Timing in Android

And How it Evolved Over the Android Versions

Robert Zetzsche – movisens GmbH

Über mich

- Robert Zetzsche (25 Jahre)
- Lebenslauf
  - Bachelorstudium Angewandte Informatik an der DHBW Karlsruhe bei der Fiducia IT AG
  - Masterstudium Informatik am KIT
- Arbeit
  - Mobile Android Engineer bei der movisens GmbH

movisens

Agenda

- Motivation
- Anwendungsfälle
- Historie
- Aktueller Stand
- Best Practices
- Fazit

Motivation

- Timing ist eine prinzipielle Anforderungen um moderne Apps zu entwickeln
  - Bspw. für Benachrichtigungen, Reminder, Hintergrundarbeiten/-aktualisierungen
- Verschiedene Anforderungen an das Timing
  - Bspw. Genauigkeit, Energiesparsamkeit, Stabilität

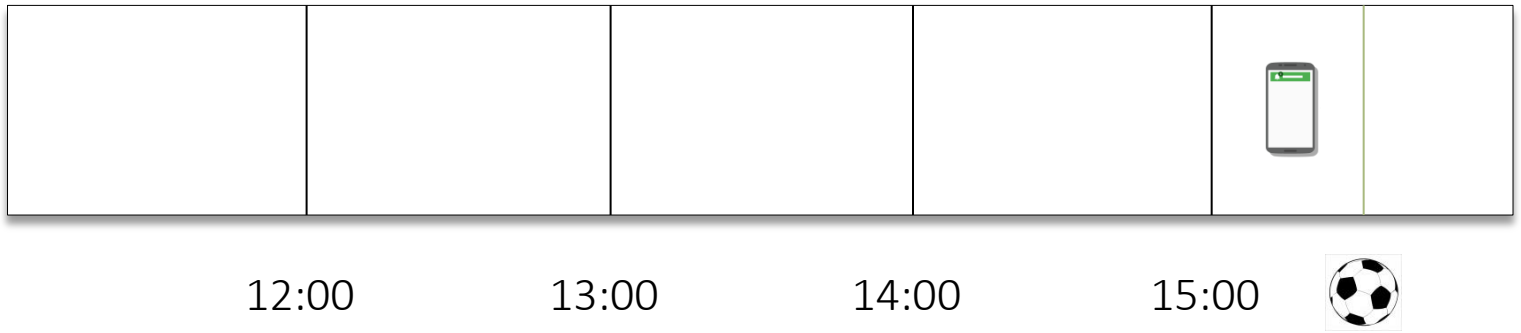
Anwendungsszenarien

Anwendungsszenarien

- Hintergrundsynchronisierungen durchführen
  - Bsp. Twitter, Facebook oder Mail Apps
  - Meist wenn kein Push möglich ist
- Sammeln von Daten im Hintergrund (interne oder externe Sensordaten)
- Automatisierungsapps
  - Bspw. Verzögertes Ausschalten von Lampen

Anwendungsszenarien

- Klassisches Beispiel: Sport-App
- Soll einem 10 Minuten vor dem Beginn der Partie eine Benachrichtigung geben



Anwendungsszenarien - AA

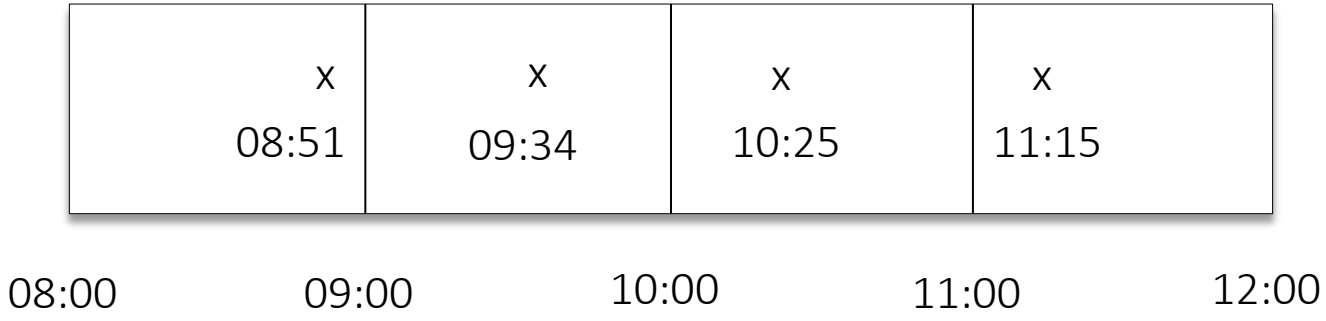
- Ambulantes Assessement
  - Datenerhebung im Alltag eines Probanden
  - Häufig in der Psychologie für Ermittlung von Gefühlen, Stimmungen, Erlebnissen etc.
  - Mittlerweile oft Computergestützt
- movisensXS zur Erstellung und Ausführung solcher Studien auf Smartphones
  - Statistische Datenauswertung mit Daten

