Developing Mobile Device Management for 15 million devices (case study)

Rim KHAZHIN

whoami

- software architect @ btt ltd
- space technologies research institute
- Ericsson mobility world
- underwater photographer

why am I giving this talk?

- share our research
- describe undocumented Android
- share experience

plan of this talk

- what is MDM?
- managing Android:
  - using API functions
  - rooted
  - with vendor support
- optimization

Mobile Device Management system

- purpose: controlling device(s)
- typical features:
  - profile delivery: wifi pass, b/w list, email, vpn
  - policy: password strength, camera disabled
  - application control

in two words:

- restricts
- controls

“Fatih” project

- ordered by Turkish Ministry of Education
- 15 million devices delivered at 30k public schools
- free wifi Internet to all public schools
- running since mid 2013

about this project

“Fatih” project requirements

- deliver and manage 15 million devices
- control applications
- control hardware
- manage by profile, location, group

btt

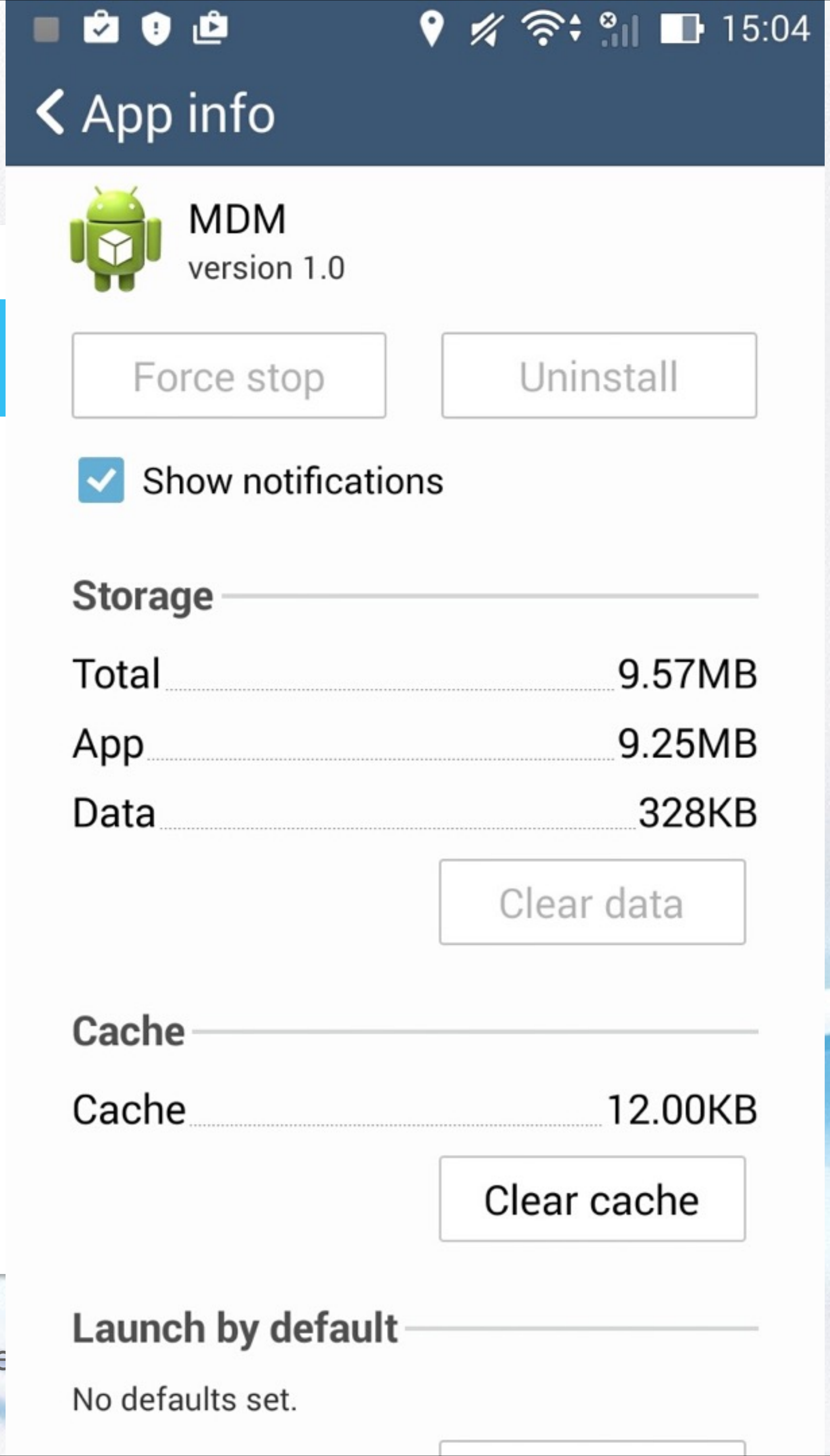
design

MDM application design

- MDM is an app
- protect MDM
    - undetectable, unstoppable, unremovable
    - prevent rooting
    - detect if rooted

make app unstoppable?!

- device administration permission
  - app is unstoppable!
  - and unremovable!



App info

MDM
version 1.0

Force stop | Uninstall

✓ Show notifications

Storage

Total	9.57MB
App	9.25MB
Data	328KB

Clear data

Cache

Cache	12.00KB

Clear cache

Launch by default

No defaults set.

device administration API

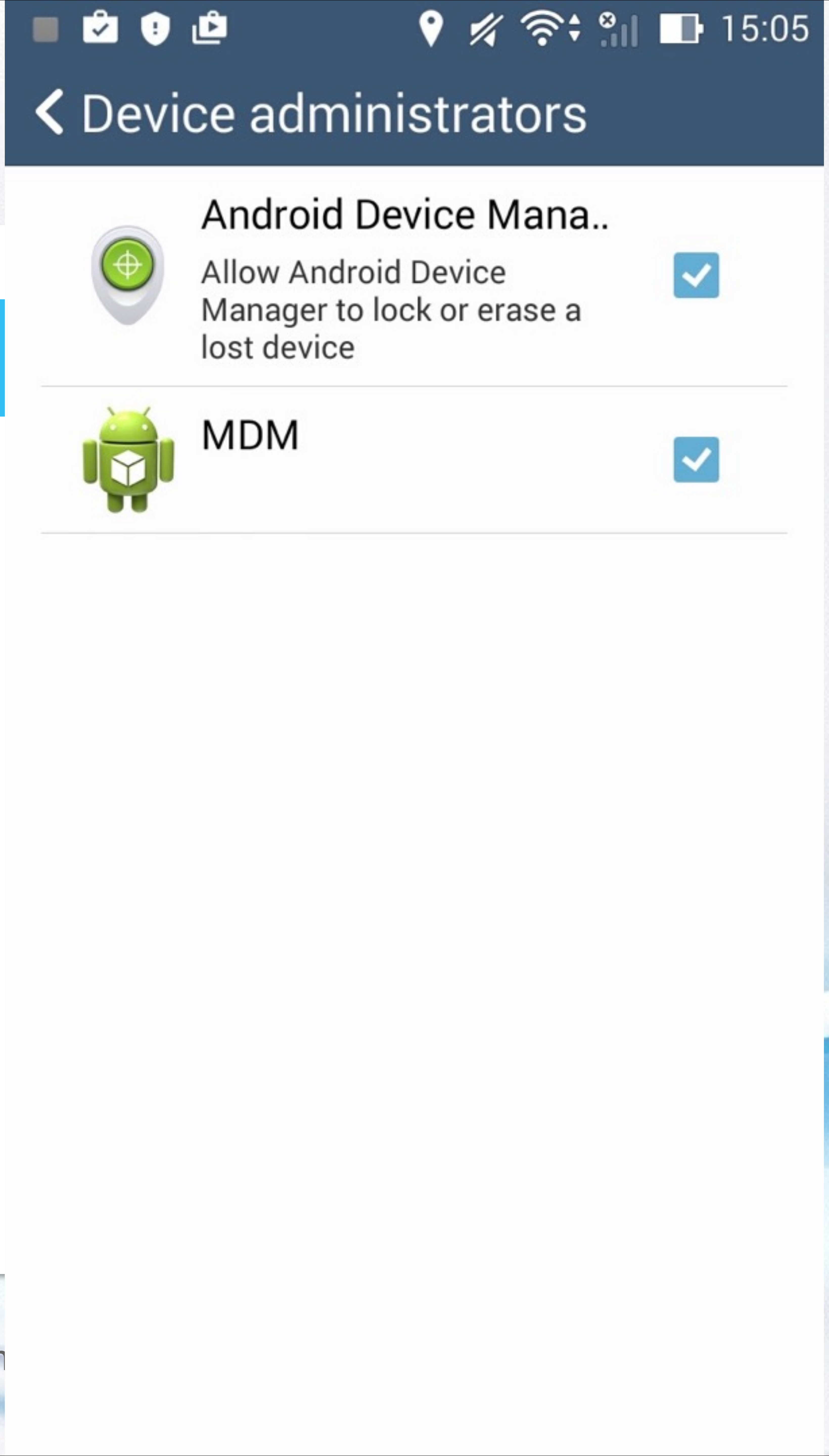
- password strength policy
- set new password
- lock, wipe, encrypt, disable camera

