Mobile Middleware

Applications and Service Case Studies

Contents

Introduction

- Mobile software is a growing area
  - Billions of downloads from Apple AppStore
  - Development processes, tools, APIs are crucial for the ecosystem
  - Integration with Web resources
- Key applications
  - Voice
  - Multimedia
  - Messaging
  - Web sites, mashups, services
  - Location-based services
- Forthcoming features
  - Context-awareness, adaptability, smart spaces

Mobile Service Development

- The mobile landscape is fragmented
  - Heterogeneous device base
  - Many different wireless technologies
- The situation is challenging for the developer
  - Many APIs
  - Many middleware platforms
  - APIs evolve over time
- Current challenge of the industry pertains to improving the development processes
- HTML5 and hybrid apps, cross-platform toolkits

Mobile Services Overview

Consumer Domain

Information

Dynamic content
• News
• Weather
...

Reference content
• Phone books
• Catalogues
• Dictionaries

Communication

Messaging
• SMS
• email
...

Advertising
• Sponsored Alerts
• Mobile Promotion
• Permission Marketing

Mobile Emergency Service
•Tracking

Modern apps combine these!

Entertain

Games & Gambling
• Stand-alone Games
• Betting

Audio
• Ringtones
• MP3
...

Video
• Photographs
• Video-Clips

Transaction

Tailing
• Auctions
• Sales
• Ticketing

Finance
• Brokerage
• Banking
...

Payment
• Micro
• Macro

External

M-SCM
• **Fleet Management**
• **Tracking**

M-CRM
• **Sales**
• **Service**

Internal

M-Workforce
• **Calendar**
• **Email**
• **Groupware**

Business Domain

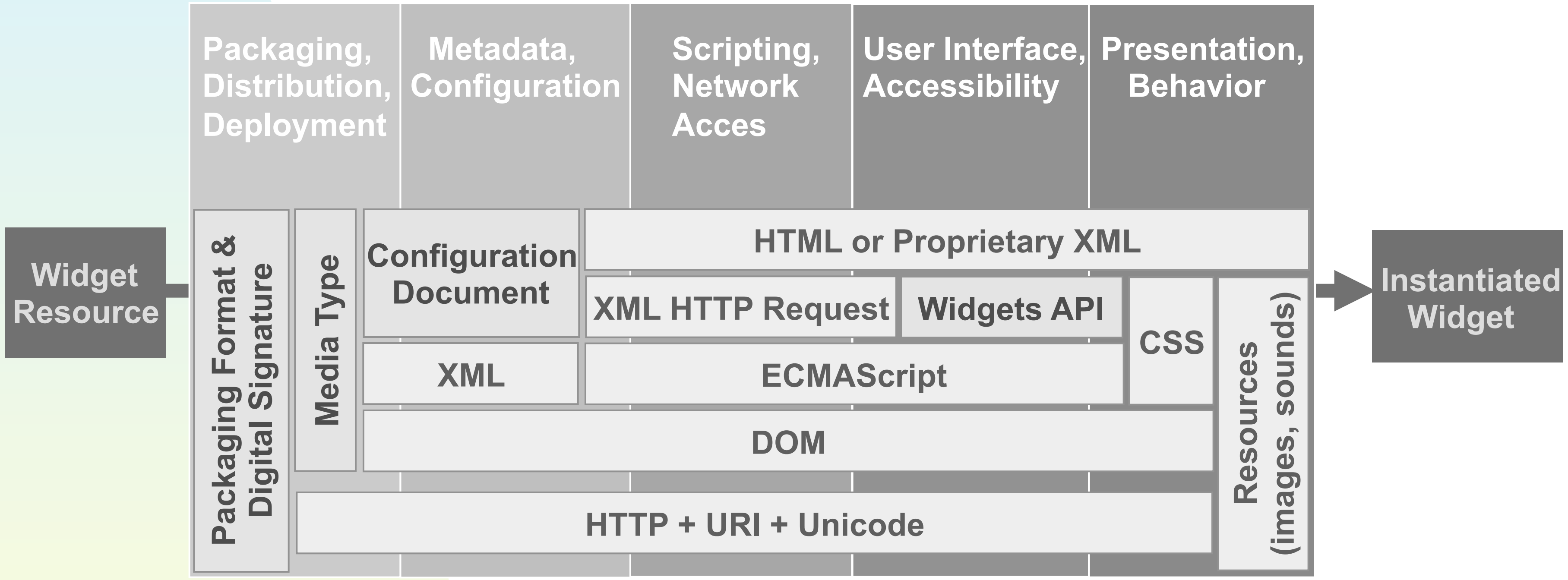
Network centric mobile application types