Anwendungsszenarien - movisensXS

- Forscher bestimmt über Ablaufplan die Zeit
  - Bspw. festen oder randomisierte Zeitpunkte
- Exakter Zeitpunkt für die Ausführung enorm wichtig
  - Restrospektive Verzerrung
  - Minderung der Vergleichbarkeit zwischen Probanden
  - Randomisierung nicht mehr garantiert

Anwendungsszenarien – Stratofizierte Randomisierung

- Algorithmus zur Findung von Randomisierten Zeitpunkten
  - Bsp: 08:00-12:00, 4x , 20 Minuten Mindestabstand



Historie

Historie – Android < 4.3

- Alarm Manager wurde mit API Level 1 eingeführt

> “These allow you to schedule your application to be run at some point in the future. […] Registered alarms are retained while the device is asleep (and can optionally wake the device up if they go off during that time)[…]. The Alarm Manager holds a CPU wake lock as long as the alarm receiver's onReceive() method is executing.”
>
> https://developer.android.com/reference/android/app/AlarmManager.html

Historie – Android < 4.3

- Alarm Manager wurde mit API Level 1 eingeführt

  Methoden:

  set(int type, long trigger, PendingIntent operation)

  setRepeating(int type, long trigger, long interval, PendingIntent operation)

  setTimeZone(String timeZone)

  cancel(PendingIntent operation)

  setInexactRepeating(int type, long trigger, long interval, PendingIntent op)

  Type:

  ELAPSED_REALTIME_WAKEUP, ELAPSED_REALTIME, RTC, RTC_WAKEUP

Historie – Android 4.4

- API Level 19 führte neues Verhalten ein
  - Alarme verschiedener Apps werden zusammengefasst (“reasonably similar times”)
  - set() und setRepeating() sind **ab jetzt** inexakt
  - setWindow() und setExact() neu eingeführt
  - Wdh. Alarme müssen über Rescheduling umgesetzt werden

Historie – Android 5

- JobScheduler bzw. GCMNetworkManager eingeführt mit API-Level 21
  - GCMNetworkManager ab API-Level 9
- Energieeffiziente Möglichkeit um Hintergrundtasks auszuführen
- Bedingungen können über Kriterien gesetzt werden
  - Netzwerk-Status
  - Batterie-Status
  - Deadlines

JobScheduler/ GCMNetworkManager

```
OneoffTask task = new OneoffTask.Builder()
        .setService(MyTaskService.class)
        .setTag(TASK_TAG_WIFI)
        .setExecutionWindow(0L, 3600L)
        .setRequiredNetwork(Task.NETWORK_STATE_UNMETERED)
        .build();

mNetworkManager = GcmNetworkManager.getInstance(this);
mNetworkManager.schedule(task);
```

Historie – >= Android 6

- API Level 23 führte erneut neues Verhalten ein
  - Einführung von Doze Mode und App Standby
  - Alarme werden nicht mehr exakt ausgeführt

setAlarmClock(AlarmManager.AlarmClockInfo info, PendingIntent op)

