Mobile Application Development MTAT.03.262

Mobile Web Server

Presented by Chii Chang
PhD, Research Fellow
chang@ut.ee
www.ut.ee/~chang
Outline

• Mobile web server
  • Exercise 1
• Mobile web service discovery
• Mobile web server research
  • Exercise 2

Ask question anytime during the lecture.
Part 1.

Mobile Web Servers
What is Mobile Web Server?

• A **mobile web server** is a server-side container (e.g. HTTP server socket) hosted on the application of mobile devices (e.g. smartphone, tablet, media player etc.). It allows the application to provide **mobile web services**.

• **Mobile web service** is provided by a mobile web server, which means:
  • The service does **not have a static connection** because the host device is a moving object.
  • The service may or **may not have a static IP** address.
  • The service **state is highly dynamic**.
  • The **Quality of Service (QoS)** and the **reliability** are **hard to be controlled**.
What is Mobile Web Server?

Classic Mobile Web Server stack

- Web Page
- Web App
- Etc.

- HTTP Service Method
- HTTP Server Socket
- TCP Server Socket
- IP Network
Why Mobile Web Servers are Needed?

Healthcare
• Real-time & on-demand sensing.
• Connecting wearable devices.
Why Mobile Web Servers are Needed?

Internet of Vehicles

• Smartphone as the low cost hub for connected legacy vehicles (Menard & Miller 2011).

• Sensing and integration
  • Wi-Fi monitoring
  • Bluetooth monitoring
  • GPS
  • On Board Diagnostics (OBD) integration
  • Video & audio
  • Wearable connection
  (Rodrigues et al. 2011; Albertengo et al. 2014)
Why Mobile Web Servers are Needed?

Crowd Sensing

Field Working

Data Broker

P2P
Why Mobile Web Servers are Needed?

Internet of UAV (Unmanned Aerial Vehicle)

- Useful for close platform-based UAV that provides only API

Request / Response
Why Mobile Web Servers are Needed?

Phonebot
Devices are Powerful!

Data source: Geekbench 4 (browser.geekbench.com)
History of Mobile Web Server Research

2002


2004-2008


2011-2013


2018


- Chang, C (2018). University of Tartu, Estonia

- Rest

- Rest+ WfMS

- Trust Resource-aware

- Distributed processing

• Lightweight DCE Engine

- Soap

- Soap+ ESB

• Distributed processing

- App Code

- Resource-aware
HTTP Web Server

- Host OS
- TCP / UDP Socket Server
- HTTP Socket Server
- Application Runtime Environment
- Web Application (.NET, JSP, PHP, Node, Python, ObjC etc.)
- Optional: User Interfaces (Web forms, Web pages, specific applications)

Common HTTP Web Server
XML Web Service Server

SOAP Client Application

Request/Response

Discover

Publish

SOAP Messaging

WSDL

Web Application (.NET, JSP, PHP, Node, Python, ObjC etc.)

HTTP Socket Server

TCP / UDP Socket Server

Application Runtime Environment

Host OS

UDDI
Mobile Web Server with ESB

Requester

Atomic Service

Composite Service

Service Bus

Sensor Application

Location Application

Adaptor

Sensor

Mobile Web Server
Mobile Web Server with Workflow Engine

Utilisation example of Mobile Workflow Server

Dynamic Code Execution

Request Package
- Configuration Description
- Context Interpretation Algorithm Source Code
- Program Dependencies / Libraries

RESTful Interface

Code Execution Manager

Data Manager
- Create
- Raw Sensory Data

Work Package
- Configuration Description
- Data
- Source Code
- Libraries

Requester

Receiver
Classic XML Web Services are heavyweight?

- Because TCP causes overhead?
Devices Profile for Web Services (DPWS)

- OASIS Open Standard (2009 —).
- Used in Windows Vista natively.
- **SOAP-over-UDP**.
- WS-Discovery. (support LAN/subnet discovery).
- WS-Eventing. (support publish/subscribe).
- Well-known APIs:
  - WS4D ([http://ws4d.org/](http://ws4d.org/))
  - SOA4D ([https://forge.soa4d.org/](https://forge.soa4d.org/))

- **Drawback**: SOAP is still heavyweight.
RESTful Services

• SOAP vs. REST

REST was introduced by Roy Fielding in his PhD thesis (Fielding; 2000), and is currently an alternative to SOAP-based Web service design models. The major difference between REST and SOAP is the request style. In SOAP, request messages have been formatted in an XML document. In REST, a request is simply sent to a URL with an HTTP request method: GET, POST, PUT or DELETE.

