MOBILE SERVER PLATFORM

Architectures and Protocols for Future M2M Ecosystems

Fahad Aijaz

ComNets Research Group
RWTH Aachen University

March 16, 2012
Outline of Talk

• Research Area in a Nutshell
• Motivation and Research Gap
• The Mobile Server Platform
  – The Mobile Web Server Layer
    • Architecture, Payload and Performance Aspects
    • Multimedia Extensions and Topologies
  – The SLA Framework Layer
    • Components and Life Cycles
• Performance Analysis
• The Software Application Layer
  – MEDICARE – A Context-Aware Health Care Application
  – Social Network in Pocket (SNiP)
• Conclusion
Today’s Internet and the WWW

→ High-tech Web Servers.
→ Hosts Web Service and Resources.
→ Transparent Access to the Clients.
→ Neutral towards diverse clients.

Internet of Things = M2M

M2M Terminal

CONSUMER + PROVIDER

M2M Terminal

CONSUMER + PROVIDER

Web Server

TRANSPARENT ACCESS

WEB SERVICES
- Specialized functions
- Internal process
- Access interface

RESOURCES
- Private Data
- Multimedia
- Websites

Publish/Search/Outsource

Web Service Broker/Cloud Computing

M2M Services
Motivation and Research Gap

Mobile devices are not only mobile phones!

PROPERTIES
→ Only request-response based.
→ Immediate service invocation, only.
→ Short-lived executions.
→ Respond on same network infrastructure (blocking call!).
→ Runtime service management is not possible.

Industrial Automation
Automotive / Telemetric
Smart Energy Grids
Health Care Sector
Home Automation
Social Networking
Web 2.0 Services

Transform into

Mobile Server Platform

M2M Domain Requirements
M2M Applications

Provides a general middleware architecture and protocol stack for rapid development of M2M applications

This research gap has been confirmed through literature!
The Mobile Server Platform
(Conceptual Overview)

The MSP provides a set of strategically connected architectures
Offers a Web Services provisioning solution for M2M apps
The platforms architecture is categorized into logical layers

- Notification
- Solicit-Response
- Request-Response
- One-way

The MSP offers a set of strategically connected architectures for Web Services provision for M2M apps. The platforms architecture is categorized into logical layers:

**Software Application Layer**

**Service Interaction Strategies**

- SOAP Access Interface
- REST Access Interface

**SLA Framework Layer**

**Mobile Web Server Layer**

- SOAP
- REST
- SIP
- RTSP
- RTP

**Mobile Interfaced Mobile Web Services**

- HTTP/TCP
- RTP/UDP
- IEEE 802.15.4

**Synchronous Mobile Web Services**

**Asynchronous Mobile Web Services**

**Applications’ private Data**

- Health Care
- Social Networks

**Service Interaction Strategies**

- Asynchronous Service Access Protocol
- SLA Life Cycles
- Web Service Agreements
- Agreement Protected Services

**Server Endpoints**

- Server Process Classification
- Multimedia Delivery & Control
- Multi-Interfaced Services

**OASIS**

Asynchronous Server Endpoints

Extends
The Mobile Web Server Layer
(Classification of Server Processes)

Mandatory process to consume asynchronous Mobile Web services

Involved Server Endpoints

Service Factory

Service Observer

Service Instance

CreateInstance

Create with Observer EPR

Request with Observer EPR

Response with Instance EPR

Completed

Service Computation Result
(normal completion)

Notification Listener

scheduled invocation

immediate invocation

mutually exclusive service invocations

solicit-response / notification

1a

1b

2

3a

3b

4

Observer

Factory

Instance

ASYNCHRONOUS
MOBILE WEB SERVICE

© Fahad Aijaz, ComNets, RWTH Aachen University
The Mobile Web Server Layer
(Classification of Server Processes)

Only possible subsequent to the Service Creation Process

Service Creation Process

Service Management Process

Request with Observer EPR

Response with Instance EPR

Notification Listener

GetProperties
Unsubscribe
Subscribe
SetProperties
ChangeState

Service Instance

mutually exclusive service states

GetProperties
StateChanged

STATES OF AN
ASYNCHRONOUS MOBILE WEB SERVICE

open.running
open.notrunning
open.notrunning.suspended
closed.abnormalCompleted.terminated
closed.abnormalCompleted
closed.completed
closed.abnormalCompleted.aborted

ASYNCHRONOUS
MOBILE WEB SERVICE

CONTROLLABLE
INTERNAL

Events and State Notifications
(solicit-response / notification)
The Mobile Web Server Layer
(SaaR Design Approach)

