



# **Mobile Middleware**

## **Applications and Service Case Studies**

# Contents

---

- Summary of patterns and cases
- Web applications and Firefox OS
- Conclusions
- Assignment status

# Patterns for Mobile Computing

- Three categories
  - ◆ distribution
  - ◆ resource management and synchronization
  - ◆ communications
- Distribution patterns pertain to how resources are distributed and accessed in the environment.
  - ◆ remote facade, data transfer object, remote proxy, and observer
- Resource management and synchronization
  - ◆ session token, caching, eager acquisition, lazy acquisition, synchronization, rendezvous, and state transfer
- Communications
  - ◆ connection factory, client-initiated connections, multiplexed communication

# Revisiting Patterns 1/4

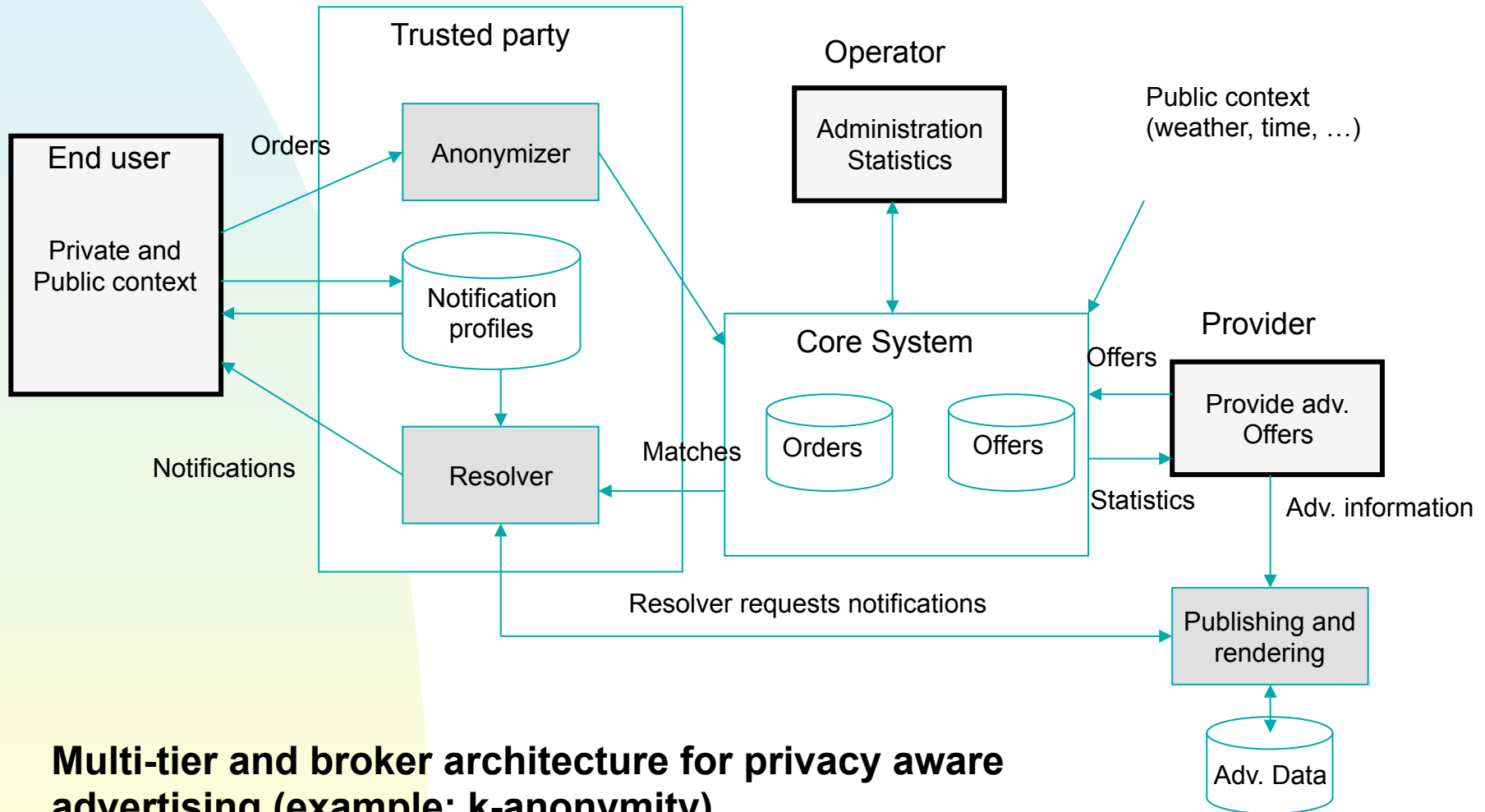
- Widgets
  - ◆ Widgets can employ a number of patterns, typically Remote Proxy and Broker are pertinent. Observer for updates.
- Location Awareness.
  - ◆ Rendezvous and Synchronization are crucial. This can be achieved using a Remote Proxy pattern and the Connection patterns. The Remote Facade pattern is often applied to minimize the number of remote calls needed. Eager Acquisition can be used to anticipate future information needs.