Figure 2.5(a) and (b) illustrate the difference between REST and SOAP using a product order service example. In SOAP (Figure 2.5(a)), each function is an individual operation. In REST (Figure 2.5(b)), three functions, create order, update order and delete order, are sent to the same URL with different HTTP methods and parameters. For example, to get product detail from a SOAP-based Web service (see Listing 2.1), the request message consists of two parts: a HTTP request method and the SOAP message envelope.

Reference:
Chang, C. (2013), Service-Oriented Mobile Social Network in Proximity, PhD Dissertation, Monash University, Australia.
Request Message

SOAP vs. REST

Listing 2.1: SOAP-based request example

GET / HTTP/1.1
Host: www.example.com
Content-Type: application/soap+xml; charset=UTF-8
Content-Length: {length}

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
    <!-- Header information here -->
  </env:Header>
  <env:Body>
    <m:getProductDetail xmlns:m="http://www.example.com/">
      <productID>21</productID>
    </m:getProductDetail>
  </env:Body>
</env:Envelope>

Conversely, a RESTful Web service (REST-based Web service model) request will be a simple HTTP GET request with the corresponding URI, which is shown in Listing 2.2.

Listing 2.2: REST-based request example

GET /product/21
Host: www.example.com
Content-Type: application/x-www-form-urlencoded

A RESTful Web service requires much less data transaction between requester and provider (Pautasso et al.; 2008). However, if additional content needs to be attached with the request message such as standardised semantic annotations, which involves substantial URI namespace management, a SOAP-based Web service is more flexible than a REST-based Web service (Zur Muehlen et al.; 2005). For example, in a P2P-based MSNP environment, a requester intends to retrieve an 'event picture of current location' from other participants by multicasting the request message to other peers. If the requester uses a SOAP request message, the request message can embed semantic annotations to help other participants to understand the request method. Conversely, such flexibility is not supported by REST because HTTP method-based REST does not use standard document-based messages.

Chang, C. (2013), Service-Oriented Mobile Social Network in Proximity, PhD Dissertation, Monash University, Australia.
Web Servers for Devices

• iJetty (Web application container).
• Nodejs
• Tomcat (Web application container).
• CocoaHTTPServer.
• NanoHTTPServer.
• Many others.
HTTP is not Lightweight?

- Emerging constrained network environments.
- HTTP-based RESTful services are too heavy.
Constrained Application Protocol (CoAP)

• Internet Engineering Task Force (IETF) - RFC 7228 standard.
• RESTful
• UDP
• Many API/SDK in different languages and platforms. C, C++, Java, C#, Python, C# Go, JavaScript, Objective-C, Ruby, Swift etc.
• Used in OCF – IoTivity, OMA – Lightweight M2M etc.
• Core protocol standard for constrained WSN, CPS, IoT, M2M etc.
• Return resource description from “/.well-known/core”.
# CoAP vs. HTTP

<table>
<thead>
<tr>
<th>Feature</th>
<th>CoAP</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>UDP</td>
<td>TCP</td>
</tr>
<tr>
<td>Network layer</td>
<td>IPv6 &amp; 6LoWPAN</td>
<td>IP layer</td>
</tr>
<tr>
<td>Multicast support</td>
<td>supports</td>
<td>N/A</td>
</tr>
<tr>
<td>Architecture model</td>
<td>request/response &amp; publish/subscribe</td>
<td>request/response</td>
</tr>
<tr>
<td>Synchronous communication</td>
<td>N/A</td>
<td>required</td>
</tr>
<tr>
<td>Overhead</td>
<td>Less</td>
<td>More</td>
</tr>
<tr>
<td>Application</td>
<td>WSN/IoT/M2M</td>
<td>General</td>
</tr>
</tbody>
</table>

Constrained mobile Sensing as a Service (mSaaS) Reference Architecture

For more information:
Exercise 1

• HTTP Server
  • http://kodu.ut.ee/~chang/Exec_HTTP.pdf

• CoAP Server & Client
  • http://kodu.ut.ee/~chang/Exec_CoAP.pdf
Part 2.

Mobile Web Service Discovery
CHAPTER 2. A REVIEW OF MOBILE WEB SERVICES

A typical example of central registry is Universal Requester Provider Registry Search Register Interact. Figure 2.2: Central registry-based Web service infrastructure

Description, Discovery and Integration (UDDI) which was introduced as a core element of Web service infrastructure at the early stage to provide a central registry for global Web service providers to publish and list their services. However, UDDI did not gain much interest from the industries, and ended up losing support from major companies such as IBM and SAP.

2.2.1.2 Index

An index-based Web service infrastructure is similar to central registry-based architecture in terms of service description metadata publishing. The major difference between index-based architecture and central registry-based architecture relates to who controls the published service metadata. In an index-based Web service architecture, there can be multiple index service providers providing the indexing mechanism, which actively searches services and also allows other service providers to publish/register their services.