- Streaming Media
  - Challenges: high jitter, low throughput
  - buffering, layered encoding
  - Tradeoffs between energy / network utilization / buffer size
- Mobile Commerce
  - Challenges: high latency, security
  - Adaptive design, minimized comms.
- Pervasive Gaming
  - Challenges: latency variations, real-time requirements
  - CPU intensive
- Web Browsing
  - Challenges: low throughput, high load
  - Caching, improved protocols (SPDY, …)

Widgets

- Widgets are lightweight Web applications / mobile apps
  - HTML, Cascading Style Sheets (CSS), RSS, Javascript, and AJAX
- Differences exist in:
  - the packaging format
  - the security model
  - the APIs
- WidSets is a simple service developed by Nokia that provides mobile users with information that is normally accessed via the Internet
  - WidSets is based on widgets that utilize RSS feeds to retrieve current information from the Web
  - Obsolete by now
- Homescreen widgets in Android

W3C Widgets



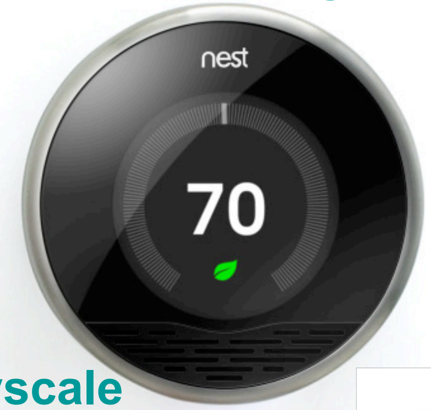
Widget User Agent

Android Widgets

- To create your own widget, extend View or a subclass (typically RemoteView).
- Java implementation file
  - This is the file that implements the behavior of the widget. BroadcastReceiver is used for updates.
- XML definition file (AndroidManifest.xml)
  - An XML file in res/values/ that defines the XML element used to instantiate your widget, and the attributes that it supports.
- Layout XML [optional] (AppWidgetProviderInfo)
  - An optional XML file that defines the layout of the widget and other UI related parameters.
- Widget onReceive should finish within 5secs. Long-running activities can be implemented in a service that updates the widget.

Internet of Things

- M2M traffic solutions (security, healthcare, energy, …)
- Cosm (Pachube) Web service for connecting sensor data
  - www.cosm.com
- There gateway for home automation and monitoring
  - http://therecorporation.com/fi
- Rymble By Symplio
  - http://www.rymble.com/
- NEST learning thermostat
- Withings products
  - http://www.withings.com/en/bodyscale
- Karotz By Aldebaran Robotics
  - http://www.karotz.com/home
- Green Goose
  - http://greengoose.com/
- Google Q
- And many emerging products based on 802.15.4, WiFi, RFID and NFC, and the power of the cloud (REST-based interfaces)



CoAP Protocol

- Constrained **machine-to-machine (M2M)** web protocol
- HTTP for the Internet of Things
- **Follows the Representational State Transfer (REST) architecture**
- Asynchronous transaction support, reliable unicast and best-effort multicast
- Basic proxy and caching capabilities
- Low header overhead and parsing complexity
- URI and content-type support
- UDP binding (may use IPsec or DTLS)
- Support for resource discovery

CoAP and HTTP

- Methods are very similar to HTTP methods
- Resources are identified by URIs
- Response codes are a subset of HTTP response codes
- Options carry additional information (similar to HTTP header lines, but using a more compact encoding)

CoAP Messaging

- Messaging model
  - CON (confirmable), receiver ACK or RST
  - ACK, CON was OK, can include data
  - Reset Message (RST), problem with CON
  - Non-Confirmable (NON)
- Simple stop-and-wait retransmission reliability with exponential back-off for "confirmable" messages.
- Duplicate detection for both "confirmable" and "non-confirmable" messages.
- CoAP transactions provide
  - Reliable UDP based messaging
- CoAP methods are similar to HTTP methods
  - GET, POST, PUT, DELETE

Location-based Services I

- Location-based services
  - GPS
    - 24 satellites 20 km above the Earth
    - 4 satellites are needed (at least 3)
  - A-GPS
    - Phone gets satellite information from the mobile network
    - Works indoors
    - Energy efficient (offloading)
  - Cell-id (one basestation, three basestations + known measurement point)
  - Indoor positioning

