A Presence Server For Context-Aware Applications

Mohammad Zarifi Eslami

University of Twente
25 August 2008
Outline

- Context-aware applications
- Goal of the project
- SIP, SIP-SIMPLE, XML, PIDF
  - Subscribe, Publish, Notify
- SER – Overall architecture
- SER “PA” module
- Evaluation
- Conclusion & Future Work
- Questions?
Context-aware Applications

- Presenting context information as content to the user
  ✓ Presenting a choice of printers close to the user.
  ✓ Presenting a list of available rooms for booking for a meeting.
  ✓ Presenting heart rate, blood pressure, weight, ... (U-Care project).

- Automatically executing a service, triggering actions, or reconfiguring a system on the occurrence or change of a context
  ✓ Sending reminders when a user is in the specific location.
  ✓ Advice human being when doing exercise based on his healthcare factors (U-Care).

- Deciding upon and performing an action on behalf of the user (based on context information and user preferences)
  ✓ Rejecting a phone call when a user is on the meeting.
  ✓ Playing appropriate music based on human being mental state

- Attaching context to information for later retrieval
  ✓ Attaching date/time/weather information when taking a picture for advanced process of printing picture later.

- ...
Goal of the project

1. Obtaining context information
2. Read and process this data
3. Storing the valuable information
4. Sending the relevant context
5. Removing expired Watchers and context information (Publish messages)

Context is any information that can be used to characterized the situation of an entity (a person, place, or object)
Session Initiation Protocol (SIP)

✔ Strongly associated with **IP telephony**, but there are additionally uses.

✔ A **text-based** protocol, similar to **HTTP** for initiating interactive communication sessions between users.

✔ Creating, modifying, and terminating **sessions** (voice, video, chat, interactive games, virtual reality, etc.) with one or more participants.

✔ SIP utilizes **request-response** messages

✔ **Why** SIP: Powerful, easy to implement, scalable, open standard, a text-based protocol (human-readable), supports mobility and IMP
SIP for Instant messaging and presence leveraging extensions (SIP-SIMPLE)

- An extension of SIP for Instant Messaging and Presence (IMP).
- An open standard, for exchanging messages within a SIP session
- Provide a subscription based framework for an event notification

**Watcher**: interested in learning about updates to presence information (using Subscribe messages), A **notifier** (known as presentity): provides presence information to interested watchers (using Notify message), A **Presence User Agent** (PUA): provides presence information for a presentity (using Notify messages).

*SIP-SIMPLE is a method for distributing context information among entities.*
Extensible Mark-up Language (XML)

✓ Used to **share data** across different information systems (e.g., Internet)
✓ A **strict** language in comparison with HTML
✓ Can be used on a wide variety of platforms with a wide variety of tools
✓ Flexible, readable both by humans and machines, easy to create and extend

Context modeling can be represented as an XML document

```xml
<healthiness>
  <heart beat>98</heart beat>
  <blood pressure>135mmHg,84mmHg</blood pressure>
  <blood sugar>5mM/L</blood sugar>
</healthiness>
```
Presence Information Data Format (PIDF)

- Provides a means for transferring presence information in a domain without modification.
- A well formed XML document
- Covers the minimal model of Instant Messaging and Presence Protocol (IMPP), using some basic elements
- Extensible and flexible

```xml
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:location="http://it.kth.se/~moze/schemas/mohammad.xsd"
"entity="sip:ccsleft@130.237.15.238"
<tuple id="6sJ8J0">
<status><basic>open</basic>
<healthiness>
<heart beat>98</heart beat>
<blood pressure>195mmHg, 114mmHg</blood pressure>
<blood sugar>5mM/L</blood sugar>
</healthiness>
</status>
<note>Blood pressure is high</note>
<contact>Mohammad</contact>
</tuple>
</presence>
```