Figure 2.3 illustrates an example of index-based Web service architecture. An index service can have its own approach to manage service descriptions and provide its own solution to help requesters search for services. Google search is an example of an index-based service.

2.2.1.3 Peer-to-Peer (P2P)-based Web service infrastructure is a dynamic decentralised network in which a stable registry service or index service is not used. A typical example of P2P-based Web architecture is a group of software entities operating in a MANET environment. In such an environment, peers discover/interact with one another using a specific routing algorithm.

A generic approach, which is shown in Figure 2.4, utilises the request forwarding approach, in which a requester (Peer R) sends its request message to other connected peers. The request is then forwarded to other connected peers until a matching provider is found. The provider then sends a response to the requester.

Figure 2.3: Index-based Web service infrastructure

Figure 2.4: P2P-based Web service infrastructure

Peer-to-Peer

Figure 2.4: P2P-based Web service infrastructure

Chang, C. (2013), Service-Oriented Mobile Social Network in Proximity, PhD Dissertation, Monash University, Australia.
Peer-to-Peer
## A Comparison of Mobile P2P Service Discovery Technologies

<table>
<thead>
<tr>
<th></th>
<th>Apache River</th>
<th>JXTA</th>
<th>Bonjour / Zero-configuration</th>
<th>UPnP</th>
<th>DPWS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Topology</strong></td>
<td>Centralised</td>
<td>Decentralised</td>
<td>Decentralised / centralised</td>
<td>Decentralised / centralised</td>
<td>Decentralised</td>
</tr>
<tr>
<td><strong>Scope</strong></td>
<td>Local</td>
<td>Global</td>
<td>Local / Global (centralised)</td>
<td>Local / Global (centralised)</td>
<td>Local</td>
</tr>
<tr>
<td><strong>Addressing</strong></td>
<td>—</td>
<td>—</td>
<td>Self-assigned addressing</td>
<td>DHCP; self-assigned addressing</td>
<td>WS-addressing</td>
</tr>
<tr>
<td><strong>Naming</strong></td>
<td>—</td>
<td>128 bit unique ID</td>
<td>IETF RFC 6762 – multicast DNS</td>
<td>DNS; IP</td>
<td>WS-addressing</td>
</tr>
<tr>
<td><strong>Discovery</strong></td>
<td>Lookup service</td>
<td>Super-peer Advertisement</td>
<td>IETF RFC 6763 – DNS-based Service Discovery (DNS-SD)</td>
<td>SSDP</td>
<td>WS-Discovery; WS-Metadata Exchange</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Java Object Class Definition</td>
<td>XML</td>
<td>—</td>
<td>XML</td>
<td>WSDL</td>
</tr>
<tr>
<td><strong>Invocation</strong></td>
<td>Java RMI</td>
<td>Pipes</td>
<td>—</td>
<td>SOAP</td>
<td>SOAP; WS-Transfer</td>
</tr>
<tr>
<td><strong>Eventing</strong></td>
<td>—</td>
<td>Pipes</td>
<td>—</td>
<td>GENA</td>
<td>WS-Eventing</td>
</tr>
<tr>
<td><strong>Presentation</strong></td>
<td>Java Class</td>
<td>—</td>
<td>—</td>
<td>HTML</td>
<td>—</td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td>JVM</td>
<td>Cross Platform</td>
<td>Platform Independent</td>
<td>Cross Platform</td>
<td>Platform Independent</td>
</tr>
</tbody>
</table>

Background of Zeroconf

• Zero-configuration (Zeroconf) =
  Multicast DNS (mDNS) +
  DNS-Service Discovery (DNS-SD)

• 2011, Apple submitted Bonjour (Zeroconf implementation of mDNS + DNS-SD) to IETF as an ongoing standard protocol.

• 2013, IETF published the two technologies as two separated standards, RFC 6762 (mDNS) and RFC 6763 (DNS-SD).
Utilisation of Zeroconf

- Zeroconf is supported by Apple’s SDKs and devices by default.
- IBM Security Network Intrusion Prevention System.

  - **Google**
    - Physical Web API is using Zeroconf for Wi-Fi-based approach.
    - Chromecast uses mDNS.
    - Android SDK supports Zeroconf (Network Service Discovery [NSD]) by default.
    - Cloud Print – Privet uses Zeroconf for local mode.

- HP Universal Print Driver uses mDNS.

- Allseen – AllJoyn (*a framework for M2M) uses Zeroconf for local service discovery.

- Avahi (Linux implementation of Zeroconf) is installed by default on Debian and Ubuntu.

- Many other IoT/CPS protocols are also using Zeroconf......
Multicast DNS

• Ease-of-use auto-configuration IP networking.
• IPv4 or IPv6.
• Example domain: "_http._tcp.", "_ssh._tcp.", "_coap._udp." etc.
• Example address: "ThinkPad-XXX.local.:80" or "site.com.:80".
• Uses UTF-8, and only UTF-8, to encode resource record names.
• Allows names up to 255 bytes plus a terminating zero byte.