# Revisiting Patterns 2/4

- Generic Mobile push. This application case is similar to Mobile Server, Location Awareness, Mobile Advertisement, and Mobile Video.
- Mobile Push Email. Reachability is vital also in this application scenario. This is achieved using the Client-initiated Connection, Remote Proxy, and Rendezvous patterns.
- Facebook Chat. Multi-tier, client-initiated connection, Lazy synchronization (contacts), Rendezvous, and Remote Proxy.

# Mobile Advertisement Example

- The central entities are the end user, the trusted party, the operator, and the provider
- The trusted party manages end user profiles and anonymizes user profiles and other data so that other parties cannot determine user preferences
- The operator is responsible for running the core system that stores orders
- When an order and offer match, a notification is generated towards the end user
- The provider is the advertiser and responsible for the offers and providing advertisement information that can be then delivered to end users.



# Revisiting Patterns 3/4

## ■ Mobile Advertisement.

- ◆ This application requires a combination of patterns, namely Client-initiated connections, Rendezvous, Synchronization, Caching, Remote Proxy, and Broker.
- ◆ The connections ensure reachability of the mobile terminals and allow to the advertisement system to synchronize advertisements and impressions with the mobile device (if they are stored on board).
- ◆ Rendezvous is needed to keep track of the current location of the device.
- ◆ Remote proxy is needed to handle the connections.
- ◆ The Broker is used to provide indirection (and privacy) between different components in the system.



# Revisiting Patterns 4/4

- Mobile Video. This application can utilize the Client-initiated Connection and Multiplexed Connection for enabling continuous media delivery to the client.
  - ◆ Video-on-demand can be Cached, and video stream buffering can be seen a variant of the Eager Acquisition pattern.
- Mobile Server.
  - ◆ Reachability is vital in this application and it is achieved using the Client-initiated Connection, Remote Proxy, and Rendezvous patterns. Caching can be used at the Remote Proxy to improve performance.



# Applications: revisiting Web apps

---

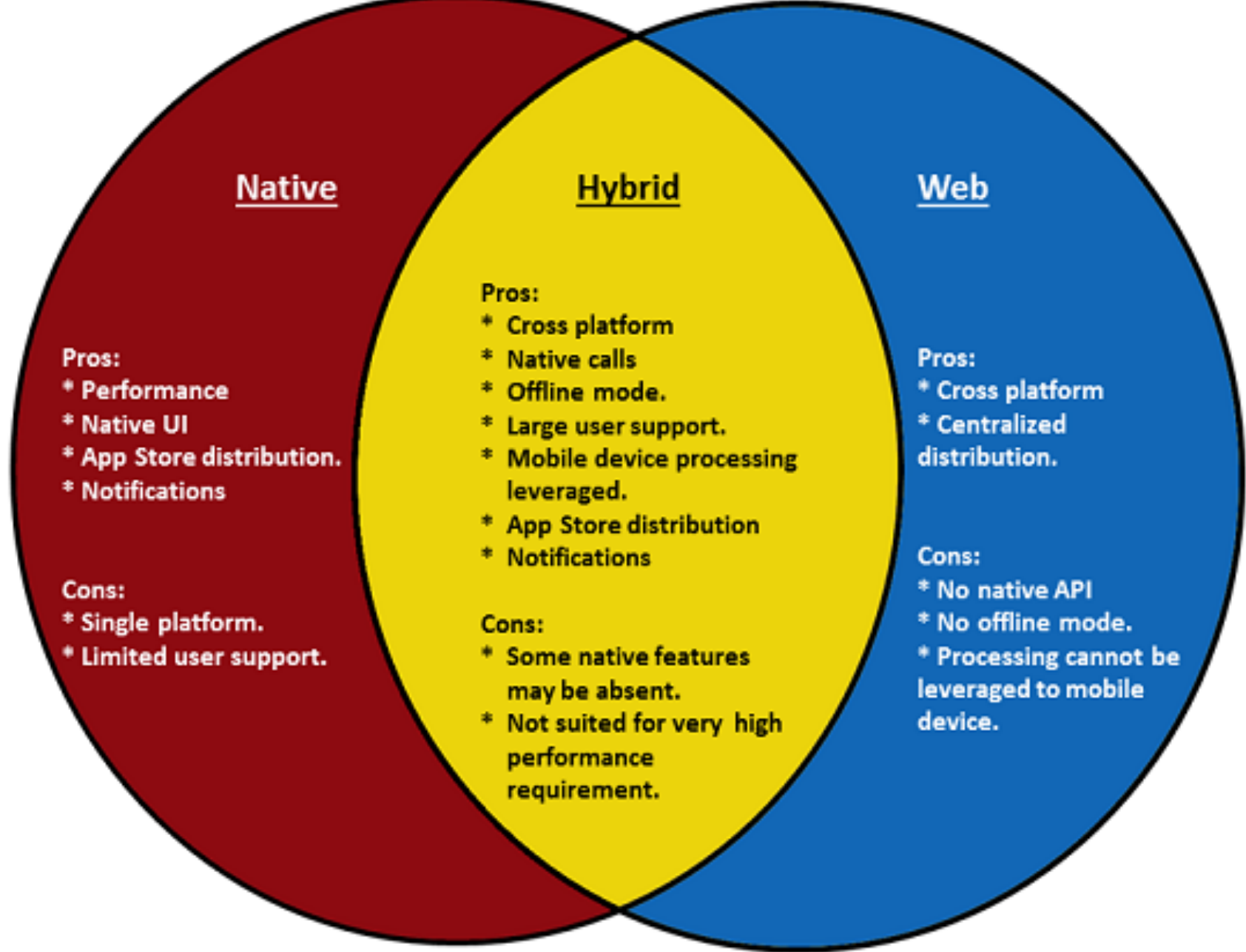
- The current state is fragmented
- Difficult to achieve portability
- Certain patterns are pervasive (model view control and others)
- Solutions?

