# Platform-Based Development: Network Communication

BS UNI studies, Spring 2019/2020

Dr Veljko Pejović
Veljko.Pejovic@fri.uni-lj.si

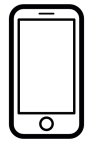
University *of Ljubljana*
Faculty *of Computer and Information Science*

# Course Administration

- Next week's lecture (Wed May 20th) starts at 17:00
- Lab 8 due Friday, May 15th, 7pm
- Mini App 3 due Sunday, May 24th, 23:59
- Final Exam
  - No specific guidelines from the University, yet
  - Exchange students – please inform us via email if you are not in Slovenia

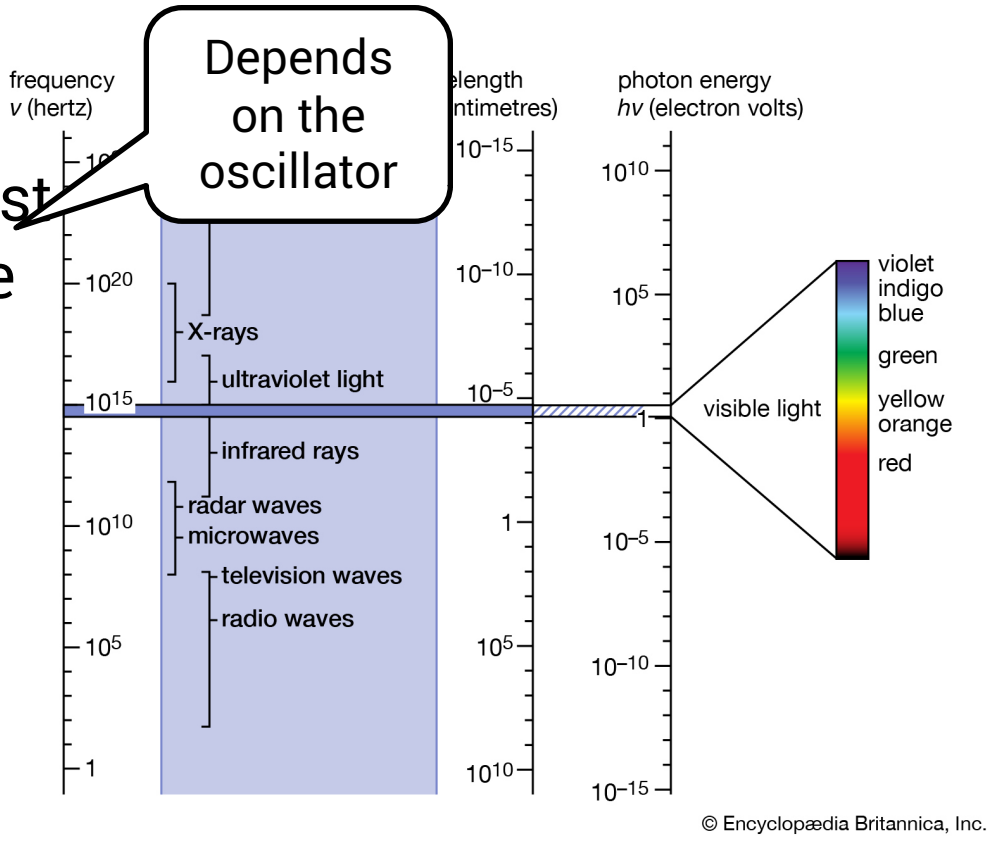
# Wireless Data Transmission

- Concept:
  - Information is encoded as the variability of the electromagnetic fields
  - Waves are propagated from a sender (Tx) to a receiver (Rx)
    - Transmitter circuit – includes an oscillator, DAC
    - Receiver circuit – includes an oscillator, ADC
    - Antennas – convert electric current to EM waves and vice versa



