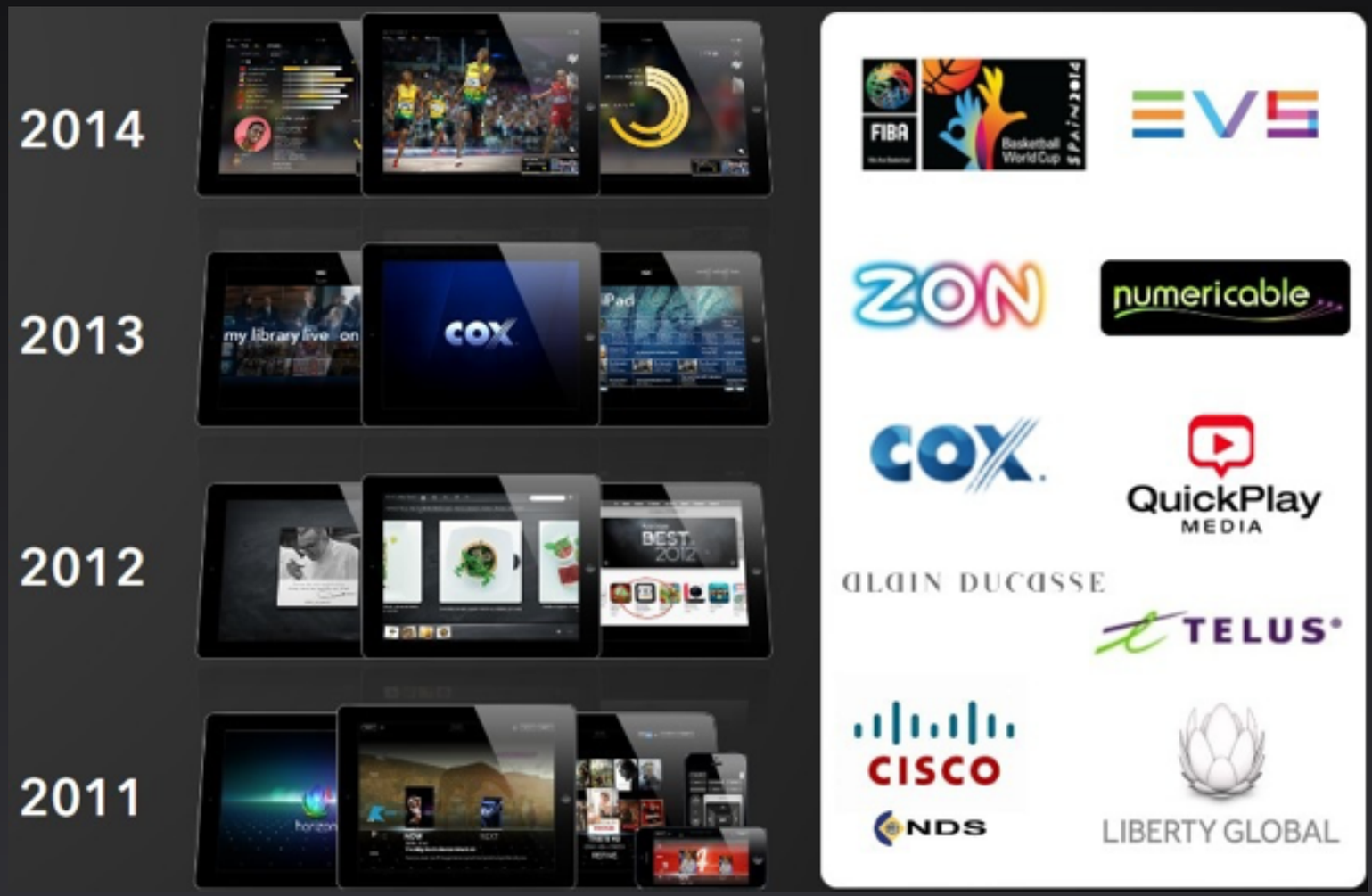
# iBeacons

kinan.arnaout@intellicore.net - Octobre 2015

Intellicore
INTERACTION DESIGN

Intellicore
INTERACTION DESIGN

- www.intellicore.tv




- http://www.intellicore.tv
- http://playrz.com/
- iBeacon Solutions

# Content

- What is an iBeacon?
  - Bluetooth LE
  - How does it work?
  - Examples / FAQ
- Let’s practice
  - Compatible devices
  - iBeacon API