# Web Apps

- Emerging as an alternative to native applications
- Hybrid usage: Web content to native application interfaces
- Web content can partially solved portability issues
- **Survey: Android Programmers Shifting Toward Web Apps**
  - ◆ *CNet (03/20/12) Stephen Shankland*
  - ◆ [http://news.cnet.com/8301-30685\\_3-57400136-264/survey-android-programmers-shifting-toward-web-apps/](http://news.cnet.com/8301-30685_3-57400136-264/survey-android-programmers-shifting-toward-web-apps/)

# HTML5

- HTML 5 is the next version of HTML
  - ◆ The first public working draft of the specification available in January 2008 and completion expected soon
- Improvements
  - ◆ Web Socket API, advanced forms, offline application API, and client-side persistent storage (key/value and SQL).
- HTML 5 support divides the platforms.
  - ◆ The iPhone platform has a very good support for HTML 5
  - ◆ Also Windows Phone and Android support it
  - ◆ <http://mobilehtml5.org/>



# Firefox OS

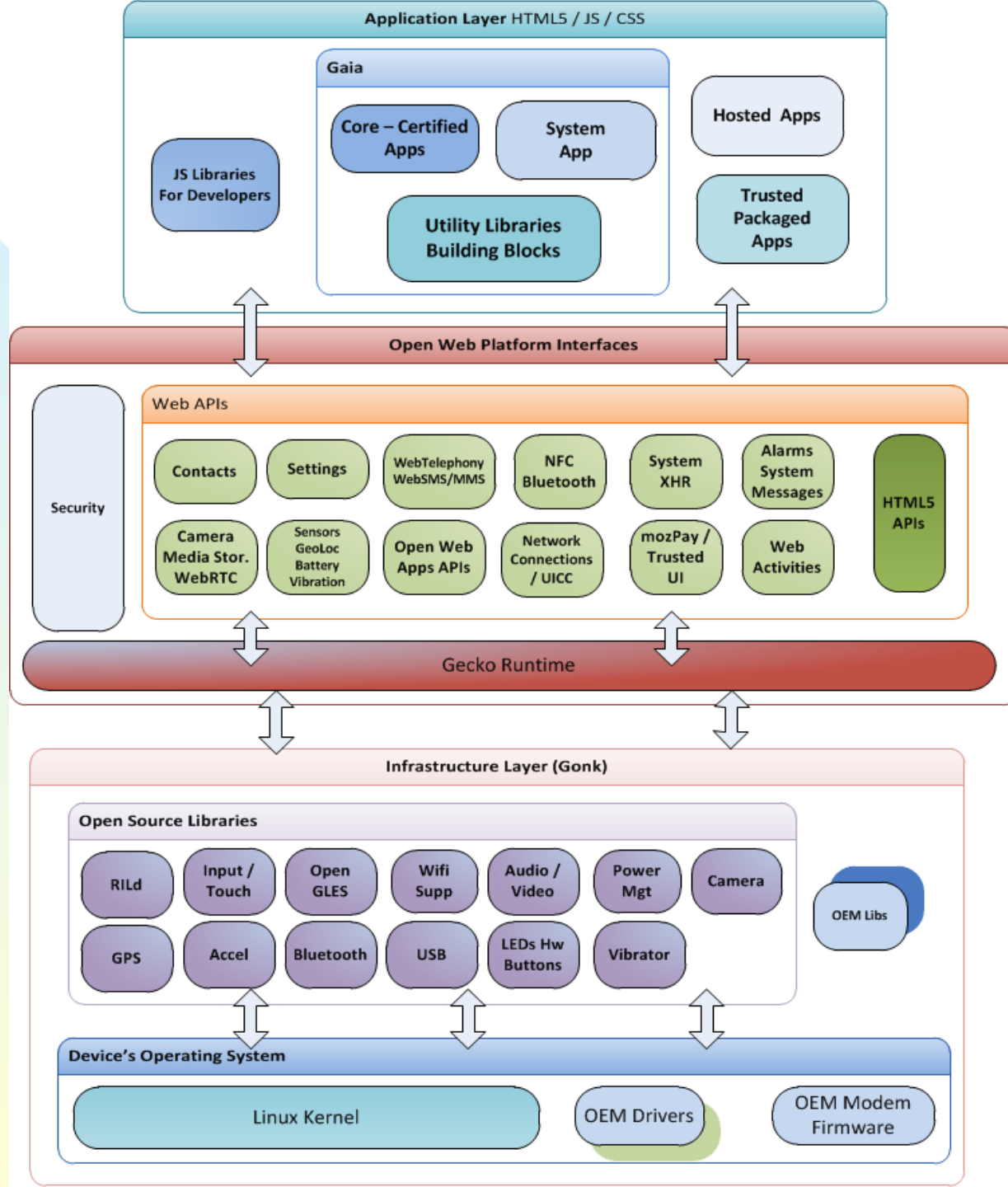
- Standalone OS for open WeB
- Applications are developed using standard Web based technologies
  - ◆ Everything is a Web app
  - ◆ HTML5 and JavaScript
- 1. Create app, 2.Create manifest (.webapp), 3. publish/install
- Mozilla Marketplace
  - ◆ Built-in carrier billing
- <https://hacks.mozilla.org/2012/11/firefox-os-video-presentations-and-slides-on-the-os-webapis-hacking-and-writing-apps/>

