Mobile Middleware Course

Mobile Platforms and Middleware

Sasu Tarkoma
iOS APNS

1. The app registers for push notifications
2. The app receives the device token
3. The app sends the token to your server
4. When something interesting happens, your server sends a push notification to the APNS server
5. APNS sends the push notification to your app

Your App
- App ID
- Provisioning Profile
- Device Token

Your Server
- SSL Certificate
- Private Key

APNS Server

Source: http://www.raywenderlich.com/3443/apple-push-notification-services-tutorial-for-ios-part-12/push-overview
Apple Push Notification Service

- APNS usage involves the following steps:
  - Service or application developer connects to the APNS system using a unique SSL certificate. The certificate is obtained from Apple with the developer identifier and application identifier.
  - Applications obtain deviceTokens that are then given to services.
  - The APNS is used to send one or more messages to mobile devices. The push operation is application and device specific and a unique deviceToken is needed for each destination device.
  - The service or application disconnects from APNS.
Android: Google Cloud Messaging

Windows 8 Push Messaging

Summary of Push Services

- Very similar in design for Android, iOS and WP
  1. Client-initiated connection with push servers: TCP and TLS, fallback to HTTP/HTTPS
  2. Registration phase to obtain URI and token
  3. Delegation of URI and token to 3rd party services
  4. 3rd party servers push content to mobiles through push servers (URI and token needed)
Discussion

- The current state is fragmented
- Difficult to achieve portability
- Certain patterns are pervasive (MVC 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
HTML5

- HTML 5 is the next version of HTML
  - The first public working draft of the specification available in January 2008 and completion expected around 2012

- 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/
The Open Mobile Terminal Platform (OMTP) group defines requirements and specifications that aim towards simpler and more interoperable mobile APIs.

BONDI defines requirements governing Device Capability access by JavaScript APIs to promote interoperability and security of implementations.

The 1.1 release of BONDI is compliant with the W3C Widgets: Packaging and Compliance specification.
BONDI Architecture

Website

Browser

Widget

Widget User Agent

Web Engine (WebKit, ...)

JavaScript Extension

JavaScript API and Access Control

Application Invocation, Network, Messaging, Communication Log, Media Gallery, Media Recording, Personal Information, Persistent Data, Location, User Interaction, Device Status, System Events, Policy Management, APIManagement, Extensions

Device Capability Access Control

Operating System

BONDI code
### Native

**Pros:**
- Performance
- Native UI
- App Store distribution
- Notifications

**Cons:**
- Single platform.
- Limited user support.

### Hybrid

**Pros:**
- Cross platform
- Native calls
- Offline mode.
- Large user support.
- Mobile device processing leveraged.
- App Store distribution
- Notifications

