Doing Work in the Background: Where Implementation meets Theory

Darryn Campbell
SW Architect, Zebra Technologies
@darryncampbell

December 19th, 2019

Doing work in the background: Where implementation meets theory

Who am I?

- Developer advocate / Software architect for Zebra Technologies
  - Android OEM developing task specific devices
- Responsible for our Android developer kits & APIs

@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

Problem statement

Over the years there has been continual change in this area:

						android
Running in the background	**Job Scheduler**	**Doze mode**	**Doze “on the go”**	**Background restrictions**	**Bucket-based restrictions & Restricted apps**	**Cannot start activity from background**

@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

Advice from Google – I need to run a task in the background, how should I do it?

Local Deferrable Work & Sync	**WorkManager**
Is online triggered?	**Firebase Cloud Messaging & Work Manager**
User expects to run immediately without interruption?	**Foreground Service**
Needs to execute at an exact time?	**Alarm Manager**

@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

Advice from Google – I need to run a task in the background, how should I do it?

Local Deferrable Work & Sync	**WorkManager**
Is online triggered?	**Firebase Cloud Messaging & Work Manager**
User expects to run immediately without interruption?	**Foreground Service**
Needs to execute at an exact time?	**Alarm Manager**

@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

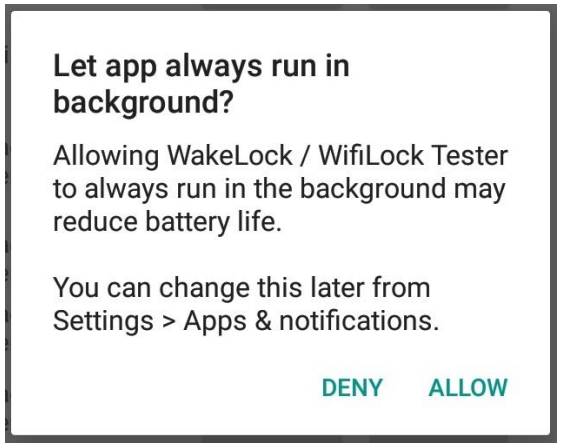
Customer 1: Warehouse device tracker

- Problem statement:
  - Think “Find My Device” but within the 4 walls
  - No external network access:
    - No Google accounts or FCM
    - No WAN connection
  - Show real-time device locations in a list and on a map