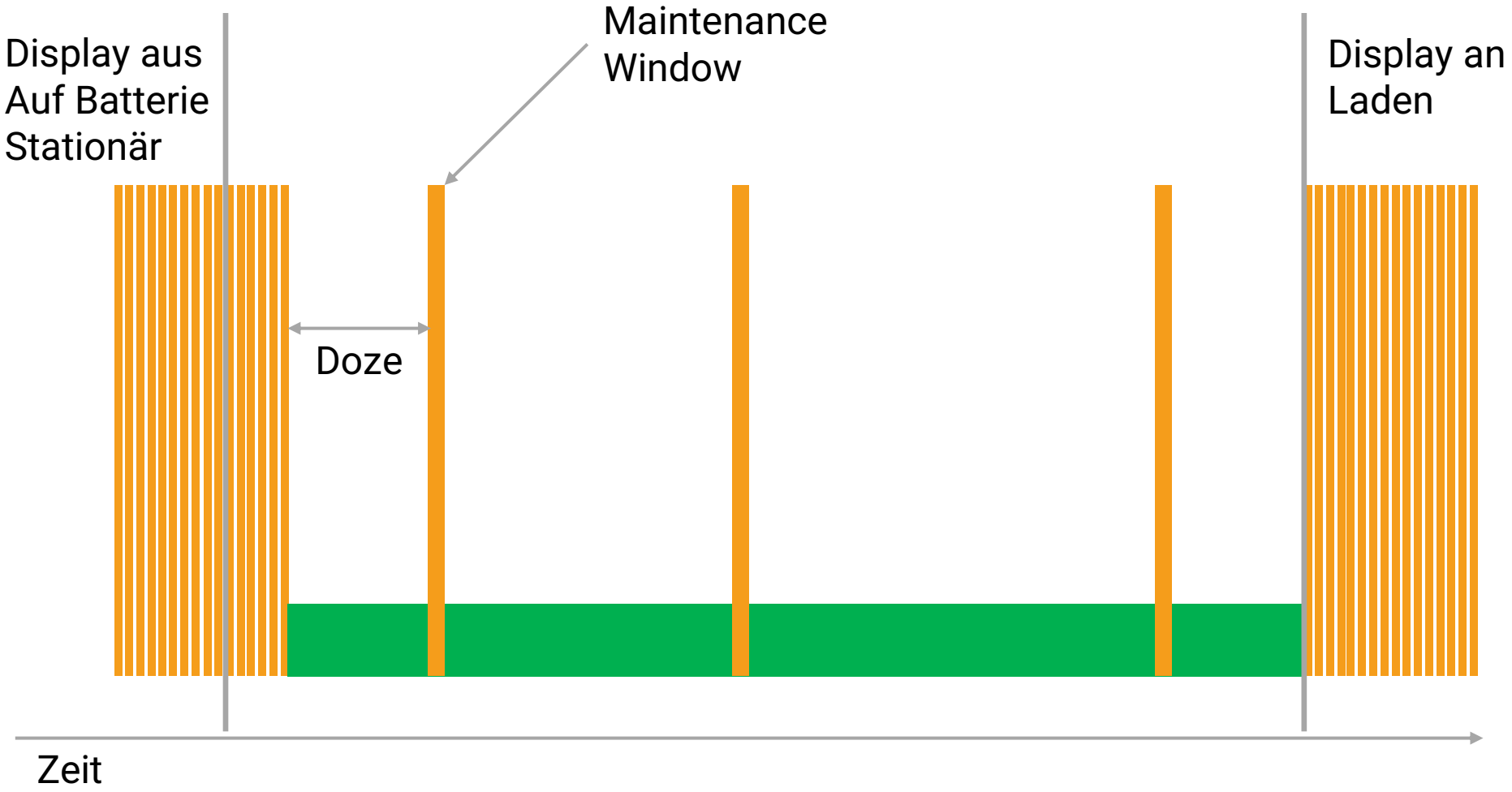
setAndAllowWhileIdle(int type, long triggerMillis, PendingIntent op)

setExactAndAllowWhileIdle(int type, long triggerMillis, PendingIntent op)