Access Interface

| Service Provider specifies the mapping semantics for each method |

**S** - State
Every resource changes the client’s state

**T** - Transfer
Resources are transferred using HTTP

**R** - Representational
Network resources
Example: HTML, XML, JPG, GIF...

**SYNCHRONOUS URL ELEMENTS**

HTTP: // IP : PORT / SERVICE / METHOD

**Mandatory** | **Optional**

**ASYNCHRONOUS URL ELEMENTS**

HTTP: // IP : PORT / SERVICE / METHOD / STRATEGY / ENDPOINT / OPERATION

**Mandatory** | **Optional**

Synchronous Access URL

Asynchronous Access URL

**Service Logic**

- **CREATE**
- **READ**
- **UPDATE**
- **DELETE**
The Mobile Web Server Layer
(Payload Optimization)

XML payload comparison of the operations offered by Service Factory Endpoint

- CreateInstanceRs: ≈ 82% reduction
- CreateInstanceRq: ≈ 67% reduction
- ListInstanceRs: ≈ 67% reduction
- ListInstanceRq: ≈ 83% reduction
- GetPropertiesRs: ≈ 67% reduction
- GetPropertiesRq: ≈ 95% reduction

JSON/REST results in much optimized payload than the XML/REST
The Mobile Web Server Layer
(Basic Analysis of XML and JSON Payloads)

XML Encoding

\[
S_{XML} = 5 + (2 \times B_{root\_name}) + \sum_{e=1}^{n} 5 + (2 \times B_{element\_name})
\]

7 bytes allocated by the XML root tag of 1 byte

JSON Encoding

\[
S_{JSON} = 2 + \sum_{e=1}^{n} \left( 7 + B_{element\_name} \right)
\]

2 bytes allocated by the start and end braces of JSON element
The Mobile Web Server Layer
(Basic Analysis of XML and JSON Payloads)

Difference trend with $2^5$ elements

\[ S_{XML} - S_{JSON} \]

JSON results in less data transmission as number of elements increase.

Lengthy element names result in much rapid increase is payload.

JSON/REST for M2M terminals $\rightarrow$ Less computation requirements.
The Mobile Web Server Layer
(Multimedia Extensions and Topologies)

Integration of RTSP and RTP

Extended RTSP States in Asynchronous Server

Factory Instance Observer

Multi-Interfaced Mobile Web Services

Software Application Layer

Mobile Web Server Layer

RTSP OPTIONS and DESCRIBE methods are exposed as synchronous mobile Web Services.

The RTSP SETUP, PLAY, PAUSE and TEARDOWN are exposed as states of asynchronous services.

Delivery Resource \rightarrow Asynchronous Services
Control Resource \rightarrow Synchronous and Asynchronous Services
Multimedia Delivery Strategy \rightarrow RTP/UDP
Multimedia Control Strategy \rightarrow RTSP/TCP

Offers a Web Service provisioning solution through M2M terminals

© Fahad Aijaz, ComNets, RWTH Aachen University
### Works with every network where TCP/UDP are enabled!

1. **Server establishes a permanent TCP tunnel with the IAG → TCP hole punching**
2. **The client sends RTSP SETUP to the IAG → IAG relays it to server → SCP response**
3. **Only client sends UDP to IAG → IAG is not introducer gateway → sends keep-alive to client**
4. **Client sends RTSP PLAY to IAG → IAG relays it to server → Service Management Process**
5. **Server indirectly delivers media over the RTP/UDP channel that is relayed by the IAG**
6. **Client may send RTSP PAUSE, TEARDOWN via IAG → Service Management Process**
The functions of the SLA architecture are classified into 4 distinct life cycles. The SLA life cycles are executed based on the incoming mobile Web Service requests. The SLA negotiation is based on the Web Service Agreement standard of the Open Grid Forum. The standard is optimized to define the SLA messaging for mobile nodes. The SLA framework is compatible with the REST and SOAP access interfaces. Protects both, the synchronous and asynchronous services.

The life cycles utilize the properties of the Synchronous and Asynchronous Servers.
The SLA Framework Layer
(Life Cycles)

Template Acquisition Life Cycle

A) Reads and manipulates the template
B) Generates a UUID for the client
C) Saves a copy against the UUID

REST REQUEST
http://xpert.comnets.de:9090/FetchTemplate

A) Associated agreement template with validity
B) Must be used before expiry
C) Automated deletion

1. MOBILE WEB SERVICE
   CLIENTS

2. MOBILE WEB SERVICE
   HOST

PROTECTED
MOBILE WEB SERVICE

AC
AE

Agreement
Creation
Phase

Agreement
Evaluation
Phase

Agreement
Offer

Agreement
Template

NOTIFICATION

NEGOTIATED
AGREEMENT

Notation
The SLA Framework Layer
(Life Cycles)

Template Acquisition Life Cycle

1. **MOBILE WEB SERVICE**
   - **CLIENTS**
   - **MOBILE WEB SERVICE HOST**
   - **PROTECTED MOBILE WEB SERVICE**

   **Template Acquisition Life Cycle**
   - **Agreement Creation (AC)**
     - Template Acquisition Life Cycle
   - **Agreement Offer Life Cycle**
     - **Agreement Evaluation (AE)**

