Mobile Application Design and Development

CS 5520

Adrienne Slaughter

Summer 2019

Northeastern University
College of Computer and Information Science

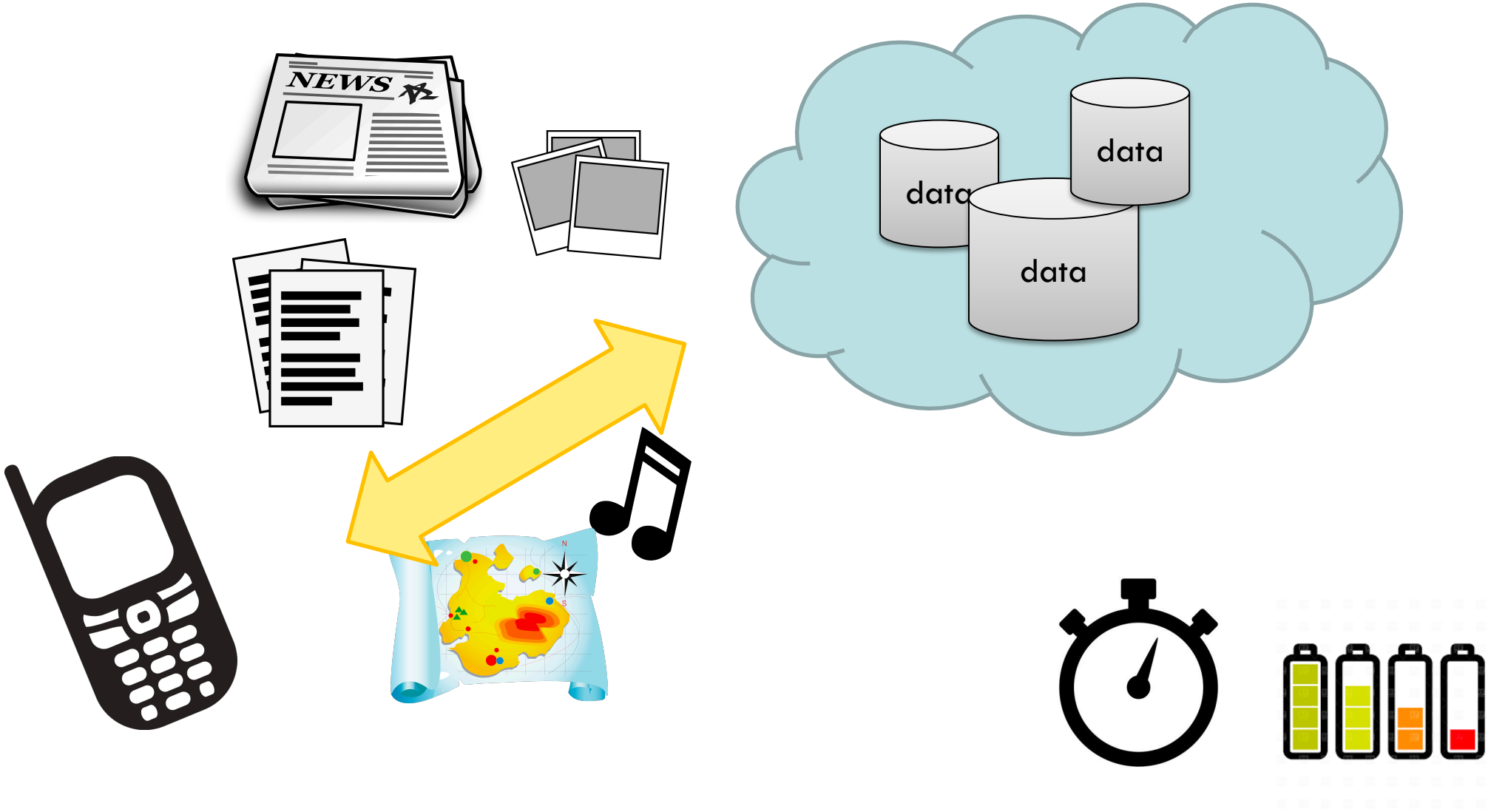
Topics

- Networking
  - Ways to get stuff from over the network
- We'll also touch on:
  - Authentication flows
  - Storing/Retrieving Preferences
  - Adapting to multiple versions of Android
- And review:
  - Async tasks
  - Handlers

NETWORKING

What is networking?