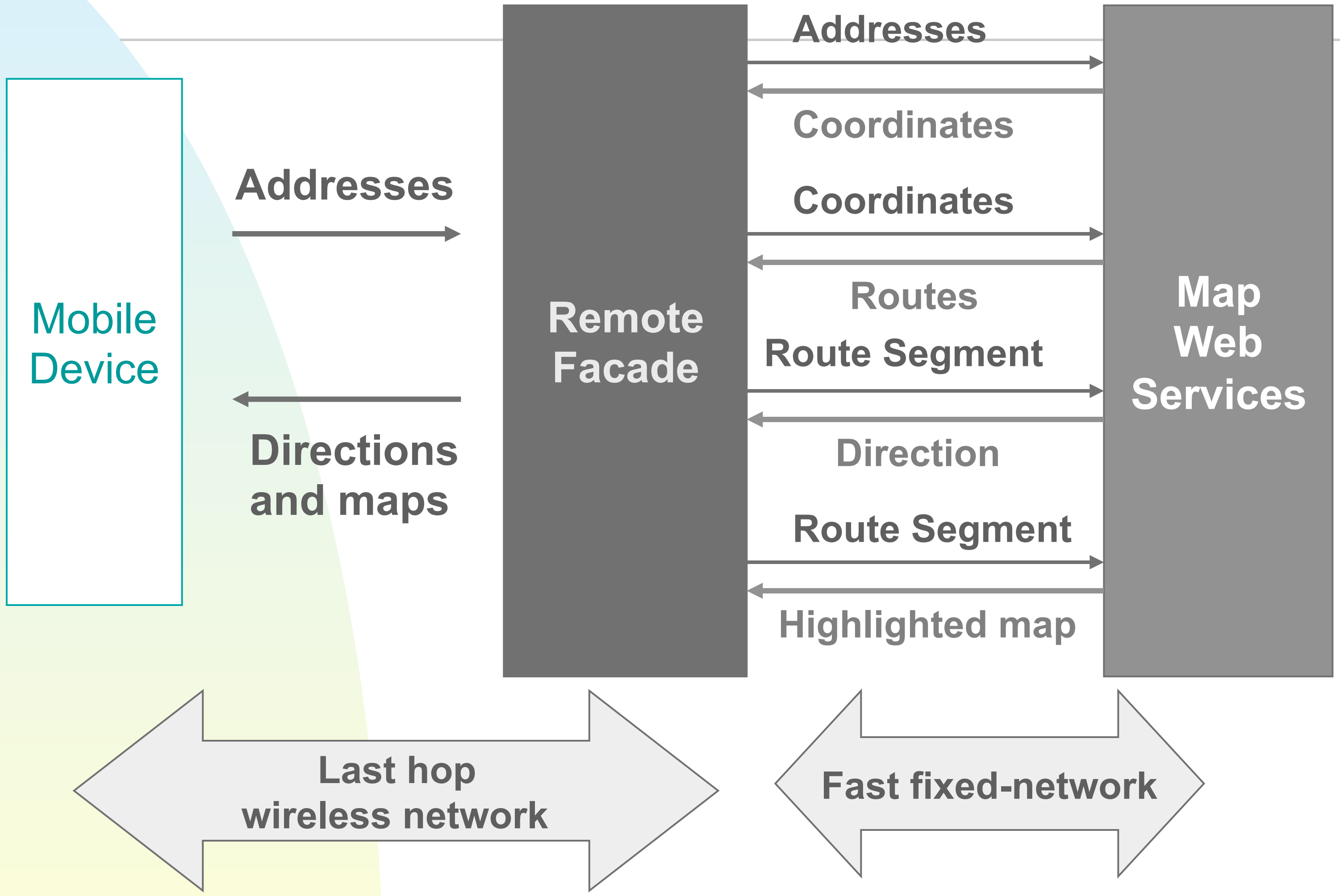
Location-based Services II

- Geocoding: to calculate a location's latitude and longitude coordinates, including street addresses and intersections, street blocks, postal codes, …
- Reverse geocoding: to get location information given latitude and longitude
- Geotagging: to add map annotations

- Applications
  - Friend finding and communities
  - Dynamic content services
  - Pedestrian and city use
  - Outdoor and satellite maps
  - Alerts for traffic, POI, safety, speed alerts
  - Collaborative location-aware sensing