# Terms

---

- **Gaia**
  - ◆ The user interface of the Firefox OS platform.
- **Gecko**
  - ◆ This is the Firefox OS application runtime
  - ◆ HTML, CSS, and JavaScript.
- **Gonk**
  - ◆ Gonk is the lower level operating system of the Firefox OS platform.
  - ◆ Linux kernel and userspace hardware abstraction layer (HAL).





Source: [https://developer.mozilla.org/en-US/docs/Mozilla/Firefox\\_OS/Platform/Architecture](https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture)

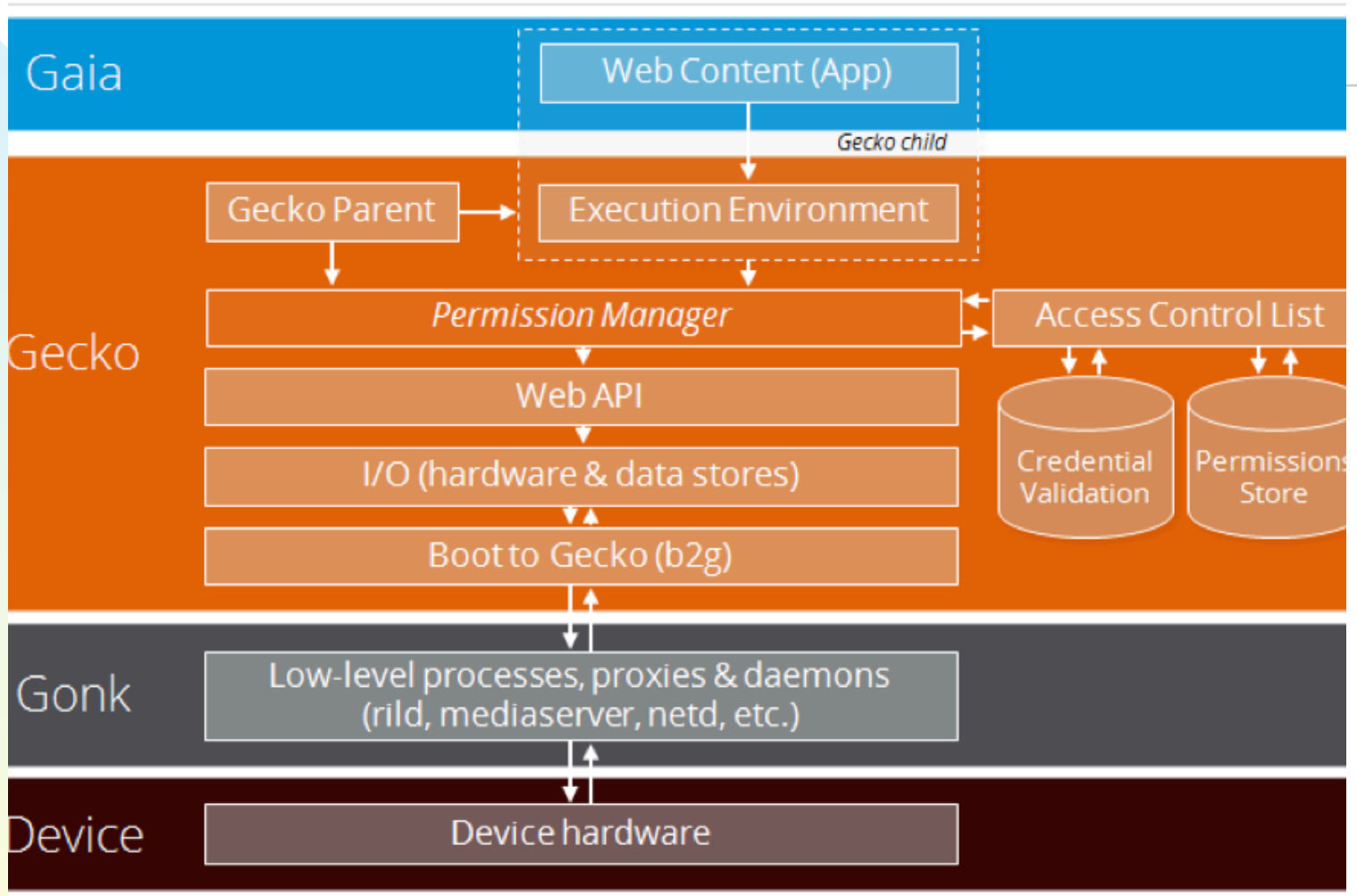
# Security

- Protect OS, apps, user data
- Principle of least permissions
- Each app runs in its own sandbox
- Sandbox includes also data, cookies, etc.
- Apps cannot start other apps
- Apps can share content if the proper permissions have been set (certified apps)
- More information:
  - ◆ [https://developer.mozilla.org/en-US/docs/Mozilla/Firefox\\_OS/Security/Security\\_model](https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Security/Security_model)