# Wireless Data Transmission

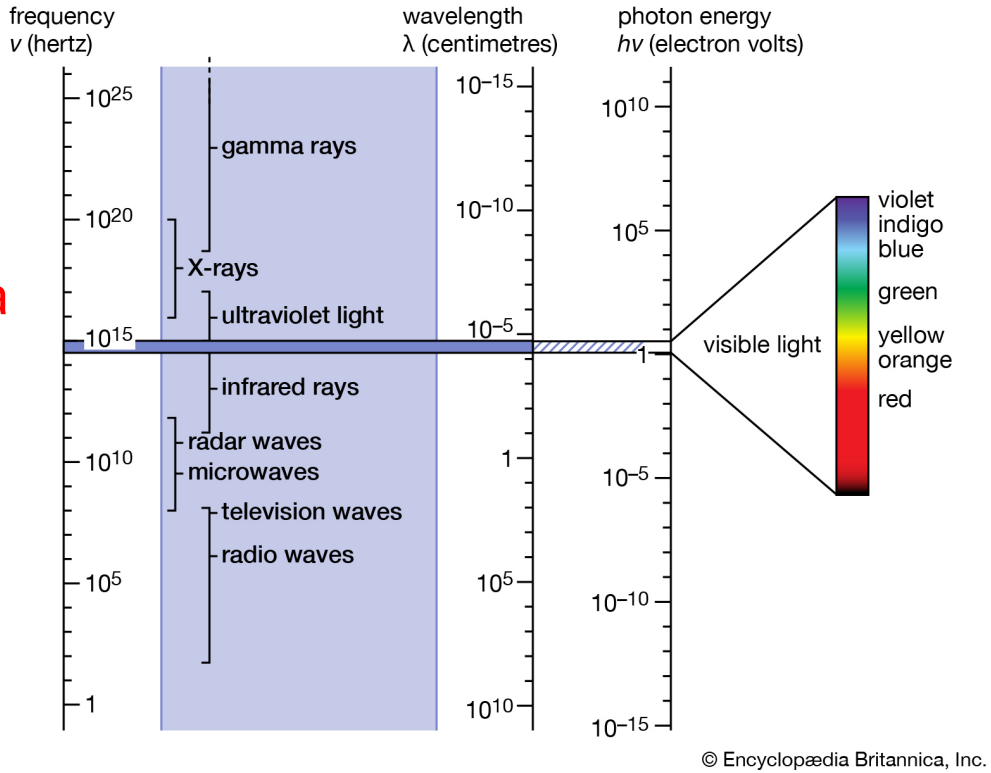
- Basic parameters:
  - Frequency ($f$) – how fast does the wave oscillate
  - Wavelength ($\lambda$) – distance between consecutive wave repetitions (inverse of frequency)
  - Bandwidth – the width of a range of frequencies used for the transmission




Depends on the ADC and DAC

# Wireless Data Transmission

- Physical properties:
  - Antenna length should be proportional to $\lambda$
    - higher $\lambda$, longer antenna
  - The irradiated power drops with the square of frequency
    - higher $f$, shorter range
  - More bandwidth – more information transmitted (higher throughput)



# Smartphone Wireless Interfaces

- Near Field Communication (NFC)
  - Very low power
    - Tags don't even need to be powered
  - Very short range (~10cm)
  - Low throughput (~400 kbps)
  - Applications: security tags, location-based services, payment systems

# Smartphone Wireless Interfaces

- Bluetooth/Bluetooth Low Energy (BLE)
  - Low power
    - ~10 mW for BLE
  - Short range (~10 m)
  - Low throughput (~1 Mbps)
  - Applications: connection with peripherals, wearables (smartwatch), medical equipment, vehicle systems

# Smartphone Wireless Interfaces

- WiFi
  - Medium power consumption
    - ~100 mW
  - Relatively short range (~100 m)
  - High throughput (~100s Mbps)
  - Applications: home entertainment, large downloads, system updates
  - WiFi is usually an unmetered network (more data transferred ≠ more cost)

# Smartphone Wireless Interfaces

- Cellular network
  - Medium-high power consumption
    - ~200 mW
  - Long range (~1000 m)
  - Varying throughput
    - ~40 kbps with 2G
    - ~1 Gbps with 5G
  - Applications: ubiquitous connectivity, real-time updates, Voice-over-IP (VoIP), smart metering (occasionally)

# Building Wireless Solutions

- Wireless interface selection impacts:
  - Capabilities of your app
    - How much data will you be able to pull/push
    - Coverage area
    - If peer-to-peer, maximum range between peers
  - Cost for the user
    - Monetary cost for using cellular services
  - Power consumption
    - Think about low power solutions or a mix of high power and low power communication when possible