# What is an iBeacon?

# Bluetooth LE

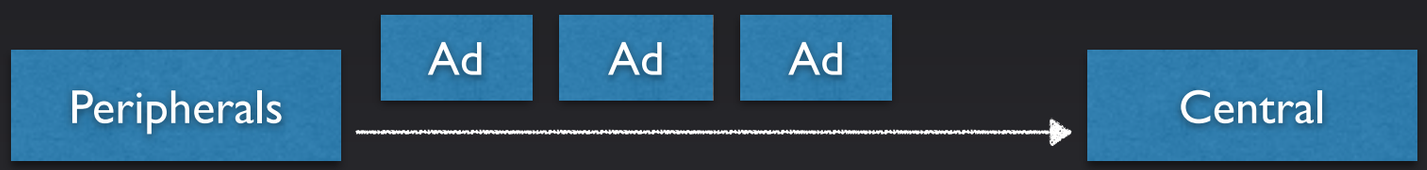
- Bluetooth Low Energy / Bluetooth Smart
- Invented by Nokia in 2006 (Wibree)
- Wireless personal area network technology used for transmitting data over short distance
- Bluetooth version 4.0
- Cardio & temperature monitors / smart watchs / etc.

# Bluetooth LE

- No pairing
- Short connection time
- Cheaper (60-80%)
- Large number of slaves
- Lower data transfer speed
- Low power consumption to detect and broadcast (up to 3 years with simple coin battery cell)

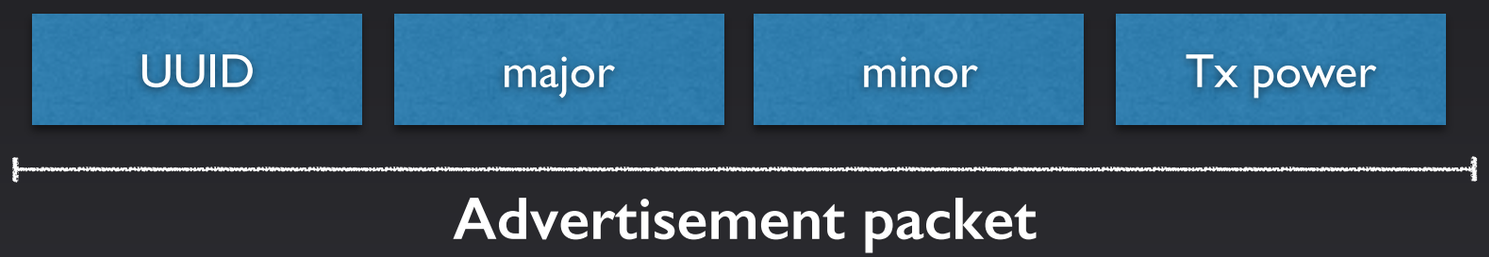
# Bluetooth LE

- Advertisement : small data packet
- Broadcast at regular interval
- One way communication



# How does it work?

- Standard for BLE advertising
- 4 main pieces of information
- Devices as peripherals or central

