پارامتر های مورد نیاز مربوط به کیفیت سیگنال و مقدار آن را اندازه گیری کرد.

-شناسه های منحصر به فرد گوشی:

برای به دست آوردن شناسه های منحصر به فرد گوشی همانند پارامتر مقدار و کیفیت سیگنال ابتدا لیست اطلاعات سلول را در هر نسل دریافت میکنیم و سپس متناسب با هر نسل اقدام به محاسبه ی این شناسه ها میکنیم:

برای مثال این بار نسل چهارم تلفن های همراه که همان LTE میباشد را توضیح میدهیم و بقیه ی نسل ها همانند همین نسل قابل محاسبه هستند:

```
for (CellInfo cellInfo : cellInfoList) {
    if (cellInfo instanceof CellInfoLte) {
        cellSig = ((CellInfoLte) cellInfo).getCellSignalStrength().getDbm();
        cellsigdbm = getRsrpLevel(cellSig);
        cellID = ((CellInfoLte) cellInfo).getCellIdentity().getCi();
        cellMcc = ((CellInfoLte) cellInfo).getCellIdentity().getMcc();
        cellMnc = ((CellInfoLte) cellInfo).getCellIdentity().getMnc();
        cellPci = ((CellInfoLte) cellInfo).getCellIdentity().getPci();
        cellTac = ((CellInfoLte) cellInfo).getCellIdentity().getTac();
    }
    if (cellInfo instanceof CellInfoWcdma)
```

یکی از پارامتر های مورد نیاز در هر سلول شناسه ی سلول میباشد که این شناسه ی سلول یک شناسه ی منحصر به فرد میباشد که این پارامتر را از طریق getci()انجام دادیم البته لازم به ذکر است که در ابتدا با توجه به نسل چهار بودن گوشی اطلاعات مربوط به این نسل را از تمام اطلاعات سلول جدا میکنیم و سپس بخش getcellidentity() هویت نسل مورد نظر را برایمان مشخص میکند و سپس با استفاده از صدا کردن آن تابع getci() شناسه ی سلول را دریافت میکنیم.

پارامتر دوم و سوم که mcc و mnc است مربوط به plmn میباشد که منظور mobile country code و mobile network code میباشد و همچنین پارامتر pci و Tac را هم برای این نسل محاسبه میکنیم.

-شبکه ی موبایل استفاده کننده و phone type

این پارامتر ها را با استفاده از کتابخانه ی telephony manager محاسبه میکنیم که مشخص میکند کاربر از چه نسل از شبکه ی موبایل استفاده میکند و نوع تلفن که از چه نوع نسلی میباشد را هم مشخص میکند همچنین با استفاده از این پارامتر ها نسل مورد استفاده را هم مشخص میکنیم.

-طول و عرض جغرافیایی:

مکان دقیق کاربر با استفاده از نقشه قابل پیاده سازی میباشد.

-زمان ثبت رخداد:

با استفاده از کلاس calander این پارامتر را اندازه گیری میکنیم.

-پارامتر های مربوط به شبکه:

این پارامتر ها که مربوط به شبکه است منظور response time download speed upload ping latency jitter throughput httpresponse speed را با استفاده از کتابخانه ی httpok پیاده سازی کردیم لازم به ذکر است که برخی از این پارامتر ها در برخی از ورژن های اندروید پشتیبانی نمیشود برای همین قابل مشاهده برای همه ی نسل های تلفن همراه نمی باشند.

روش کار به این صورت است که ابتدا به یک سایت دلخواه ریکوئست میزنیم و تا رسیدن پاسخ و ریسپانس این وبسایت میزان زمانی که طول میکشد تا این رخداد انجام شود را اندازه گیری میکنیم.