# Android Networking Support

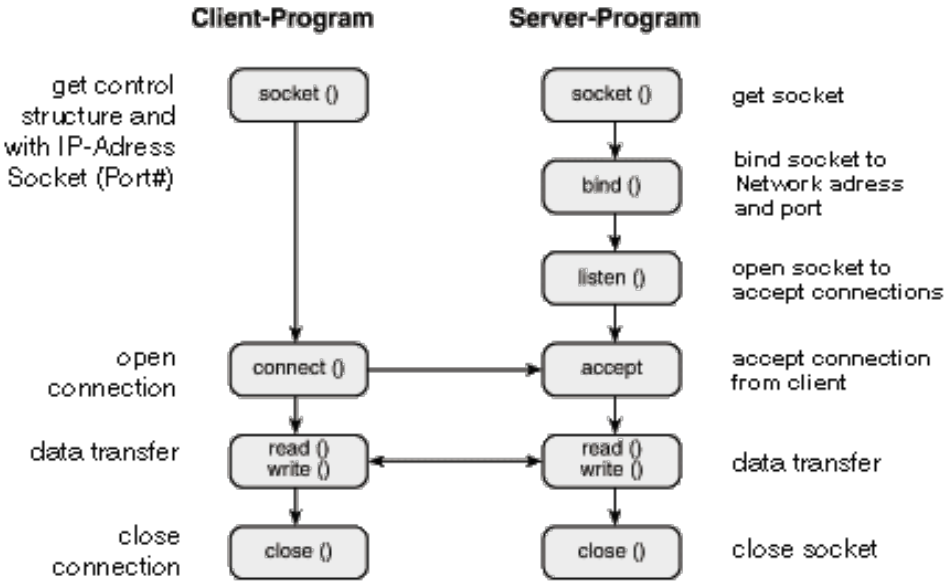
- Manage different physical interfaces (e.g scan for networks, associate with a network, get info about the network/link, etc.)
  - NfcManager
  - BluetoothManager
  - WifiManager
  - TelephonyManager
- ConnectivityManager
  - Monitors network connections, manages failovers, notifies when connectivity changes

# Networking Abstractions

- Sockets
  - Standard Java sockets:
    - Socket (TCP)
    - DatagramSocket (UDP)

Refresh your socket programming knowledge to understand Android networking.

However, socket programming on Android is rarely used – Web abstractions!




TCP sockets

src: https://www.wut.de/e-58www-16-apus-000.php

# Web Abstractions

- Http(s)URLConnection
  - (Dis)connect using HTTP(S)
  - Send HTTP requests, obtain responses
  - Connection pooling
    - Sockets can be reused
  - Response caching
  - Cookie management
  - Supports secure communication via Transport Layer Security (TLS)

# Http(s)URLConnection Example

# Web Abstractions

- OkHttp (a third-party library)
  - Advanced HTTP client
    - Includes pretty much all the HttpsURLConnection functionalities
  - Automatic network connection recovery
  - Retries
  - Data compression

Fun fact: the current version of HttpURLConnection is based on an earlier version of OkHttp!

# OkHttp Example

# Web Abstractions

- Retrofit (a third-party library)
  - REST Client for Android
    - Define a model
    - Define possible REST operations
    - Define converter
    - Define adapter
    - Define authentication mechanism
    - Build client!
  - Uses OkHttp under the hood

Remember REST: stateless, cacheable, client-server, layered architecture for Web services

# Web Abstractions

- Volley
  - REST Client for Android
  - Compared to Retrofit:
    - Does not treat REST API calls as simple java methods (more complex to write)
    - Retrofit has more response parsing options
    - Volley has in-built support for image loading
    - Volley has a good caching mechanism
    - Volley supports retries and backoffs
    - Retrofit supports multipart uploads

