Mobilne komunikacije

Mobilne aplikacije

Agenda

Radio talasi

- Radio talasi su oscilacije elektromagnetnog polja u vremenu i prostoru.
- Slabljenje radio talasa zavisi od njegove frekvencije i karakteristika medija kroz koji se prostire.

Radio uređaji

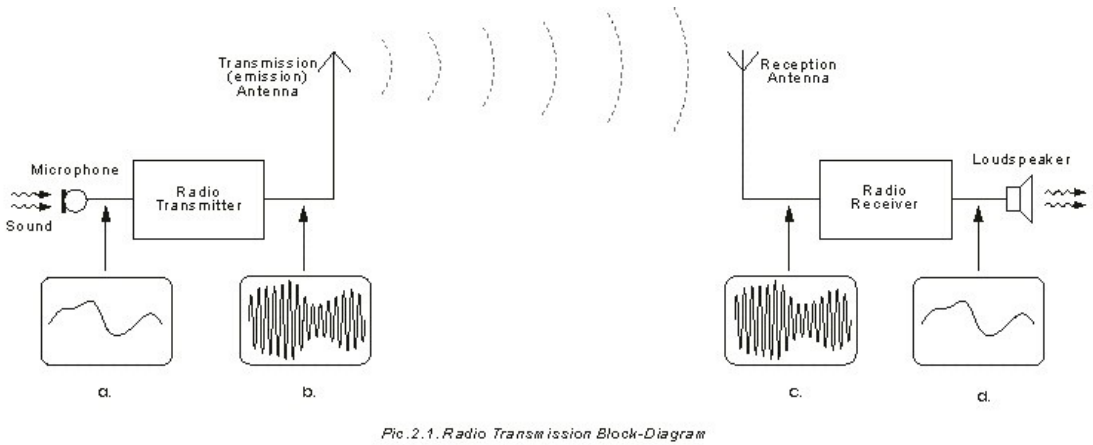


Figure 1: Prijemnik i predajnik.

- Pošiljalac
- Predajnik
- Antena
- Medij
- Antena
- Prijemnik
- Primalac

Analogni signal

Figure 2: Analogni signal.

- Analogni signal je kontinualni signal koji informacije prenosi kao promena amplitude, frekvencije i faze.

Digitalni signal

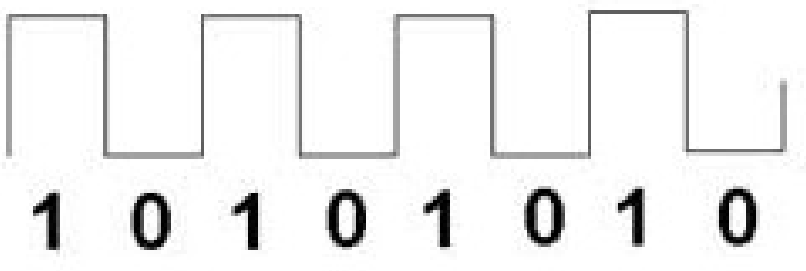


Figure 3: Digitalni signal.

- Digitalni signal je diskretan signal koji informacije prenosi kao niz znakova (logičkih nula i jedinica).

Modulacija

Modulacija je proces kojim se signal prilagođava karakteristikama prenosnog medija (transponuje se u područje frekvencija pogodnih za prenos radio talasima).

- Analogni postupci koriste se za molulisanje analognog signala
  - Amplitudna modulacija
  - Frekventna modulacija
  - Fazna modulacija
- Digitalni postupci koriste se za modulisanje digitalnog signala
  - Pulsna kodna modulacija
  - Delta modulacija

Agenda

Ćelijski sistem

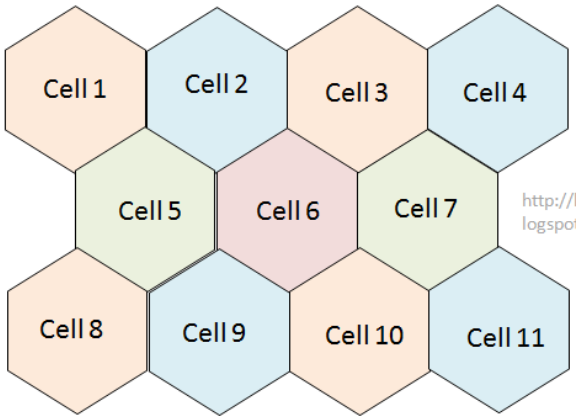


Figure 4: Ćelijski sistem.