Example: Android API

- LocationProvider class (can use Criteria to select a suitable provider)
    - network: uses 3G/WiFi for positioning, works indoors
    - gps: uses GPS for outdoor positioning
    - Passive: allows application to get passive location information based on other apps
- Can use LocationProvider to figure out if a specific provider is enabled (and then can prompt user to enable it by using an intent).
- Permissions: ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, INTERNET


Mobile Device
Addresses
Directions and maps
Remote Facade
Addresses
Coordinates
Coordinates
Routes
Route Segment
Direction
Route Segment
Highlighted map
Map Web Services
Last hop wireless network
Fast fixed-network

Email

- Simple Mail Transfer Protocol (SMTP) protocol for sending messages
- The Internet Message Access Protocol (IMAP) supports polling and notifications
- The server sends a notification to a client to inform that there is data available
- This allows flexible retrieval of messages and gives the client the control of whether or not to download new message data.

Mobile Push Email

- BlackBerry
- Microsoft DirectPush
- Apple iPhone OS 3.0

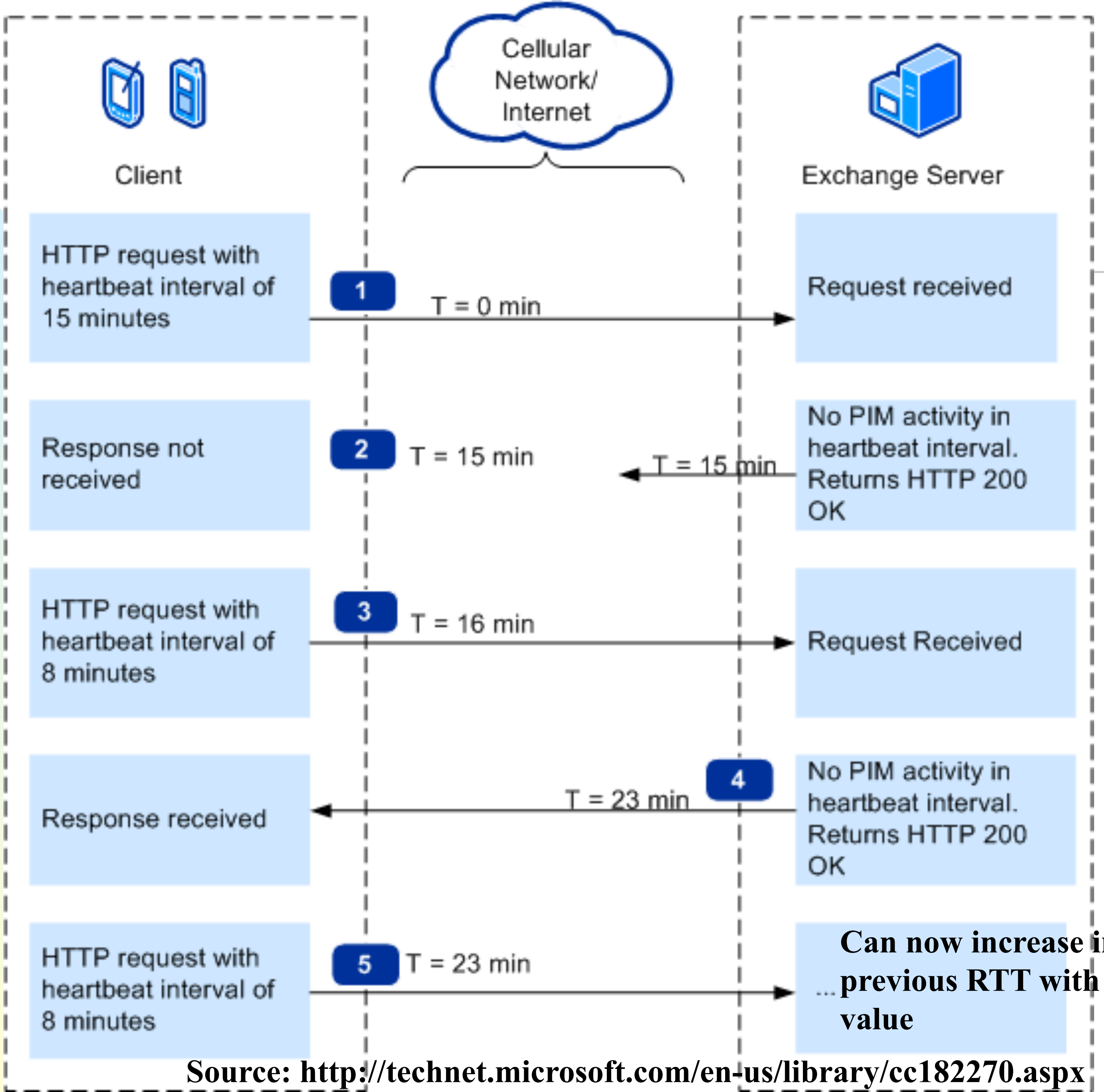
- Implementation
  - Custom server in access network
  - IMAP IDLE
  - Long-lived client-initiated connection
  - SIP (in the future?)