getting device administration permission

- bind BIND_DEVICE_ADMIN permission
- extend DeviceAdminReceiver
- listen to ACTION_DEVICE_ADMIN_ENABLED intent

security -> device administrators

- view device administrators
- remove permission


15:05

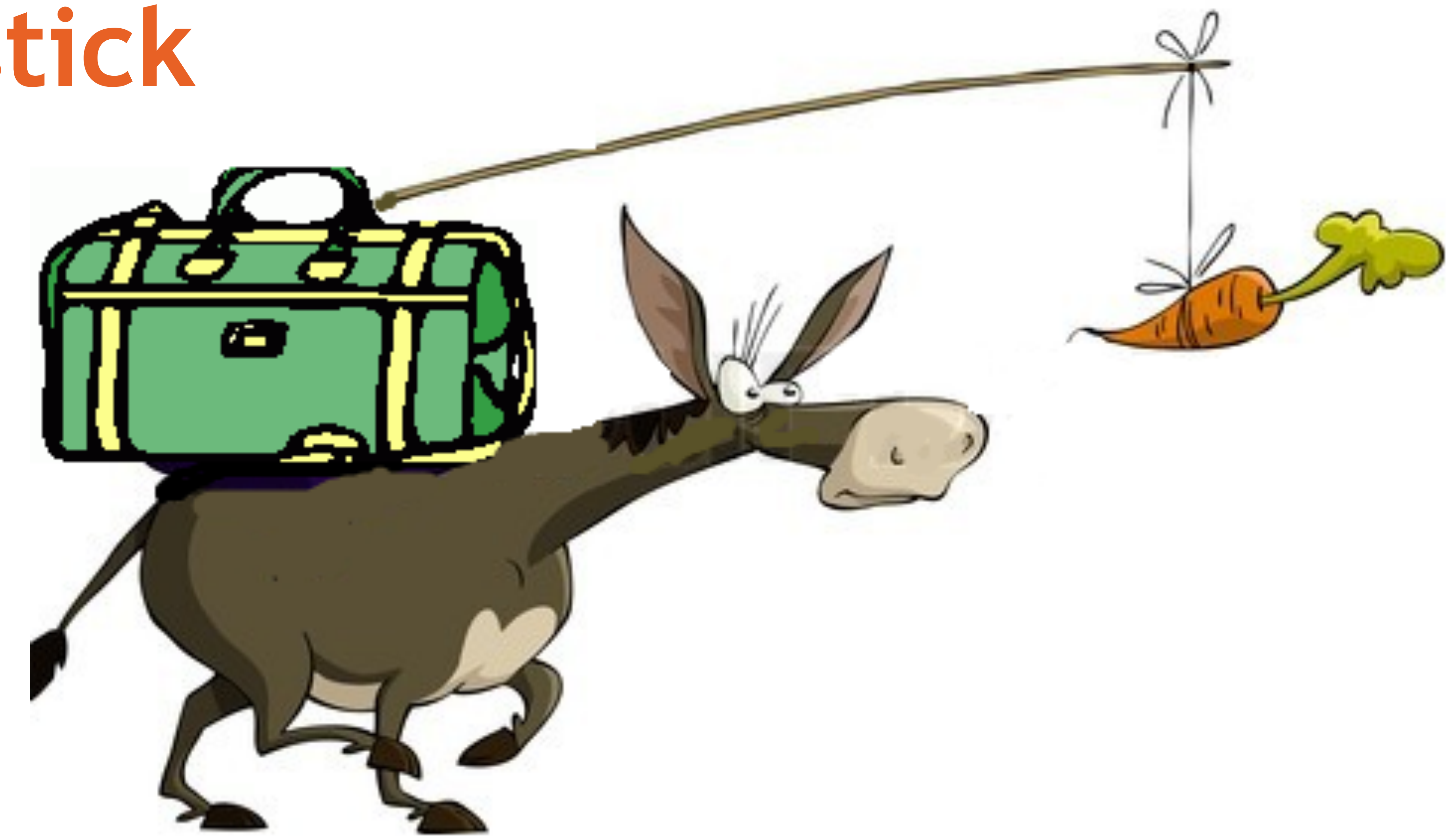
Device administrators

Android Device Mana..
Allow Android Device Manager to lock or erase a lost device ✓