- Određena teritorija je podeljena na ćelije.
- Svakoj ćeliji je dodeljen skup frekvencija koje su izabrane tako da minimizuju interferenciju sa susednim ćelijama.
- Skup frekvencija može se koristiti i u drugim ćelijama, sve dok te ćelije nisu susedne.
- Kada mobilni telefon pređe iz jedne ćelije u drugu ćeliju dok je poziv u toku, mreža će izdati naredbu mobilnom telefonu da promeni kanal (frekvenciju) i u isto vreme preusmeriti poziv na novi kanal.

Usluge GSM-a

- Telefonija (prenos govora)
- Short Message Service (SMS)
- Multimedia Messaging Service (MMS)
- Prenos podataka

SIM

- Subscriber Identity Module (SIM) je pametna kartica koja omogućava mobilnom telefonu da promeni pretplatnika i pretplatniku da promeni mobilni telefon.
- Implementira Java Card specifikaciju da bi omogućio interoperabilnost aplikacija.
- Pruža sigurno skladištenje:
  - Integrated Circuit Card Identifier (ICCID) - identifikator kartice
  - International Mobile Subscriber Identity (IMSI) - identifikator pretplatnika
  - i kriptografskog ključa (Ki) koji se koristi za autentifikaciju pretplatnika.

SIM

- Takođe skladišti:
  - Personal Identification Number (PIN) - lozinku za uobičajenu upotrebu
  - Personal Unblocking Code (PUK) - lozinku za otključavanje PIN-a
  - Service Provider Name (SPN)
  - Local Area Identity (LAI)
  - broj SMSC-a
  - broj za hitne slučajeve
  - spisak uluga kojima pretplatnik može da pristupi
  - itd.
- Nudi i dodatne funkcije kao što je skladištenje telefonskog imenika i tekstualnih poruka.

Agenda

Telefonija

- Osnovna usluga mobilne mreže je mobilna telefonija (prenos govora).
- Međutim, iz bezbednosnih razloga nije moguće napraviti "in call" aktivnost.
- Android API omogućava:
  - korišćenje podrazumevane "in call" aktivnosti za obavljanje telefonskih poziva
  - pristup podacima o telefonu (tip, identifikator, verzija softvera, telefonski broj)
  - pristup podacima o SIM kartici (stanje, država i ime operatora, serijski broj)
  - pristup podacima o mreži (država, identifikator operatora, ime mreže, tip mreže)
  - pristup podacima o prenosu podataka (stanje i trenutna aktivnost)
  - reagovanje na promenu stanja mreže, poziva, lokacija ćelije, snage signala, aktivnosti i stanja prenosa podataka, itd.

AndroidManifest.java

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application ... >
    <uses-permission android:name="android.permission.CALL_PHONE" />
  </application>
</manifest>
```

ExampleActivity.java

```
public void dialNumber() {
  Intent intent =
            new Intent(Intent.ACTION_DIAL, Uri.parse("tel:1234567"));
  startActivity(intent);
}