Doing work in the background: Where implementation meets theory

Customer 1: Warehouse device tracker

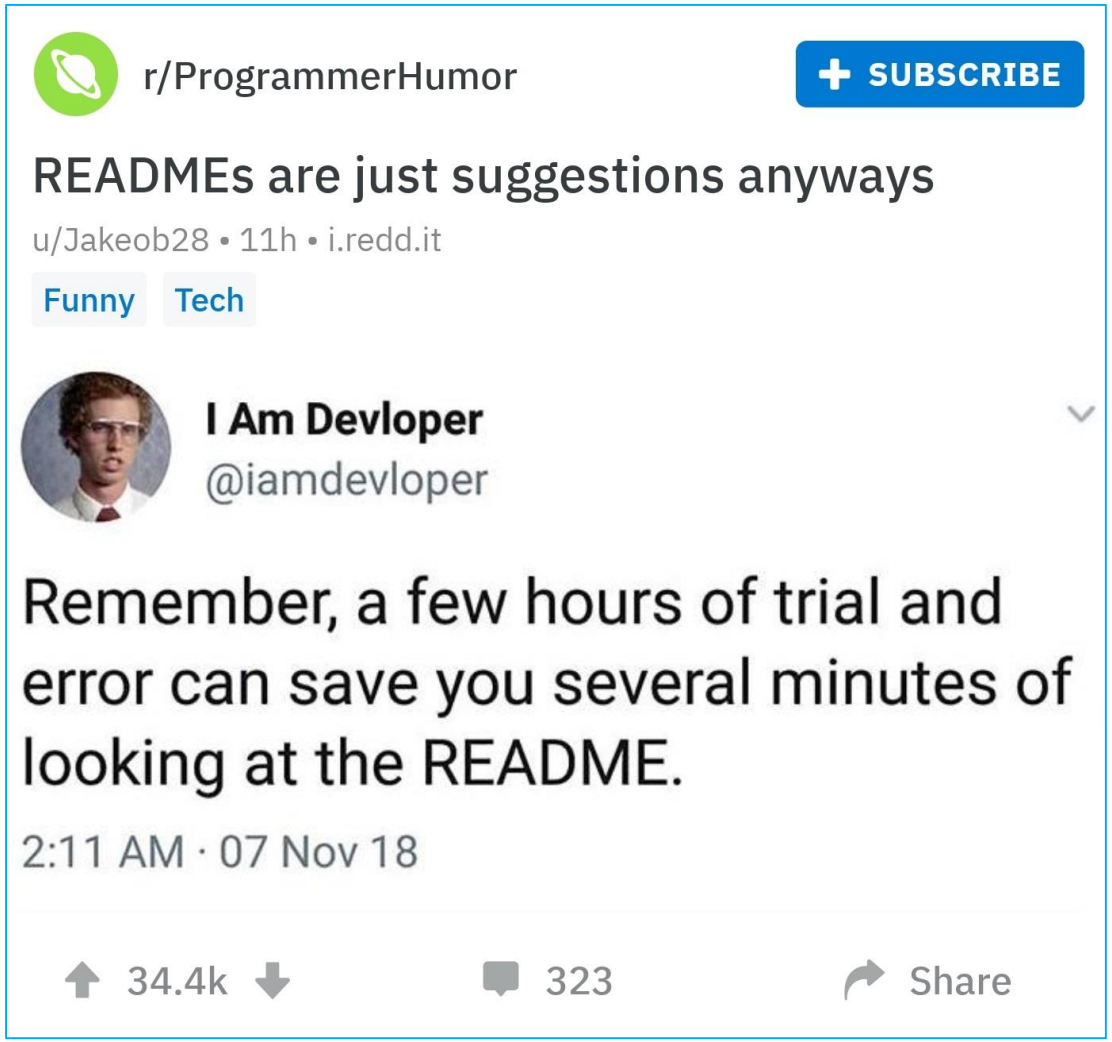
- Initial concept:
  - Device needs constant connection to communicate location every 5 minutes
    - Therefore: Whitelist the app to avoid doze mode
    - Whitelisting required the user to agree to disable battery optimization manually
  - PoC saw battery consumption increase by 7%!!



Doing work in the background: Where implementation meets theory

Customer 1: Warehouse device tracker

- Final concept:



Doing work in the background: Where implementation meets theory

Customer 1: Warehouse device tracker

- Final concept:
  - This was a perfect example where following Google's guidance was the right thing to do
    - Use Work Manager and schedule a periodic job to report device location
  - Lessons learned:
    - Discuss requirements with the client – the 5 minute reporting interval was being treated as an *exact* interval
    - Spend 30 minutes reading articles on power management rather than hours debugging why your app is non-responsive or battery drain is excessive

Doing work in the background: Where implementation meets theory

Customer 1: Warehouse device tracker

- Segue: WorkManager
  - Great getting started guides at https://developer.android.com/topic/libraries/architecture/workmanager
  - This is not a presentation about Work Manager – Google have some great presentations on YouTube for that.

Doing work in the background: Where implementation meets theory

Customer 1: Warehouse device tracker

- Segue: WorkManager
  - Used to perform tasks in the background that should happen when your application is not in the foreground
  - Takes account of all Android's background restrictions now and in the future
  - Replaces and supersedes previous classes such as Firebase JobDispatcher & JobScheduler
  - Work can be periodic or can run a single time
  - Work will be scheduled to run at a time that is most power efficient e.g. during a doze mode window
  - Developer can add conditions such as 'device on power' or 'device has wifi'