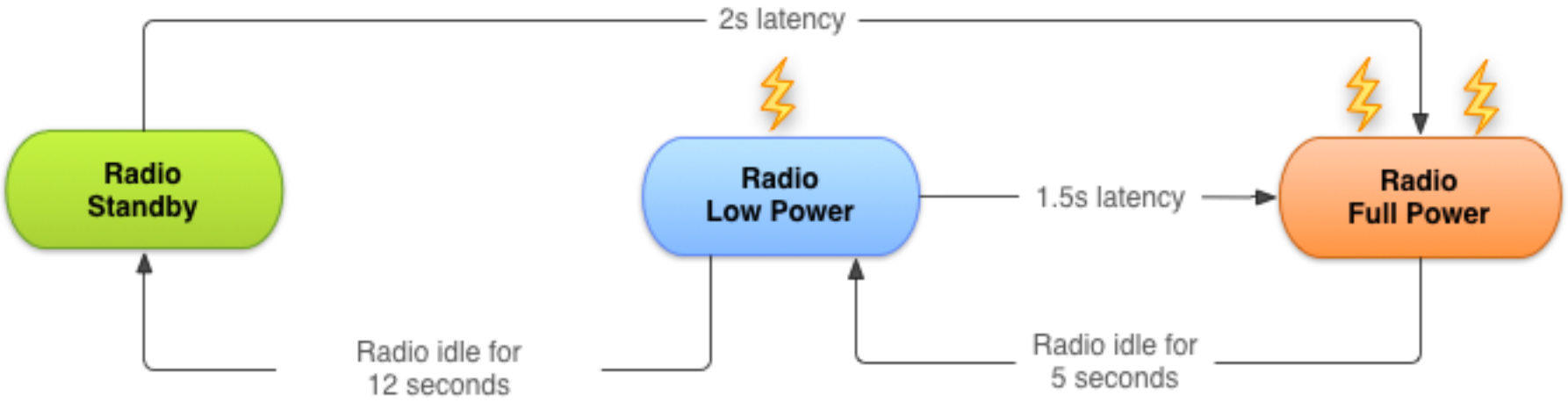
What are some relevant issues?


NEWS
data
data
data

Being safe with user data

- Think about what data you're collecting, and why.
- Send over SSL
- Consider a network security configuration and use custom Certificates/Certificate Authorities
- More info: https://developer.android.com/training/articles/security-tips.html#Networking

Radio State Machine (example: AT&T 3G)



- Standby is lowest power state
- Full Power is highest power draw state, but is required for data transfer
- Low Power state draws less power, but has a quick transition to full power state/data transfer mode

Networking Overview: 4 ways to "use the network"

- Viewing content by opening a browser
- Viewing content by embedding a web view
- Getting data from a web service
  - By analogy, sending data to a web service
- Managing network connections on Android
  - Power-aware data transfer design strategies
  - Using Volley, Cronet
  - DownloadManager
    - https://androidclarified.com/android-downloadmanager-example/

Providing content to your users, Strategy 1

- Open a web page in a browser using an intent
- Pros:
  - Simple!
  - Take advantage of existing content and display
  - Use a browser the user is familiar with
- Cons:
  - You're sending users out of your app
  - You have no control over the experience

Demo

- Mod7, LaunchWebActivity

Providing content to your users, Strategy 2

- Embed a WebView within your app
- Pros:
  - Relatively simple
  - It's easy to build a website– there are many existing tools and technologies to support it
  - You keep users within your app
- Cons:
  - Little control within your app over how the information is displayed
- Also: management of the web content is now out of the app developer's control.