# App security model

- Web content
- Web apps
  - ◆ Website with manifest
  - ◆ Can request access to: Geolocation API, Sensor API, Alarm API, FM Radio
- Privileged apps
  - ◆ ZIP file with manifest & signed
  - ◆ Access to medium risk APIs
- Certified apps (system apps)
  - ◆ Access to high risk APIs
  - ◆ Direct access to: Background services, WebSMS, WebTelephony, WebBluetooth, MobileConnection API, Power Management API, Push, ...

# Security model





**Using HTML5, CSS and JavaScript together with a number of APIs to build apps and customize the UI.**

**APIs: <https://hacks.mozilla.org/2013/02/using-webapis-to-make-the-web-layer-more-capable/>**

# Regular APIs

---

- Vibration API
- Screen Orientation
- Geolocation API
- Mouse Lock API
- Open WebApps
- Network Information API
- Battery Status API
- Alarm API
- Web Activities
- Push Notifications API
- WebFM API
- WebPayment
- IndexedDB
- Ambient light sensor
- Proximity sensor
- Notification
- FMRadio

# Privileged APIs

---

- Device Storage API
- Browser API
- TCP Socket API
- Contacts API
- systemXHR

# Certified APIs

---

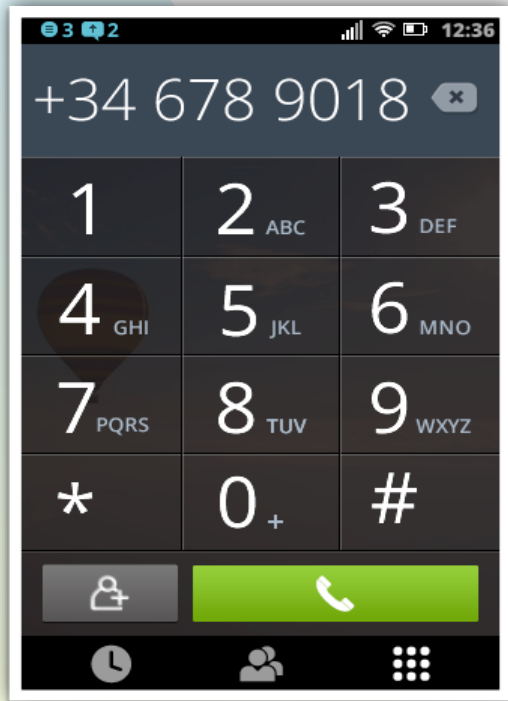
- WebTelephony
- WebSMS
- Idle API
- Settings API
- Power Management API
- Mobile Connection API
- WiFi Information API
- WebBluetooth
- Permissions API
- Network Stats API
- Camera API
- Time/Clock API
- Attention screen
- Voicemail



# Planned APIs

---

- Resource lock API
- UDP Datagram Socket API
- Peer to Peer API
- WebNFC
- WebUSB
- HTTP-cache API
- Calendar API
- Spellcheck API
- LogAPI
- Keyboard/IME API
- WebRTC
- FileHandle API
- Sync API



# WEBTELEPHONY

*// Telephony object*

**var** tel = navigator.mozTelephony;

*// Check if the phone is muted (read/write property)*

console.log(tel.muted);

*// Check if the speaker is enabled (read/write property)*

console.log(tel.speakerEnabled);



---

```
// Place a call
```

```
var cal = tel.dial("123456789");
```

```
// Events for that call
call.onstatechange = function (event) {
  /*
    Possible values for state:
    "dialing", "ringing", "busy", "connecting", "connected",
    "disconnecting", "disconnected", "incoming"
  */
  console.log(event.state);
};

// Above options as direct events
call.onconnected = function () {
  // Call was connected
};

call.ondisconnected = function () {
  // Call was disconnected
};
```