MDM ✓

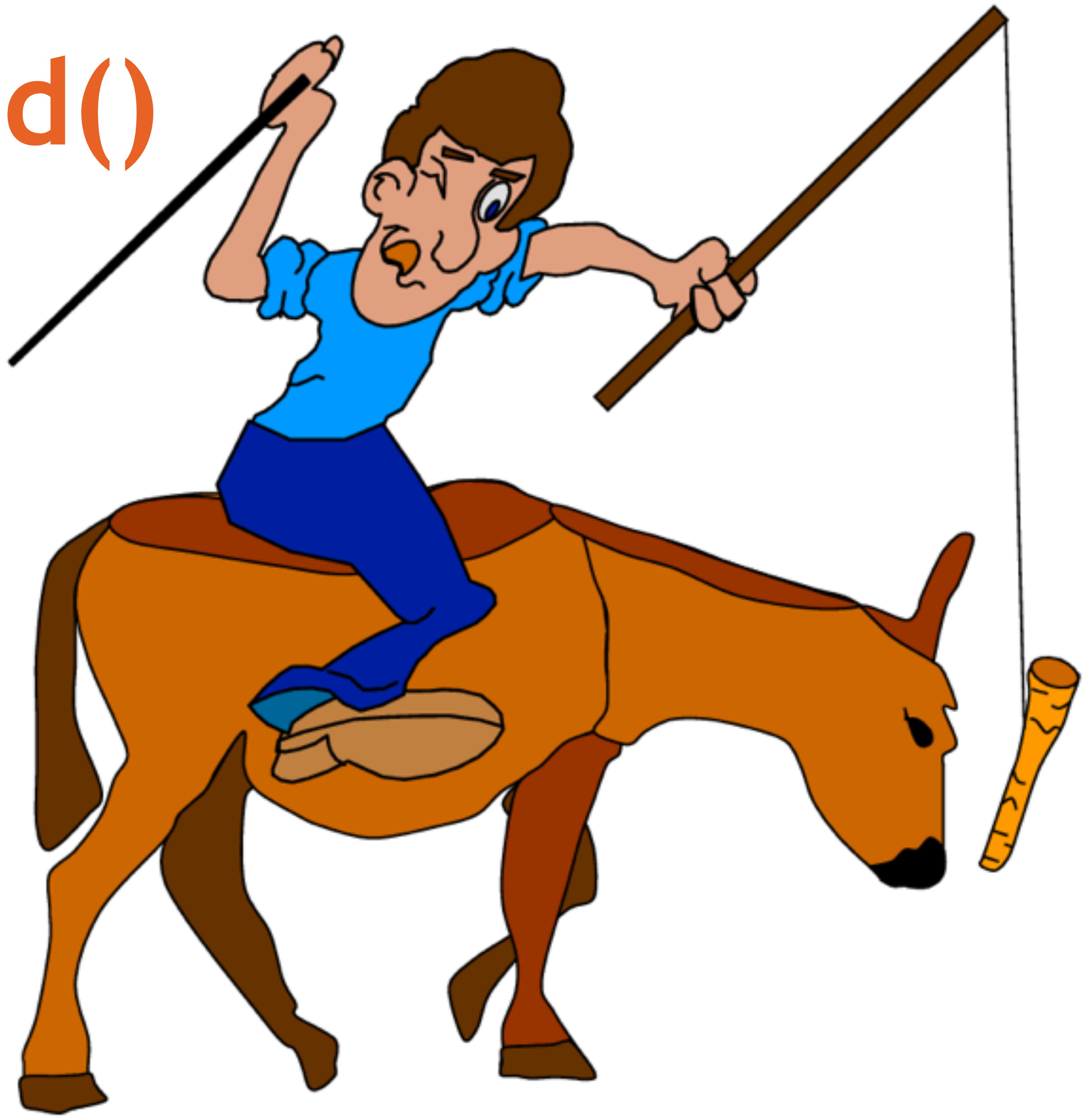
prevent removing admin permission

- offer carrot on a stick
    - wifi settings
    - email account
    - vpn settings



if permission removed!

- DeviceAdminReceiver.onDisabled()
  - disable accounts
  - show warning
  - notify system administrator



prevent removing admin permission

- use custom launcher
- what is “launcher”?

custom launcher

- an application
- device home screen
- lists and launches other apps
- keyword: lists and launches

use custom launcher to:

- show only allowed apps
- hide settings app
  - show your own modified Settings

developing a launcher

- **Intent filter**

```
<intent-filter>
    <category android:name="android.intent.category.HOME" />
    <category android:name="android.intent.category.DEFAULT" />
```

- **full-screen app**

```
<activity
   android:theme="@android:style/Theme.Wallpaper.NoTitleBar.Fullscreen"
```

making launcher default

- click home button
- select your launcher
  - tick “Use by default for this action” checkbox

application management

- list apps using PackageManager
  - every app is not runnable
  - getLaunchIntentForPackage()
- launch apps

application management

- **installing apps**

```
Intent promptInstall = new Intent(Intent.ACTION_VIEW)
    .setDataAndType(Uri.parse("file:///RestaurantMenu.apk"),
                        "application/vnd.android.package-archive");
```

- **deleting apps**

```
Intent intent = new Intent(Intent.ACTION_DELETE);
intent.setData(Uri.parse("package:com.facebook.messenger"));
```

remember the carrots

- don’t restrict too much
- give good carrots:
  - wifi access. Don't give the password!
  - corporate accounts: disable account if MDM gets removed

hard-core Android

but how do we REALLY control the device?

- unremovable
    - system application
- undetectable
    - core application

what is a system application?