Demo

- Mod7, WebViewActivity

Working with Web Services

- Data transfer strategy
  - Pulling data from the web
  - Pushing data to the web
- Uses HTTP connection
  - Android has included traditional, common HTTP networking classes
- Must be done off the main thread
  - Can either use an AsyncTask or new Thread

Working with Web Services, cont. (2/2)

- Pros:
  - Developer has great control over when to transfer data
  - Developer has great control over display of data
  - Many sources– this is a common way to access/save data now
  - Super flexible
- Cons:
  - More responsibility for the developer
  - More code to manage
  - More ways to make the experience slower, drain the battery

Demo

- Mod7, WebServiceActivity

Understanding Network Management on Android

- Differences between Wi-Fi and Cellular networks
- How to check the network status
- How to store user preferences
- How to respond to network changes

Wi-Fi vs Cellular Networks

	Wi-Fi	Cellular
Power consumption	Uses less power for data transfer*	Uses more power, especially with 4G Usually automatically disabled when connected to Wi-Fi (by the device)
Speed	Typically faster than cellular (greater bandwidth)	Typically slower than Wi-Fi; network speed depends on carrier and network type (3G, 4G, LTE, …)
Cost	Usually free	Metered (user has a plan; network carriers usually throttle data after a certain limit)
Availability	Tends to be more private, familiar locations (home, work, school, cafés) Useful when in one place for a while Smaller area of service	Available in public, places where there isn't Wi-Fi (e.g., a park), more remote locations. Stays connected while traveling Larger area of service

* Wi-Fi uses less power for data transfer, but if the device is not connected to a Wi-Fi network when Wi-Fi networks are available, the battery will drain while trying to connect to one of the networks.

How to check network status

- Using ConnectivityManager
- There's a difference between before Android 21 (Lollipop) and after
- Generally:
  - Ask the ConnectivityManager about the current network
  - If connected to a network:
    - Decide on network/data transfer operations based on the network type that you are connected to.

Demo

- Mod7, NetworkInfoActivity

User Preferences around Networking

- A best practice is to let your user specify preferences of using different networks
- Can use a nifty thing called PreferenceActivity
- Basically, want to:
  - Identify the types of data transfer you do in the app
  - Allow the user to specify which network to use for each type of data transfer

Example: Let's brainstorm

- Mail App
- Music Streaming app
- News reader app
- Single player game

Demo: Setting user preferences

- Mod7, NetworkInfoActivity & NetworkSettingsPrefActivity

Reacting to changes in the network connection

- Challenge:
  - Network changes after you check network status & user preferences
- Scenario:
  - User launches your app at home
  - Your app checks and sees that it is on wi-fi, and the user has said you can download data on wi-fi but not cellular
  - User leaves home and transitions to cellular
  - Does your app keep downloading? Or stop?

Reacting to changes in the network connection, cont.

- Use a BroadcastReciever
- BUT:
  - Declaring in the manifest means….:
- Goal:
  - When the network changes:
    - Check the new network
    - Check the user preferences
    - If they are consistent, continue.
    - Otherwise, stop.

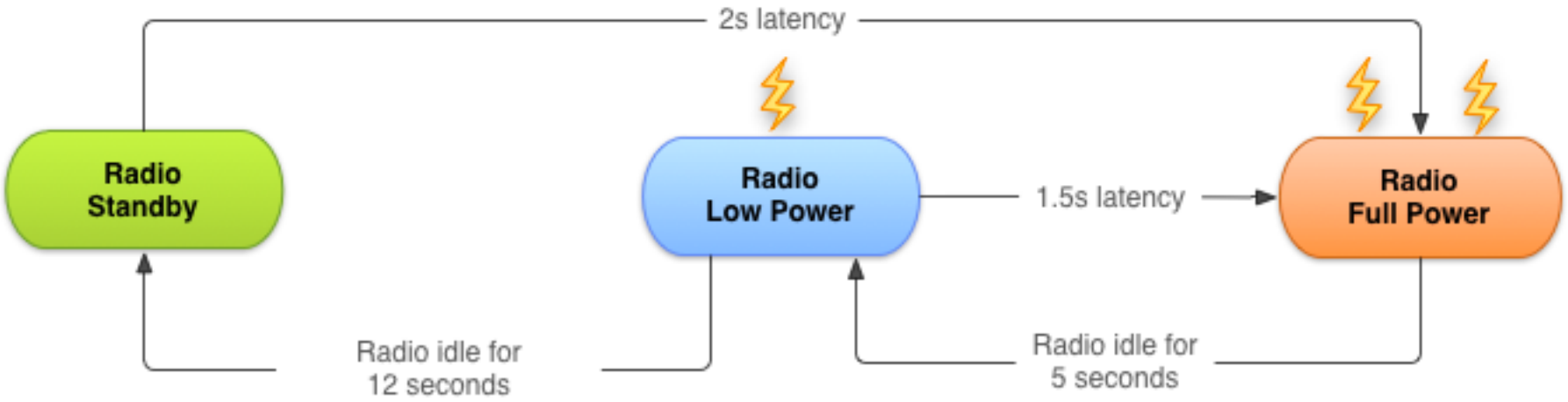
Demo

- Mod7, NetworkInfoActivity,

Optimizing Data Transfer

- Radio state machine
- Approaches:
  - Bundling network requests
  - Prefetching
  - Reducing Connections