Using basic elements

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
"entity="sip:ccsleft@130.237.15.238"
<tuple id="6sJ8J0">
<status>
<basic>open</basic>
</status>
<note>Blood pressure is high</note>
<contact>Mohammad</contact>
</tuple>
</presence>
```

Using extended elements

PIDF can carry context information in the body of Publish and Notify messages.
✓ A high-performance, configurable, open source SIP server.
✓ Can receive and process SIP messages to enable intelligent VoIP Services
✓ Open source & now a commercial system; iptel.org website is the main entry for further information
✓ Used to register Watchers in a database (MySQL), enabling SIP messages (Subscribe, Publish, and Notify) to be routed between clients, service agents, applications, and etc.
A Presence Agent (PA) module for SER

✓ A new module for SER to implement a presence server.
✓ Initially: designed and implemented my own module
✓ Iptel.org had recently add a new module for SER as a PA (only presence event)
✓ Extended their source code, to support different kind of events (such as location)

SER has different modules for different purposes, such as cpl, acc, mysql, PA, etc.

Ser.cfg controls which modules should be loaded and how the module should behave

M1: Subscribe, M2: OK, M3: Notify, M4 : OK, M5: Publish for updates
M6: OK, M7: Notify, M8: OK, M9: Notify (for expiration of the Publish), M10: OK
Evaluation

- **To determine the efficiency of the proposed presence server**
  - ✓ Six different scenarios ranging from simple to complex
  - ✓ Checking for correct functioning of the server
    
    *(the correct Notify messages ➔ interested Watchers)*

  - ✓ Measuring the performance of the server in terms of response time
  - ✓ Determine the scalability of the server

    *(i.e., multiple Watchers trying to access the same context)*

- **Some expected results**
  - ✓ Server works correctly in each of the different scenarios
  - ✓ One unit in the ’Expire’ field of both the Subscribe and Publish messages means approximately 1 second at the server
Repeating the Notify message

- The Notify message is retransmitted, when the OK is missing from Watchers
  ✓ Retransmission continue until the Subscribe message expires
  ✓ When there is a limited number of Watchers → Not a big problem
  ✓ If there are a lot Watchers → it can be a real bottleneck

<table>
<thead>
<tr>
<th>No.</th>
<th>Time (seconds)</th>
<th>Source</th>
<th>Destination</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>8.878187</td>
<td>130.237.238.63</td>
<td>130.237.15.238</td>
<td>Request: SUBSCRIBE sip:ccsleft@130.237.15.238</td>
</tr>
<tr>
<td>M2</td>
<td>8.883079</td>
<td>130.237.15.238</td>
<td>130.237.238.63</td>
<td>Status: 200 OK</td>
</tr>
<tr>
<td>M3</td>
<td>8.884546</td>
<td>130.237.15.238</td>
<td>130.237.238.63</td>
<td>Request: NOTIFY sip:Sub1@130.237.238.63</td>
</tr>
<tr>
<td>M3</td>
<td>9.363678</td>
<td>130.237.15.238</td>
<td>130.237.238.63</td>
<td>Request: NOTIFY sip:Sub1@130.237.238.63</td>
</tr>
<tr>
<td>M3</td>
<td>10.36364</td>
<td>130.237.15.238</td>
<td>130.237.238.63</td>
<td>Request: NOTIFY sip:Sub1@130.237.238.63</td>
</tr>
<tr>
<td>M3</td>
<td>12.36361</td>
<td>130.237.15.238</td>
<td>130.237.238.63</td>
<td>Request: NOTIFY sip:Sub1@130.237.238.63</td>
</tr>
<tr>
<td>M3</td>
<td>16.36348</td>
<td>130.237.15.238</td>
<td>130.237.238.63</td>
<td>Request: NOTIFY sip:Sub1@130.237.238.63</td>
</tr>
</tbody>
</table>
Basic performance

✓ The server can accept and handle the Subscribe and Publish messages within less than 10ms

Only Subscribe messages