- runs with system UID

```
USER      PID   PPID  VSIZE  RSS    NAME
root       1     0     888    740   /init
root       2     0     0      0     kthreadd
root       157   1     883620 45152 zygote
keystore   163   1     4712   1048  /system/bin/keystore
radio      871   157   920240 31748 com.android.phone
bluetooth  886   157   896776 21828 com.mediatek.bluetooth
system     901   157   903968 29880 com.btt.mdm
u0_a8      923   157   954192 33456 com.android.launcher3
u0_a2      974   157   905620 25408 com.android.contacts
```

developing a system application

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    coreApp="true"
    package="com.btt.mdm"
    android:sharedUserId="android.uid.system">
```

- core application
- use system privileges

permissions

```
<permission android:protectionLevel=["normal" | "dangerous" |
                    "signature" | "signatureOrSystem"] />
```

"signatureOrSystem" A permission that the system grants only to applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission.

obtaining the permissions

- “System” permission
  - put app to system ROM
  - /system/app/
- “Signature” permission:
  - sign the app with platform key

prevent removing Device admin permission?

- disable settings menu
  - compile Settings from sources
  - mind vendor-specific features

how to control hardware?

- disable bluetooth use
  - for ALL apps!
- disable external memory card, wifi, …
  - for ALL apps!
- no functions to disable camera for ALL apps!

reverse-engineer Android

- android source
  - http://source.android.com/source/building.html
- find settings application source
  - android-source/packages/apps/Settings/
- find bluetooth control functions

hardware control functions

- **camera control**

```
SystemProperties.set(SYSTEM_PROP_DISABLE_CAMERA, value);
```

- **bluetooth control**