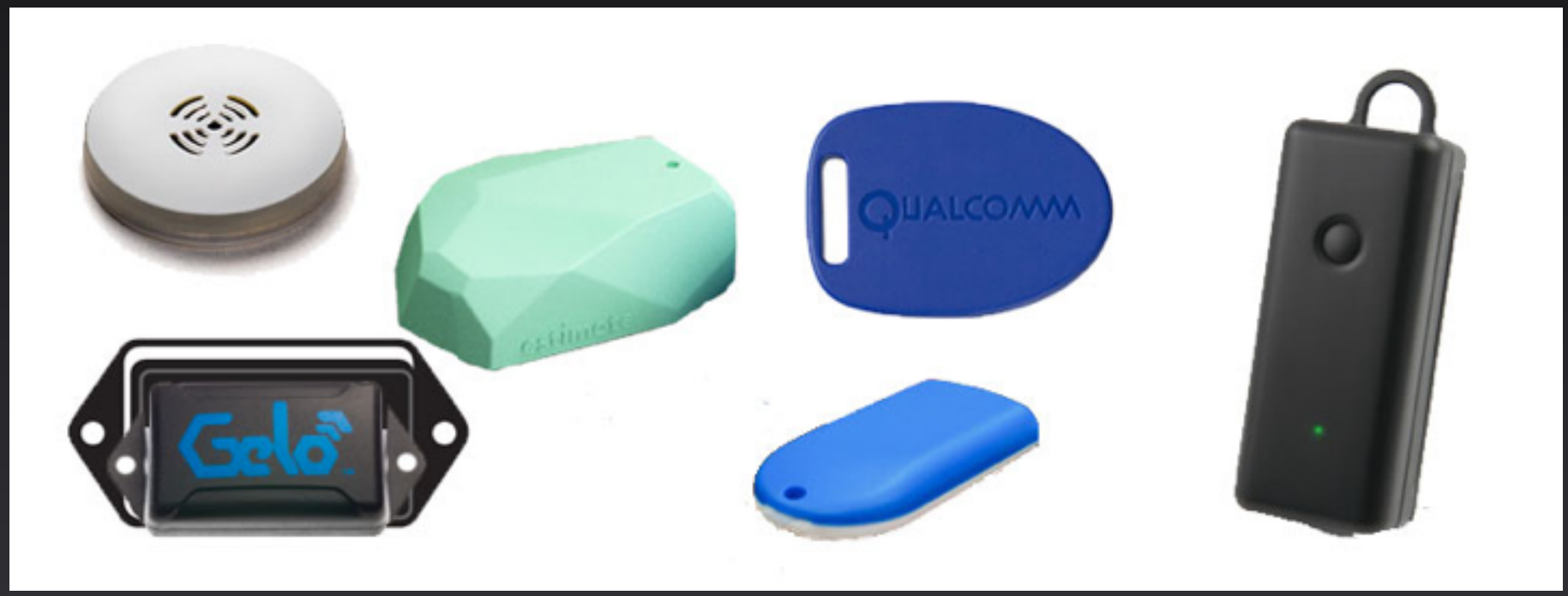


# How does it work?

- Scan for beacons identified with the same UUID
- Detect if the device is in the region of one or more beacons (~50m)
- Determine the proximity of a beacon (ranging)
- Differentiate beacons with minor and major

# How does it work?

- Chipset implementing bluetooth LE and iBeacon standard
- Embedded in small devices



# How does it work?

- Indoor positioning system
- Geofencing
- Location awareness for apps
- Deliver contextual content to user based on location

# How does it work

- Using advertisement packet information

  - UUID : 16 bytes string identifying a large group of related beacons
  - Major : 2 bytes number describing a subset of beacons
  - minor : 2 bytes number to identify a particular beacon
  - Tx power : determine proximity (distance)

# How does it work

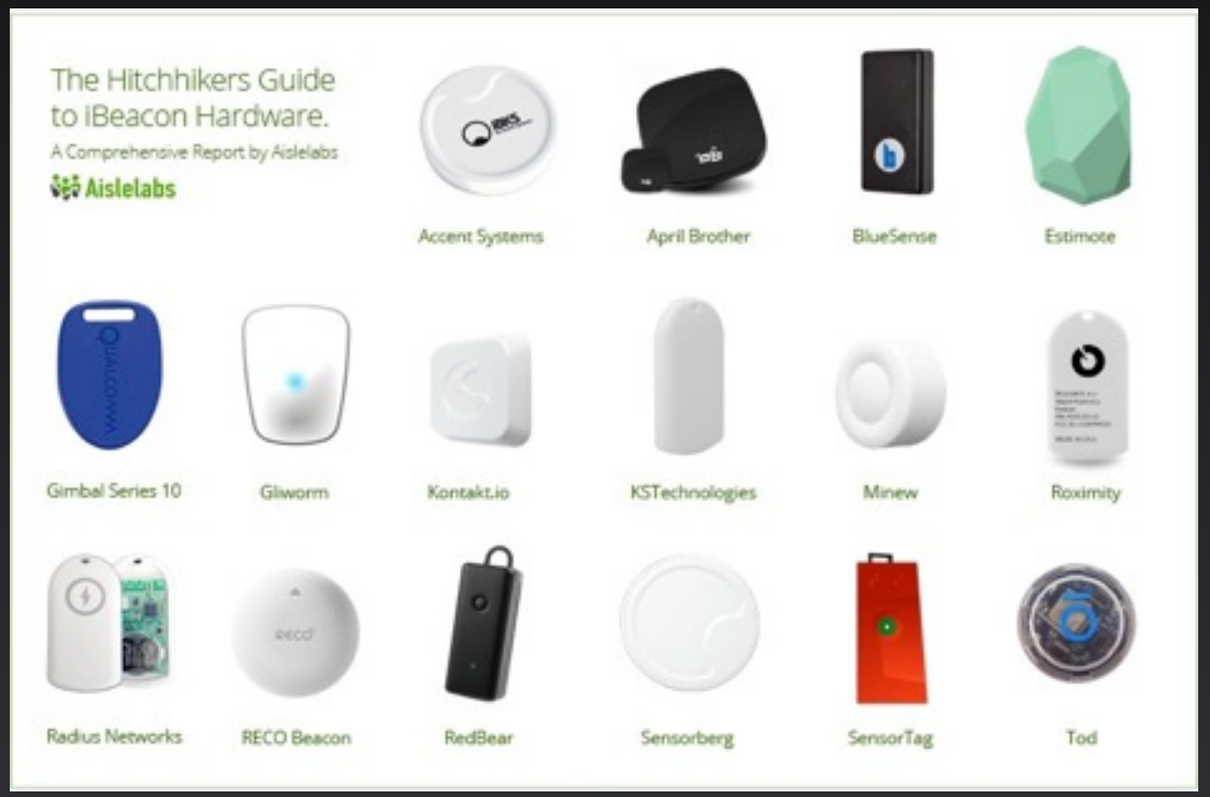
- 3 kinds of proximity (distance)
  - immediate
  - near
  - far