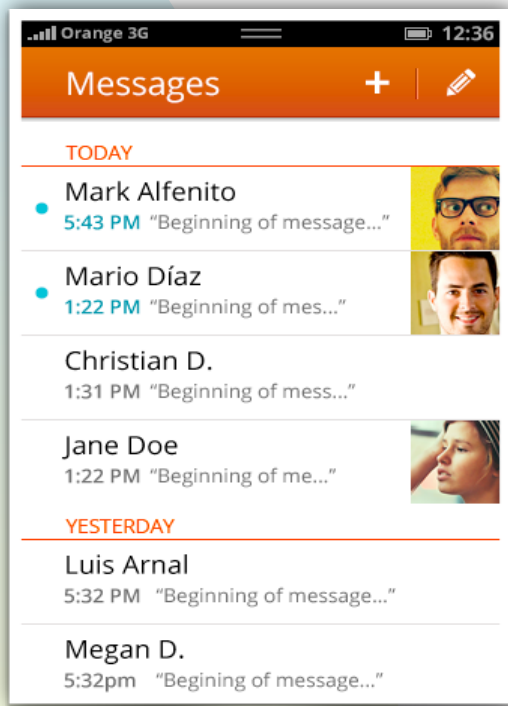
```
// Receiving a call
tel.onincoming = function (event) {
    var incomingCall = event.call;

    // Get the number of the incoming call
    console.log(incomingCall.number);


    // Answer the call
    incomingCall.answer();
};

// Disconnect a call
call.hangUp();

// Iterating over calls, and taking action depending on their changed status
tel.onscallschanged = function (event) {
    tel.calls.forEach(function (call) {
        // Log the state of each call
        console.log(call.state);
    });
};
```




# WEBSMS



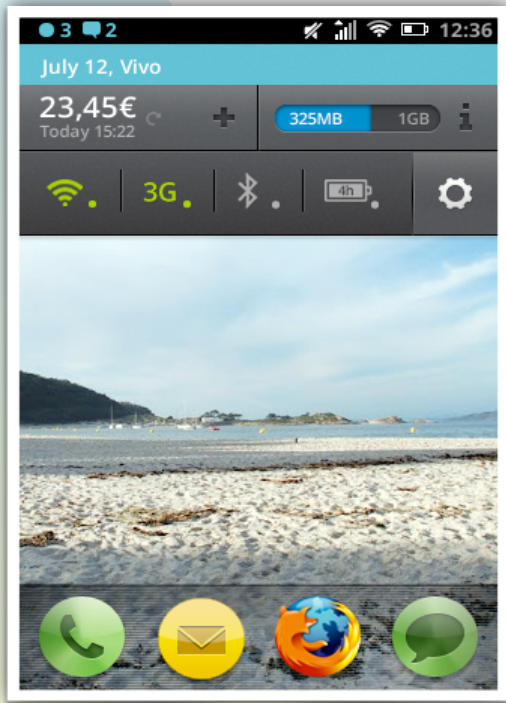
```
// SMS object
var sms = navigator.mozSMS;

// Send a message
sms.send("123456789", "Hello world!");
```





```
// Recieve a message
sms.onreceived = function (event) {
  // Read message
  console.log(event.message);
};
```




# BATTERY STATUS API

```
var battery = navigator.mozBattery
if (battery) {
    var batteryLevel = Math.round(battery.level * 100) + "%",
        charging = (battery.charging)? "" : "not ",
        chargingTime = parseInt(battery.chargingTime / 60, 10),
        dischargingTime = parseInt(battery.dischargingTime / 60, 10);

    // Set events
    battery.addEventListener("levelchange", setStatus, false);
    battery.addEventListener("chargingchange", setStatus, false);
    battery.addEventListener("chargingtimechange", setStatus, false);
    battery.addEventListener("dischargingtimechange", setStatus, false);
}
```



# **VIBRATION API**

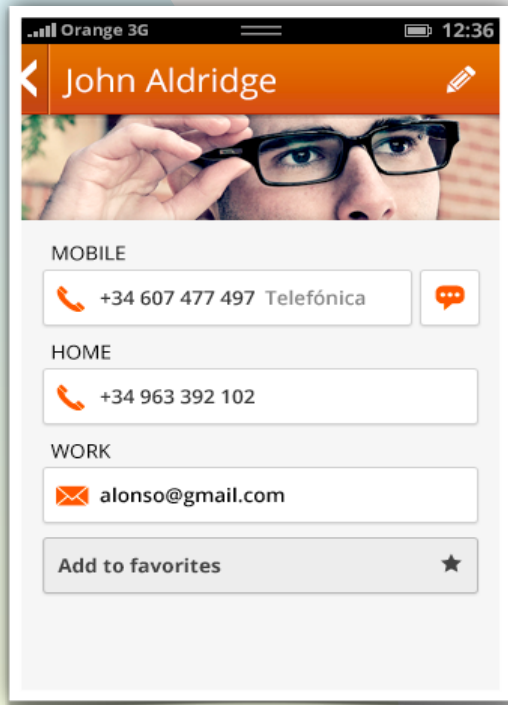


```
// Vibrate for one second  
navigator.mozVibrate(1000);
```

```
// Vibration pattern [vibrationTime, pause,...]  
navigator.mozVibrate([200, 100, 200, 100]);
```

```
// Vibrate for 5 seconds  
navigator.mozVibrate(5000);
```

```
// Turn off vibration  
navigator.mozVibrate(0);
```



# CONTACTS API

```
var contact = new mozContact();
contact.init({name: "Tom"});

var request = navigator.mozContacts.save(contact);
request.onsuccess = function() {
    console.log("Success");
};

request.onerror = function() {
    console.log("Error")
};
```



# SETTINGS API



```
var settings = window.navigator.mozSettings;  
settings.getLock().set({"background":  
"pretty.png"});
```