```
BluetoothAdapter mBluetoothAdapter =
    BluetoothAdapter.getDefaultAdapter();
mBluetoothAdapter.disable();
```

these functions:

- undocumented
- hidden
- unavailable
- modify kernel-level params

app is compiled. now what?

- root the device
- manufacturer’s support

what do we get from manufacturers?

- Android (Samsung, LG, General Mobile, etc)
  - MDM API
- Sony Open Devices
- Apple MDM
  - built-in

MDM API

- Hardware control
- Application management
  - Install application (silent)
  - Remove application (silent)
- Control submenus of Settings

procedure

- join Enterprise Developer Program
- get you app signed by vendor
- security check

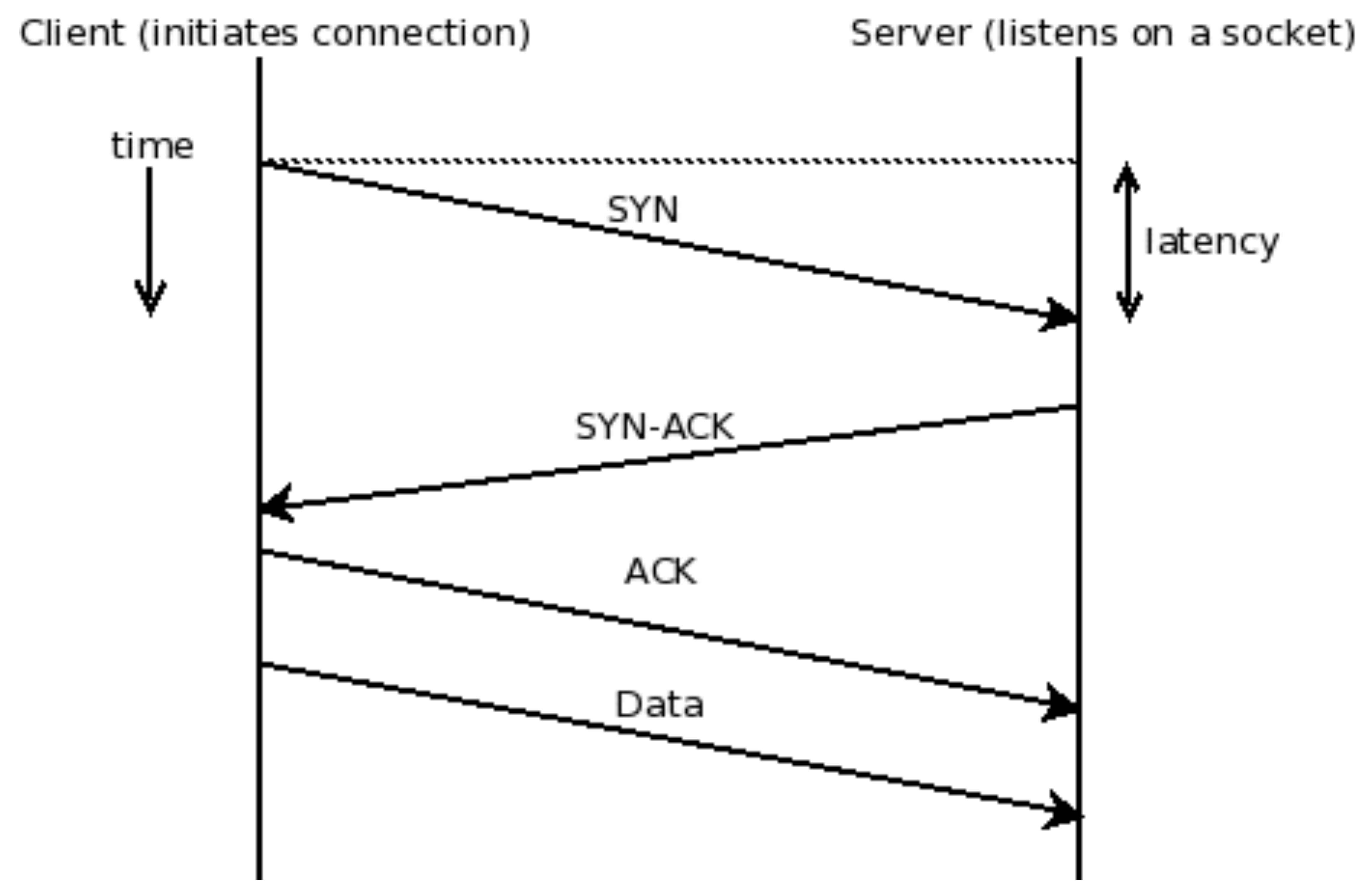
scaling

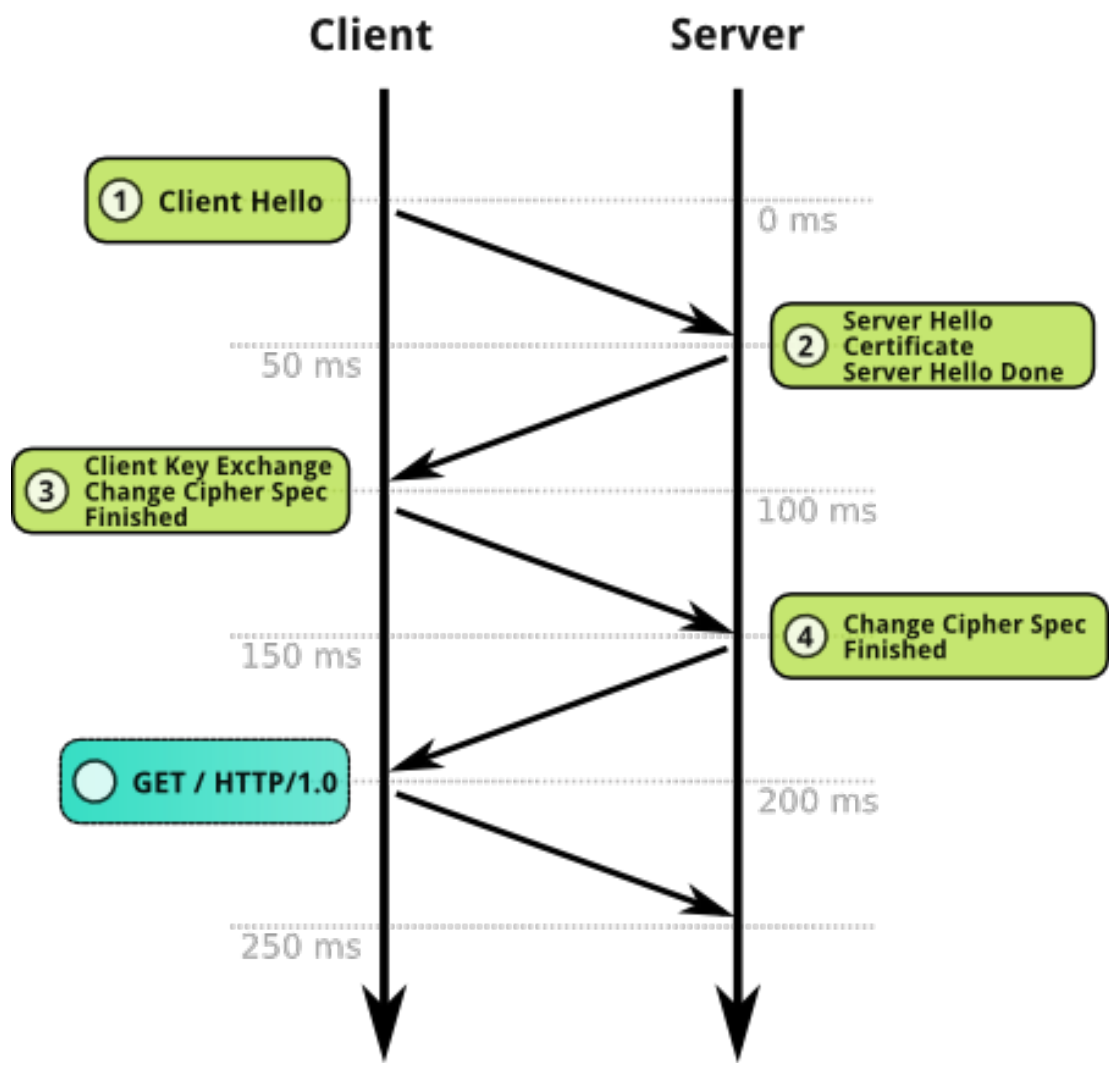
app requirements

- low battery
- low bandwidth
- low latency

how do you do this?

- minimal number of transactions


Client (initiates connection)
Server (listens on a socket)
time
SYN
latency
SYN-ACK
ACK