# To sum up

- Bluetooth device any app can discover if it knows the right UUID
- A set of beacons can describe a region
- Each beacon provides major, minor and proximity

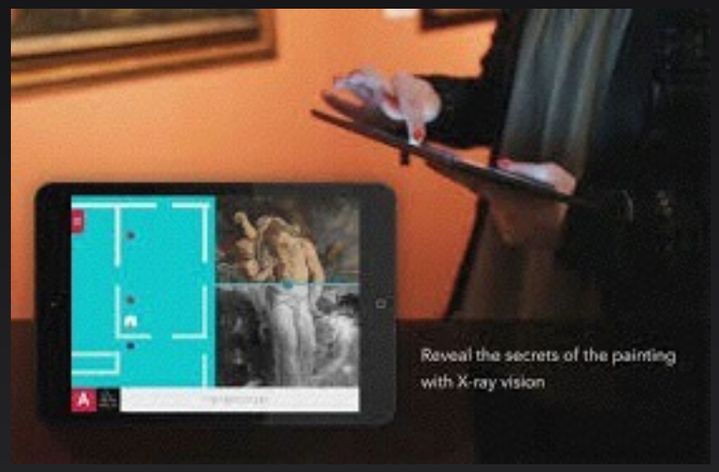
# iBeacons benchmark

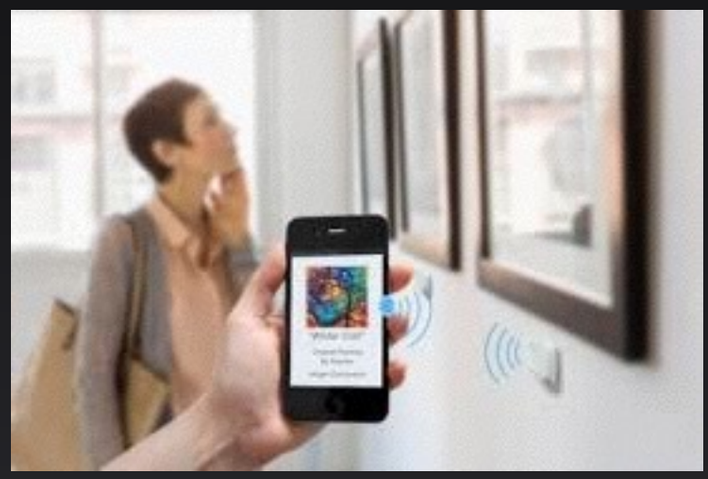
- http://www.aislelabs.com/reports/beacon-guide/