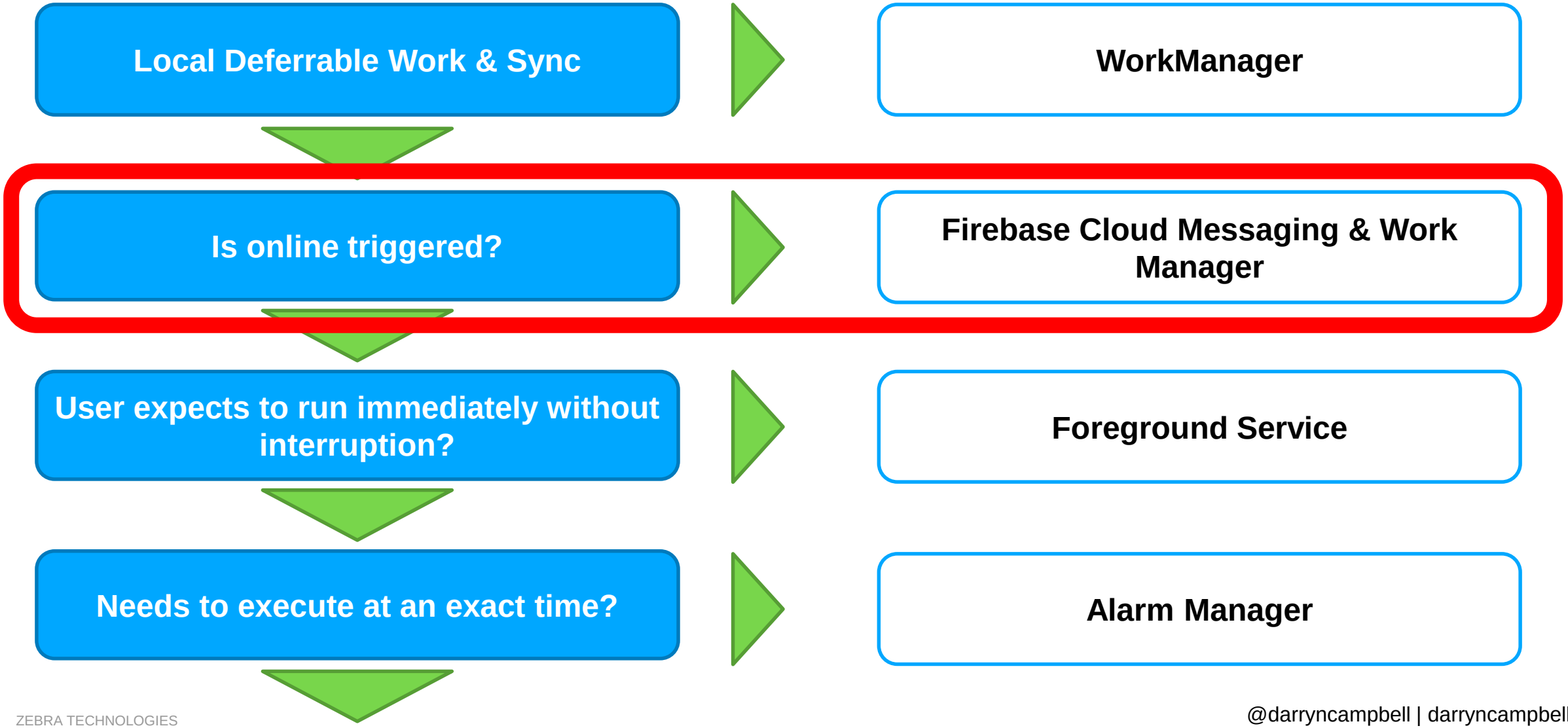
Doing work in the background: Where implementation meets theory

Customer 1: Warehouse device tracker

```
public class UploadWorker extends Worker {

OneTimeWorkRequest uploadWorkRequest = new OneTimeWorkRequest.Builder(UploadWorker.class)
        .setConstraints(new Constraints.Builder().setRequiresCharging(true).build())
        .build()

  @Override
  public Result doWork() {
    // Do the work here - in this case, upload the images

WorkManager.getInstance(myContext).enqueue(uploadWorkRequest);

    // Indicate whether the task finished successfully with the Result
    return Result.success()
  }
}
```

Doing work in the background: Where implementation meets theory

Advice from Google – I need to run a task in the background, how should I do it?



@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

Customer 2: Device update in the field

- Problem statement:
  - Device OS (Android) needs to be updated in the field
  - Devices are not managed by a carrier, a server needs to push a .zip file containing the update to the device
    - To avoid strain on network, devices need to be updated in batches
    - Devices need to be updated during downtime to minimize interruption
    - .zip file can be in excess of 1GB

Doing work in the background: Where implementation meets theory

Customer 2: Device update in the field

- Initial concept:
  - “OSUpdate” application on the device maintains a foreground service
    - Always ready to receive instructions to update automatically
    - Download can happen without worrying about background restrictions or doze mode
    - All devices now have a continual notification for a function that happens monthly
  - Need to educate user that the notification should be ignored