BlackBerry

- Blackberry devices have become popular among business users in part because they support desktop style email usage experience with almost instant delivery of messages
- Blackberry devices utilize a custom enterprise server that is connected to the traditional e-mail system
- The enterprise server monitors the e-mail server and then can pull new messages and send them to the Blackberry device using push over the wireless network

DirectPush

- Microsoft introduced the DirectPush Technology with Windows Mobile 6
- Mobile devices that support DirectPush utilize a long-lived HTTPS request to the Exchange server
- The Exchange server monitors activity on the users mailbox
- Details
  - http://technet.microsoft.com/en-us/library/cc182260.aspx


Cellular Network/ Internet
Client
Exchange Server
HTTP request with heartbeat interval of 15 minutes
1
T = 0 min
Request received
Response not received
2
T = 15 min
T = 15 min
No PIM activity in heartbeat interval. Returns HTTP 200 OK
HTTP request with heartbeat interval of 8 minutes
3
T = 16 min
Request Received
Response received
T = 23 min
4
No PIM activity in heartbeat interval. Returns HTTP 200 OK
HTTP request with heartbeat interval of 8 minutes
5
T = 23 min
...

Can now increase interval, one previous RTT with the same value

Source: http://technet.microsoft.com/en-us/library/cc182270.aspx

Hearbeat settings

- The heartbeat starts at the default rate.
- The direct push algorithm on the device then dynamically adjusts the heartbeat interval to maintain the maximum time between heartbeats without exceeding the time-out value.
- The rate adjustment is based on the following configuration parameter settings (increments are maximum, can be smaller set by tuning component).
  - HeartbeatDefault (480s=8min)
  - HeartbeatIncrement (300s=5min)
  - HeartbeatMin (480s=8min)
  - HeartbeatMax (1680s=28min)

Why adaptive?

- Two different things:
  - Network timeout
  - Server data available rate
- Too large value → network timeout breaks connection (should be below network timeout)
- Too small value → too many polling operations
- Polling operations delay data retrieval
- Thus dynamic setting of the polling taking the network/server into account

IMAP IDLE

- This solution relies on the existing IDLE (RFC 2177) command to provide instant e-mail notification on the client device
- The IDLE command is often used to signal the ability of a client to process notifications sent outside of a running command
- This can be used to provide a similar user experience to push

Apple Push Notification Service

- APNS usage involves the following steps:
  1. 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.
  2. Applications obtain deviceTokens that are then given to services
  3. 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.
  4. The service or application disconnects from APNS.