// ACTION_DIAL opens the dialer app with no permissions required
// ACTION_CALL initiates phone call and requires permission requests
    statically (in manifest file) and dynamically (at run-time)
    because protection level for this permission is 'dangerous'
```

Agenda

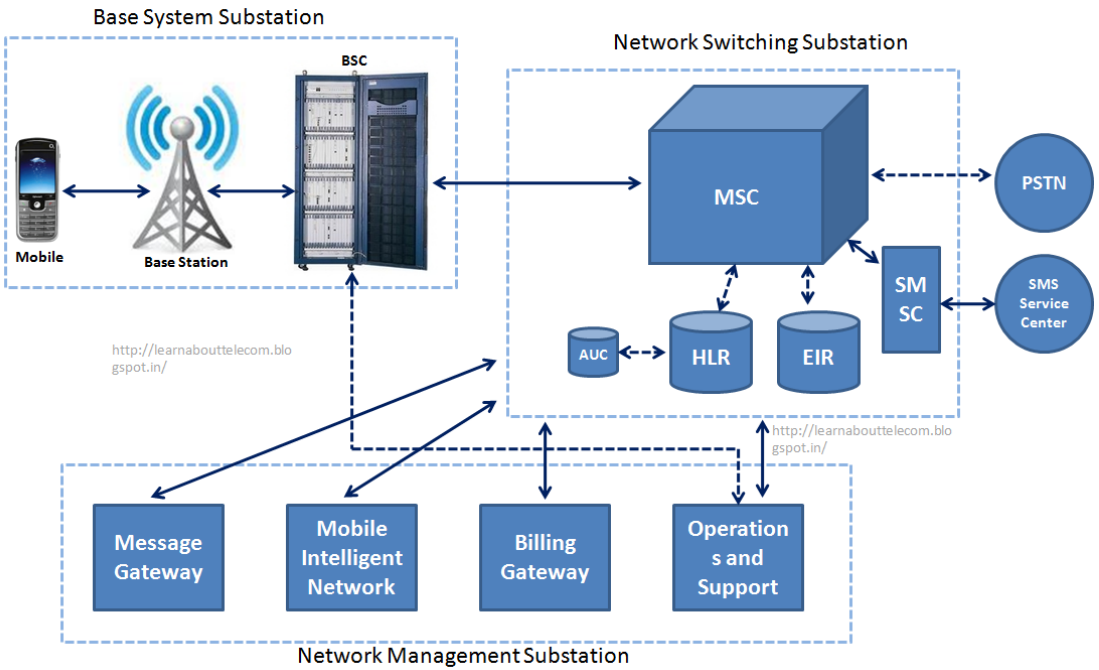
SMS

- Short Message Service (SMS) je tehnologija koja omogućava slanje i primanje kratkih poruka između mobilnih telefona.
- Podržavaju je (skoro) svi mobilni telefoni.
- Jedna SMS poruka može da sadrži najviše 140 bajtova (160 znakova ukoliko se koristi 7-bitno kodiranje, 70 znakova ukoliko se koristi 16-bitno kodiranje).
- Pored teksta, SMS poruke mogu da prenose i binarne podatke.

PDU mode SMS

- Konkatenirane SMS poruke ili PDU (protocol data unit) mode SMS poruke mogu da sadrže više od 140 bajtova.
- Mobilni telefon pošiljaoca deli dugačku poruku u manje delove i šalje svaki deo kao pojedinačnu SMS poruku.
- Mobilni telefon primaoca spaja pojedinačne SMS poruke u dugačku poruku.

Arhitektura SMS sistema



- Short Message Service Center (SMSC)
- SMS Gateway

Figure 5: Arhitektura SMS sistema.

SMSC

- Short Message Service Center (SMSC) upravlja SMS operacijama celularne mreže (njegova glavna funkcija je rutiranje SMS poruka).
- Kada mobilni telefon pošalje SMS poruku, ona stiže u SMS centar. SMS centar prosleđuje SMS poruku primaocu.
- Ako je primalac nedostupan (mobilni telefon je isključen ili nema domet), SMS centar skladišti poruku i prosleđuje je primaocu kada postane dostupan.

SMS Gateway

- Pre nego što stigne do odredišta, SMS poruka može da prođe kroz SMS gateway i druge SMS centre.
- SMS Gateway omogućava različitim operaterima mobilne telefonije da povežu SMS centre i razmenjuju SMS poruke.

MMS

- Multimedia Messaging Service (MMS) je proširenje SMS-a.
- MMS omogućava formatiranje teksta i slanje i primanje multimedijalnih poruka (poruka koje sadrže fotografije, audio i video).

SMS i Android

Postoje dva načina slanja SMS poruka:

- putem podrazumevane SMS aplikacije ili
- putem SMSManager servisa.

ExampleActivity.java (slanje kroz podrazumevanu aplikaciju)

```
public void sendSMS() {
  Intent smsIntent = new Intent(Intent.ACTION_SENDTO, Uri.parse("sms:+381641234567"));
  smsIntent.putExtra("sms_body", "Please call me as soon as possible");
  startActivity(smsIntent);
}

```

ExampleActivity.java (slanje kroz podrazumevanu aplikaciju)

```
public void sendMMS() {
  // Get the URI of a piece of media to attach.
  Uri attachedUri = Uri.parse("content://media/external/images/media/1");
  // Create a new MMS intent
  Intent mmsIntent = new Intent(Intent.ACTION_SEND);
  mmsIntent.putExtra("sms_body", "Please see the attached image");
  mmsIntent.putExtra("address", "+381641234567");
  mmsIntent.putExtra(Intent.EXTRA_STREAM, attachedUri);
  mmsIntent.setType("image/png");
  startActivity(mmsIntent);
}

```

AndroidManifest.xml (slanje putem SmsManager servisa)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application ... >
    <uses-permission android:name="android.permission.SEND_SMS"/>
  </application>
</manifest>

<!-- This permission should be requested in run-time as well because
     its level is 'dangerous' -->
```