# Best Practices in Android Networking

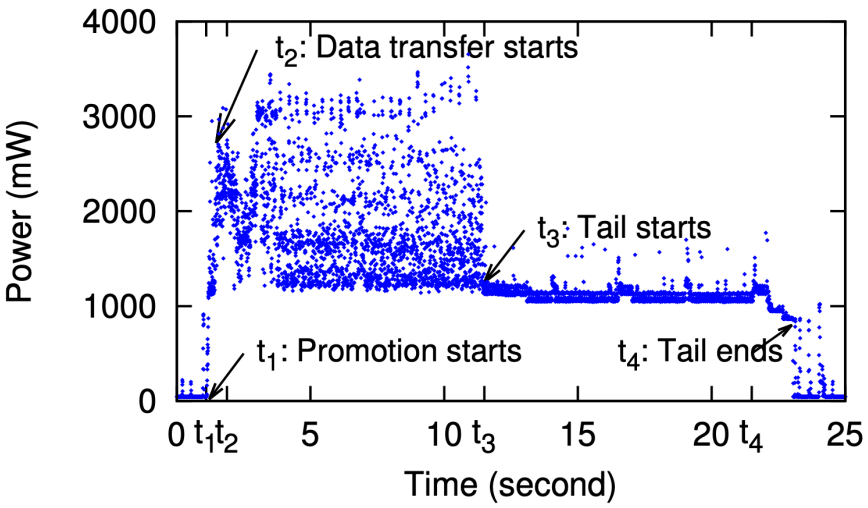
- Run network operations on a separate thread
  - You must do this in Android API 11+
- Reduce the amount of data transferred
  - Low resolution content when possible
  - Compress the data
  - Design a REST API that allows intelligent querying
    - Sends you what you need, not more
  - Cache static content
  - Cache dynamic content and check expires/last-modify
    - Caching directories: getExternalCacheDir, getCacheDir

# Best Practices in Android Networking

- Push, don't pull
  - Rather than checking the server for new data, get notified when new data is available
    - Firebase Cloud Messaging
- Reuse network connections
  - Rather than open/close frequently
  - But don't leave them hanging on forever
- Secure data and the connection
  - Use SSL and use it right
  - Minimize transfer of sensitive data

# Best Practices in Android Networking

- Energy-efficient networking
  - Wireless radio power consumption is not directly proportional to the amount of sent data because of the power tail (especially 3G and LTE)
    - Instead of short frequent transfers, bundle the data together and send less frequently
  - Use WorkManager
    - Leaves the phone in the doze mode




**"Power states of LTE"**

from

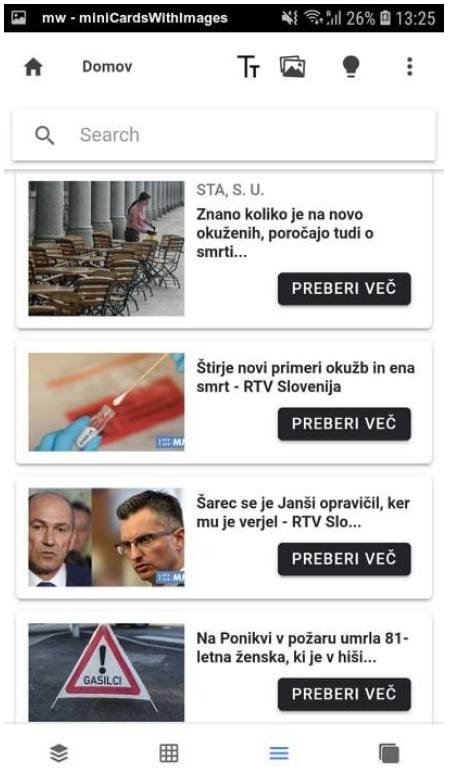
"A Close Examination of Performance and Power Characteristics of 4G LTE Networks" by Huang et al.

# Best Practices in Android Networking

- Adapt to the physical connection
  - Reduce data transfer when on a slower network
  - Prefetch more data when on WiFi
    - Predict user requests: monitor behavior, most popular content
  - Postpone non-critical downloads and uploads to periods when a user is on WiFi
  - Detecting connectivity
    - ConnectivityManager
    - TelephonyManager

# Best Practices Case Study – News Reader App

- Balancing UX, data and energy
  - Option 1: download headlines only after a news category has been selected
    - saves data
    - radio drains energy – always active
    - poor user experience
  - Option 2: download a set of headlines for most common categories, load articles shown on the page in a background thread
    - less frequent requests allow the radio to sleep
    - smoother user experience
    - data used for content that may never be seen