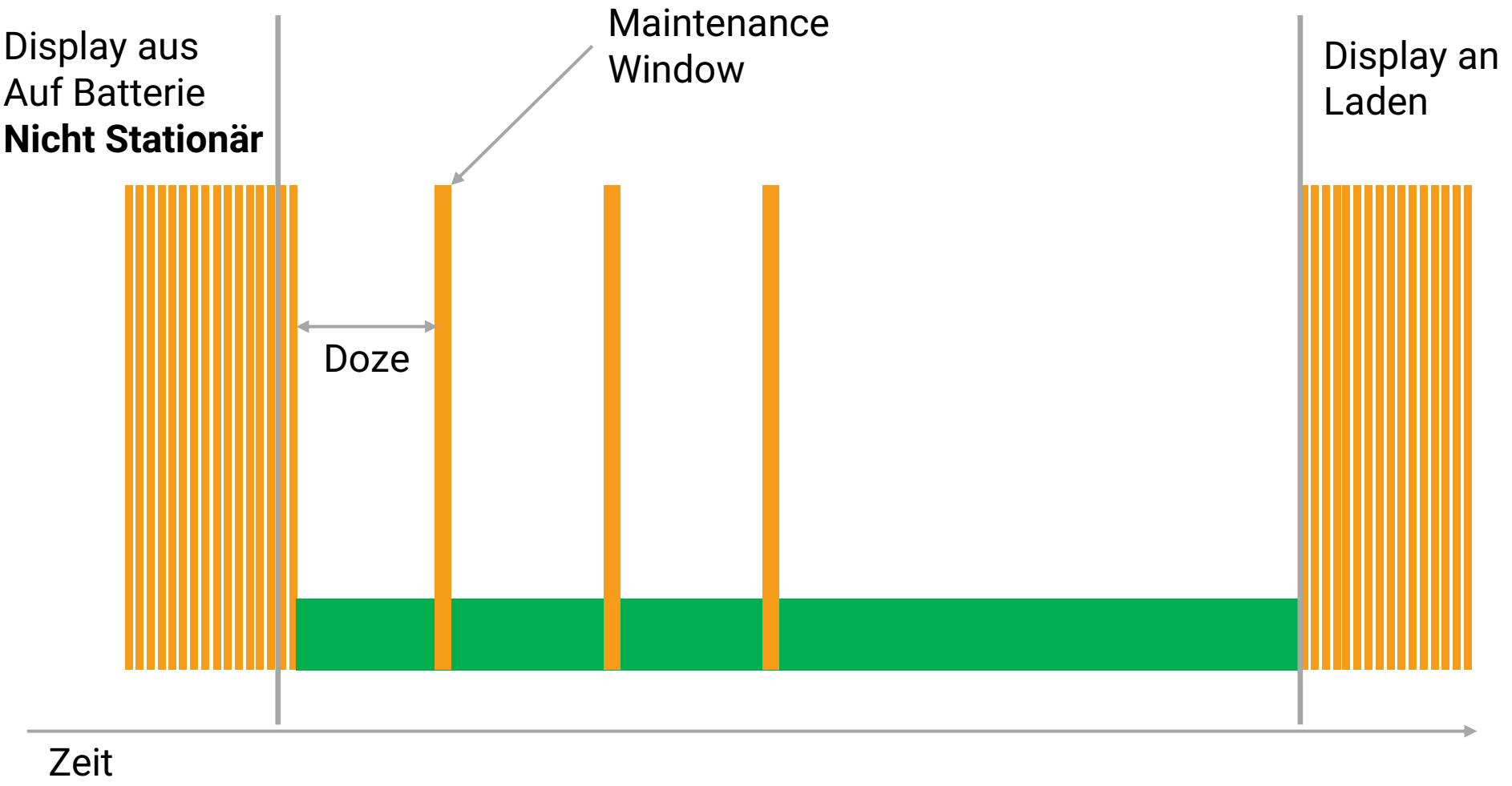
Android Doze Mode

- Neuer Energiesparmodus
- Einschränkungen:
  - Zugang zu Netzwerk und CPU intensiven Services
  - Zugang zum Netzwerk allgemein
  - Alarme, Jobs und Syncs werden verhindert
  - Wakelocks werden ignoriert
  - Wi-Fi Scans werden nicht durchgeführt

