Challenges in determining device location

From Week 6,
COMP4216/COMP5216

THE UNIVERSITY OF SYDNEY

Challenges in determining device location

- Location adds **"Context"** to every action.
- The biggest challenge - User is moving.

- Various sources to determine the location.
  - GPS
  - Assisted-GPS
  - Cell towers
  - …
  - …

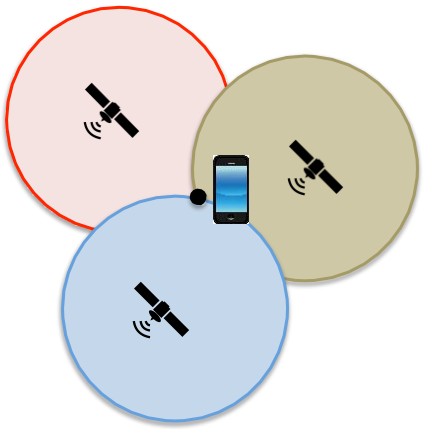
- **Each source comes with different accuracy, availability, resource requirement and efficiency**

- Dependent on the environmental factors.

The Global Positioning System (GPS) Location

- **Use the signals received from satellites for localisation.**
- Each location in the world is covered by at least four satellites.
- User device receives the signal and measure the time lag to estimate the distance to each satellite.

- No data connection is required.
- Longer time-to-fix.
  - Identifying the satellites
  - Synchronizing the clocks.



GPS Accuracy

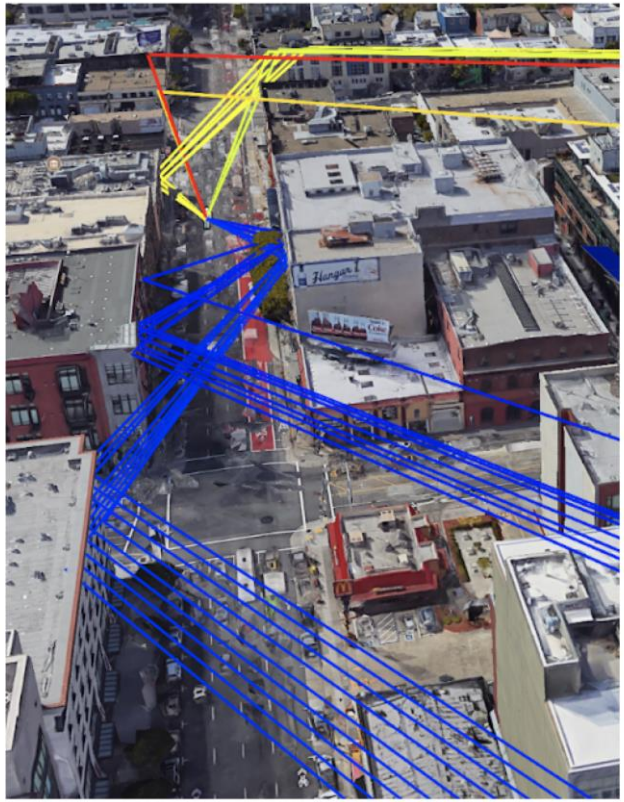
- Standard Positioning Service (SPS)
  - Available to all users
  - No restrictions or direct charge
  - high-quality receivers have accuracies of 3m and better horizontally
  - **In the level of 5-10m in worst case.**

- Precise Positioning Service (PPS)
  - Used by US and Allied military users
  - Use more satellites than public service

GPS Location

- Smartphones can connect to multiple constellations to improve the accuracy: GPS or GLONASS & use the combined result.
  - Many apps in Google Play Store to check the status of GPS signals

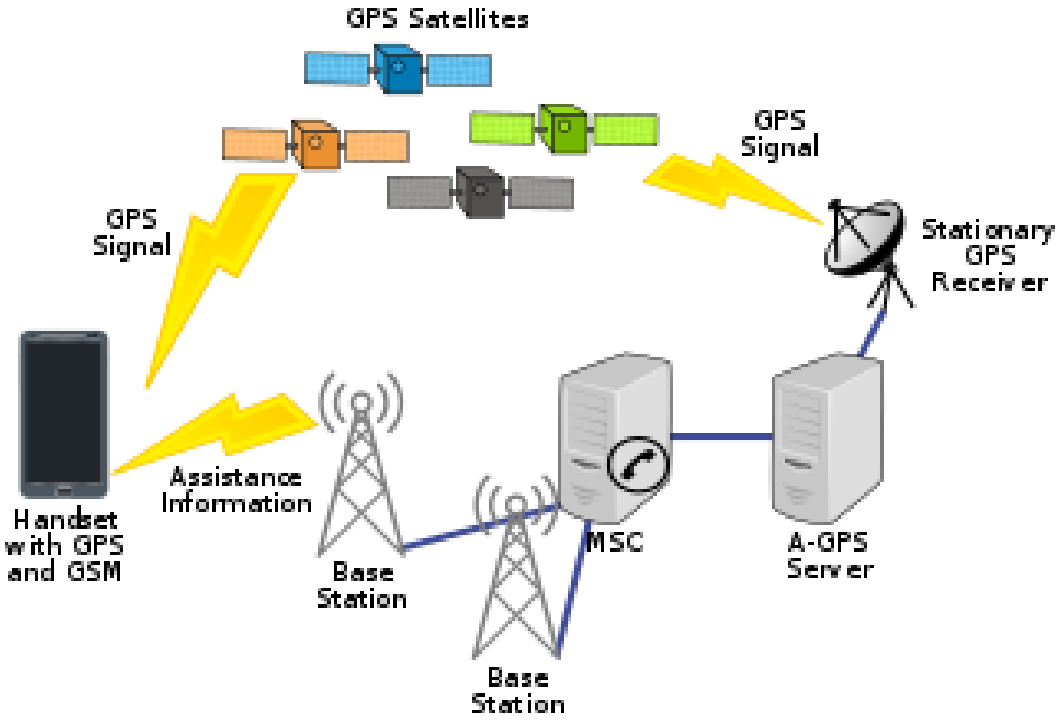
- However, GPS is not available everywhere, especially indoors.



GPS signals bouncing off facades in an urban environment.

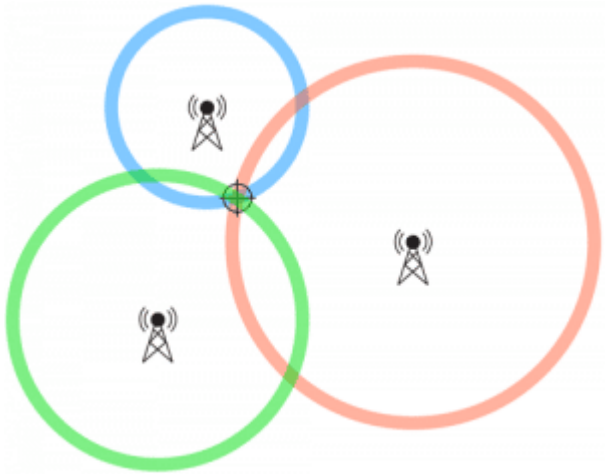
Assisted GPS (HW)

- Tries to address some of the problems in GPS.
- Faster set-up time by getting satellite information through data connection.
- Lower energy consumption



Network Location

- Use near by cell tower & WiFi access point information to query a geo tagged database.