پارامتر های اضافه تری هم از شبکه دریافت کردیم که این پارامتر ها را در تصویر زیر مشاهده میکنید:

```
nettype_text = "Network Type = " + nettype_text;
MobileRxBytes = "Mobile Rx = " + MobileRx + " KB";
getMobileTxBytes = "Mobile Tx = " + MobileTx + "KB";
WifiRx_text = "Wifi Rx =" + WifiRx + " KB";
WifiTx_txt = "Wifi Tx = " + WifiTx + " KB";
traffic = "throughput = " + "IP Address = " + ipAddress + "Wifi BSSID = " + wifi_bssid + " Wifi Rssi = " + wifi_rssi +
        " Wifi SSID  = " + wifi_ssid + " Wifi Mac Address  = " + mac_address + " Link Speed  = " + linkspeed + " RxLinkSpeedMbps = "
        + rx_mbps + "Mbps \n" + "TxLinkSpeedMbps = " + tx_mbps + "Mbps";
http = "Http response time = " + http_rtt;
latency_txt = "Latency = " + late + "ms ";
jitter_txt = "jitter = " + jitter + "ms ";
ping_txt = "Ping = " + Ping;
```

2.کار با پایگاه داده و ذخیره اطلاعات:

در این پروژه از RoomDatabase استفاده کردیم که setup های اولیه مربوط به استفاده از این پایگاه داده را انجام میدهیم و سپس dao این دیتابیس را پیاده سازی میکنیم همانند تصویر زیر :

```
@Dao
public interface ApplicationDao {

    @Query("SELECT * FROM Parametrs")
    List<Parametrs> getAll();

    @Query("DELETE FROM Parametrs")
    void deleteAll();

    @Insert
    void insertAll(Parametrs... parametrs);
}
```

نکته ی حائز اهمیت در پیاده سازی این دیتابیس آن است که برای insert کردن به این پایگاه داده ما اعضای یک کلاس را در این پایگاه داده ذخیره میکنیم.

کلاسی را تحت عنوان Parameterمیسازیم در این کلاس یک کانستراکتور میسازیم که تمام پارامتر های مورد نیاز را در دیتابیس ذخیره میکنیم سپس برای هم memberدر این کانستراکتور یک ستون در پایگاه داده میسازیم.

3. افزودن نقشه

برای نمایش اطلاعات روی یک نقشه باید یک MapsActivity ایجاد کرد. برای این کار باید از طریق زیر اقدام کرد:

app -> New -> Google -> Google Maps Activity

با اینکار سه فایل در پروژه ایجاد می شود:

- MapsActivity.java
- activity_maps.xml
- google_maps_api.xml

در MapsActivity.java تمام الگوریتم های مربوط به نقشه و نمایش کیفیت سلول و غیره قرار دارد. وظیفه ی activity_maps.xml نمایش نقشه است. در google_maps_api.xml باید API key را قرار داد. این کلید برای دسترسی به Google Map Server نیاز است. این مقدار از رجوع به این سایت بدست می آید. برای استفاده از نقشه نیاز است که Google Play services SDK نصب شده باشد.

2. تنظیم permission ها برای دسترسی به location

برای دسترسی به location کاربر GPS تلفن باید روشن باشد. پس باید آن را در وابستگی ها اضافه کرد. کد زیر در buid.gradle(Project: Map) به همین منظور اضافه شده است:

```
implementation 'com.google.android.gms:play-services-maps:17.0.0'
implementation 'com.google.android.gms:play-services-location:17.0.0'
```

از آنجایی که دسترسی به location و اطلاعات آن در خارج از محدوده ی دسترسی های یک اپلیکیشن اندروید است، برای دسترسی داشتن به آن باید permission دسترسی به آن را درخواست کند. برای گرفتن دسترسی ابتدا در AndroidManifest.xml اعلام می شود که این اپلیکیشن به این دسترسی نیاز دارد. اینکار از طریق افزودن کد زیر صورت می گیرد:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

سپس در زمان اجرا (run time) از کاربر درخواست می شود که این permission ها را بدهد. اینکار از طریق افزودن کد زیر در Activity صورت می گیرد:

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            fetchLocation();
        }
    }
}
```

3. نمایش location روی نقشه

برای کار با location باید کتابخانه ی زیر را import کرد:

```
import static
com.google.android.gms.location.LocationServices.getFusedLocationProviderC
lient;
```

locationProvider ساختار داده ای است که شامل پارامتر های کیفیت خدمات است. با استفاده از دستور زیر یک location Provider ایجاد می شود:

```
fusedLocationProviderClient = getFusedLocationProviderClient(this);
```

سپس تابع زیر را فراخوانی می کنیم:

```
fetchLocation();
```

این تابع به دریافت آخرین location و نمایش آن روی نقشه می پردازد.

```
private void fetchLocation() {
```

در ابتدای این تابع ابتدا مجددا permission دسترسی به location چک می شود.