Android Doze Mode



Android Doze Mode



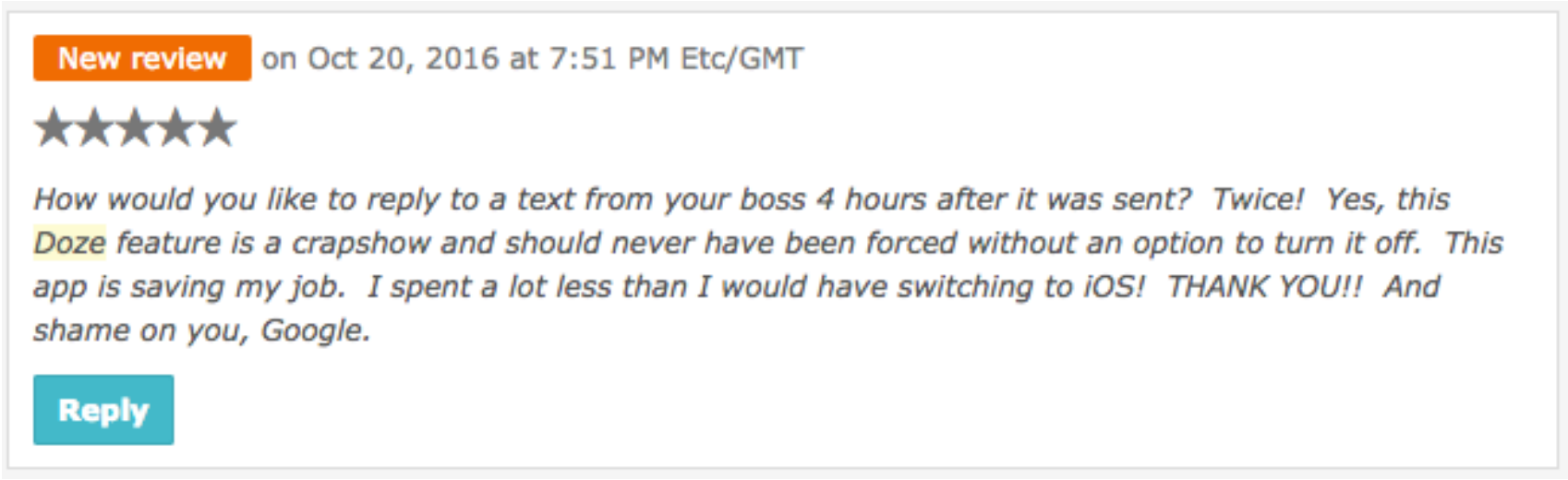
Android App Standby

- Energiesparmechanismus für selten benutzte Anwendungen
- Bedingungen:
  - Nutzer hat die App nicht explizit gestartet
  - App hat keinen Prozess im Vordergrund
  - App generiert keine Benachrichtigung au dem Lockscreen oder der Benachrichtigungsleiste
- Im Batteriebetrieb bekommen Apps im App-Standy ca. 1x am Tag Zugriff auf das Netzwerk

How to break the doze?

Why breaking the doze mode?

- Steigert die Akkulaufzeit spürbar
  - Verfeinerung mit Android Nougat (Light und Deep Doze Mode)
- Änderung der Funktionsweise von alten Apps



How to break the doze?

- Google bzw. Firebase Cloud Messages
  - High Priority Flag weckt Gerät zuverlässig
- Probleme:
  - Nicht immer nutzbar (bspw. Dt. Datenschutz)
  - Kein Internet führt ebenfalls zum nicht aufwachen

How to break the doze?

```
{
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
  "priority" : “high",
  "notification" : {
    "body" : "This week’s edition is now available.",
    "title" : "NewsMagazine.com",
    "icon" : "new",
  },
  "data" : {
    "volume" : "3.21.15",
    "contents" : "http://www.news-magazine.com/world-
week/21659772"
  }
}
```

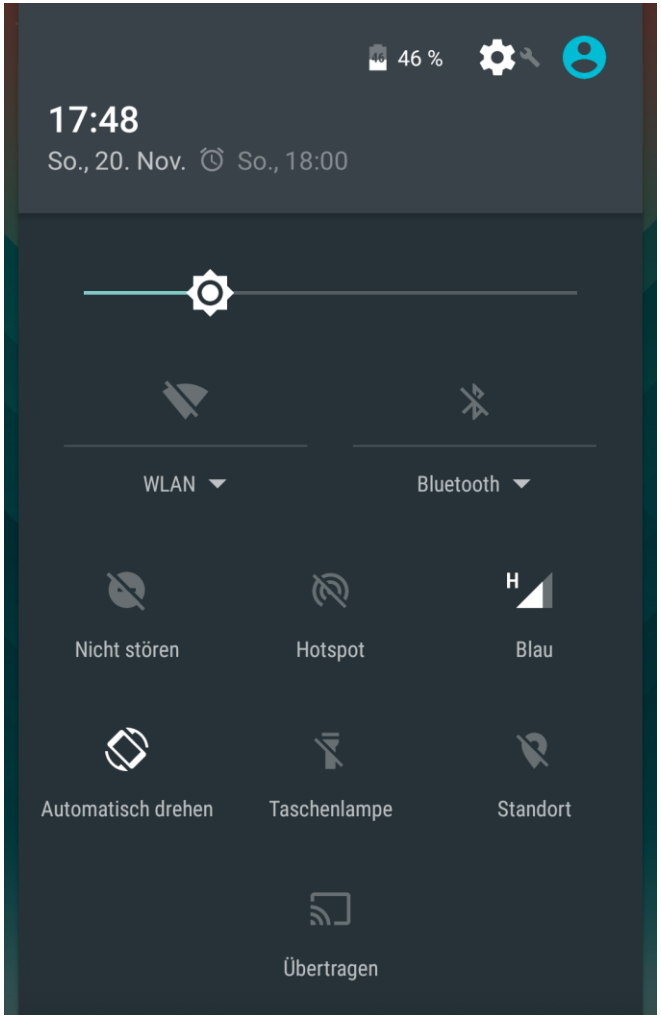
How to break the doze?