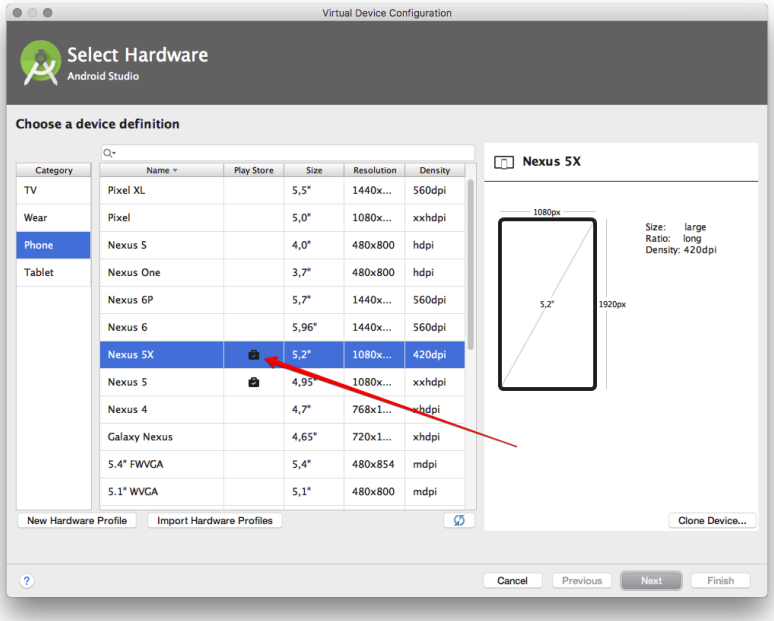


- Energy Efficient.
- Needs a user data connection.
- Who provide data to the geo DBs?
  - Everyone
  - It has been happening for sometime now https://www.zdnet.com/article/how-google-and-everyone-else-gets-wi-fi-location-data/

https://android-developers.googleblog.com/2020/12/improving-urban-gps-accuracy-for-your.html

How to get location in Android

- **FusedLocationProviderClient**
  - Current method [**Recommended by Google**].
    - https://developer.android.com/training/location
    - Google Play Services
    - Provides a much higher level view for the developer.
    - Automatically changing the appropriate Location Provider, e.g. GPS or WiFi
    - Better accuracy and power management.
    - Setting Permissions




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

Frequency & Latency

- `setinterval()` method
  - The interval at which *location is computed for your app.*
  - Larger the better for battery
- setFastestInterval() method
  - sets the **fastest** rate in milliseconds at which your app can handle location updates.
  - Unless your app benefits from receiving updates more quickly than the rate specified in setInterval(), you don't need to call this method.

```
mLocationRequest = new LocationRequest();
mLocationRequest.setInterval(10);
mLocationRequest.setFastestInterval(10);
mLocationRequest.setPriority
                    (LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
```

Accuracy

- Specify location accuracy using the `setPriority()` method
- **`PRIORITY_HIGH_ACCURACY`**
  - Most accurate
  - Use as many providers as necessary (GPS, WiFi, Cell-towers, etc.)
- **`PRIORITY_BALANCED_POWER_ACCURACY`**
  - Accurate location
  - Rarely uses GPS.
- **`PRIORITY_LOW_POWER`**
  - Coarse (city-level) accuracy
  - Mostly using on cell towers
- **`PRIORITY_NO_POWER`**
  - Passive location
  - Rely on location computed by other apps

Mobile Computing

COMP5216

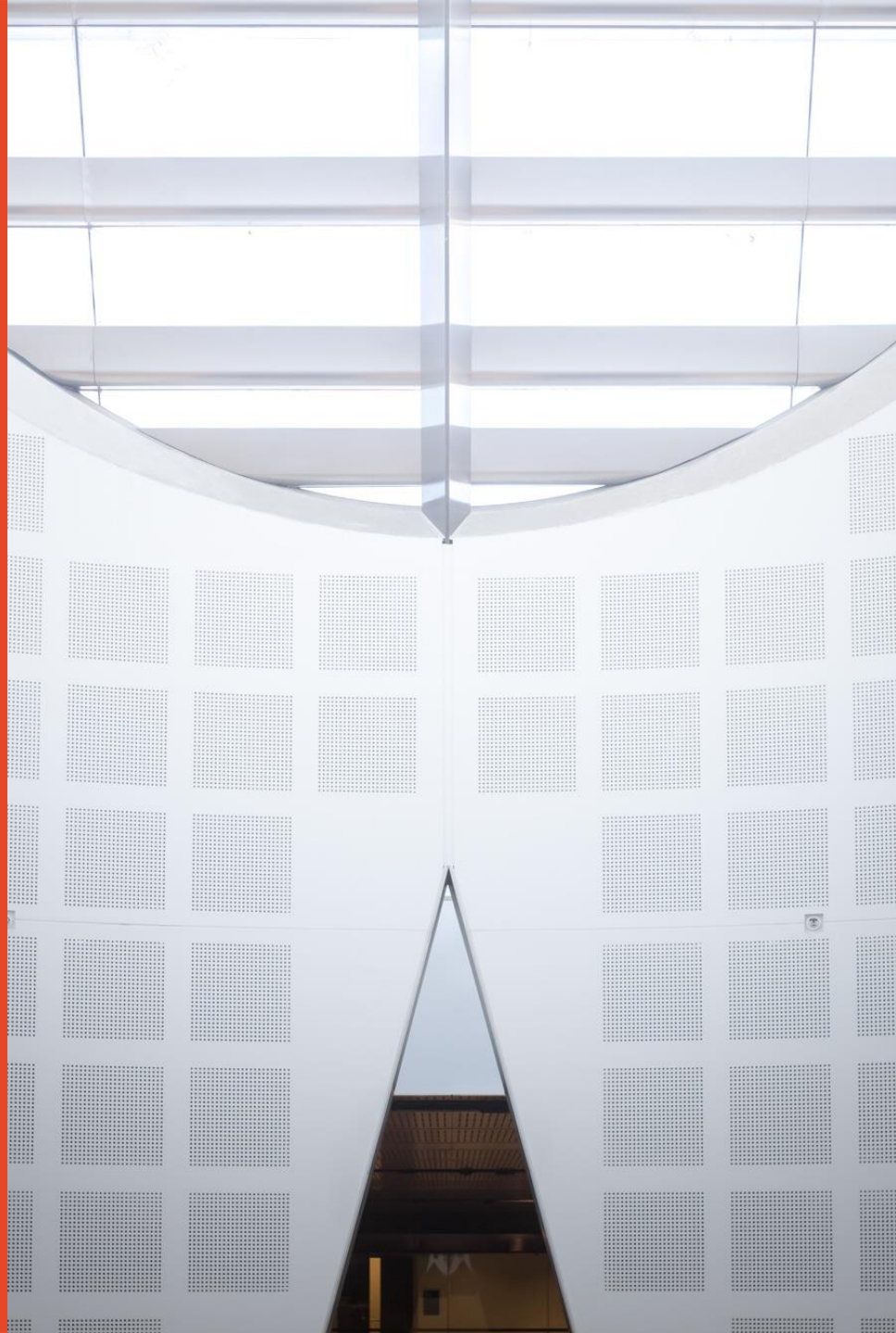
Week 07

Semester 2, 2023

Dr. Thilina Halloluwa

School of Computer Science

THE UNIVERSITY OF SYDNEY



Announcements

- For project and assignments
  - Please don't think you can reuse an app developed by past students. It will be captured by Turnitin similarity check !