ExampleActivity.java (slanje putem SmsManager servisa)

- Do API level 31:

```
public sendSMS() {
  SmsManager smsManager = SmsManager.getDefault();
  String to = "+381641234567";
  String message = "Android supports programmatic SMS messaging!";
  smsManager.sendTextMessage(to, null, message, null, null);
}
```

- Od API level 31:

```
public sendSMS() {
  SmsManager smsManager = getSystemService(SmsManager.class);
  String to = "+381641234567";
  String message = "Android supports programmatic SMS messaging!";
  smsManager.sendTextMessage(to, null, message, null, null);
}
```

SMS

Parameters	Meaning
destinationAddress	The address to send the message to
serviceCenterAddress	The service center address (or null to use the default SMSC).
text	The body of the message to send.
sentIntent	If not null this PendingIntent is broadcast when the message is successfully sent, or failed.
deliveryIntent	If not null this PendingIntent is broadcast when the message is delivered to the recipient.

Table 1: Parametri `sendTextMessage` metode.

MMS/Data/Multipart Messages

- `sendDataMessage` (sends a data based SMS to a specific application port)
- `sendMultimediaMessage` (sends an MMS message)
- `sendMultipartTextMessage` (sends a multi-part text based SMS)

Agenda

Networking

- Umrežavanje mobilnih uređaja je slično kao i kod računara i računarske opreme.
- Bez obzira na tip konekcije za razmenu podataka, na raspolaganju su nam protokoli sa viših OSI nivoa.
- Nekim lokacijama nije potrebno pristupati često (slika, audio zapis, video klip i sl.) dok se neke lokacije mogu intenzivno koristiti (npr. mail server, web servisi).

Primer preuzimanja podataka sa neke web adrese

```
InputStream is = null;
try {
  URL url = new URL("https://some.address.com/some/path");
  URLConnection conn = url.openConnection();
  is = conn.getInputStream();
  ByteArrayOutputStream os = new ByteArrayOutputStream();
  byte[] buff = new byte[1024];
  int read;
  while ((read = is.read(buff)) > 0) {
    os.write(buff,0,read);
  }
  return os.toString();
} catch (Exception e) {
  e.printStackTrace();
} finally {
  is.close();
}

```

Web servisi

- Za udaljeno izvršavanje operacija nad podacima se obično koriste REST servisi.
- Podaci se najčešće prenose u JSON formatu što olakšava njihovu serijalizaciju/deserijalizaciju.
- Iako je ovaj način komunikacije relativno jednostavan za implementiranje, u praksi se koriste gotove biblioteke koje dodatno olakšavaju korišćenje REST servisa.

Retrofit

- Retrofit je HTTP klijent za Android i Javu.
- Omogućava rad sa sinhronim i asinhronim pozivima, pri čemu je na Android platformi asinhrona varijanta praktičnija jer se na taj način ne blokira glavna UI nit.
- Retrofit biblioteka ne vrši serijalizaciju i deserijalizaciju JSON objekata, pa je u te svrhe potrebno uključiti i druge biblioteke u projekat.

build.gradle

```
dependencies {
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    ...
}

/*
  converter-gson enables retrofit to use the gson library.
  converter-gson depends on gson library that performs
                 (de)serialization of JSON objects.
*/

```

Model podataka

- Za konverziju između JSON i Java objekata potrebno je u projektu definisati model podataka.
- Ovaj model predstavljaju POJO klase sa odgovarajućim anotacijama koje povezuju atribute ovih klasa sa odgovarajućim poljima u JSON objektima.
- Na adresi `https://www.jsonschema2pojo.org/` se nalazi online alat za automatsko generisanje anotiranih klasa na osnovu uzorka JSON podataka. Potrebno je izabrati:
  - Source Type: JSON
  - Annotation Style: Gson

Book.java

```java
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

public class Book {

    @SerializedName("id")
    @Expose
    private Integer id;

    @SerializedName("title")
    @Expose
    private String title;

    @SerializedName("author")
    @Expose
    private String author;

}
```

Endpoint

- Nakon što smo povezali Java model sa JSON modelom potrebno je definisati endpoint pomoću metoda u Java interfejsu.
- To se takođe postiže pomoću anotacija uz metode i njihove parametre tako što označavaju na koji način učestvuju u HTTP request-u i response-u.
- Pri tome se putanje do endpointa navode u relativnom zapisu dok se osnovni deo URL-a setuje pri inicijalizaciji Retrofit instance.