@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

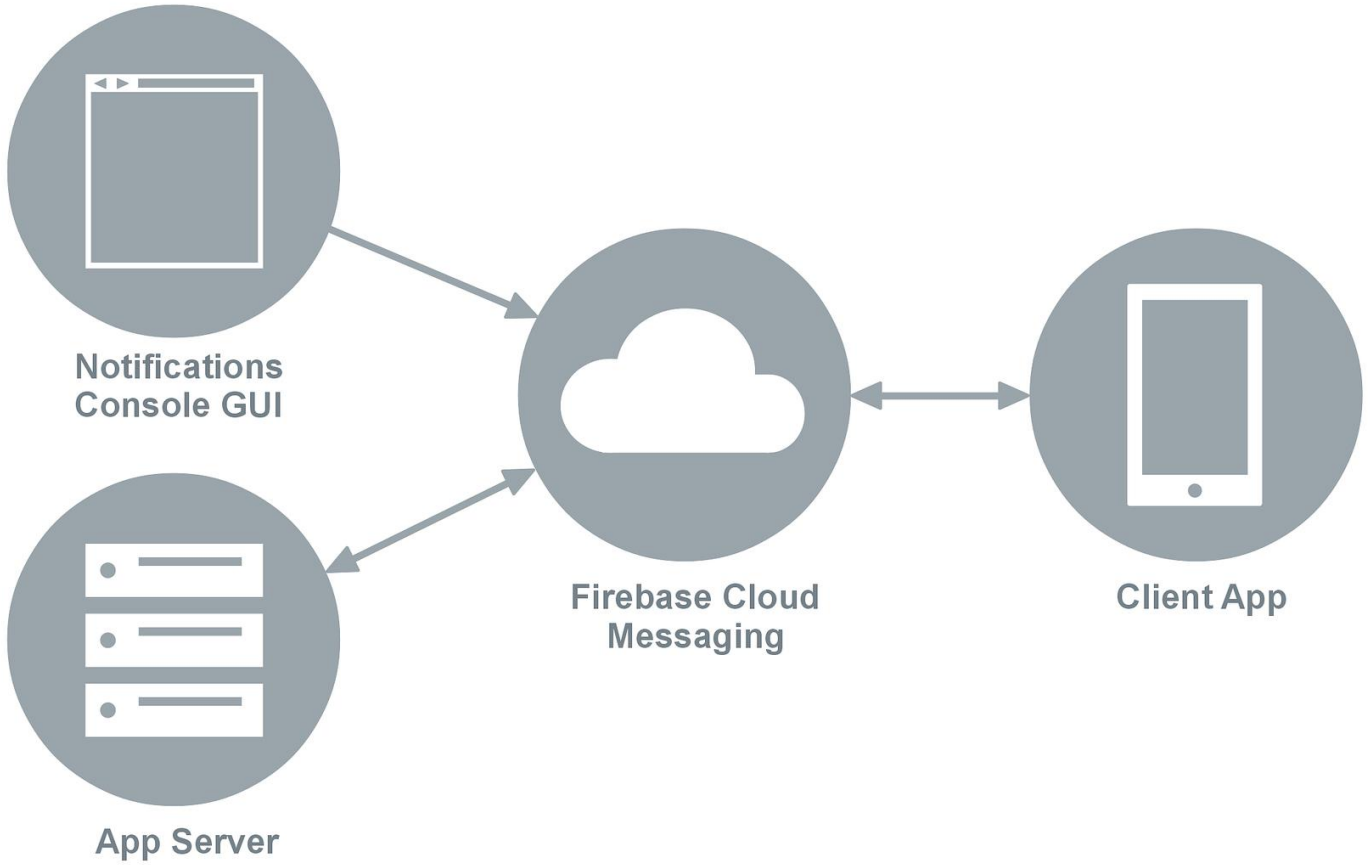
Customer 2: Device update in the field

- Final concept:
  - OS Updates are initiated by a Firebase Cloud Message
    - High priority message received regardless of power restrictions
    - 1 or 2 messages a month – no worries about bucket-based restrictions
    - No need to maintain a persistent notification - Power efficient
  - Downloading the update is managed by Android's built-in Download Manager
    - Works 99% of the time but some edge cases for updating unattended devices