- **Programming Help Desk**
  - It will be conducted by the tutor Kshitiz Bhargava
  - Book a timeslot kshitiz.bhargava@sydney.edu.au

Outline

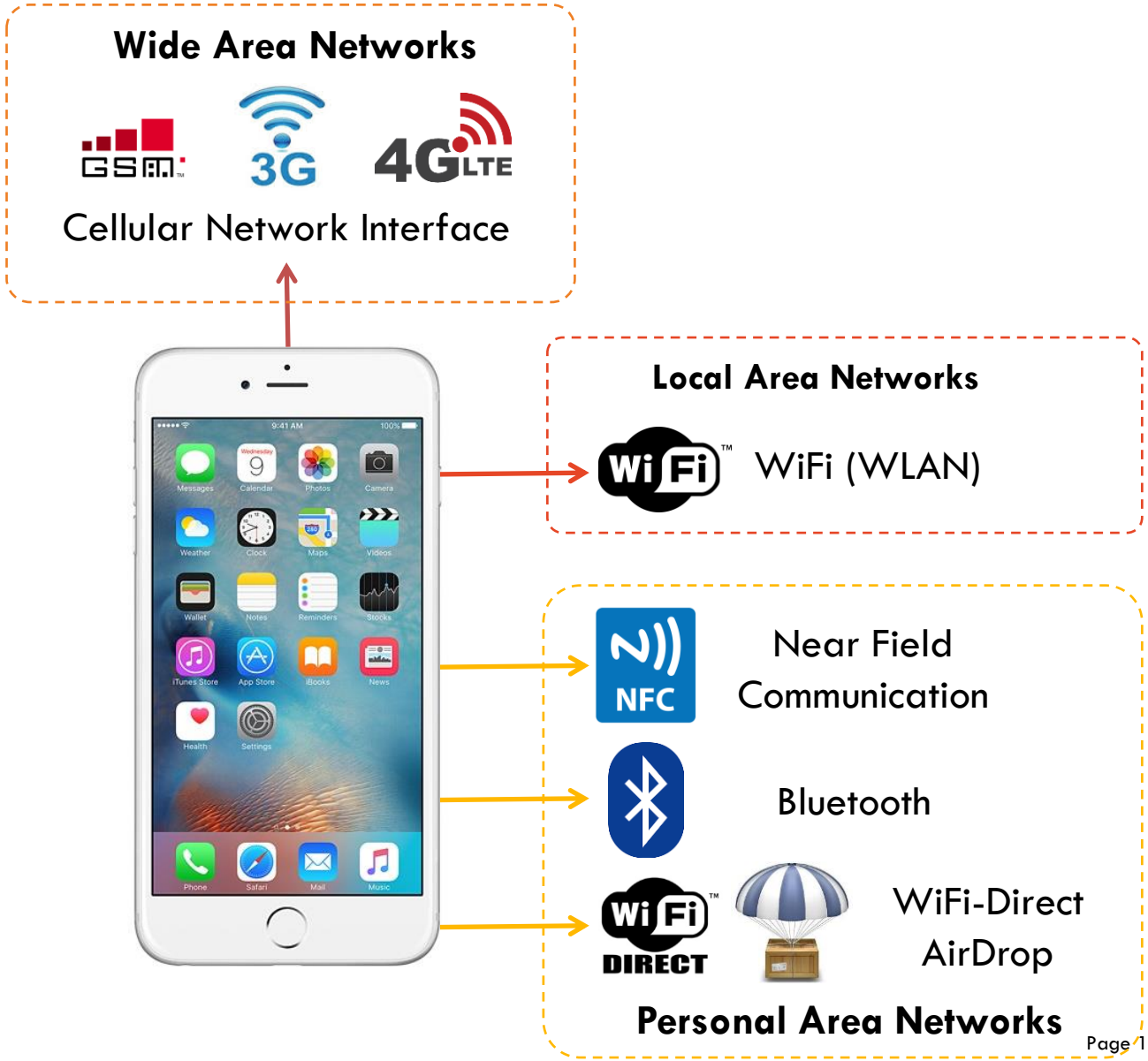
- Networking Challenges
- What can we do as developers to optimize networking cost ?
  - Selecting the right content
  - Offloading to cheap networks
  - Reduce data usage
  - Reuse data
- Tools for Network Debugging
  - Android Profiler
  - Network Inspector

Networking Challenge

Network Conditions

Intermittent Connectivity

Network Handover



Factors to consider

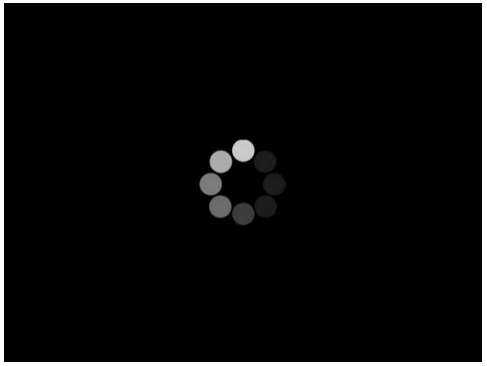
There are several important considerations to ensure a reliable, efficient, and secure user experience

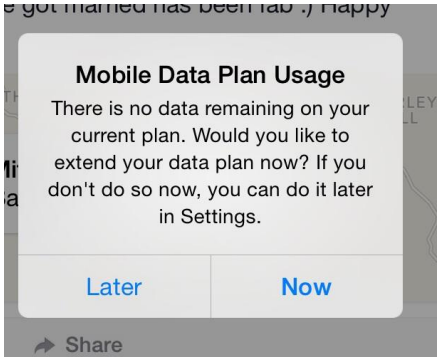
- **Type of Network (Range)**
  - Location of the end hosts
  - Mobility
- **Usage Cost**
  - For the network operator
  - For the consumer
- **Speed**
  - Real time or delay-tolerant
  - User expectations
- **Privacy and Security**
  - Public content or personal data
  - Location of the end hosts
- **Energy**
  - Smartphone energy consumption

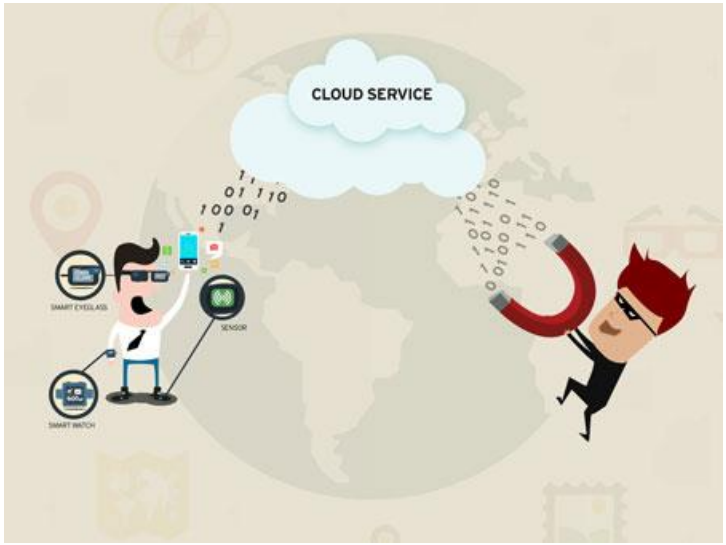
E.g. Sharing accelerometer data from smartwatch to smartphone

- What options we have?
- Pros and Cons (HW)

What to avoid ?







Best Practices for Mobile Networking

What can we (developers) do ?