BooksEndpoint.java

```java
import java.util.List;
import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.GET;
import retrofit2.http.POST;
import retrofit2.http.Path;

public interface BooksEndpoint {

    @GET("books")
    Call<List<Book>> getBooks();

    @GET("books/{id}")
    Call<Book> getBook(@Path("id") int id);

    @POST("books")
    Call<Book> createBook(@Body Book book);

}
// return type of methods is Call<T>
// where T is a type of an object the method should return.

```

Anotacije

Annotation	Description
@GET @POST @PUT @DELETE	Type of HTTP method
@Path	Named replacement in a URL path segment
@Query	Query parameter appended to the URL
@Body	Controls the request body
@Header	Specifies the header parameter
@Headers	Predefines the header parameters

Table 2: Objašnjenja anotacija endpoint-a.

Instanciranje Retrofit objekta

- Da bi smo koristili Retrofit potrebno je da najpre instanciramo objekat klase `Retrofit` putem `Builder`-a.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

- `Builder`-u smo setovali:
  - osnovni URL (na koji će se dodavati relativne putanje definisane u interfejsu endpoint-a) i poželjno je definisati ga kao konstantu

```
public static final String BASE_URL = "http://.../";
```

  - konverter koji će se koristiti za (de)serijalizaciju JSON objekata (u ovom slučaju to je konverter iz Gson biblioteke).

Kreiranje poziva

- Na osnovu interfejsa koji smo definisali, Retrofit omogućava kreiranje objekta koji reprezentuje servis i omogućava pozivanje metoda koje smo deklarisali.

```
BooksEndpoint be = retrofit.create(BooksEndpoint.class);
```

- Nad ovim objektom kreiramo poziv neke od metoda na sledeći način:

```
Call<List<Book>> call = be.getBooks();
```

- Objekat tipa `Call<T>` reprezentuje poziv ka servisu i dozvoljava sinhrono i asinhrono izvršavanje.
- Za sinhrono izvršavanje se koristi metoda `execute()`.
- Za asinhrono izvršavanje se koristi metoda `enqueue()`.

Sinhrono izvršavanje poziva

```
Response<List<Book>> response = call.execute();
List<Book> books = response.body();
```

Asinhrono izvršavanje poziva

```
call.enqueue(new Callback<List<Book>>() {
    @Override
    public void onResponse(Call<List<Book>> call, Response<List<Book>> response) {
        // success
        List<Book> books = response.body();
    }
    @Override
    public void onFailure(Call<List<Book>> call, Throwable t) {
        // failure
    }
});

```

Inicijalizacija Retrofit biblioteke i korišćenje asinhronog poziva

```
// base url is defined as a global constant
public static final String BASE_URL = "http://some.address.com/";

// retrofit instance is created using the Builder
// base url and (de)serializer are initially set
Retrofit retrofit = new Retrofit.Builder()
  .baseUrl(BASE_URL)
  .addConverterFactory(GsonConverterFactory.create())
  .build();

// the object representing service is created using retrofit and the interface
BooksEndpoint be = retrofit.create(BooksEndpoint.class);
// request is created and stored in object representing this call
Call<List<Book>> call = be.getBooks();

// the request is sent and response is processed asynchronously
call.enqueue(new Callback<List<Book>>() {
  @Override
  public void onResponse(Call<List<Book>> call, Response<List<Book>> response) {
    int status = response.code();
    List<Book> books = response.body();
    // ...
  }
  @Override
  public void onFailure(Call<List<Book>> call, Throwable t) {
    // the request is failed
  }
});
```

Asinhrono učitavanje slika

- Kada je potrebno da se neka grafička datoteka sa veba učita u pozadini korisno je upotrebiti Picasso biblioteku.
- Najpre je potrebno uključiti odgovarajuću zavisnost:

```
implementation 'com.squareup.picasso:picasso:2.5.2'
```

- Zatim se učitavanje slike u neki ImageView pogled može postići na sledeći način:

```
ImageView iv = findViewById(R.id.imageView);
Picasso.with(context).load("http://...").into(iv);
```