# **DEVICE STORAGE API**

```
var storage = navigator.getDeviceStorage("videos"),
    cursor = storage.enumerate();

cursor.onerror = function() {
    console.error("Error in DeviceStorage.enumerate()", cursor.error.name);
};

cursor.onsuccess = function() {
    if (!cursor.result)
        return;

    var file = cursor.result;

    // If this isn't a video, skip it
    if (file.type.substring(0, 6) !== "video/") {
        cursor.continue();
        return;
    }

    // If it isn't playable, skip it
    var testplayer = document.createElement("video");
    if (!testplayer.canPlayType(file.type)) {
        cursor.continue();
        return;
    }
}
```



# **SCREEN ORIENTATION API**

```
// Portrait mode:  
screen.mozLockOrientation("portrait");
```

```
/*
```

Possible values:

"landscape"

"portrait"

"landscape-primary"

"landscape-secondary"

"portrait-primary"

"portrait-secondary"

```
*/
```



# **NETWORK INFORMATION API**




```
var connection = window.navigator.mozConnection,  
    online = connection.bandwidth > 0,  
    metered = connectrion.metered;
```



# KEYBOARD API





---

```
var keyboard = window.navigator.mozKeyboard;  
keyboard.sendKey(0, keyCode);
```

# Firefox OS

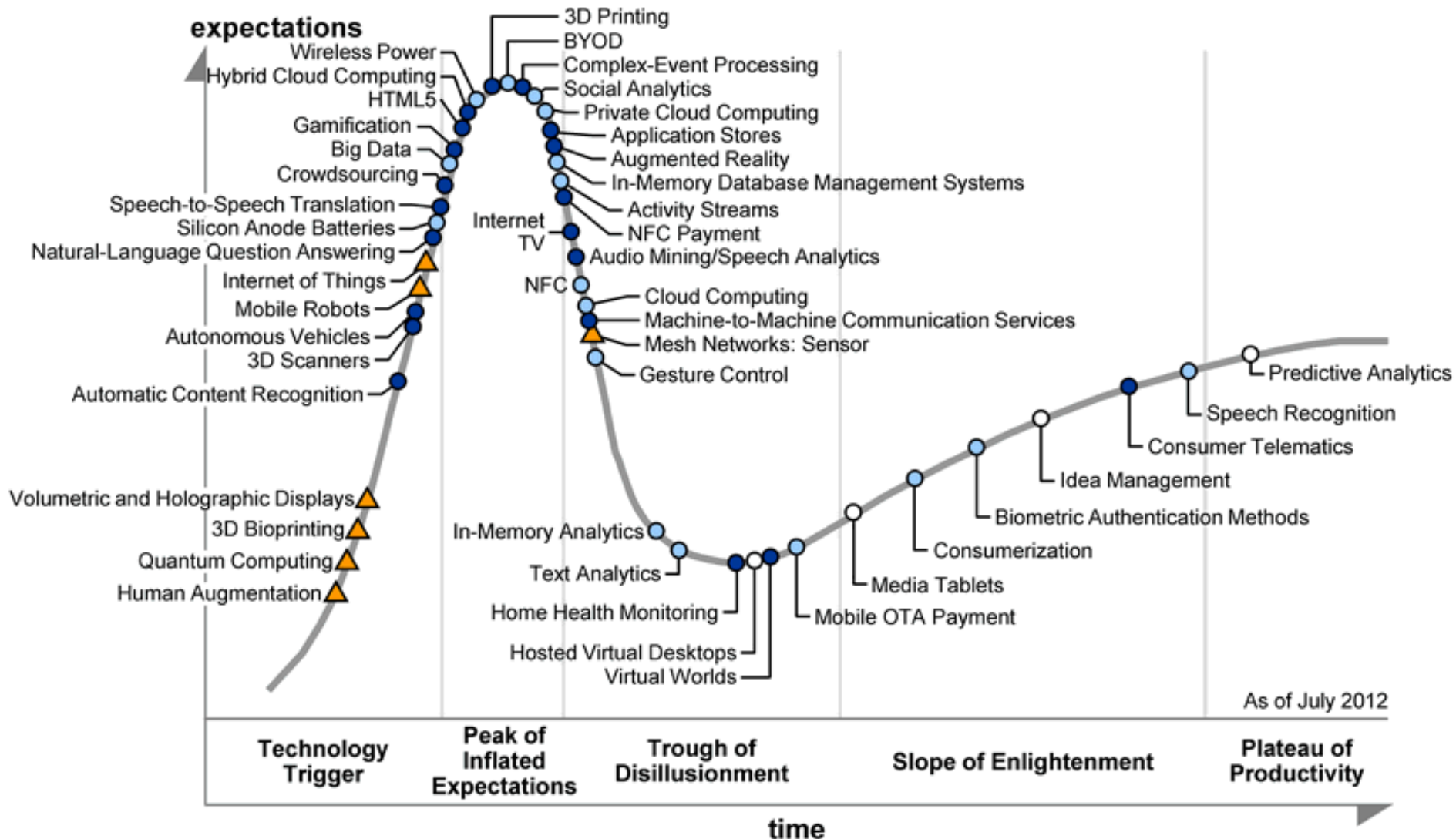
---

- HTML5 and JavaScript based mobile apps
  - ◆ Contrasts native apps on Android, iOS, and WP
- Standardized APIs to the OS and middleware