Three type of regular communication:

1. **User-initiated**
2. **App-initiated**
3. **Sever-initiated**

Use of Data in mobile computing

Do you really need this data ? Yes

1. **Select** the right content for the device, the user and **adapt** dynamically

2. **Offloading** cellular (default) to alternative networks

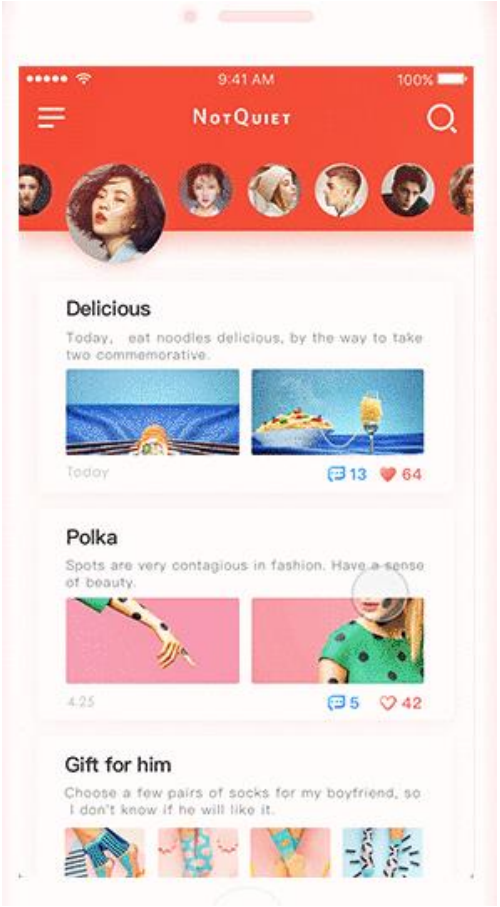
3. **Reduce** the amount of data

4. **Reuse** the data as much as possible

- Energy Management [TBD]
- Secure networking [TBD]

(1) Selecting the right content

- User expectation dependent on context;
  - Users' activity, e.g. running or sitting in the living room.
  - Time of the day.
- Request appropriate content
- Examples for bad practices?
  - Downloading same image for thumbnail and wallpaper
  - Hard-coding to download Full HD video all the time.
    - Every devices does not support Full HD viewing
    - Every network does not provide enough bandwidth to download Full HD



(1) Adapting to the conditions

- Check before downloading/uploading
  - User activity
  - Connected network type
  - Available network bandwidth
  - Location
  - Time of the day

- Slow connection → Download lower resolution media
- DASH Video Players (Dynamic Adaptive Streaming over HTTPS)
  - E.g. YouTube – Varying video quality according to the available network bandwidth

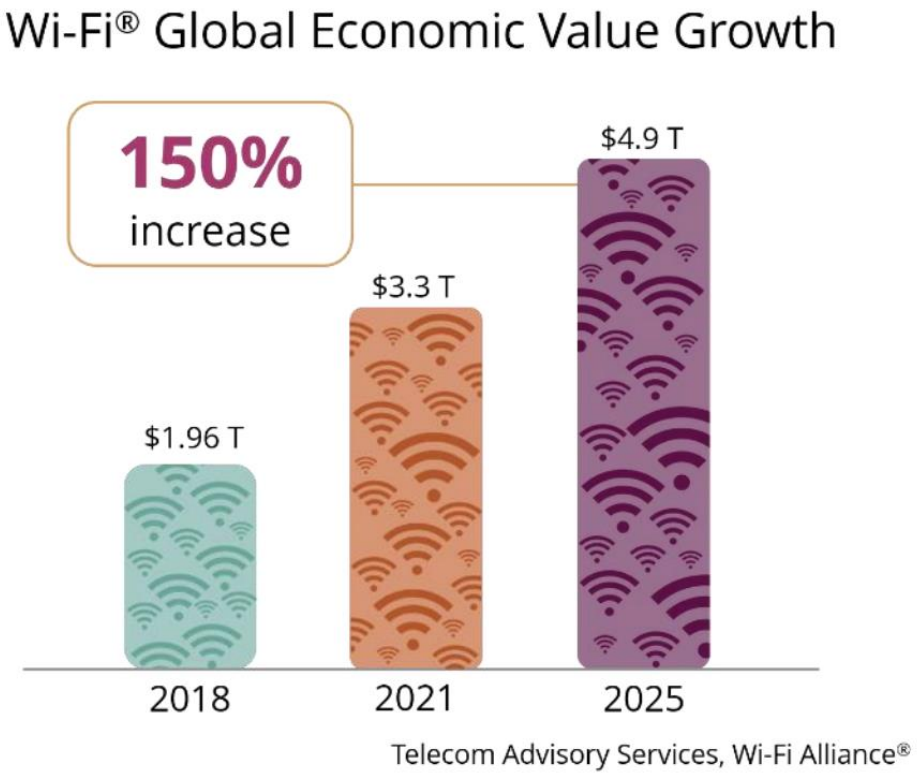

WHAT ABOUT
MPEG-DASH?

(2) Offloading

- Reactive offloading
- Predictive offloading

(2) Reactive Offloading

- Offload to another network when it is available.
- Cellular to WiFi, WiFi to Bluetooth, Bluetooth to USB



The results demonstrate an increase of almost $3 trillion in value, or 150 percent growth, from 2018 to 2025, underscoring that Wi-Fi technology is one of the dominant economic engines of the digital economy.

(2) Reactive Offloading

- All most all current smartphones automatically prioritise WiFi networks over Cellular (3G/4G) networks

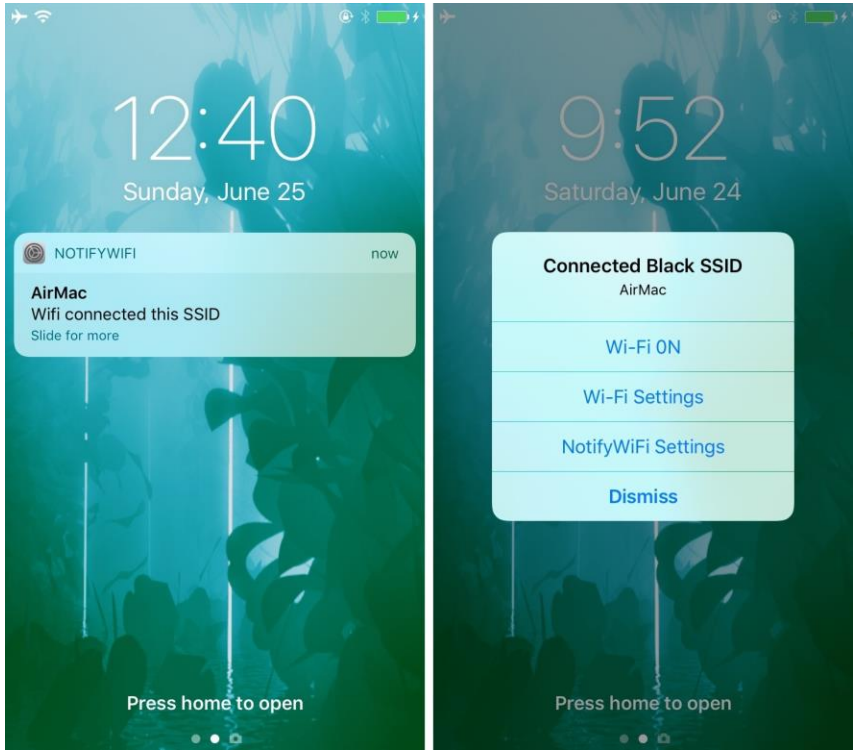
Benefits of WiFi offloading

- Faster connections (depends on the location)
- Lower cost
- Lesser battery drain
- Overall improved quality of user experience
- Reduces cellular network capacity issues

(2) Reactive Offloading

- To automatically connect to WiFi, the available WiFi network has to be **in the previously connected list of networks.**

- **Reactive WiFi offloading practices**
  - Wait until the device is connected to WiFi to transfer;
    - Large files
    - Delay tolerant content, e.g. software updates.
  - Push notifications to the user to connect to WiFi (after a timeout)
  - Scan the available WiFi networks and offer the user option the switch to WiFi



(2) Predictive Offloading

- **Predict** near future WiFi availability and **delay** the transmission.
- Only works for….
  - **Non-real time content** (delay-tolerant content)
    - Social networking content.
    - Software updates.
    - Environmental monitoring, data collection.