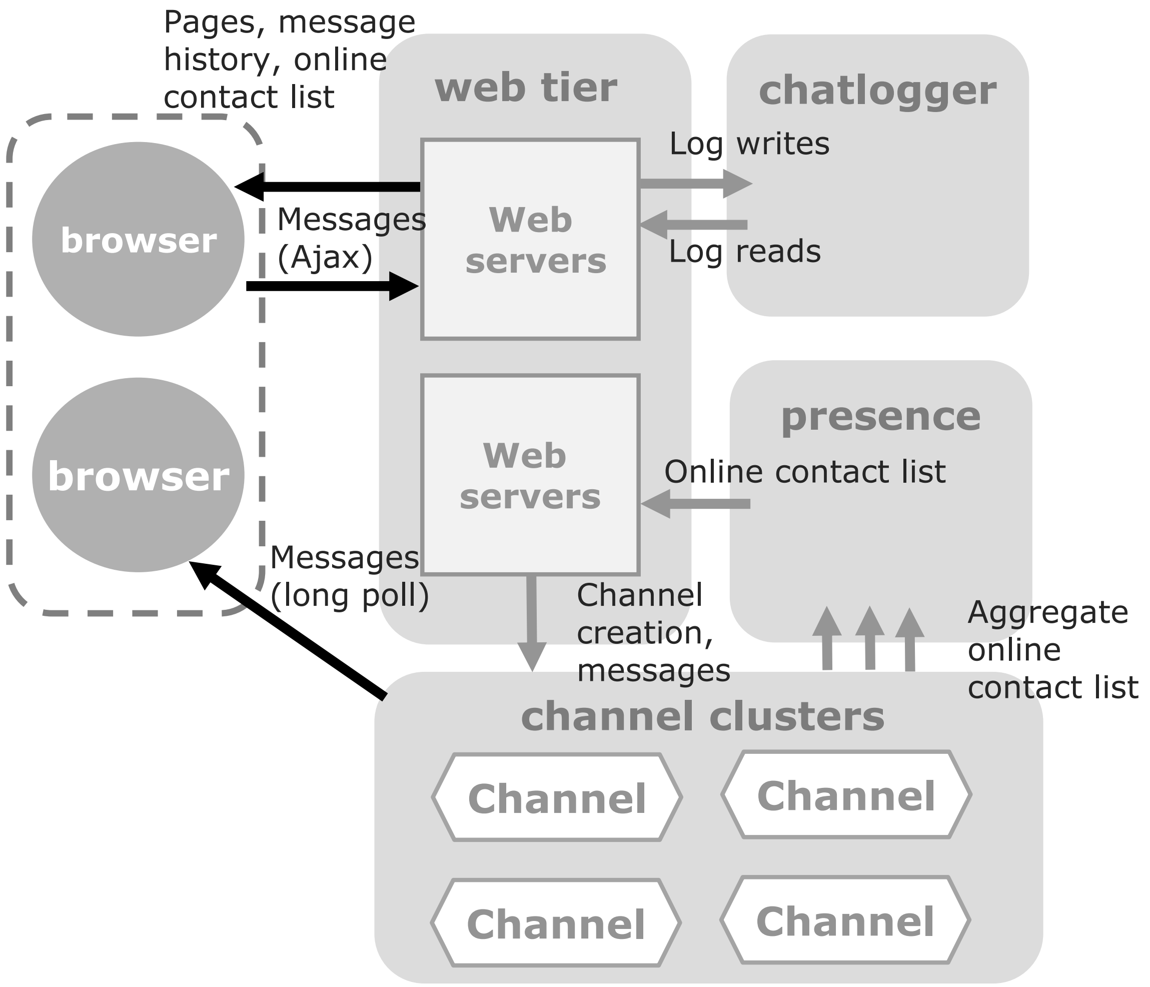
Facebook Chat

- Web-based app that interacts with the backend
- Channel clusters, each cluster is responsible for a subset of users
- Incoming message sent to the channel cluster responsible for the destination
- Regular AJAX for sending messages
- AJAX polling for presence updates
- AJAX long-polling for messages

System details

- Engineering challenges
  - Delivery time minimized with AJAX long-polling
  - Long-lived connection management
  - Status updates (transitions between states generate a lot of traffic)
- Custom web server written in Erlang