Data


Client
Server
1 Client Hello
0 ms
2 Server Hello
Certificate
Server Hello Done
50 ms
3 Client Key Exchange
Change Cipher Spec
Finished
100 ms
4 Change Cipher Spec
Finished
150 ms
GET / HTTP/1.0
200 ms
250 ms

server side

- memory
- CPU
- network bandwidth
- example: 15 million devices sending 1KB each

how to reduce?

- few requests
- small packets
- Google spdy protocol
  - faster!
  - great for poor network!

optimize network operations

- handle connection exceptions
- random wait period
  - use AlarmManager, set PendingIntent
  - setInexactRepeating()
- limited retry count

optimize network operations

- **check the connectivity status**

```
ConnectivityManager cm =
    (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
```

- **listen to network connection changes**

```
<action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
```

sending commands to device

- send push notification
- device connects over HTTPS
- verify SSL certificate

server optimization

microservices

- separate service for each function:
  - send message
  - send ‘like’
  - upload image
  - get messages

microservice workflow

- parse and validate message
- authenticate user
- no business logic

background services

- sending push notifications
- sending emails
- resizing images
- processing video

database optimization

- stored procedures
  - speed
  - security
  - business logic

debugging (shared) production server

- identify single request out of millions of requests?
- log all requests
    - turn on when needed
    - for a single microservice
    - turn off when done

testing is important!

- what happens if 1% of 100 customers complain?
- what happens if 1% of 15mln customers complain?
- is bug-free software possible?
- well-tested software is

conclusion

- android administration
- scaling
- optimization
- don't over-engineer!
- release the app

questions?

http://google.com/+RimKhazhin