- **Predict** future demand and effectively **pre-load** the content when the user is connected to WiFi.
  - News
  - YouTube, Facebook videos

(2) Predictive Offloading

- Regular behavioral patterns of users
  - Users have regular weekly patterns
- Time based predictions
- User and location targeted content delivery
  - Location based advertisement distribution
  - User targeted content, e.g. Facebook videos

(3) Reduce Data

- **Throttle user requests**
  - Download only essentials
  - Don't upload everything collected

Data Compression

- Specially for text based files. e.g. HTML, JavaScript, CSS, .txt
  - Files less than 1KB do not benefit from compression.
- The compression and decompression have to be supported by the end-points.
  - Common algorithms - GZIP, DEFLATE

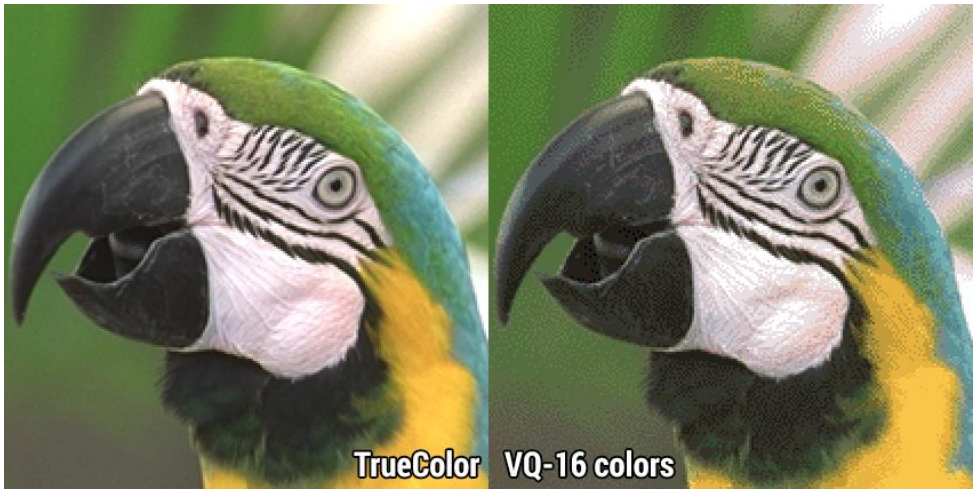
https://www.w3.org/Graphics/PNG/RFC-1951

https://developer.android.com/develop/ui/views/graphics/reduce-image-sizes

https://medium.com/@duhroach/reducing-png-file-size-8473480d0476

(3) Reduce Data - Image compression

- Trade size and color
  - More colors → larger size



- Adjust quality to around 75
  - Significantly smaller image size for insignificant visual difference

- **WebP** provides better compression than JPEG and PNG
  - https://developers.google.com/speed/webp/
  - WebP lossless images are 26% smaller compared to PNGs
- **AVIF**
  - Android 12 (API level 31) and higher support images that use the AV1 Image File Format (AVIF) dramatically improves image quality for the same file size when compared to older image formats, such as JPEG.

(3) Reduce Data - Image compression

- Use WebP whenever possible
- If not, use;
  - PNG – if image needs transparency or image is simple in color and structure
  - JPG – if image is complex

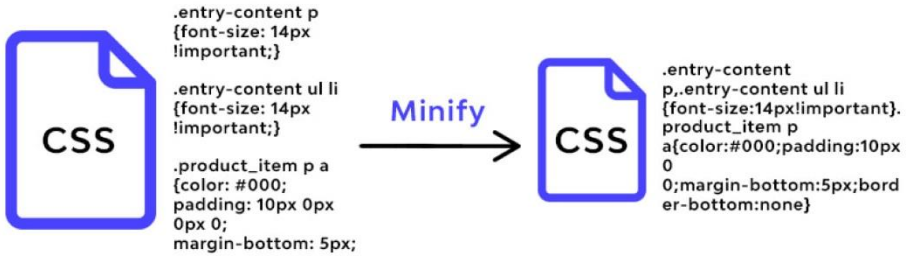
JPG : 201 k
PNG : 636 k

JPG : 82 k
PNG : 50 k

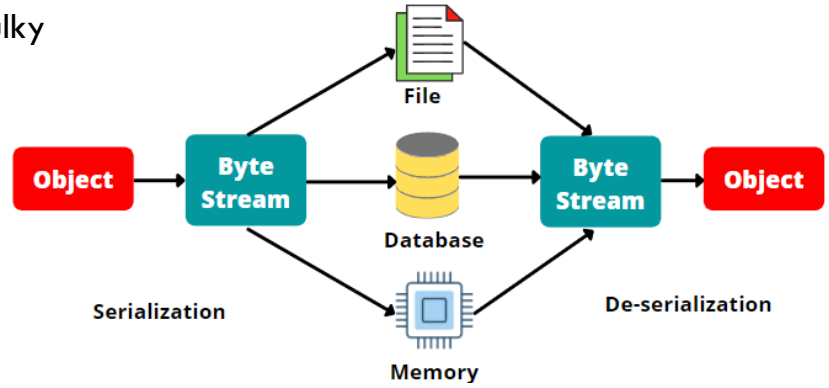
The decision between PNG and JPG often comes down to the complexity of the image itself. The image on the left has many small details, and thus compresses more efficiently with JPG. The image on the right, with runs of the same color, compresses more efficiently with PNG.

(3) Reduce Data - Other forms of data reductions

- **Minification** of JS, CSS and HTML
  - Remove unnecessary characters
    - JS – e.g. UglifyJS [https://github.com/mishoo/UglifyJS2]
    - CSS – e.g. CSSNano [https://github.com/ben-eb/cssnano]
    - HTML – HTMLminifier [https://github.com/kangax/html-minifier]
  - General minification tools. e.g. eMinifier
  - Compress: GZIP compression on server
  - Offline compression using Zopfli or 7-Zip.



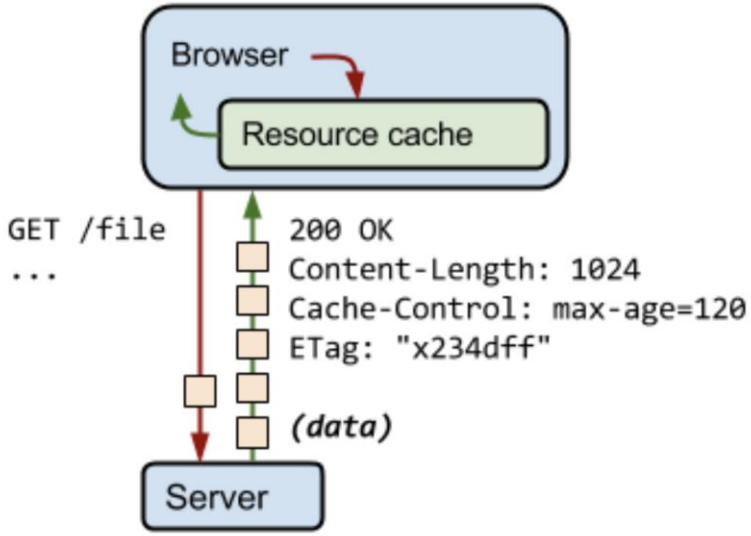
- **Serialization**
  - JSON, XML are serialization methods, but bulky and slow
  - Use customized structure to minimize data
  - FlatBuffers - an efficient cross platform serialization library
    - https://google.github.io/flatbuffers/



(4) Reuse Data

- Design the app and the communication protocol to reuse data, if you control both the client and the server.
  - **Static content** – Cache until updated
  - **Dynamic content** – Cache until expires