# Examples

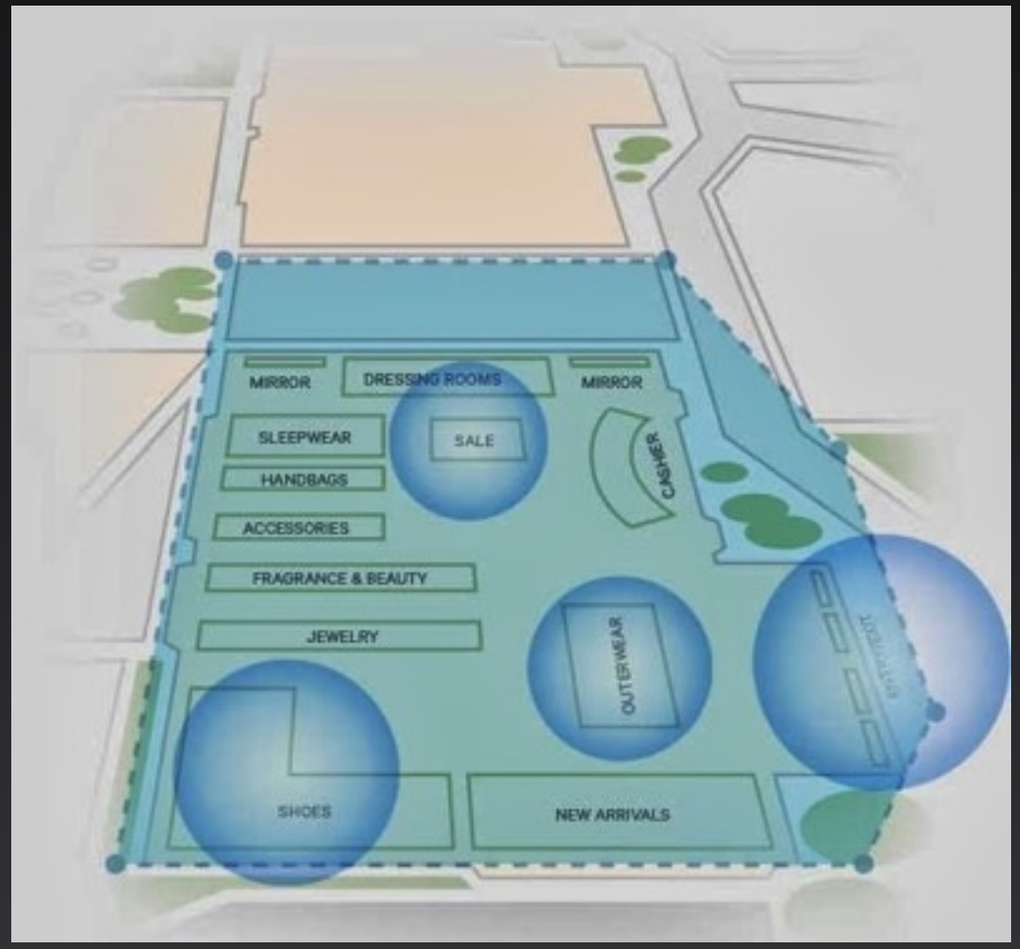
- Museums

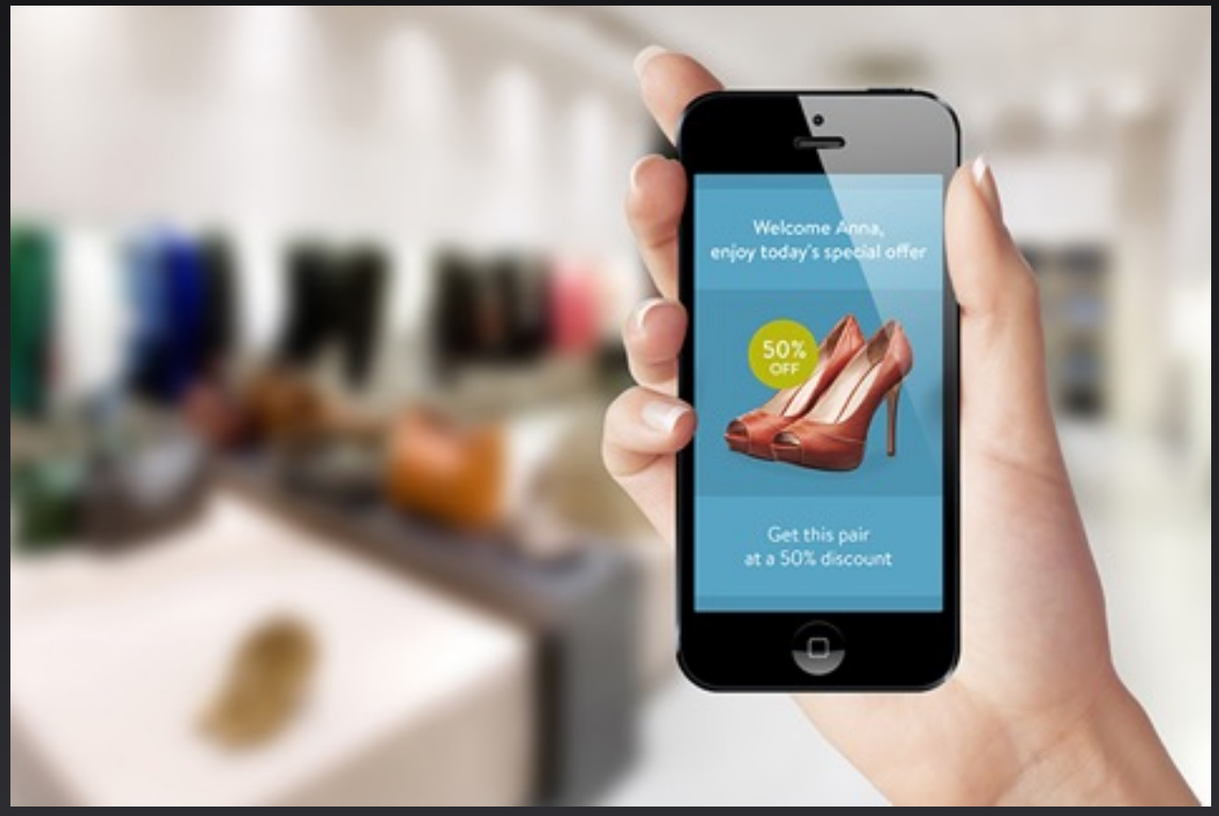



# Examples

- Stores



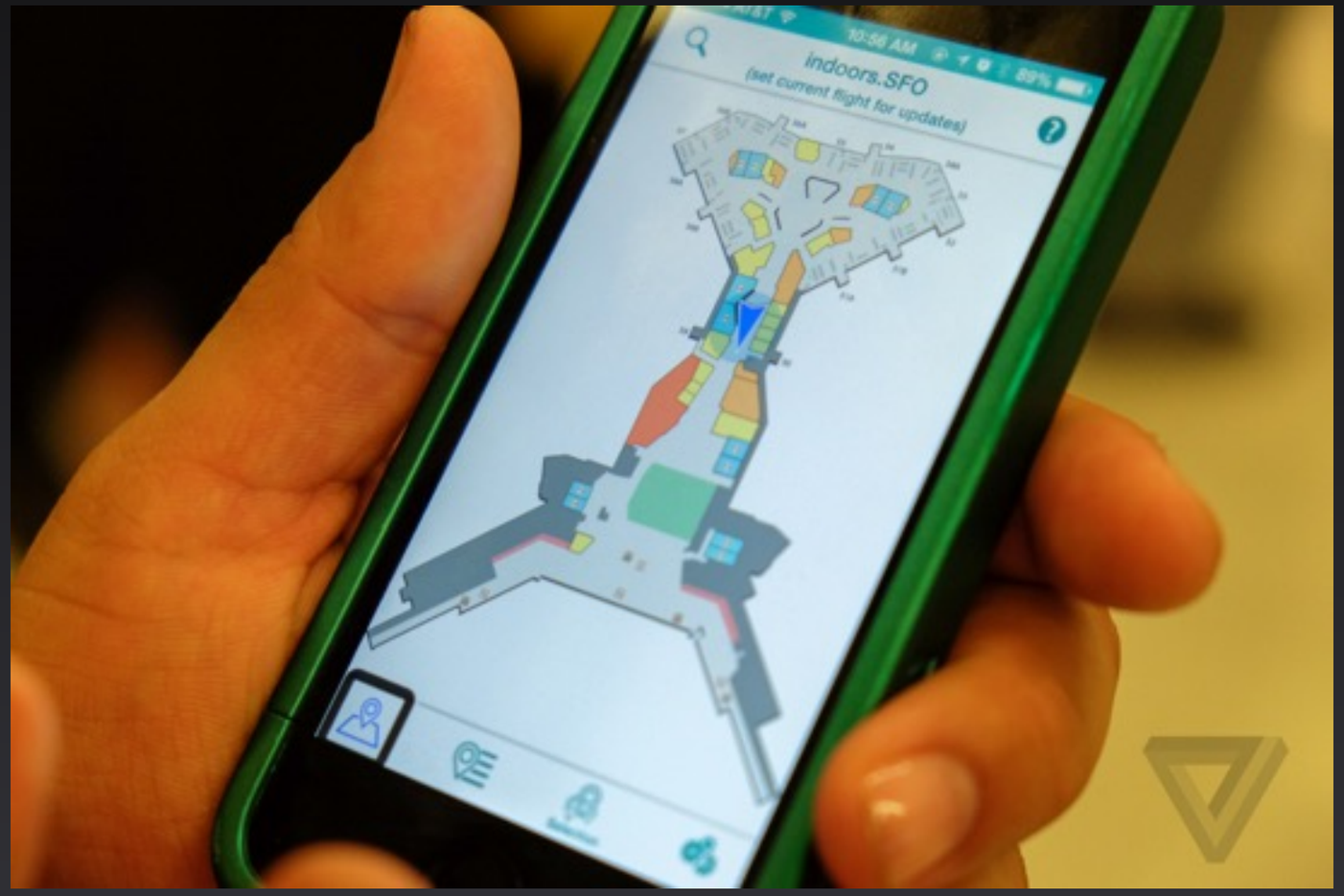



# Examples

- Stadiums

# Examples

- Airports



# Pros & cons

- Barrier to wide adoption : customers have to turn on bluetooth, accept location services on the relevant app and opt-in to receive in-store or indoor notifications
- Can not detect iBeacon without UUID
- No precise location (more accurate than GPS though)
- Indoor location
- Clean API paired with GPS in CoreLocation (and Apple full support)
- Cheaper hardware

# FAQ

- A beacon can be precisely located :

  Wrong : signal strength and environment factors
- Beacon can push information :

  Wrong : the app receive major/minor (had to retrieve data from server or in app database)
- Beacon can be detected in background :

  True : the app will detect if the device enters or exits a location described by beacons (latency)
- Other platforms :

  True

# Let’s practice

# Compatible devices

- Devices compatible with Bluetooth 4.0