Doing work in the background: Where implementation meets theory

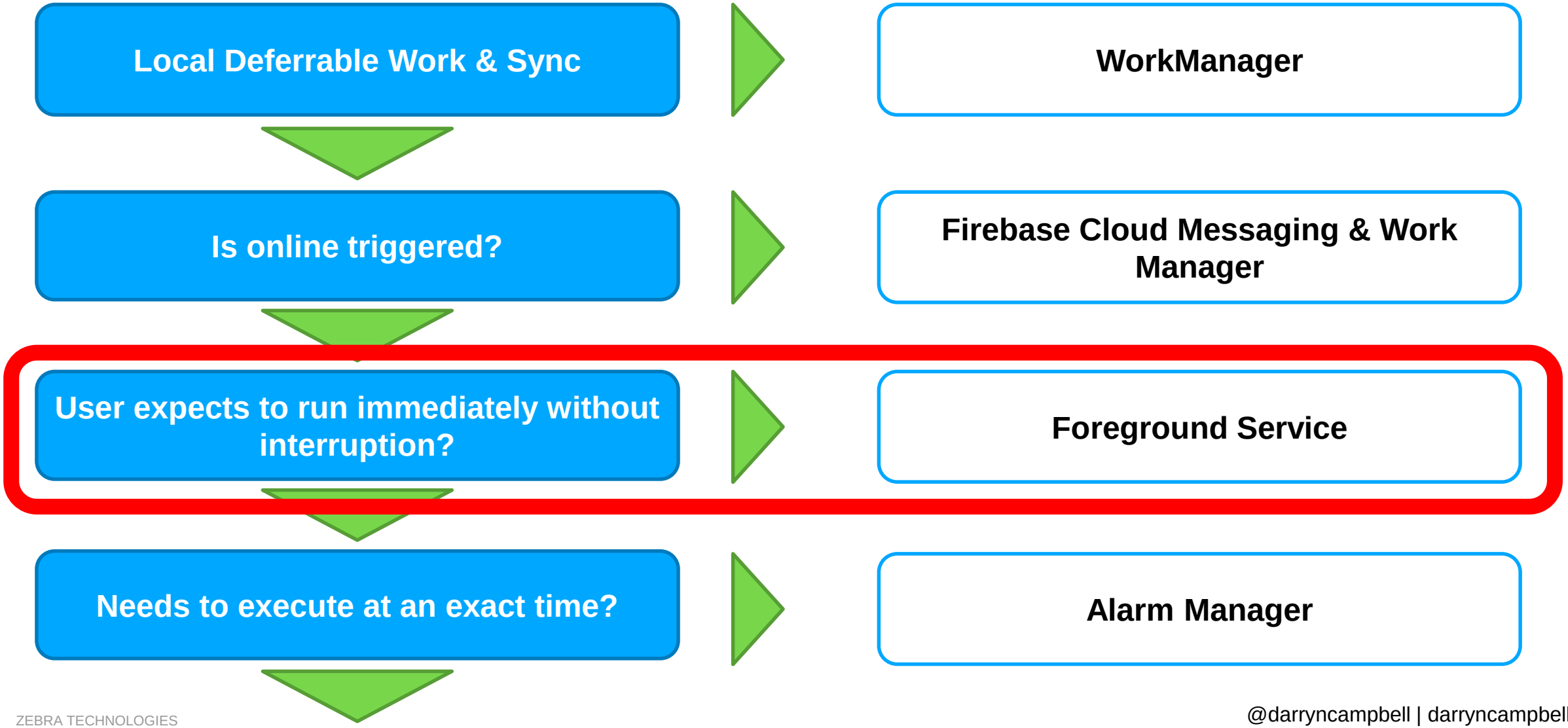
Customer 2: OS Update

- Segue: Firebase Cloud Messaging (FCM)



Doing work in the background: Where implementation meets theory

Advice from Google – I need to run a task in the background, how should I do it?



Doing work in the background: Where implementation meets theory

Customer 3: Healthcare task assignment system

- Problem statement:
  - Within a hospital
    - Assign tasks to nurses / doctors / staff
    - Some messages VERY time sensitive
    - Emergency tasks could (literally) be a matter of life of death
  - Customer not comfortable with Firebase Cloud Messages
    - FCM “very reliable but outages do occur”
    - Round-trip-time to external FCM unacceptable

Doing work in the background: Where implementation meets theory

Customer 3: Healthcare task assignment system

- Initial concept:
  - System deployed and working well under Lollipop
  - Uptime guaranteed by hospital IT administrator
  - Any changes to software / hardware seen as risk

- Customer moved to Marshmallow
  - Deployed software stops working when in doze
  - Customer uses <u>foreground service</u> as a workaround