```
    if (ActivityCompat.checkSelfPermission(
            this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
            this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
```

```
        return;
    }
```

سپس همان طور که گفته شد با استفاده از locationProvider ای که قبلا تعریف شده، آخرین location دریافت می شود.

```
    Task<Location> task = fusedLocationProviderClient.getLastLocation();
    task.addOnSuccessListener(new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(Location location) {
            if (location != null) {
                currentLocation = location;
                SupportMapFragment supportMapFragment =
(SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
                assert supportMapFragment != null;
```

سپس با دستور زیر این activity را به onMapReady() پاس داده می شود:

```
                supportMapFragment.getMapAsync(MapsActivity.this);

            }
        }
    });
}
```

سپس در onMapReady() با استفاده از دستورات زیر محل آخرین location و رنگ کیفیت سلولی که تلفن در آن location به آن سلول متصل بوده نمایش داده می شود:

```
@Override
public void onMapReady(GoogleMap googleMap) {
```

هر location یک مختصات جغرافیایی دارد. برای نمایش location باید مختصات جغرافیایی آن را استخراج کنیم. این کار توسط LatLng صورت می گیرد. LatLng نقطه ای در مختصات جغرافیایی است.

```
    LatLng latLng = new LatLng(currentLocation.getLatitude(),
currentLocation.getLongitude());
```

سپس با دستور زیر به قسمتی از نقشه که مختصات استخراج شده در آن قرار دارد حرکت می کند. با متحرک کردن این تغییر، دیگر کاربر نیاز به دنبال کردن مسیر حرکت خود روی نقشه ندارد و مکان او به طور خودکار در نقشه حرکت می کند. بدین ترتیب رابط کاربری ساده تر می شود و کاربر حس بهتری را تجربه خواهد کرد.

```
    googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
```

میزان بزرگی و zoom روی نقشه برابر با 17 قرار داده شده. 15 در سطح خیابان ها و 20 در سطح ساختمان ها است و 17 مابین این دو است. بنابراین مسیر حرکت بهتر نمایش داده می شود.

```
    googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,
17));
```

برای اینکه آخرین مکان کاربر از مکان های قبلی وی متمایز باشد، بر روی LatLng آخرین مکان با دستور زیر یک Marker اضافه می شود. هر بار قبل از اجرای مجدد تابع fetchLocation() با دستور remove() آن حذف می شود تا هر بار فقط یک Marker روی نقشه باشد.

```
    current = googleMap.addMarker(new
MarkerOptions().position(latLng).title("Current Location"));
```

در جای دیگری از کد، رنگ بر اساس کیفیت سلول تعیین می شود و در متغییر color ذخیره می شود:

- 1- به معنای عدم وجود سیگنال با رنگ مشکی
- 0 به معنای کیفیت خیلی ضعیف با رنگ قرمز
- 1 به معنای کیفیت ضعیف با رنگ نارنجی
- 2 به معنای کیفیت متوسط با رنگ زرد
- 3 به معنای کیفیت خوب با رنگ سبز کم رنگ
- 4 به معنای کیفیت عالی با رنگ سبز پر رنگ

با دستور addCircle() روی نقشه یک دایره به مرکز LatLng و شعاع 10 و رنگ تعیین شده رسم می شود که نمایانگر کیفیت سرویس است.

```
    // Set Color
    if (color == 4) // Excellent    dark green
        googleMap.addCircle(new
CircleOptions().center(latLng).radius(10).strokeColor(Color.rgb(0,100,0)).
fillColor(Color.rgb(0,100,0)));
    if (color == 3) // Good         green
        googleMap.addCircle(new
CircleOptions().center(latLng).radius(10).strokeColor(Color.GREEN).fillCol
or(Color.GREEN));
    if (color == 2) // Fair         yellow
        googleMap.addCircle(new
CircleOptions().center(latLng).radius(10).strokeColor(Color.YELLOW).fillCo
lor(Color.YELLOW));
    if (color == 1) // Poor         orange
        googleMap.addCircle(new
CircleOptions().center(latLng).radius(10).strokeColor(Color.rgb(255,165,0)
).fillColor(Color.rgb(255,165,0)));
    if (color == 0) // Very Poor    red
        googleMap.addCircle(new
CircleOptions().center(latLng).radius(10).strokeColor(Color.RED).fillColor
(Color.RED));
    if (color == -1) // No Signal   black
        googleMap.addCircle(new
CircleOptions().center(latLng).radius(10).strokeColor(Color.BLACK).fillCol
or(Color.BLACK));
}
```

4. بروزرسانی location و اطلاعات

در این اپلیکیشن موقعیت کاربر، کیفیت سلول و سایر اطلاعات شبکه در هر لحظه و دائما اندازه گیری می شوند و در پایگاه داده ثبت می شوند و روی نقشه نمایش داده می شوند. این کار ها در تابعی به نام startLocationUpdates() صورت می گیرد. در این تابع به روز رسانی دائمی پیاده سازی شده است. این تابع در onCreate() در MapsActivity.java فراخوانی شده است.