- iOS devices from iPhone 4S, new iPad, iPad mini, ...
- Android 4.3 (Nexus 4/5/7, Samsung galaxy S3/S4/S5/Note2/Note3, Optimus G/G2, HTC One/Butterfly, etc.)

# iBeacon API

- A layer on top of CoreBluetooth exposed by CoreLocation
- Monitoring & ranging
- Background detection
- Display notification

# iBeacon API

- Ranging beacons
- Device as a beacon
- Monitoring beacons
- Detect in background and notify

# Ranging beacons

- PBBeaconDetectionTableViewController
- Import CoreLocation
- Implement CLLocationManagerDelegate
- Create a LocationManager, NSUUID and CLBeaconRegion (viewDidLoad)
- Start ranging beacons in the CLBeaconRegion previously created (in viewDidAppear)
- Stop ranging (in viewDidDisappear)
- Location  manager delegate :

```
- (void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray *)beacons
inRegion:(CLBeaconRegion *)region
```

- Complete UITableView data source methods

# Device as a beacon

- PBAdvertiseBeaconViewController
- Import CoreLocation and CoreBluetooth
- Implement CBPeripheralManagerDelegate
- Create CBPeripheralManager object and initialize in viewWillAppear
- When switch is enable, retrieve data from form, create CLBeaconRegion and start advertising (using CBPeripheralManager object)
- When switch is disabled, stop advertising (or when viewWillDisappear)

- UUIDGEN command in terminal to generate new UUID

# Monitoring beacons

- See « Ranging beacons »
- Create CLBeaconRegion setting notifyOnEntry and notifyOnExit attributes
  - Call startMonitoringForRegion
  - Listen to callbacks

```
- (void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion *)region;
- (void)locationManager:(CLLocationManager *)manager didExitRegion:(CLRegion *)region;
```

  - Start Ranging when in entering a region (and stop when exiting)

# Background & notifications

- CLBeaconRegion notifyEntryStateOnDisplay attribute
- Location service will keep track of device state
    - In appDelegate
    - Use CLLocationManagerDelegate to get the callback

```
- (void)locationManager:(CLLocationManager *)manager didDetermineState:
(CLRegionState)state forRegion:(CLRegion *)region
```

    - Create UILocationNotification
    - Check state value to set notification body

# Questions?