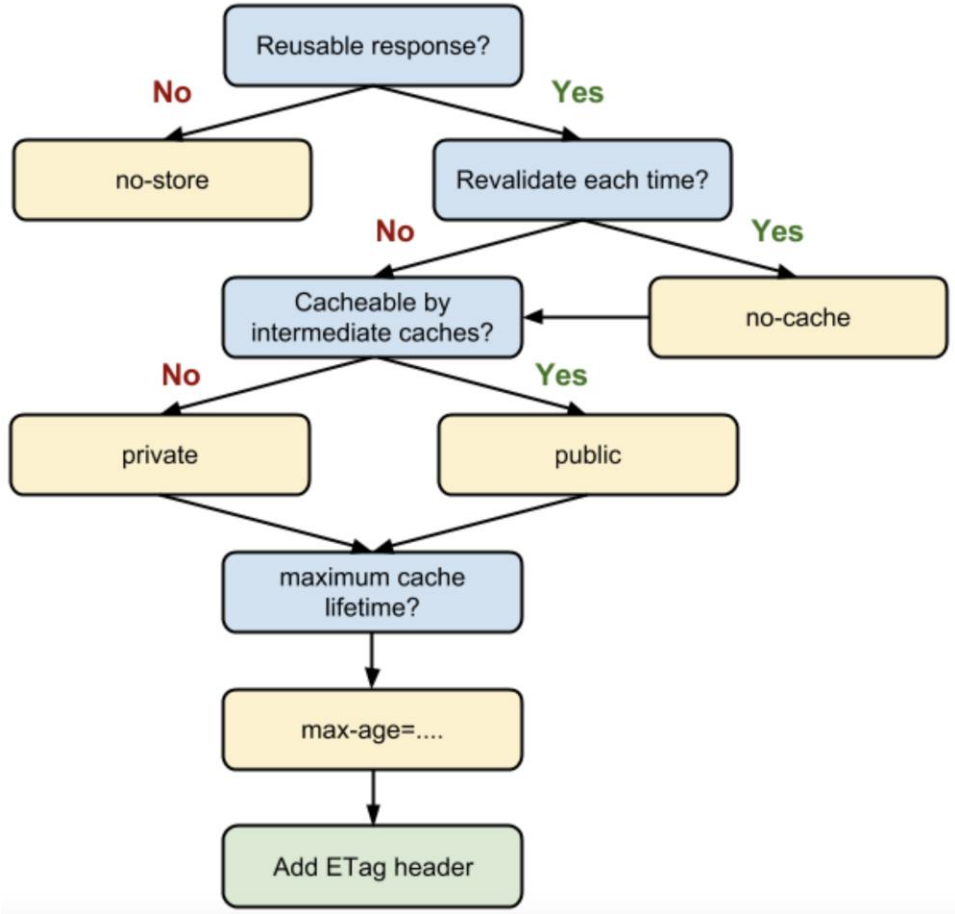
- For web-content, HTTP provides number of flexible cache-control parameters.



- https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching

(4) Reuse Data - HTTP Caching

- HTTP Header cache-control directives:
  - `no-store`- cannot cache the content
  - `no-cache`- can not use unless checking with the server
  - `public`- can be cached
  - `private`- can be cached for only one user
  - `max-age`- maximum time in seconds that the cached content can be used
  - **Immutable** - can cache it indefinitely without revalidation.



(4) Reuse Data - Cache Replacement Policies

- Storage is not unlimited, **not possible to cache everything…!**
- Memory cache vs Disk cache
- Use the most suitable cache replacement policy for the app.
  - FIFO - First In First Out
  - **LRU - Least Recently Used**
  - **LFU - Least Frequently Used**

- Android Caching support
  - https://developer.android.com/reference/android/net/http/HttpResponseCache
  - https://developer.android.com/reference/android/util/LruCache
  - https://developer.android.com/topic/performance/graphics/cache-bitmap

- Image caching/loading libraries
  - Glide - https://github.com/bumptech/glide
  - Picasso - http://square.github.io/picasso/

Tools for Network Debugging

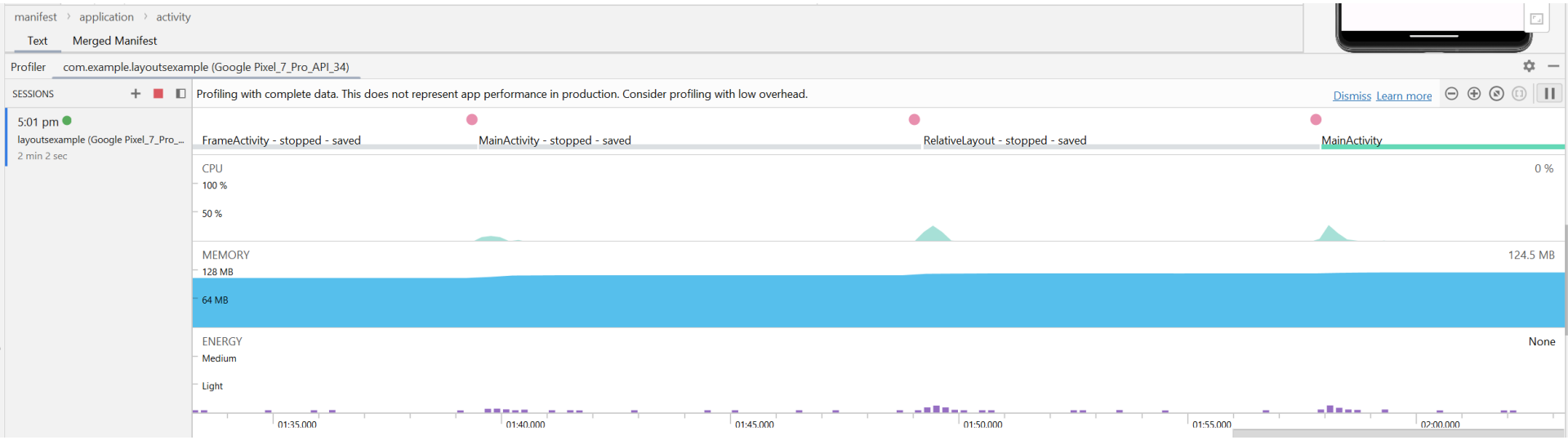
Android Profiler

- Available from Android Studio 3.0
  - **View** > **Tool Windows** > **Profiler**
- Great tool to ensure efficient/in-efficient usage of resources of your app
  - CPU Profiler
  - Memory Profiler
  - Energy Profiler
- Enable Advanced Profiling
  - **Run** > **Edit Configurations** > **Profiling** > **Enable advanced profiling**

