

# Android alkalmazásfejlesztés

Kommunikáció Androidon  
Mobilinternet  
Wifi

OE-NIK

2012. március 24.

**Sicz-Mesziár János**

sicz-mesziar.janos@  
nik.uni-obuda.hu



# Kommunikációs eszközök Androidon

## ◎ Wi-Fi

- Ma már minimum: a / b / g szabványok, újabbakon már n szabvány is

## ◎ Mobilinternet

- EDGE, UMTS, HSPDA, ...
- Rosszabb lefedettség, mint GSM esetén

## ◎ GSM / SMS / GPRS

- Jó lefedettség, klasszikus, már megszokott
- Elsődlegesen beszédátvitel volt a cél

## ◎ Bluetooth

- Egy hálózatban 1 masterhez max. 7 másik eszköz csatlakozhat

## ◎ NFC

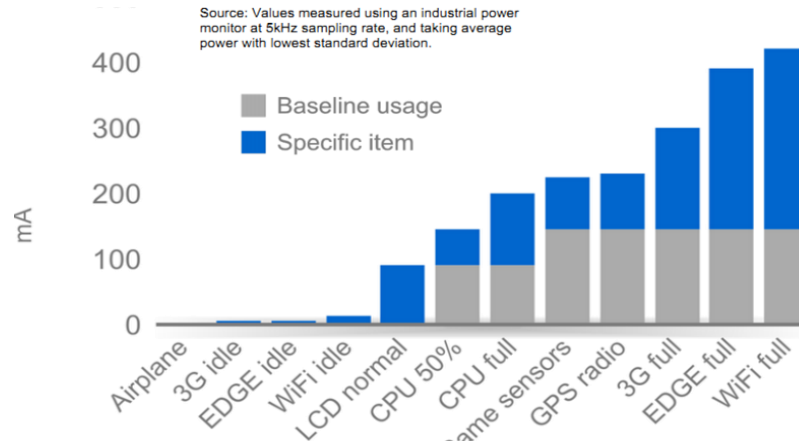
- Nagyon rövid hatótávú kommunikáció
- Még kevés készülékben található meg

## ◎ USB

- Vezetékes adatátvitel

# Mobilinternet vs. Wi-Fi

## ◎ Fogyasztás →



## ◎ Frekvencia

- Wi-Fi: 2.4 / 5 Ghz
- Mobilinternet példák:  
UMTS/HSDPA/HSUPA ↔ 850/900/1900/2100 MHz  
UMTS ↔ 2100/1900/850 combo

## ◎ Sebesség tekintetében

(elméleti maximális)

Wi-Fi (IEEE szabvány szerint)	
802.11a	54 Mbit/s
802.11b	11 Mbit/s
802.11g	54 Mbit/s
802.11n	600 Mbit/s

Mobilinternet (átvitel technológia szerint)		2G
HSCSD	9,6 – 57,6 Kbit/s	2.5G
GPRS	< 171,2 Kbit/s	
EDGE	< 473 Kbit/s	
UMTS	1,8 – 7,2 Mbit/s	3G
HSDPA	1,8 - ... - 14,4 - 28,8 Mbit/s	3.5G
LTE	< 326 Mbit/s	4G

\* csak a fontosabbak lettek felsorolva, ennél jóval több létezik, lásd [Network Type konstansok itt!](#)

# Android támogatás

- ◎ OSI modell miatt nem szükséges különbséget tenni
  - Adatátvitel módja detektálható, lásd: [Google I/O 2009](#)
- ◎ Internet használathoz jogot kell kérnünk:
  - android.permission.INTERNET
- ◎ Wi-Fi
  - Kezelése a [Wi-Fi API-kon](#) keresztül
  - Hivatalosan ad-hoc kapcsolódás nem engedélyezett
  - Wi-Fi Tethering Android 2.2 óta
  - [Wi-Fi Direct](#) támogatás [Android 4.0](#) óta
- ◎ IP szabvány szállítási rétegében:
  - **TCP** : kapcsolat orientált, csomag megérkezést megerősíti, és a csomagok sorrendjéről is gondoskodik
  - **UDP** : gyors, apró üzenetváltások, de nem sorrendtartó, valamint a csomagok megérkezése nem garantált

# Fájl letöltése URL alapján

- ◎ A URLConnection egy könnyebb súlyú megoldás 😊
  - Fájl letöltéséhez jobb választás szemben egy HttpClient-el!
  - Oka: [lásd itt!](#)
- ◎ Adott URL tartalmának letöltése egy fájlba:

```
URL url = new URL("http://nik.uni-obuda.hu/malk/");
File malkFile = new File("/sdcard/malk.html");
URLConnection ucon = url.openConnection();
InputStream is = ucon.getInputStream();
FileOutputStream fos = new FileOutputStream(malkFile);
byte[] buffer = new byte[1024];
int len = 0;
while((len = is.read(buffer)) != -1)
    fos.write(buffer, 0, len);
fos.close();
is.close();
```