   **Fetch Template**
   - A) Reads and manipulates the template
   - B) Generates a UUID for the client
   - C) Saves a copy against the UUID
   - REST REQUEST
     - http://xperia.comnets.de:9090/FetchTemplate

   **Agreement Offer Life Cycle**
   - A) Associated agreement template with validity
   - B) Must be used before expiry
   - C) Automated deletion

   **Agreement Offer**
   - **MOBILE WEB SERVICE**
     - **HOST**
     - **PROTECTED MOBILE WEB SERVICE**

   **Agreement Creation (AC)**
   - a) Agreement creation phase is completed
   - b) Agreement evaluation phase is started

   **Agreement Evaluation (AE)**
   - **MOBILE WEB SERVICE**
     - **HOST**
     - **PROTECTED MOBILE WEB SERVICE**

   **Notification**
   - a) Agreement creation phase is completed
   - b) Agreement evaluation phase is started

   **Negotiated Agreement**
   - A) Evaluate the agreement offer against the related template
   - B) Accept or reject the agreement offer
   - C) Save a copy (if accepted) & notify the client
The SLA Framework Layer

(Life Cycles)

Service Invocation Life Cycle

A) Service provider specifies the QoS handlers during deployment
B) Reads the service settings
C) Starts the associated QoS handlers
D) QoS handlers monitors and reacts upon QoS violations

MOBILE WEB SERVICE HOST

SLA

QoS Violations

Service Invocation + UUID

Service Response

Notification

For asynchronous services

Agreements and Agreement Templates to dispose!

Explicit disposal requests are possible only from the permitted clients through the client-controlled process.

Third-party QoS handlers are possible!
The SLA Framework Layer (Life Cycles)

**Service Invocation Life Cycle**

1. **Agreement Evaluation (AE)**
   - Service Invocation Life Cycle

2. **QoS Monitoring (QM)**

3. **MOBILE WEB SERVICE CLIENTS**
   - A) Verify and evaluate request against the negotiated agreement
   - B) Verify the usage limit & the usage interval (default)
   - C) Invoke or schedule the service
   - D) Initiate the QoS monitoring

4. **SERVICE INVOCATION + UUID**
   - **SERVICE RESPONSE**
     - A) Service provider specifies the QoS handlers during deployment
     - B) Reads the service settings
     - C) Starts the associated QoS handlers
     - D) QoS handlers monitors and reacts upon QoS violations

5. **SETTINGS**
   - For asynchronous services
   - QoS VIOLATIONS

6. **MOBILE WEB SERVICE HOST**
   - Third-party QoS handlers are possible

**Agreement Disposal Life Cycle**

1. **Disposal Monitoring (DM)**
   - Agreement Disposal Life Cycle

2. **MOBILE WEB SERVICE CLIENTS**

3. **AGREEMENT DISPOSAL + UUID**
   - **DISPOSAL RESPONSE**
     - A) Offers periodic cleanup cycles in automatic disposal
     - B) Looks for the expired agreements and templates
     - C) Takes the agreement validity (end date) and usage limit as the expiration criteria
     - D) Disposes agreements, templates and client records (e.g. UUID)
     - E) Shared process for all agreements and templates

4. **DISPOSAL RESPONSE**
   - Automatic disposal is a default process
   - Agreements and Agreement Templates to dispose

---

© Fahad Aijaz, ComNets, RWTH Aachen University

18
Performance Analysis
(Service Creation Process)

Synchronous Server

1. Web Service Request
2. Parse Request
3. Create Service Object
4. Complete

- Process: Running
- Event: Notify

Asynchronous Server

1. Web Service Request
2. Parse Request
3. Create Service Object
4. Complete

- Process: Running
- Event: Notify

- Until the service object is created!
Performance Analysis
(Service Creation Process)

Asynchronous SCP with SOAP

Synchronous SCP with SOAP

\[ d^{\text{syn}}_{\text{soap}}(x) = D_S + (e_{\text{in}} + e_{\text{st}} + e_{\text{k=syn}}) + d_{\text{obj}} \]

\[ \frac{d^{\text{syn}}_{\text{soap}}}{d^{\text{parc}}_{\text{syn}}} = \frac{d^{\text{syn}}_{\text{soap}}}{D_{\text{c}}} \]

\[ d^{\text{asyn}}_{\text{soap}}(x) = D_S + (e_{\text{in}} + e_{\text{st}} + e_{\text{k=asyn}}) + d_{\text{obj}} \]

\[ \frac{d^{\text{asyn}}_{\text{soap}}}{d^{\text{parc}}_{\text{asyn}}} = \frac{d^{\text{asyn}}_{\text{soap}}}{D_{\text{c}}} \]
Performance Analysis
(Service Creation Process)