- Nutzen von den seit API-Level 23 eingeführten Methoden
  - setAllowWhileIdle() und setExactAndAllowWhileIdle()
  - Zugriff auf Systemressourcen für 10 Sekunden
- Probleme:
  - Maximal aller 9 Minuten
  - Wakelocks müssen sofort gehalten werden

How to break the doze?

- Nutzen von det seit API-Level 21 eingeführten Methode setAlarmClock()
  - Garantiert das Ausführen des Alarms
  - Gerät geht definitiv einige Zeit vor dem Alarm aus dem Doze Mode
- Probleme:
  - System zeigt Hinweis über diesen Alarm an

How to break the doze?

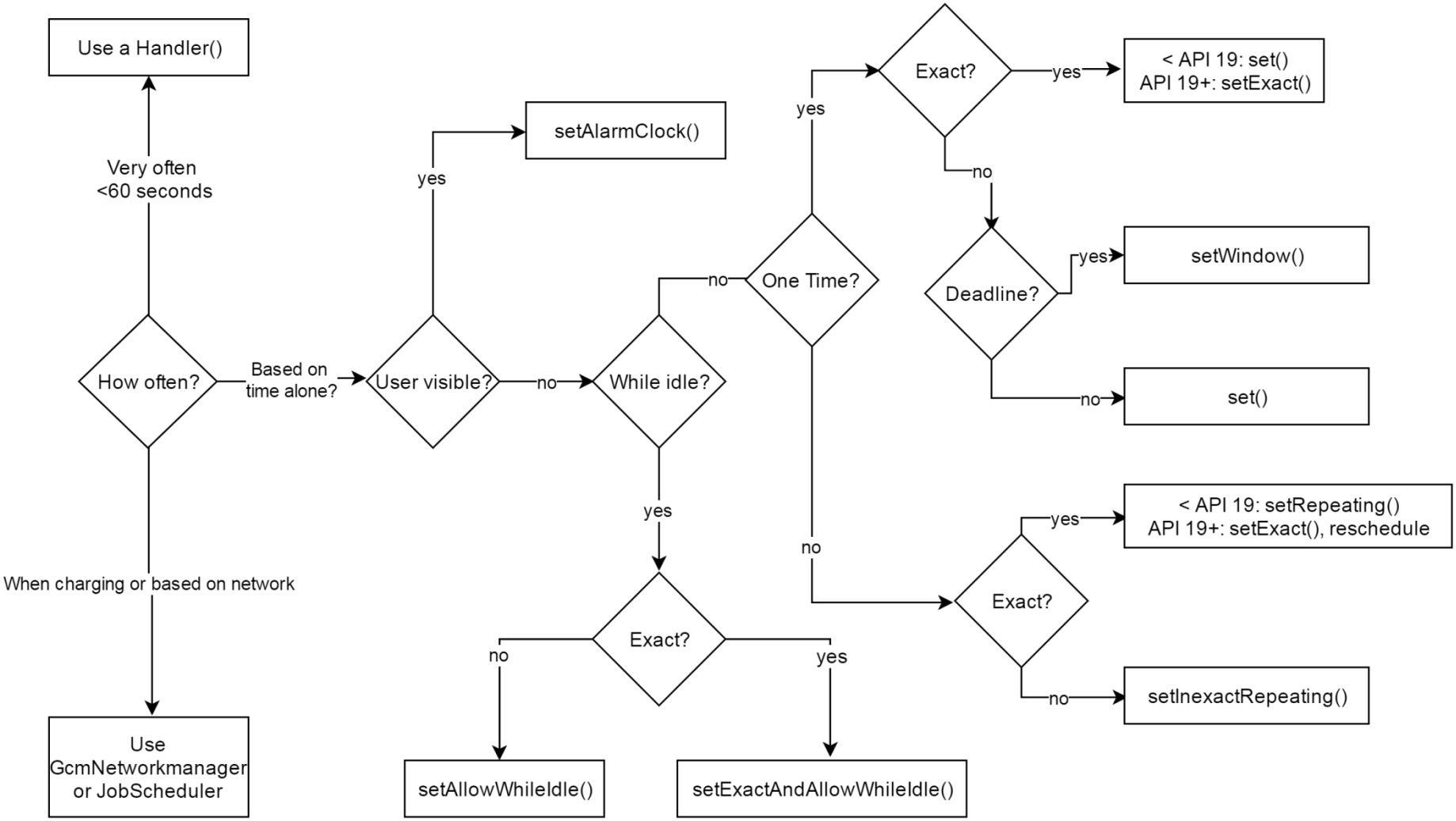


How to break the doze?

- Nutzen eines Foregroundservices
  - Laut Dokumentation soll es den Doze aufheben
  - Bug seit Android 6 verhindert das [1]
- Nutzen von Disable Battery Optimizations
  - Bedeutet sogenanntes Whitelisting
  - Hebt jedoch nicht alle Restriktionen auf
  - Evtl. Sperrung der Anwendung durch Google

[1] https://code.google.com/p/android/issues/detail?id=193802

Fragmentierung



Quelle: https://plus.google.com/+AndroidDevelopers/posts/GdNrQciPwqo?linkId=17112180