Ne felejtünk el jogosultságot kérni :  
**android.permission.INTERNET**  
**android.permission.WRITE\_EXTERNAL\_STORAGE**

# Fájl letöltése HttpClient-el

## ◎ HTTP

- HTTP protokoll a TCP/IP szállítási réteg felett (80-as port)
- Ismertebb metódusok: HEAD, GET, POST, DELETE, ...
- Válasz státuszkódok: 1xx, 2xx, 3xx, 4xx, 5xx (pl.: 200 OK)

## ◎ Java körben jól ismert Apache HTTP kliens használata

## ◎ HTTP GET kérés indítása

```
HttpClient client = new DefaultHttpClient();
HttpGet get = new HttpGet(url.getText().toString());
    get.setHeader("User-Agent", "Android-robot-1.0");
HttpResponse response = client.execute(get);
if(response.getStatusLine().getStatusCode() == 200) {
    InputStream is = response.getEntity().getContent();
    // InputStream feldolgozása...
}
```

Ne felejtünk el jogosultságot kérni az internethez:  
**android.permission.INTERNET**

# Adatküldés – HTTP POST

⦿ Mint GET esetén, csak több adatot lehet küldeni

⦿ HttpEntity-re néhány példa

- `UrlEncodedFormEntity` → Form adatok
- `StringEntity` → Egyszerű szöveg
- `InputStreamEntity` → Pl.: `FileInputStream` 😊

JSON string 😊

⦿ HTTP POST küldése adatokkal

```
HttpPost post = new HttpPost("http://pelda.hu/belepo");  
List<BasicNameValuePair> pairs = new ArrayList<BasicNameValuePair>();  
pairs.add(new BasicNameValuePair("felhasznalo",  
    username.getText().toString()));  
pairs.add(new BasicNameValuePair("jelszo",  
    password.getText().toString()));  
post.setEntity(new UrlEncodedFormEntity(pairs));  
HttpResponse response = client.execute(post);  
int status = response.getStatusLine().getStatusCode();  
// Válasz feldolgozása: státusz kód, inputstream, ...
```

`android.permission.INTERNET`

# UDP adatátvitel

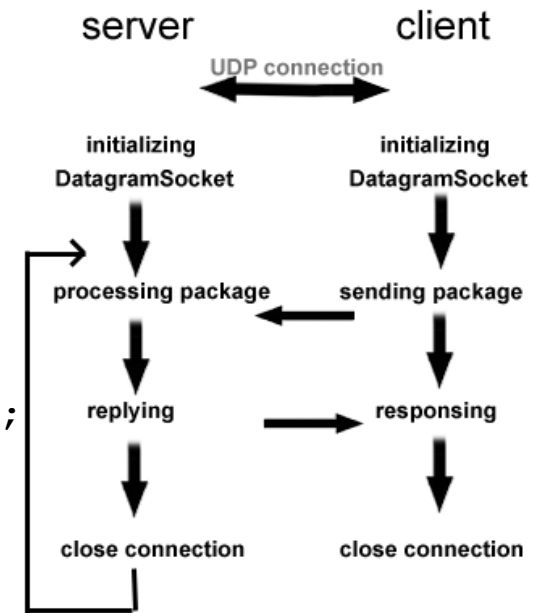
⊙ Jogosultság: `android.permission.INTERNET`

## ⊙ Szerver oldal

```
int serverPort = 50004;
InetAddress ip =
    InetAddress.getByName("192.168.1.1");
byte[] buf = new byte[1024 * 65];
DatagramPacket receivePacket =
    new DatagramPacket(buf, buf.length);
DatagramSocket socket = new
    DatagramSocket(serverPort);
socket.receive(receivePacket);      thread
receivePacket.getData();
```

## ⊙ Kliens oldal

```
String msg = "Hello UDP Package";
byte[] msgByte = msg.getBytes();
DatagramSocket socket = new DatagramSocket();
InetAddress serverIP = InetAddress.getByName("192.168.1.1");
socket.connect(serverIP, 50004);
socket.send(new DatagramPacket(msgByte, msgByte.length));
```



[Kép forrása](#)

Android 1.5 alatt még bug-os volt.



# TCP adatátvitelt

## ◎ TCP client:

```
Socket socket = new Socket();
socket.connect(new InetSocketAddress(/* Cím */,
                                   /* port */));

byte[] buffer = new byte[1024];
InputStream in = socket.getInputStream();
int len = 0;
while((len = in.read(buffer)) != -1) {           thread
    /* valamit csinálunk */
}
in.close();
```

## ◎ TCP server:

```
ServerSocket server = new ServerSocket(/* port */);
while(true) {
    Socket client = server.accept();
    /* Klient szokás új szálon kezelni a      thread
       továbbiakban */
}
```