Asynchronous SCP with XML/REST

Synchronous SCP with XML/REST

Asynchronous Access URL

- GPSLocation
- Coordinates
- Req-Res
- Factory
- CreateInstance

Synchronous Access URL

- echoString
- (optional)

HTTP: // IP : PORT / SERVICE / METHOD / STRATEGY / ENDPOINT / OPERATION

XML/REST Example

```xml
<createInstanceRq xmlns=""
  xsi:type="createInstanceRq">
  <StartImmediately xsi:type="xsd:boolean">true</StartImmediately>
  <Name xsi:type="xsd:string">GPS Location</Name>
  <Subject xsi:type="xsd:string">Create Service Instance</Subject>
  <Description xsi:type="xsd:string">Provides GPS coordinates of the host mobile</Description>
  <ContextData xsi:type="xsd:ContextData">
    <!– user-defined input XML elements –>
  </ContextData>
</createInstanceRq>
```

Mathematical Formulation

\[ d_{rest}(x) = D_s + d_{obj} \]

\[ D_m^k(X) = \frac{\sum_{i=1}^{N} d_m^k(X)_i}{N} \]

Hypothesis

\[ D_{rest}(x) < D_{soap}(x) \]

\[ D_{rest}(x) < D_{asyn}(x) \]
Performance Analysis
(Service Creation Process)

M/D/1 evaluation with increasing arrival rate

\[ \lambda = 35 \]
\[ \lambda = 50 \]
\[ \lambda = 205 \]
\[ \lambda = 380 \]

\[ \approx 6 \text{ times increase in serving capacity with XML/REST} \]

Mean waiting time with XML/REST at this arrival rate is about 2% of the one with SOAP

\[ D_{\text{rest}}(X) < D_{\text{soap}}(X) \]
The Software Application Layer
(MEDICARE – A Context-Aware Health Care Application)

Device Configuration
Mode Selection
Performance Monitor
Valves Condition
Oxygen Monitor

Equipment

Monitor
Ventilator

Light Condition
Room Temperature
Noise Detection
Presence
Door Status

Environment

Light
Heater
Door

Mobile Web Service Categories

Patient

Body Temperature
Blood Pressure
Heart Condition
Pulse Signals
Movement

Diagnostic

Mobile Web Service Categories

SPOT nodes with the MSP
(collector nodes)

Mobile Web Service Categories

Doctor’s Computer

SPOT Base Station

IEEE 802.15.4
CreateInstance...
ChangeState...

© Fahad Aijaz, ComNets, RWTH Aachen University
The Software Application Layer
(Social Network in Pocket [SNiP] – A privacy-focused social network concept)

1. Alice, a SNiP user is on vacation in Italy
2. She has many SNiP services on her mobile phone, such as, location, gallery, chat, games, video, voice...
3. She has taken pictures, recorded videos and posted comments about many wonderful places during her Italian tours
4. To socialize and share her experience, Alice allows her selected friends from her phones address book to access her SNiP services
5. Her authorized friends directly access her private data stored on her phone through the Service Access website to interact and view the mobile content that Alice has provided for shared use
6. Access denied
7. She feels satisfied, as SNiP allows her to share her private mobile content without posting it to any third-party server – PRIVACY!
8. Alice feels like her entire Social Network is in her pocket

Service Access Website
SNiP-Consumers

Disallowed SNiP-Consumers

SNiP-Provider (MSP)

Locate and Track Alice
View Alice’s picture gallery
Live Web-to-Mobile Chat
Live Web-to-Mobile Games
Live video streaming
Leave voice messages

... and more
Conclusion

Contributions of the Research

- **Novel software architecture and protocol stack, called Mobile Server Platform**
  - Offers multiple execution models, protocols and interaction strategies
  - Solution for data privacy, remote service creation and management on mobile terminals

- **The MSP is developed and evaluated with multiple access interfaces**
  - JSON/REST
    - most suitable for M2M terminals, as the payload increases
  - XML/REST
    - increases 6.0 / 7.6 times the serving capacity of Asynchronous / Synchronous Server, respectively
  - The established hypotheses are verified by M/D/1 evaluations

- **Multiple service execution models leads to new architectural extensions**
  - Multimedia control and delivery through the Mobile Web Server layer
    - Solution to bypass network constraints
  - Contracts-based services through the SLA framework layer → Solution to service protection on terminals

- **The MSP is suitable for M2M terminals and consumer devices**
  - Social Network in Pocket (SNiP) → Solution to data privacy issues
  - The MEDICARE prototype → Solution to context-aware M2M environments
End of Talk
Thank you for your attention!

Fahad Aijaz
fah@comnets.rwth-aachen.de
www.fahadaijaz.com