Fragmentierung

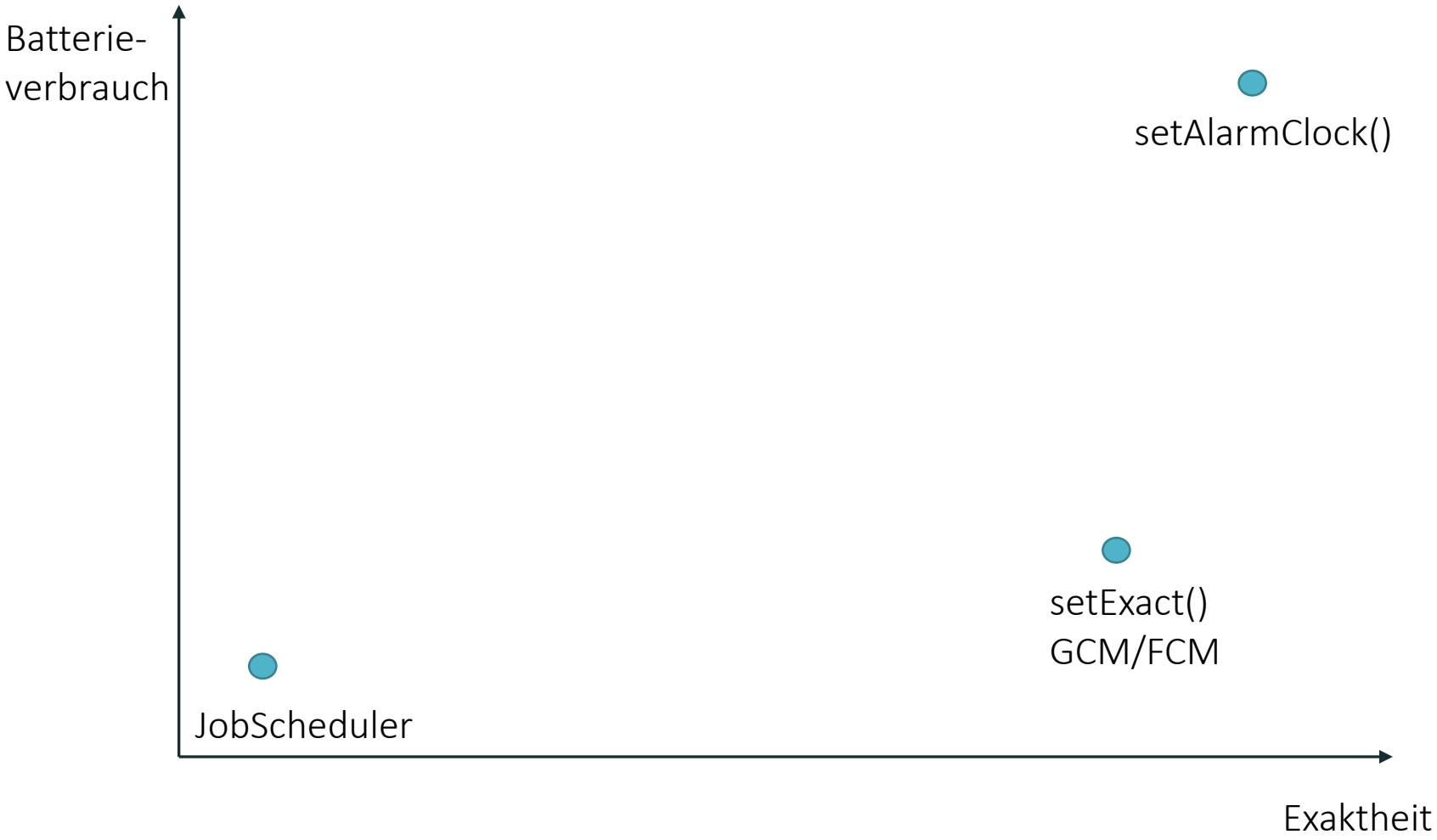
```
public void setExactAlarm(int type, long triggerAtMillis, PendingIntent operation) {
   if (Build.VERSION.SDK_INT < Build.VERSION_CODES.KITKAT) {
      alarmManager.set(type, triggerAtMillis, operation);
   } else if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
      alarmManager.setExact(type, triggerAtMillis, operation);
   } else {
      setAlarmForDozeDevices(type, triggerAtMillis, operation);
   }
}

@TargetApi(Build.VERSION_CODES.M)
private void setAlarmForDozeDevices(int type, long triggerAtMillis, PendingIntent op) {
   long trigger = triggerAtMillis;
   if (type == ELAPSED_REALTIME_WAKEUP || type == ELAPSED_REALTIME) {
       trigger = currentTimeMillis() + (triggerAtMillis - SystemClock.elapsedRealtime());
   }
   AlarmManager.AlarmClockInfo info = new AlarmManager.AlarmClockInfo(trigger, pi);
   alarmManager.setAlarmClock(info, op);
}
```

Best Practices

- Seit Android 6 müssen die Anforderungen frühzeitig bedacht werden
- Battery Life vs. Exaktheit
  - Selten Anforderungen an absolute Genauigkeit (*AllowWhileIdle* oder *AlarmClock*)
  - Hintergrundsyncing über JobScheduler oder GCMNetworkManager