Facebook Chat Architecture

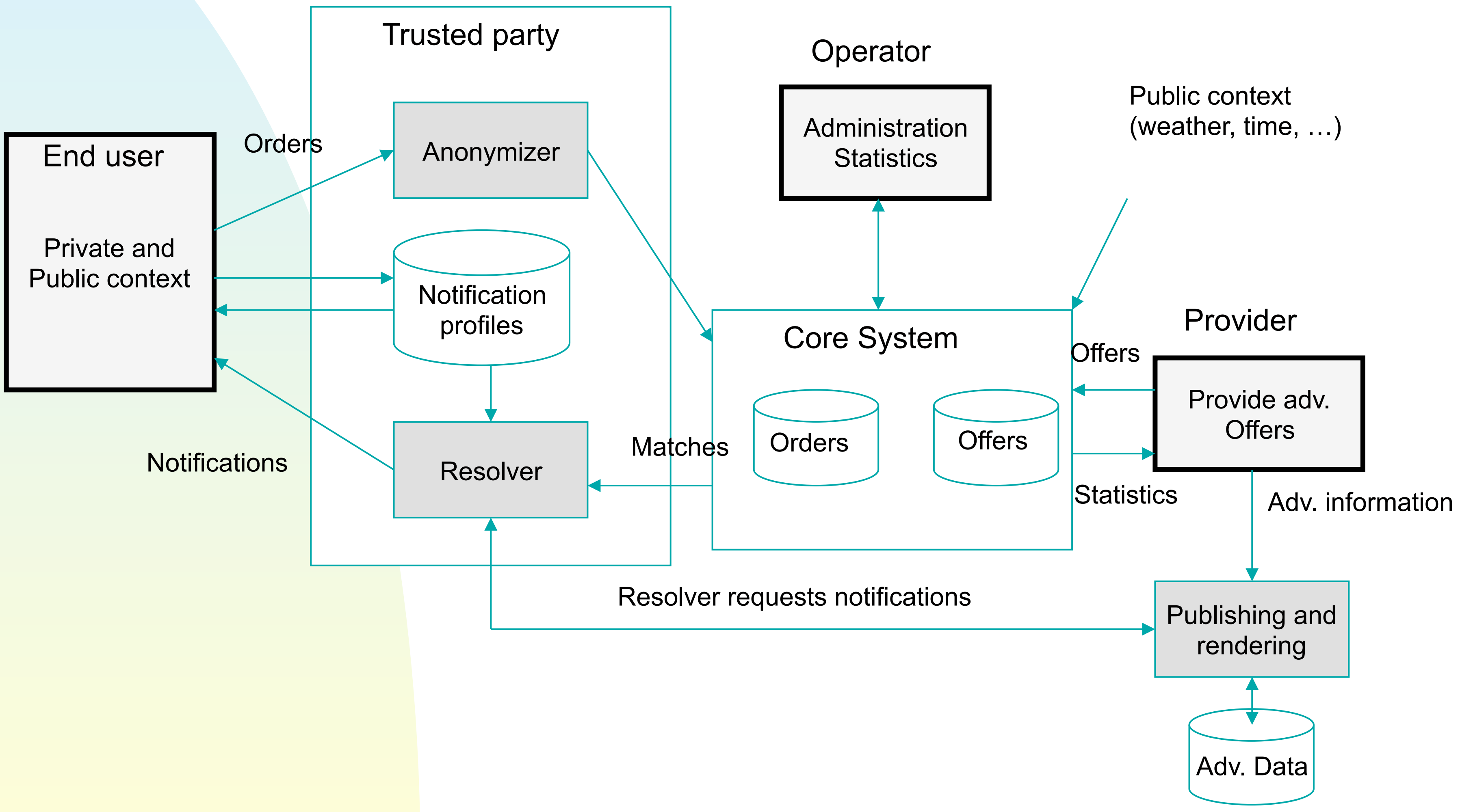


Facebook Messenger

- Social application available on mobile devices
- Chat and presence system that integrates with the backend
- Uses MQTT instead of AJAX
- MQ Telemetry Transport (MQTT) is a lightweight topic-based pub/sub protocol.

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.


Trusted party
End user
Private and Public context
Orders
Anonymizer
Notification profiles
Resolver
Notifications
Matches
Operator
Administration Statistics
Public context (weather, time, …)
Core System
Orders
Offers
Provider
Offers
Provide adv. Offers
Statistics
Adv. information
Resolver requests notifications
Publishing and rendering
Adv. Data

Mobile Video Delivery Techniques

- Server Controlled Techniques
  - **Bitrate Throttling** (send with factor x of encoding rate)
  - **Fast Caching** (use full bandwidth and large receive buffer)
- Client Controlled Techniques
  - **Bitrate Streaming** (at encoding rate)
  - **ON-OFF** using a persistent TCP connection
  - **ON-OFF** using multiple TCP connections
- **Dynamic Adaptive Streaming over HTTP (DASH)**
  - Adapt the quality of the video according the available bandwidth between a client and the server
  - Similar to HTTP Live Streaming (Apple)
- YouTube app uses a selection of these (Throttlng on iOS and Android, Bitrate on N9, ON-OFF with Android(HTML5, Fast caching on Lumia)
- Details: "M. Hoque et al. Dissecting Mobile Video Services: An Energy Consumption Perspective. In WoWMoM, June 2013".

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.
- 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 3/4

- 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, Multi-tier, Lazy synchronization (contacts), Rendezvous, and Remote Proxy.

Revisiting Patterns 2/4

- 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 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.

Recent Trends

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?
- 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