Only Publish messages

M1: Subscribe, M2: OK, M3: Notify, t1 .. t8 are 8 samples

M1: Publish, M2: OK, t1 .. t8 are 8 samples
Scalability limits

- There is a rate limit for Notify message about one per second (configurable in the server’s ser.cfg file).
- Even though the server can receive Publish messages quickly, it can not notify Watchers of updates faster than one update per second.
- The server notifies Watchers about the latest context information, which is desired.

<table>
<thead>
<tr>
<th>M5</th>
<th>58.788145</th>
<th>130.237.239.12</th>
<th>130.237.15.238</th>
<th>SIP/XML</th>
<th>Request: PUBLISH sip:ccsleft@130.237.15.238</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>M6</td>
<td>58.793403</td>
<td>130.237.15.238</td>
<td></td>
<td>SIP</td>
<td>Status: 200 OK</td>
<td>-</td>
</tr>
<tr>
<td>M5</td>
<td>59.088904</td>
<td>130.237.239.12</td>
<td>130.237.15.238</td>
<td>SIP/XML</td>
<td>Request: PUBLISH sip:ccsleft@130.237.15.238</td>
<td>Library</td>
</tr>
<tr>
<td>M6</td>
<td>59.094913</td>
<td>130.237.15.238</td>
<td></td>
<td>SIP</td>
<td>Status: 200 OK</td>
<td>-</td>
</tr>
<tr>
<td>M7</td>
<td>59.160219</td>
<td>130.237.15.238</td>
<td>130.237.239.213</td>
<td>SIP/XML</td>
<td>Request: NOTIFY sip:Sub1@130.237.239.213</td>
<td>Library</td>
</tr>
<tr>
<td>M8</td>
<td>59.230333</td>
<td>130.237.239.213</td>
<td>130.237.15.238</td>
<td>SIP</td>
<td>Status: 200 OK</td>
<td>-</td>
</tr>
<tr>
<td>M5</td>
<td>59.594796</td>
<td>130.237.239.12</td>
<td>130.237.15.238</td>
<td>SIP/XML</td>
<td>Request: PUBLISH sip:ccsleft@130.237.15.238</td>
<td>Cafeteria</td>
</tr>
<tr>
<td>M6</td>
<td>59.599527</td>
<td>130.237.15.238</td>
<td></td>
<td>SIP</td>
<td>Status: 200 OK</td>
<td>-</td>
</tr>
<tr>
<td>M5</td>
<td>60.093641</td>
<td>130.237.239.12</td>
<td>130.237.15.238</td>
<td>SIP/XML</td>
<td>Request: PUBLISH sip:ccsleft@130.237.15.238</td>
<td>Wireless</td>
</tr>
<tr>
<td>M6</td>
<td>60.095752</td>
<td>130.237.15.238</td>
<td>130.237.239.12</td>
<td>SIP</td>
<td>Status: 200 OK</td>
<td>-</td>
</tr>
<tr>
<td>M7</td>
<td>60.16002</td>
<td>130.237.15.238</td>
<td>130.237.239.213</td>
<td>SIP/XML</td>
<td>Request: NOTIFY sip:Sub1@130.237.239.213</td>
<td>Wireless</td>
</tr>
</tbody>
</table>
Large numbers of Watchers

- The server can handle a large number of Watchers (tested with 1,000,000 simulated Watchers) without crashing.
- With a large number of Watchers in the database, when a new Publish is received, the server notifies all of the Watchers correctly and each one receives the Notify with ~1ms of additional delay.

The delay in receiving Notify messages for each of the Watchers
Bottlenecks

- Increasing the number of Publish messages for different events in the SER database, increases the server’s response time
- Incorrect queries to the MySQL can cause server crashes

The relative delay in sending Notify messages, when the Publish messages is increased
Conclusions & Future work

- Conclusions
  ✓ SIP-SIMPLE distributes context information among entities, using Subscribe, Publish, Notify.
  ✓ PIDF is suitable as a context model to transfer this context information in a standard format (it has been extended to support special tags for our purposes).
  ✓ The combination of SER, an extended PA module, and MySQL database functioned as expected.

- Future work
  ✓ Add authentication of Watchers
  ✓ Add authorization of Watchers
  ✓ Add policies (for security & authorization purposes)
  ✓ Examine an alternative architecture of SIP request/reply vs. SIP-SIMPLE
Questions

Thank you for listening!

Mzarify@gmail.com