Best Practices



Fazit

- Doze Mode führt zu großer Fragmentierung
  - AlarmManager funktioniert nicht mehr
  - Keine Konfidenz in Fortbestehen des Zustands
- Es handelt sich jedoch sehr oft um Spezialfälle
  - Wecker, Sensor oder Automatisierung
- Anpassung ist extrem wichtig

Weitere Lösungen

- AlarmManagerCompat
  - Backport des ursprünglichen Verhaltens
  - https://github.com/rzetzsche/AlarmManagerCompat
- Disable Doze
  - Anwendung welche den DozeMode deaktiviert
  - https://play.google.com/store/apps/details?id=de.gerooo.disabledoze&hl=de

Danke für die Aufmerksamkeit!

Robert Zetzsche
Movisens GmbH
zetzsche@movisens.com
Twitter: @r_zetzsche
Github: @rzetzsche

Karlsruhe, 21.11.2016

Weitere Informationen

- Blogbeiträge
  - http://pguardiola.com/blog/darealfragmentation-sensors
  - http://pguardiola.com/blog/darealfragmentation-doze
  - http://pguardiola.com/blog/darealfragmentation-alarms
  - https://www.bignerdranch.com/blog/diving-into-doze-mode-for-developers/
- Google
  - https://developer.android.com/reference/android/app/AlarmManager.html
  - https://developer.android.com/training/monitoring-device-state/doze-standby.html