Radio State Machine (example: AT&T 3G)



- Standby is lowest power state
- Full Power is highest power draw state, but is required for data transfer
- Low Power state draws less power, but has a quick transition to full power state/data transfer mode

Bundling Network Requests

- Goal:
  - Transfer as much data as possible during each transfer session in an effort to limit the number of sessions you require.
- How:
  - Create a pending transfer queue of data transfers that can be delayed
  - When a data transfer that can't be delayed is requested, request all the data transfers in the queue at the same time

Approaches to minimize battery drain

- Optimize Downloads
  - Prefetching
  - Batch transfers
  - Reduce connections
  - Use Network Profiler to monitor
- Minimize effect of regular updates
- Avoid redundant downloads
- Modify downloads based on connectivity types

Prefetching Data

- Download all the data you are likely to need for a given time period in a single burst, over a single connection, at full capacity.
- Pros:
  - Reduce the number of radio activations needed to download the data
    - Conserve battery life
    - Improve latency
    - Lower required bandwidth
    - Reduce download times
    - Improves User Experience by avoiding a wait during long download times
- Cons:
  - If too aggressive, can increase battery drain and bandwidth use

Prefetching Data, cont.

- Cons:
  - If too aggressive, can increase battery drain and bandwidth use
- Prefetch depends on:
  - size of the data being downloaded
  - likelihood of it being used.
- As a rough guide: (based on above state machine)
  - for data that has a 50% chance of being used within the current user session, prefetch for around 6 seconds (approximately 1-2 Mb)
  - After that, the potential cost of downloading unused data matches the potential savings of not downloading that data to begin with.

Prefetching Data, cont.

- Generally speaking, it's good practice to prefetch data such that you will only need to initiate another download every 2 to 5 minutes, and in the order of 1 to 5 megabytes.
- Following this principle, large downloads—such as video files—should be downloaded in chunks at regular intervals (every 2 to 5 minutes), effectively prefetching only the video data likely to be viewed in the next few minutes.

Reducing Connections

- It's more efficient to reuse a connection than to create a new one
  - Also enables network to react to congestion and other network data issues.
- Challenge:
  - It's better to actively close the connection than to wait for a timeout
  - But closing the connection too early prevents you from being able to reuse it
  - More details for the developer to be aware of

Other approaches we're not covering:

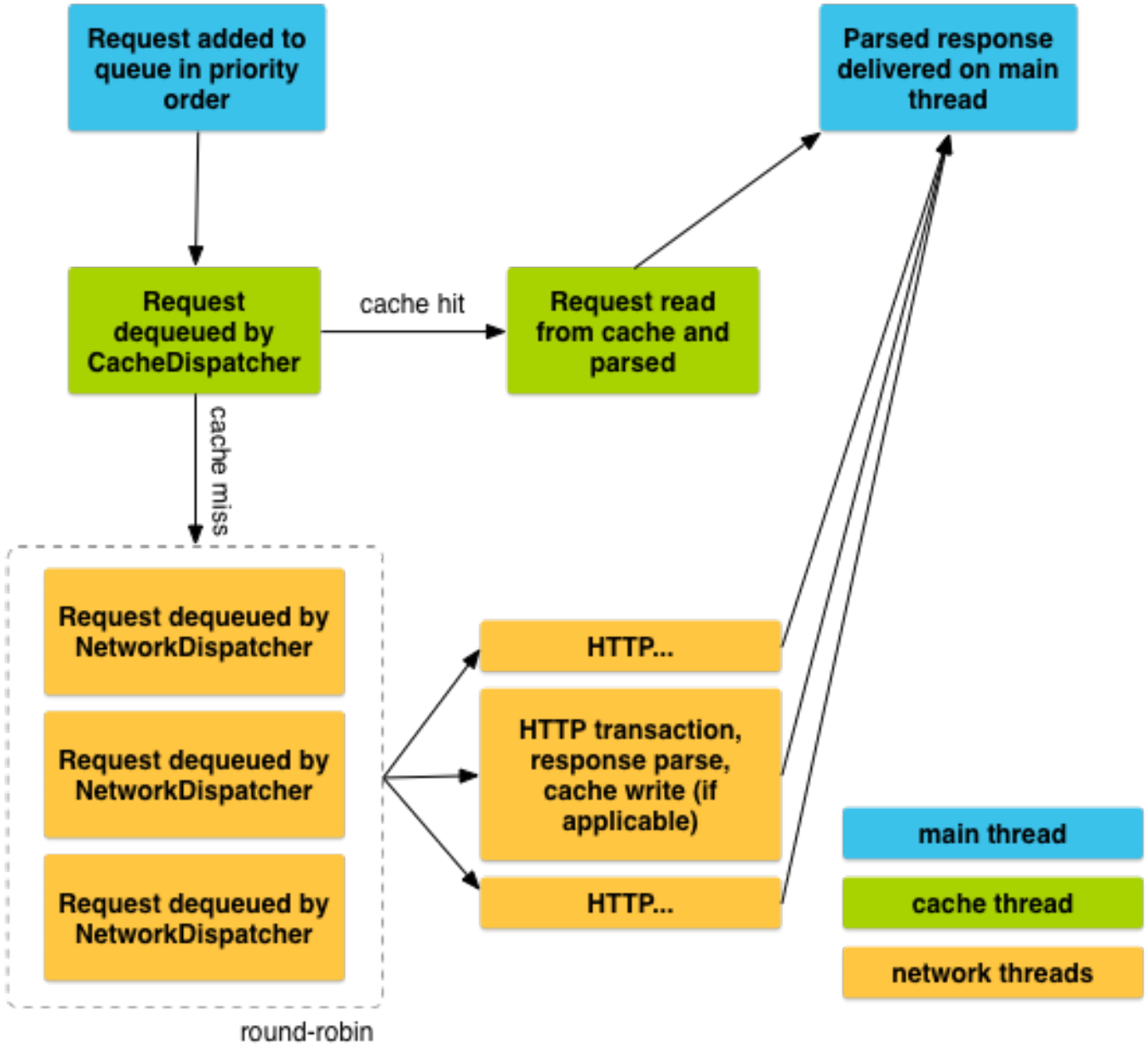
- Using Push Notifications versus polling to notify of updates
- Data Saver
  - New in Android 23
  - User specifies data preferences on the device
  - Data Saver blocks background data transfer
  - Apps can ask to be whitelisted for exemptions
- Download Manager, helping apps

NEW TOOLS TO IMPROVE NETWORKING

Volley

Volley excels at RPC-type operations used to populate a UI, such as fetching a page of search results as structured data.

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP cache coherence.
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.
- More appropriate for small network requests, not large data requests


Request added to queue in priority order
Request dequeued by CacheDispatcher
cache hit
Request read from cache and parsed
Parsed response delivered on main thread
cache miss
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
round-robin
HTTP...
HTTP transaction, response parse, cache write (if applicable)
HTTP...
main thread
cache thread
network threads

Cronet

- Chromium Network stack, provided to Android
- Incorporates technology to reduce latency and increase throughput
- To use:
  - Create a Cronet Engine
  - Create a UrlRequest.Callback to handle the response of your request
  - Create an executor to manage network tasks
  - Create a URL request
    - Pass it to the executor, which asynchronously executes the request and notifies your callback with the response when completed

Sync Adapter

- Provides framework for communication between your app and a server
  - As opposed to, e.g., requesting a URL/webpage
- You can specify triggers and scheduling for syncing to happen

Work Manager

- Makes it easy to schedule tasks to be run "deferrable, asynchronous" tasks
  - On a schedule
  - E.g.: do a big data request when the user is on wifi and not doing something else

Summary

- Radio state machine
  - 20 seconds to go from standby (low power) to standby
  - Arrange data transfers to optimize amount of data transferred while minimizing battery impact
- Approaches:
  - Bundling network requests
  - Prefetching
  - Reducing Connections