- https://developer.android.com/studio/profile/network-profiler

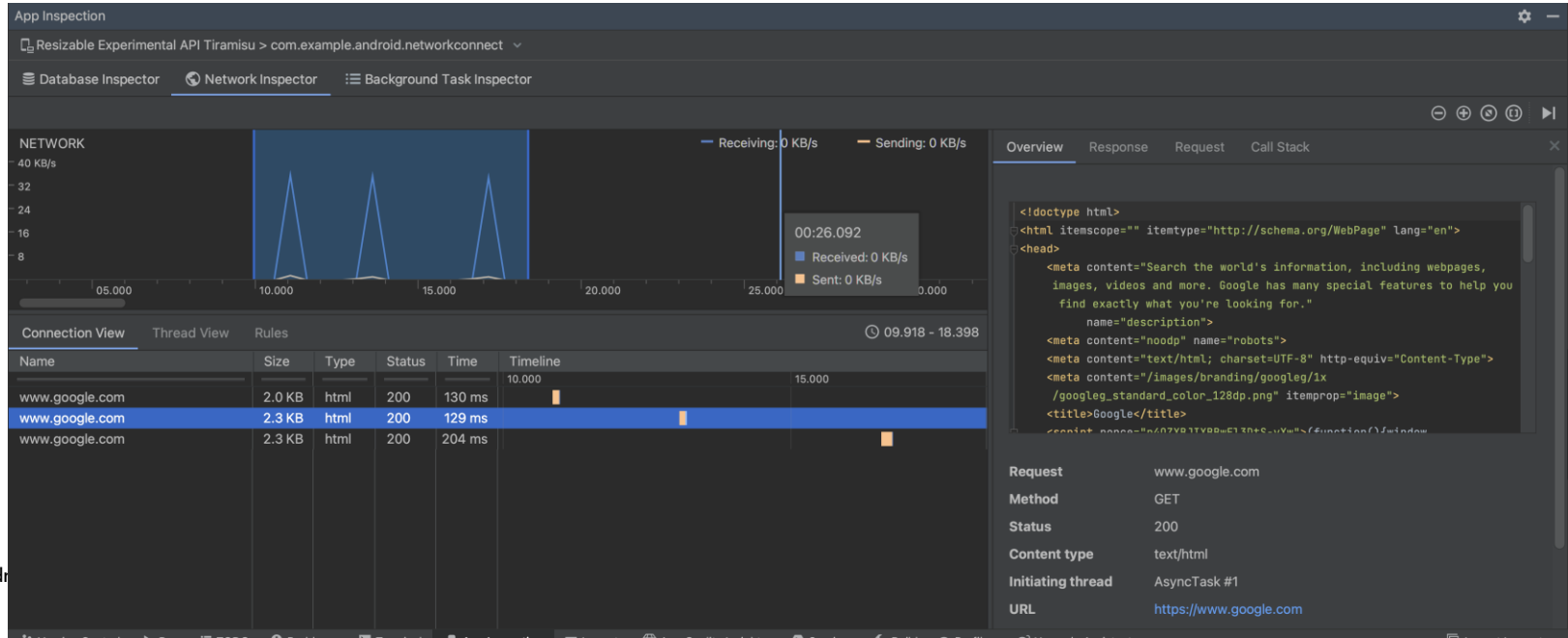
The Android Profiler

- There is a shared timeline that profiles the app simultaneously for CPU, memory, network, and energy.
- click on each of the individual timelines to begin profiling each resource in detail
- Note that to access these timelines, you need to connect Android Profiler to a running session.
  - need to connect a physical or virtual Android device to your system with debugging enabled, then start an application


manifest › application › activity
Text
Merged Manifest
Profiler
com.example.layoutsexample (Google Pixel_7_Pro_API_34)
SESSIONS
Profiling with complete data. This does not represent app performance in production. Consider profiling with low overhead.
Dismiss
Learn more
5:01 pm
layoutsexample (Google Pixel_7_Pro_...
2 min 2 sec
FrameActivity - stopped - saved
MainActivity - stopped - saved
RelativeLayout - stopped - saved
MainActivity
CPU
0 %
100 %
50 %
MEMORY
124.5 MB
128 MB
64 MB
ENERGY
None
Medium
Light
01:35.000
01:40.000
01:45.000
01:50.000
01:55.000
02:00.000

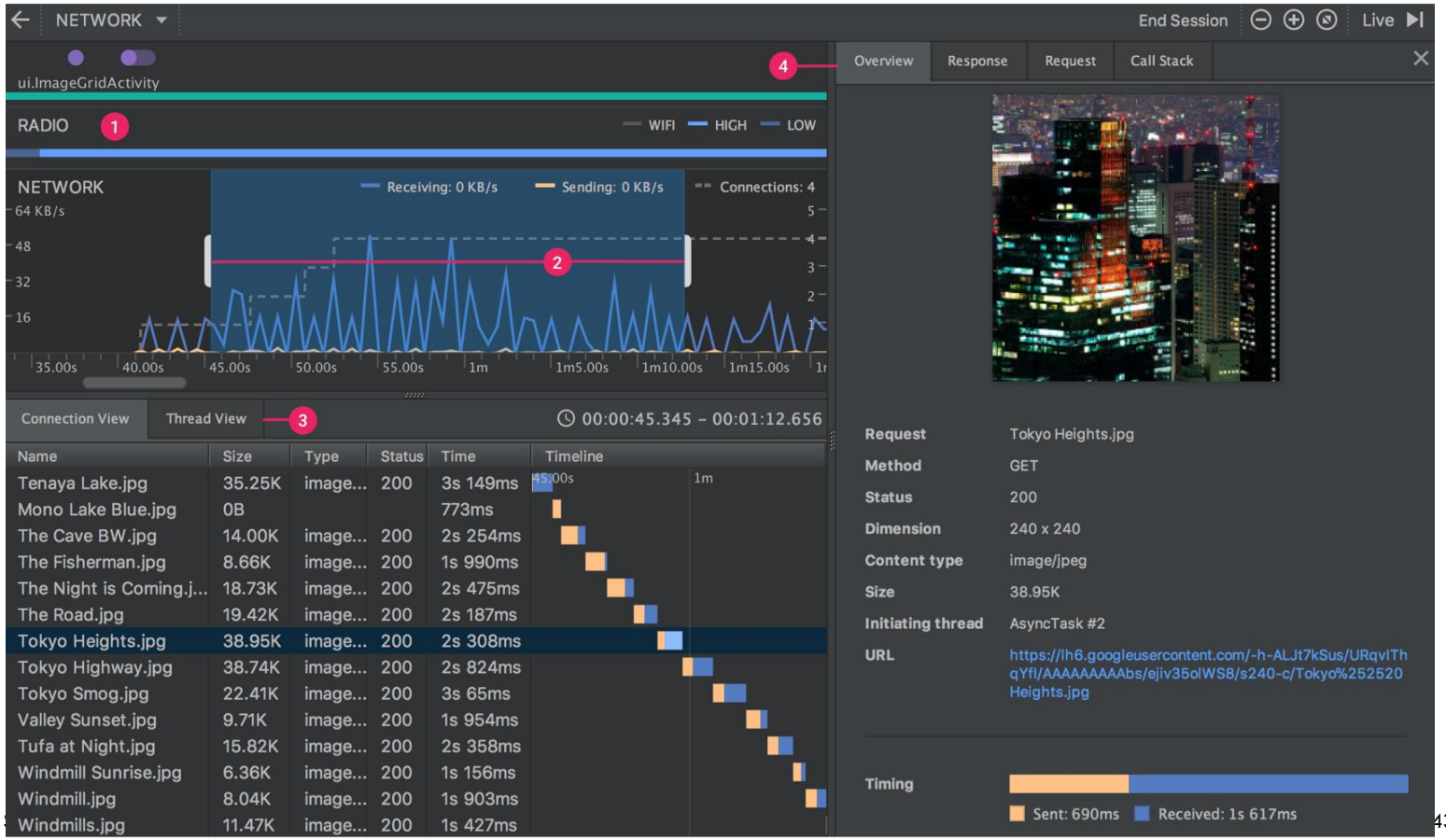
Network Inspector

- From the Android Studio navigation bar, select View > Tool Windows > App Inspection.
  - After the app inspection window automatically connects to an app process, select Network Inspector from the tabs.
- If the app inspection window doesn't connect to an app process automatically, you may need to select an app process manually.
- Select the device and app process you want to inspect from the App Inspection window