Doing work in the background: Where implementation meets theory

Customer 3: Healthcare task assignment system

- Final concept:
  - Note: This scenario falls under the acceptable use-cases for whitelisting
  - “Messaging App” “can’t use FCM because of technical dependency”

Type	Use-case	Can use FCM?	Whitelisting acceptable?	Notes
Instant messaging, chat, or calling app.	Requires delivery of real-time messages to users while device is in Doze or app is in App Standby.	Yes, using FCM	Not Acceptable	Should use FCM high-priority messages to wake the app and access the network
		Yes, but is not using FCM high-priority messages.		
Instant messaging, chat, or calling app; enterprise VOIP apps.		No, can't use FCM because of technical dependency on another messaging service or Doze and App Standby break the core function of the app.	Acceptable	
automation app	actions, such as for instant messaging, voice calling, new photo management, or location actions.			
Peripheral device companion app	App's core function is maintaining a persistent connection with the peripheral device for the purpose of providing the peripheral device internet access.	If applicable.	Acceptable	
	App only needs to connect to a peripheral device periodically to sync, or only needs to connect to devices, such as wireless headphones, connected via standard Bluetooth profiles.	If applicable.	Not Acceptable	

Doing work in the background: Where implementation meets theory

Customer 3: Healthcare task assignment system

- Final concept:
  - Device MQTT server remains running, requires CPU
    - **Hold a <u>partial wake lock</u> to ensure the CPU continues to run**
  - Device continues to have Wi-Fi access
    - **Hold a <u>WiFi lock</u> to ensure the device does not drop WiFi**
    - Only applicable for Nougat and below. Oreo and higher this is the case by default
  - Application cannot be subject to Doze mode / App Standby
    - **Application is placed on the <u>battery whitelist</u>**
  - Battery life is not a big concern for this customer

Doing work in the background: Where implementation meets theory

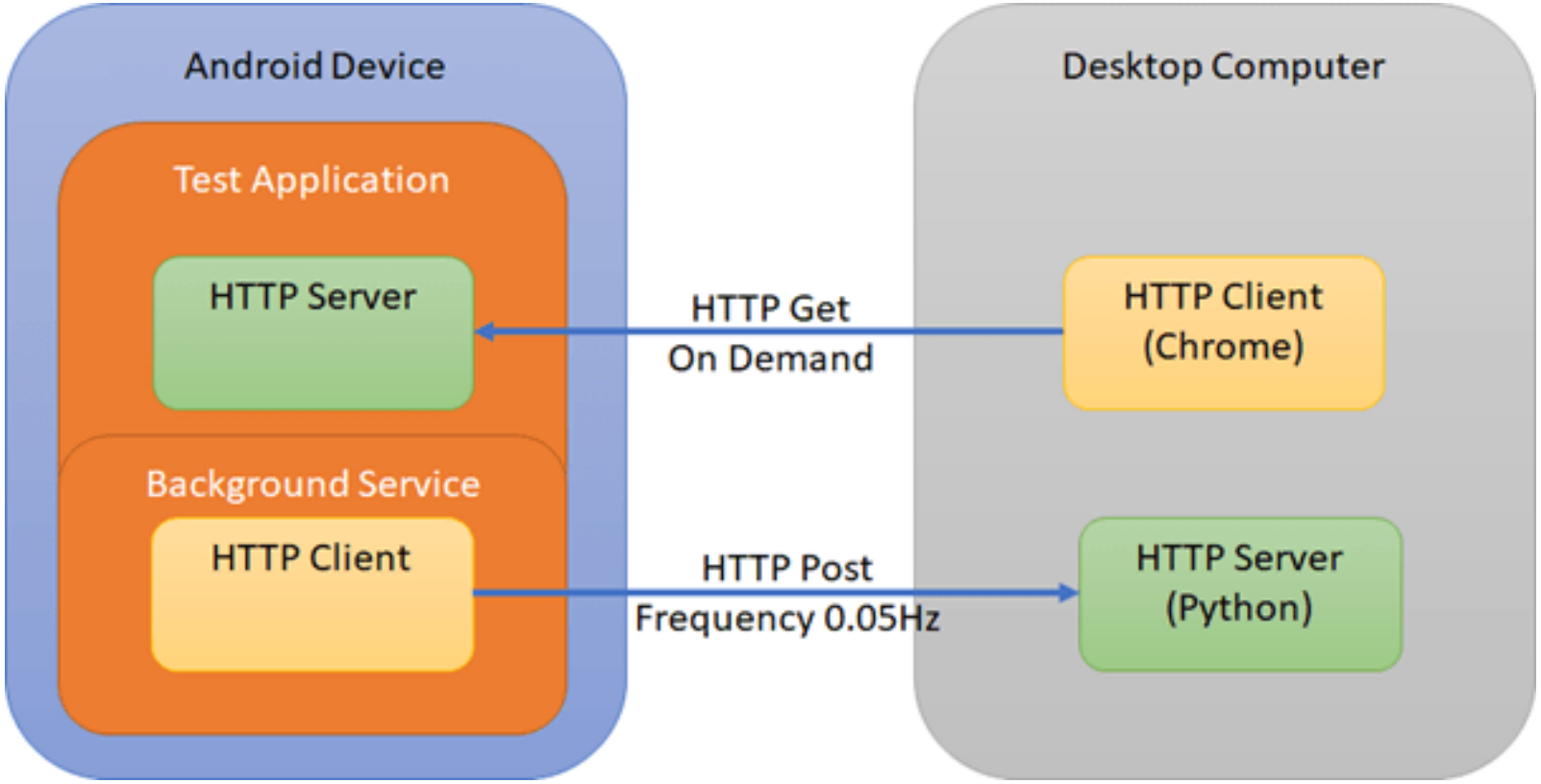
Customer 3: Healthcare task assignment system