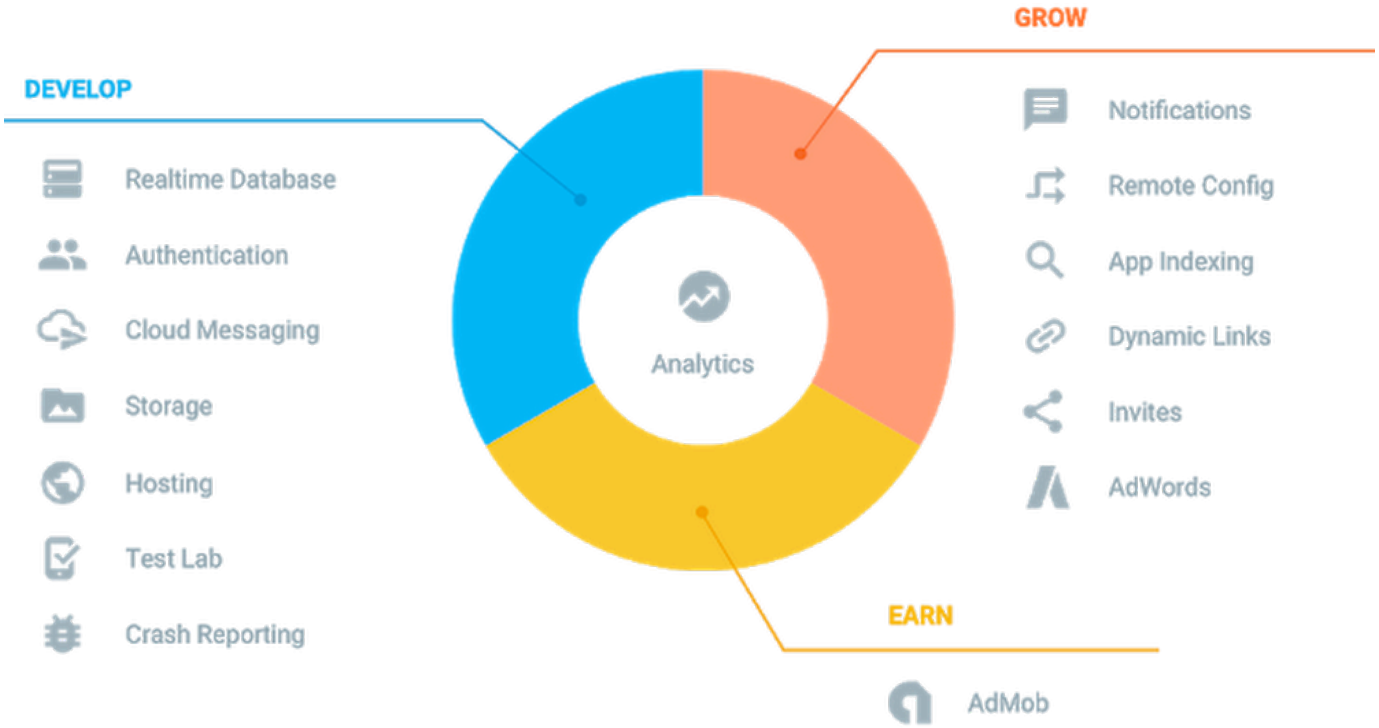

by Emir Hasanbegović

# Backend for Mobile Apps

- Android (or iOS for that matter) do not lock you into a particular backend technology
  - PHP, Node.js, Java Web apps, etc.
  - AWS, Google Cloud Platform, etc.
- Some solution easier to work with than others
  - Firebase
  - Parse Server (Back4App)

# Firebase

- Mobile and Web app development platform supported by Google



# Firebase

- Mobile and Web app development platform supported by Google
- Great for:
  - Authentication with Google ID (you have to use it)
  - Notifications (chat-like apps)
  - Crashlytics
  - Machine learning support (ML Kit)

```
implementation 'com.google.firebase:…'
```

# Parse Server

- Open source backend as a service (BaaS) platform initially developed by Facebook
  - Back4App is a Parse Server hosting platform
- Great for:
  - Building different REST APIs
  - Cron Jobs – schedule server jobs
  - User management (auto emails, social login)
  - Multiple SDKs
    - Including for Android

# Back4App

- NoSQL database
- REST API to access data
- Access via HTTP using different languages/platforms
- Different pricing tiers, but the free one is sufficient for prototyping
- Android library

```
implementation "com.github.parse-community.Parse-SDK-Android:parse:1.24.1"
```

# Back4App – Create Backend

- Go to back4app.com, log in, and create a new application
  - Manage via a dashboard
  - Add collections (tables)
  - Add custom code
  - Initiate communication (notifications)
- Get the following (and put in your Android app) in order to access the backend:
  - Application ID
  - Client Key

# Back4App Example