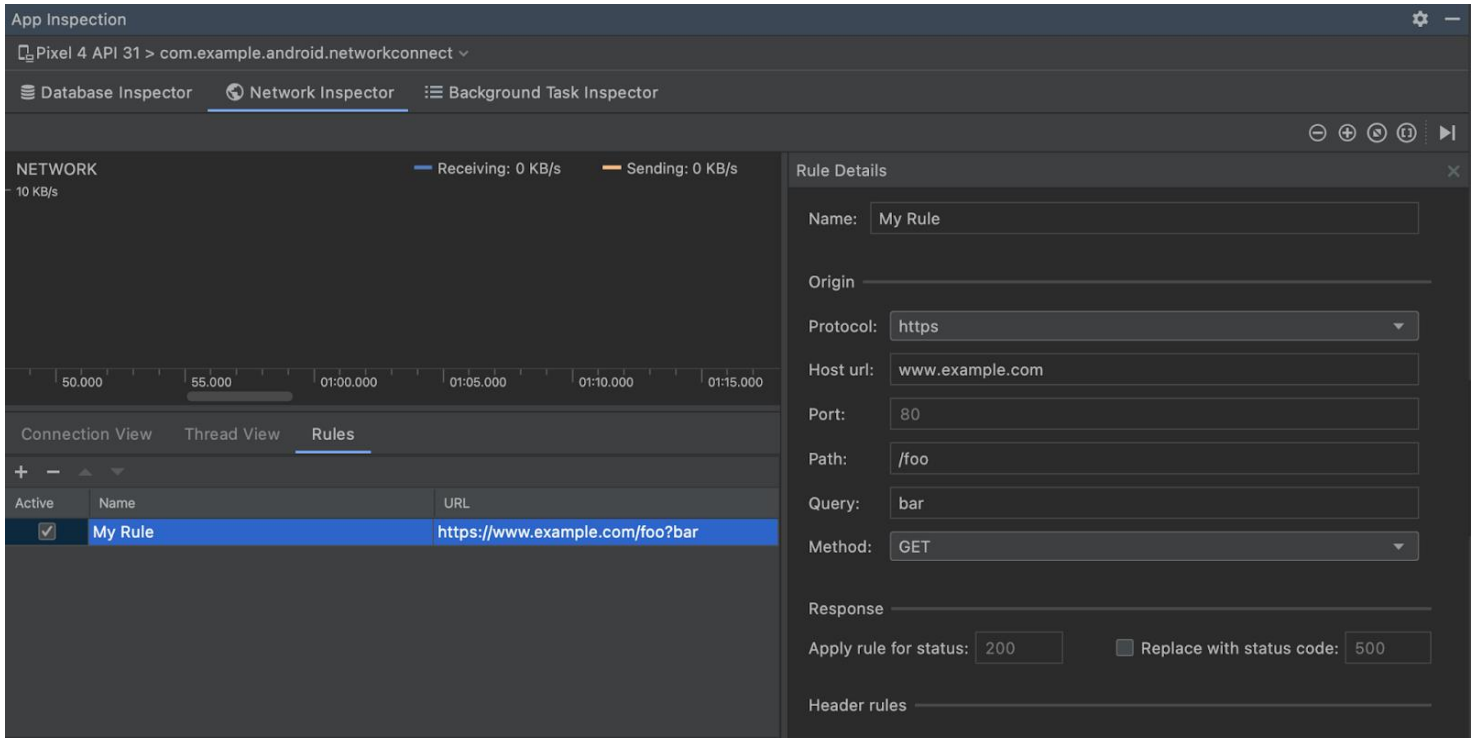
Network Inspector

- select one of the following tabs for more detail about the network activity during the selected portion of the timeline:
- **Connection View:** Lists files that were sent or received during the selected portion of the timeline across all of your app's CPU threads.



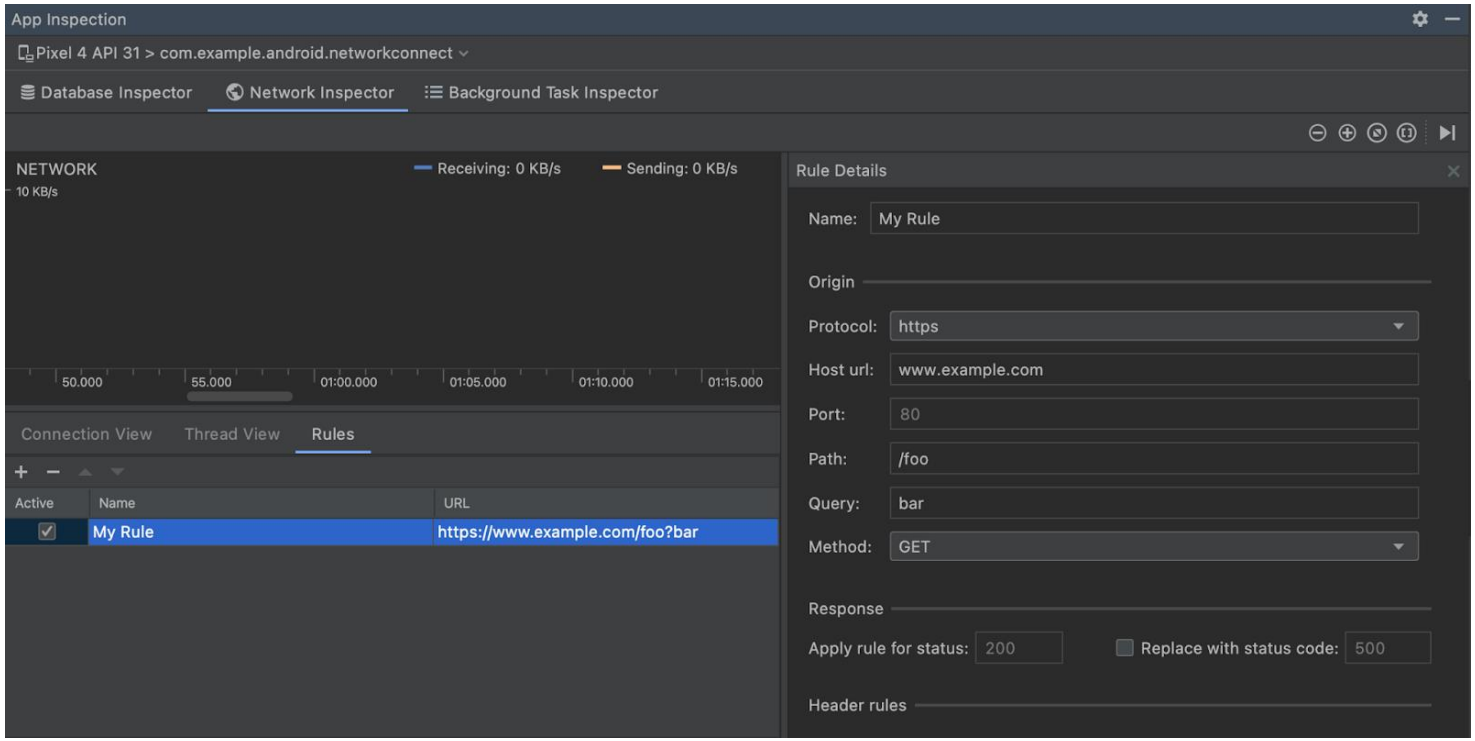
Network Inspector

- **Thread View:** Displays the network activity on each of your app's CPU threads.
- **Rules View:** Rules help test how your app behaves when encountering responses with different status codes, headers, and bodies.



Network Inspector

- select one of the following tabs for more detail about the network activity during the selected portion of the timeline:
- **Connection View**: Lists files that were sent or received during the selected portion of the timeline across all of your app's CPU threads.
- **Thread View**: Displays the network activity on each of your app's CPU threads.
- **Rules View**: Rules help test how your app behaves when encountering responses with different status codes, headers, and bodies.



https://youtu.be/41VZhwrXAKI?si=OF4_bh1JfgP3GQYP&t=423

Resources

- Android Developer Documentation
  - https://developer.android.com/guide/topics/connectivity/
- Computer Network Fundamentals
  - ***Computer Networking: A Top Down Approach, 7th Edition, Jim Kurose, Keith Ross***
- Mobile networking best practices
  - **AT&T Video Optimizer Best Practices**
  - https://developer.att.com/video-optimizer/docs/best-practices

What's Next?

- Happy learning !