**Cons:**
- Some native features may be absent.
- Not suited for very high performance requirement.

### Web

**Pros:**
- Cross platform
- Centralized distribution.

**Cons:**
- No native API
- No offline mode.
- Processing cannot be leveraged to mobile device.

*Source: http://www.developer.nokia.com/Community/Wiki/Cross_Platform-Mobile_Architecture*
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
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).
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:
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.onCallsChanged = 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);
}
DEVICE STORAGE API
```javascript
var storage = navigator.getDeviceStorage("videos"),
    cursor = storage.enumerate();

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

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

/*
Possible values:
"landscape"
"portrait"
"landscape-primary"
"landscape-secondary"
"portrait-primary"
"portrait-secondary"
*/
NETWORK INFORMATION API
```javascript
var connection = window.navigator.mozConnection,
online = connection.bandwidth > 0,
metered = connection.metered;
```
Firefox OS

- HTML5 and JavaScript based mobile apps
  - Contrasts native apps on Android, iOS, and WP
- Standardized APIs to the OS and middleware
<table>
<thead>
<tr>
<th>Feature</th>
<th>Android/Linux</th>
<th>iOS</th>
<th>FireFox OS/Linux</th>
<th>Windows Phone 8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td>Java, native code with JNI and C/C++</td>
<td>Objective-C</td>
<td>Javascript and HTML5</td>
<td>C# and .NET, various</td>
</tr>
<tr>
<td><strong>Network features</strong></td>
<td>Basic APIs: Bluetooth, WiFi, cellular, NFC Network information</td>
<td>Basic APIs: Bluetooth, WiFi, cellular, NFC Network information</td>
<td>No (only for pre-installed applications)</td>
<td>Basic APIs: Bluetooth, WiFi, cellular, NFC Network information Set connection preferences</td>
</tr>
<tr>
<td><strong>Background processing</strong></td>
<td>Yes (services)</td>
<td>Task based multitasking since version 4</td>
<td>No (planned)</td>
<td>Multitasking API</td>
</tr>
<tr>
<td><strong>Push notification</strong></td>
<td>Yes (Google Cloud Messaging)</td>
<td>Yes (iOS Push Notification)</td>
<td>Yes (Firefox Push Notification)</td>
<td>Yes (Microsoft Push Notification Service)</td>
</tr>
<tr>
<td><strong>Energy and power monitoring</strong></td>
<td>Battery status</td>
<td>Monitoring since 3.0</td>
<td>Battery status</td>
<td>Battery status</td>
</tr>
<tr>
<td><strong>HTML 5</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Open Source</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>3rd party application installation</strong></td>
<td>Certificate, Google Play</td>
<td>Certificate, Apple AppStore</td>
<td>Certificate, app stores and web sites</td>
<td>Certificate, Windows Phone Store</td>
</tr>
</tbody>
</table>
Energy Conservation Patterns and Techniques
<table>
<thead>
<tr>
<th>Low-level power management</th>
<th>Linux Power Management</th>
<th>iOS kernel</th>
<th>Windows NT</th>
<th>Linux Power Management (Gonk)</th>
<th>Kernel-side framework with power API (Power Manager), peripheral power on/off</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-level power management</td>
<td>Java class PowerManager, JNI binding to OS. Key methods: goToSleep(long), newWakeLock(...), userActivity(long...)</td>
<td>I/O Framework</td>
<td>Run-time power management framework</td>
<td>Gaia (OS Shell) Gecko runtime: Power Management Web API is non-standard and reserved for pre-installed applications</td>
<td>Applications use domain manager that follows system-wide power-state policies.</td>
</tr>
<tr>
<td>Energy conservation patterns</td>
<td>Wake lock (partial, full) is used to ensure that device stays on. Methods: Create, acquire, release, sensor batching</td>
<td>Coding patterns, multitasking API (since iOS 4), push API, coalesced updates (since iOS7)</td>
<td>Multitasking API (tasks and push notification), asynchronous events</td>
<td>Asynchronous events (system messages)</td>
<td>Active object, wakeup events, resource and domain manager</td>
</tr>
<tr>
<td>Policies</td>
<td>Wake lock specific flags and policies, system-wide power setting</td>
<td>Internal, multitasking API (since iOS 4)</td>
<td>Internal</td>
<td>Gonk and Gecko level</td>
<td>Domain manager for system-wide and domain-wide policy. Domain-specific policies are possible</td>
</tr>
<tr>
<td>Battery information</td>
<td>The BatteryManager class contains strings and constants for different battery related notifications that applications can subscribe to, includes: battery level, temperature, voltage</td>
<td>iOS 3 and later: UIDevice Class allows to query/subscribe battery info</td>
<td>Battery class provides the battery level, remaining operating time, and an event when battery is below 1%</td>
<td>W3C Battery Status API</td>
<td>Battery API (charge level, external power). Nokia Energy Profiler API</td>
</tr>
</tbody>
</table>
Libraries/frameworks for Portability

- **Apache Cordova (former PhoneGap)**
  - Use JavaScript to access a native API, like geolocation, camera, storage, etc. using a single API call irrespective of the mobile platform

- **Sench Touch**
  - An HTML5 framework and native api wrappers

- **Xamarin**
  - Xamarin compiler bundles the .NET runtime and outputs a native executable, packaged as an iOS or Android app
PhoneGap

- Uses a native shell that runs Web content
  - HTML5 and JavaScript for apps
- Allows the inclusion of Web code in a native app
- Native code loads a URL to the web app using the device’s browser
- Native APIs are exposed through JavaScript
Example: acceleration

```javascript
function onSuccess(acceleration) {
    alert('Acceleration X: ' + acceleration.x + '
    ' + '
    Acceleration Y: ' + acceleration.y + '
    ' + '
    Acceleration Z: ' + acceleration.z + '
    ' + '
    Timestamp: ' + acceleration.timestamp + '
');
}

function onError() {
    alert('onError!');
}

navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);

*Also track acceleration at regular intervals with watchAcceleration*
<table>
<thead>
<tr>
<th>Feature</th>
<th>iPhone / iPhone 3G</th>
<th>iPhone 3GS and newer</th>
<th>Android</th>
<th>Blackberry OS 5x</th>
<th>Blackberry OS 6.0+</th>
<th>WebOS</th>
<th>Windows Phone 7 + 8</th>
<th>Symbian</th>
<th>Bada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Camera</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Compass</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Contacts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>File</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Media</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Network</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Alert)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Sound)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Vibration)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Storage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Source: http://phonegap.com/about/feature/
jQuery Mobile

- jQuery Mobile is a touch-optimized HTML5 UI framework designed to make responsive web sites and apps that are accessible on all smartphone, tablet and desktop devices.

- Source: http://jquerymobile.com/
Limitations

- HTML5 has 5MB local storage limit
- Mobile browser experience is inconsistent across devices
- Performance is inconsistent across devices
- Native application code required
  - Extend CordovaWebViewClient and load the URL to the website
- JS is not compiled and visible
  - Can obfuscate
System bus

- An asynchronous system-wide event bus is a basic solution for interconnecting various on-device components.

- There is no single standard for this.
  - Android and Java ME use Java-specific events (Android Intent filtering)
  - MeeGo uses D-BUS
  - Palm's WebOS W3C Events

- One particular trend is to utilize URI-based conventions for naming system resources and services.
  - This is extensively used in Android, WebOS and the BONDI architecture. Can be used for iOS as well (URL handlers)
Summary

- Three current smartphone platforms
  - iOS, Android, WP8

- Fragmentation is a current problem
  - Device, standard, implementation

- Closed vs. Open platforms
  - Trust in apps, API access, privileges

- Cross-platform development
  - HTML5, native, or hybrid
  - Middleware support
  - Hybrid is a viable alternative