# Recent Trends

---

# Gartner Hype Curve 2012



Plateau will be reached in:

○ less than 2 years   ● 2 to 5 years   ● 5 to 10 years   ▲ more than 10 years   ⊗ obsolete before plateau

# Topics

---

- App stores
  - ◆ Apple, Nokia, Android, WP8, ...
  - ◆ In-app purchases
  - ◆ Searching, purchasing, advertising, ...
- How to do software updates
- How to support community buildup
- Push notifications
  - ◆ Dedicated push servers
  - ◆ Control plane
  - ◆ Triggers
- Inter-app communication is still in early phases
- Difficult to build on local communication context (some games do this today)

# Sensors

---

- The number of sensors will increase dramatically
- Innovative new applications
  - ◆ Pulse monitor, augmented reality, ...
- Plug-in sensors and devices
- Crowdsourcing sensor data and processing?
- Can we use basestations?
- Decentralized processing?

# Challenges

---

- Cloud integration
- Event-based program flow
- Content storage, search, and sync
- APIs and interoperability
  - ◆ Mitigating fragmentation
- Energy efficiency

# Conclusions

---

- Mobile software is mainstream
  - ◆ Appstores
  - ◆ Better tools and development environments
  - ◆ Integration with Web resources
  - ◆ Integration with other apps
  - ◆ Integration with sensors!
- Challenges include
  - ◆ Fragmentation in its many forms
    - ✦ Devices, standards, implementations
  - ◆ Access to mobile APIs
  - ◆ Practical ubicomp deployment
  - ◆ Adaptation





# Course Overview

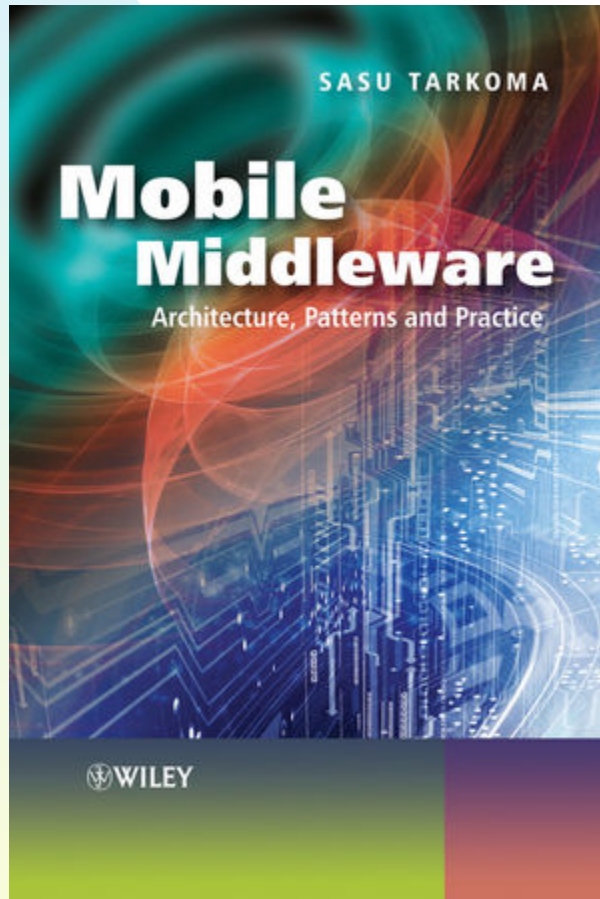
---

- 4 credit course
- Three components
  - ◆ Lectures
  - ◆ Assignment
  - ◆ Literature (three papers and course book)
- Grading based on
  - ◆ Exam (60%)
  - ◆ Assignment (40%)

# Timetable

- 13.3. Introduction and assignments.
- 20.3. Platforms, Middleware
- Assignment slot 1: Simple video player
- 27.3. Assignment slot 2: Video transmitter
- 3.4. easter
- 10.4. Patterns
- Assignment slot 3: Video server (video list/selection)
- 17.4. Applications: Carat
- 24.4. Applications and Summary
- 8.5. Assignment slot 4: Mixing table (video mixer)
- Final submission in May
- Exam 14.5. 16:00 in T1

# Course Book



- Mobile Middleware – Architecture, Patterns, and Practice published by Wiley
  - ◆ Publication date 27.3.2009
  - ◆ Available in digital form
- Several papers to read

# Included chapters

---

- Chapter 1: Introduction
- Chapter 2: Architectures (note 2.6 described old systems)
- Chapter 3: 3.1-3.3, 3.6
- Chapter 4: Principles and Patterns
- Chapter 8: Data Synchronization
- Chapter 10: Application and Service Case Studies

# Additional reading

---

- Mobile platforms survey, 2011.
- Carat: Collaborative Energy Diagnosis for Mobile Devices. UCB Tech report, March 2013.
- Analyzing Inter-Application Communication in Android. Mobisys 2011.
- K. Kumar and Y-H. Lu. Cloud computing for Mobile Users: Can Offloading Computation Save Energy? IEEE Computer, 2011.