DNS-based Service Discovery (DNS-SD)

• Query a type of services in a domain.
• Resolving host name instance.
• Instance names are relatively persistent. (i.e. A printer that has been used today should be still available tomorrow by selecting the name even the address of printer has changed.)
• Service instance name = <instance> . <service> . <domain>.
  • example: “Building 2, 1st Floor . example . com .”

Bluetooth-based Service Discovery

- Bluetooth LE Advertisement
- iBeacon by Apple
- Physical Web by Google

Programmable iBeacon

物理Web Eddystone Beacon

(Source: amazon.com)
## iBeacon vs. Physical Web

<table>
<thead>
<tr>
<th></th>
<th>iBeacon</th>
<th>Physical Web</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td>Apple products, APIs, many hardware manufacturers</td>
<td>Eddystone beacons</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Public specification managed by Apple</td>
<td>Open source; Apache 2.0 Licence</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td>Bluetooth Low Energy beacons; require Apple’s</td>
<td>Free of manufacturing</td>
</tr>
<tr>
<td></td>
<td>certification to manufacture iBeacon</td>
<td></td>
</tr>
<tr>
<td><strong>Beacon Payload</strong></td>
<td>ProximityUUID, major and minor code</td>
<td>URL formatted</td>
</tr>
<tr>
<td><strong>Alerts / Background</strong></td>
<td>Allows background notification for iOS</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Hybrid Solution – Mesh Networking
Mobile Mesh Networking

(Source: cnn.com*)

*CNN, “FireChat in Hong Kong: How an app tapped its way into the protests”
Mobile Mesh Network SDKs

- HypeLabs ([https://hypelabs.io](https://hypelabs.io))
- NewAer ([https://newaer.com](https://newaer.com))

(Source: newaer.com)
Part 3.

Mobile Web Servers Research
Do more with mobile web servers

• Internet-based mobile P2P computing
• Proximity-based mobile crowd computing
• Integrating mobile host with fog computing servers
Mobile P2P Cloud

- Infrastructure-based
- Internet-based P2P

Mobile Crowd Computing

• Infrastructure-less
• Direct P2P-based

Fog Computing

- Cloud
- Gateway
- IoT Device
- Sensor
- Onboard Computer
- Hub
- Fog
- Gateway
- Device
Proactive Mobile Fog Computing

(Soo et al. 2017)
Opportunistic Process Distribution and Migration (1/3)

Request Package
- Configuration Description
- Context Interpretation
- Algorithm Source Code
- Program Dependencies / Libraries

RESTful Interface
- CaaS Manager
- SaaS Manager

Create
- Raw Sensory Data

Work Package
- Configuration Description
- Data
- Source Code
- Libraries

Requester

MWServer
Opportunistic Process Distribution and Migration (2/3) – Process Distribution Example

<table>
<thead>
<tr>
<th>Work Package</th>
<th>Source Code</th>
<th>Libraries</th>
<th>Data</th>
<th>Configuration Description</th>
</tr>
</thead>
</table>

- **MWServer**
  - Requester
  - Output Package

- **Indie Fog Server**
  - Output
  - Work Package
  - RESTful Interface
  - Process Runtime Engine
Opportunistic Process Distribution and Migration (3/3)
– Process Migration Example

- Work Package
  - Configuration Description
  - Source Code
  - Data
  - Libraries

- Indie Fog Server
  - RESTful Interface
  - Output
  - Work Package
  - Process Runtime Engine

- MWServer

- Requester
  - Output
Project – EPIoT
Self-managed Things

• Edge Process Internet of Things (EPIoT)
  o **EPIoT-N**: EPIoT module for NodeJS
  o **EPIoT-A**: EPIoT API for Android Things
  o **EPIoT-X**: EPIoT for embedded systems
EPIIoT Example

(Chang & Srirama 2018)
Exercise 2

- Network Service Discovery (NDS; mDNS)
  - [http://kodu.ut.ee/~chang/Exec_NSD.pdf](http://kodu.ut.ee/~chang/Exec_NSD.pdf)
References


• Menard, T., & Miller, J. (2011, June). Comparing the GPS capabilities of the iPhone 4 and iPhone 3G for vehicle tracking using FreeSim_Mobile. In Intelligent Vehicles Symposium (IV), 2011 IEEE (pp. 278-283). IEEE.


• Albertengo, G., Buttazzo, W., Tragno, A., Ricca, M., Bragagnini, A., & Quasso, R. (2014, June). Smartphone enabled connected vehicles pave the way to intelligent mobility. In WTC 2014; World Telecommunications Congress 2014; Proceedings of (pp. 1-6). VDE. Chicago