```
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setupRoomDatabase();
    setContentView(R.layout.activity_maps);

    startLocationUpdates();
}
```

در این تابع یک سری تعاریف متغیر و یک loop قرار دارد. این loop هر 5 ثانیه یکبار اجرا می شود. کلیه ی توابع جهت دریافت اطلاعات و غیره در تابع getMobileInfo() فراخوانی شده اند. خود این تابع در onLocationChanged() فراخوانی شده است و خود onLocationChanged() در این loop فراخوانی می شود.
قبل از درخواست به روزرسانی مکان، اپلیکیشن باید به یک location service متصل شود و یک درخواست موقعیت مکانی (LocationRequest) را ایجاد کند. سپس می تواند بروزرسانی دائم را انجام دهد. چون این loop هر 5 ثانیه یکبار اجرا می شود، بنابراین مقدار متغیرUPDATE_INTERVAL برابر با 5000 قرار داده شده است.

```
Location currentLocation;
FusedLocationProviderClient fusedLocationProviderClient;
private static final int REQUEST_CODE = 101;

LocationRequest mLocationRequest;
long UPDATE_INTERVAL = 5 * 1000; // 5 secs
long FASTEST_INTERVAL = 5 * 1000; // 5 sec

protected void startLocationUpdates() {

    // Create the location request to start receiving updates
    mLocationRequest = new LocationRequest();
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    mLocationRequest.setInterval(UPDATE_INTERVAL);
    mLocationRequest.setFastestInterval(FASTEST_INTERVAL);

    // Create LocationSettingsRequest object using location request
    LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder();
    builder.addLocationRequest(mLocationRequest);
    LocationSettingsRequest locationSettingsRequest = builder.build();

    // Check whether location settings are satisfied
    SettingsClient settingsClient =
LocationServices.getSettingsClient(this);
    settingsClient.checkLocationSettings(locationSettingsRequest);
```

در ابتدا و قبل از اجرای loop یکبار getMobileInfo() و fetchLocation() اجرا می شوند و بار های بعدی از طریق onLocationChanged() به دریافت اطلاعات و نمایش روی نقشه پرداخته می شود. به این دلیل که تابع onLocationChanged() فقط با تغییر location اجرا می شود و در اولین بار اجرای برنامه location ای گرفته نشده است.

```
    // Start
    getMobileInfo();
```

```
    fusedLocationProviderClient = getFusedLocationProviderClient(this);
    fetchLocation();

    // Update
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }

getFusedLocationProviderClient(this).requestLocationUpdates(mLocationReque
st, new LocationCallback() {
        @Override
        public void onLocationResult(LocationResult locationResult) {
            onLocationChanged(locationResult.getLastLocation());
        }
    }, Looper.myLooper());
}
```

همان طور که گفته شد در تابع onLocationChanged() تابع getMobileInfo() فراخوانی می شود که وظیفه ی آن دریافت اطلاعات است.

```
public void onLocationChanged(Location location) {
    getMobileInfo();
```

اطلاعات location در String msg و تمام اطلاعات اندازه گیری شده در String info قرار دارند. آنها با Toast بر روی نقشه هربار نمایش داده می شوند. همزمان روی نقشه موقیت کنونی با رنگ کیفیت سلول آن و رنگ کیفیت سلول های مکان های قبلی نمایش داده می شوند.

```
    // New location
    String msg = "Updated Location: " + location.getLatitude() + "," +
location.getLongitude() + "\n" + info;
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();

    // Remove previous Marker
    current.remove();
    fusedLocationProviderClient = getFusedLocationProviderClient(this);
    fetchLocation();
}
```