- Final concept:
  - Is this sustainable??
    - Android Nougat introduced enhanced doze mode
    - Android Oreo introduced background restrictions
    - Android Pie introduced additional restrictions
    - Android 10 introduced additional, though unrelated restrictions
  - Application needs extensive testing (& potential rework) with each Android version

Doing work in the background: Where implementation meets theory

Customer 3: Healthcare task assignment system

- Segue: What if a Foreground service is unacceptable?



Doing work in the background: Where implementation meets theory

Customer 3: Healthcare task assignment system

- Segue: What if a Foreground service is unacceptable?

Whitelisting your app has implications distributing via the Play Store

Your whitelisted app may also cause excessive battery drain

Doing work in the background: Where implementation meets theory

Advice from Google – I need to run a task in the background, how should I do it?

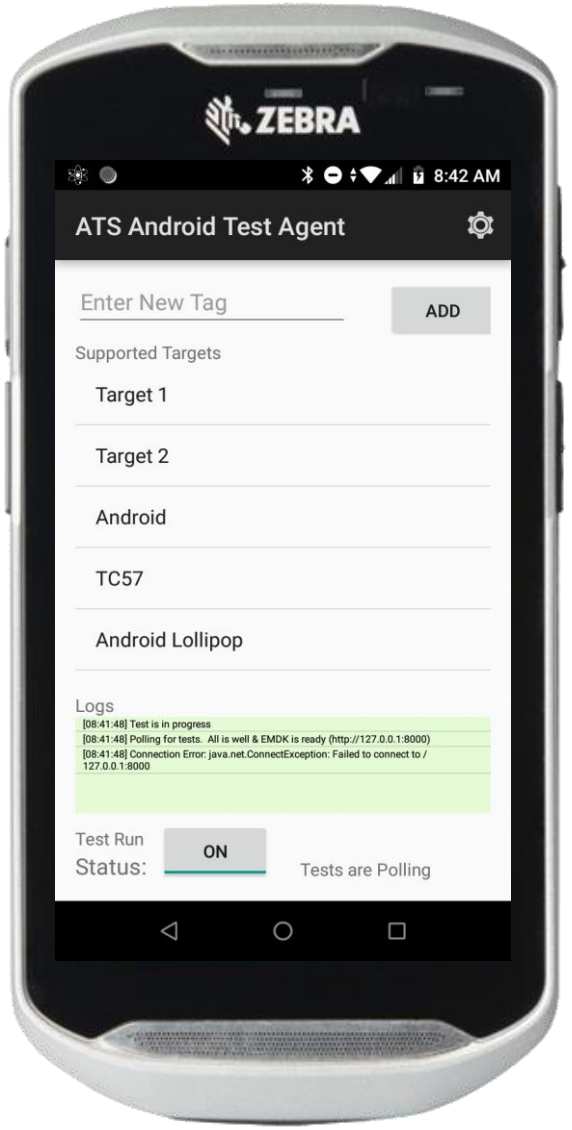
Local Deferrable Work & Sync	**WorkManager**
Is online triggered?	**Firebase Cloud Messaging & Work Manager**
User expects to run immediately without interruption?	**Foreground Service**
Needs to execute at an exact time?	**Alarm Manager**

@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

“Customer 4”: My own experience

- No customer scenarios but I do have my own story…
- Rewind to December 2013 & I’m writing my first Android app

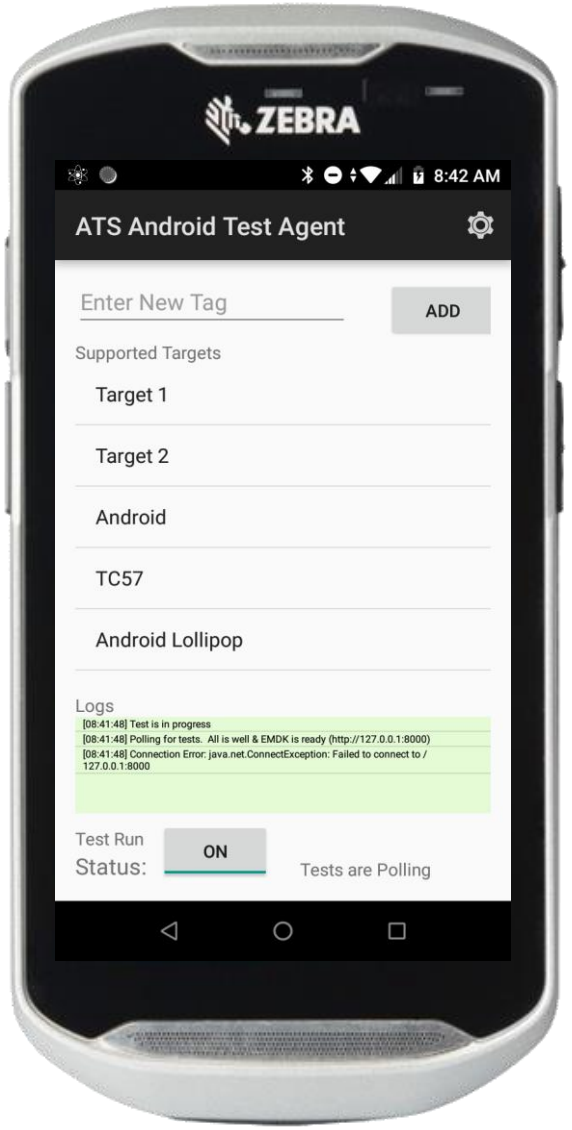


@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

“Customer 4”: My own experience

- Rewind to December 2013 & I’m writing my first Android app
  - Client sitting on Android device and runs LUA scripts
  - Polls server every x seconds to ask if there is a new script
  - Uses **AlarmManager** setInexactRepeating
    - Toss up between setInexact and setExact at the time
  - Uses **Wakeful Broadcast Receiver** to do work
    - Now deprecated
  - At the time, this was the correct way to do things (according to Stack Overflow)



@darryncampbell | darryncampbell.co.uk

Doing work in the background: Where implementation meets theory

“Customer 4”: My own experience

- What is my point??
  - AlarmManager has been around since API level 1
    - Long history and lots of experience using the API amongst the community
  - Early way of specifying an exact time to execute
  - Still underlying implementation of WorkManager on API 14 – 22
  - API 23 introduced setExactAndAllowWhileIdle()
    - Minimum period of 15 minutes
    - No benefits over WorkManager, “may take [..] liberties when scheduling in order to optimize for battery life”
  - Alarm Manager should only be used for a single task that needs to happen at a specific time, like an Alarm

Doing work in the background: Where implementation meets theory

Conclusions: Battery Historian

- I am frequently asked how to monitor device battery levels or how much extra battery use will result from e.g. taking a wake log. Battery Historian is your fried.

Doing work in the background: Where implementation meets theory

Further reading

- Google series of blogs on Power management features: https://android-developers.googleblog.com/search/label/Power%20series
- My own blogs on background restrictions:
  - Pie / 10: http://www.darryncampbell.co.uk/tag/background-restrictions/
  - Oreo / Nougat / Marshmallow: https://developer.zebra.com/blog/keeping-your-android-application-running-when-device-wants-sleep-updated-android-oreo
- My own test app for background behaviour: https://github.com/darryncampbell/WakeLock_WifiLock_Exerciser

Questions?

http://developer.zebra.com

@darryncampbell | darryncampbell.co.uk

Thank You