Design of Semantic Information Broker for Localized Computing Environments in the Internet of Things

Ivan V. Galov, Aleksandr A. Lomov, Dmitry G. Korzun

Petrozavodsk State University
Department of Computer Science

Supported by the Ministry of Education and Science of the Russian Federation and by the Russian Fund for Basic Research

AMICT’2015 Conference
May 14, 2015, Petrozavodsk, Russia
Introduction: Internet of Things (IoT)

- Multitude of physical and digital objects in our daily life
- Localized IoT environments appear everywhere
- Environment inhabitants perceive “smart services”
Smart Spaces: The M3 Architecture

- Multidevice, Multidomain, Multivendor
- Infrastructure: Semantic Information Broker (SIB) maintains smart space content in RDF triples
- Application: Knowledge Processors (KPs, agents) run on IoT devices
- Interaction: Blackboard and Pub/Sub
- Smart space: KPs share ad-hoc knowledge and reason over it to construct services
Existing SIB Implementations

- **Smart-M3 SIB**: the first official prototype

- **RIBS**: targets resource limited devices

- **OSGi SIB**: higher level of modularity and portability (Java-based)

- **RedSIB**: evolution of Smart-M3 SIB with Redland triplestore
## SIB Implementations: Properties

<table>
<thead>
<tr>
<th></th>
<th>Smart-M3 SIB</th>
<th>RIBS</th>
<th>OSGi SIB</th>
<th>RedSIB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td>C</td>
<td>ANSI C</td>
<td>Java</td>
<td>C</td>
</tr>
<tr>
<td><strong>Triplestore</strong></td>
<td>Piglet</td>
<td>Bitcube</td>
<td>Jena</td>
<td>Redland</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td>glib library, SSAP, WQL</td>
<td>lightweight, KSP, constant access latency</td>
<td>SPARQL and reasoning support</td>
<td>improved subscription, SPARQL support</td>
</tr>
<tr>
<td><strong>Drawbacks</strong></td>
<td>no SPARQL support, performance problems</td>
<td>cubical memory consumption</td>
<td>resource-demanding, incompatible with other SIBs</td>
<td>performance problems</td>
</tr>
</tbody>
</table>

Research prototypes, unsuitable for localized IoT environments
Crucial SIB Properties

- **Simplicity**: SIB architecture is easy to elaborate, evolve and understand by third-party developers.

- **Extensibility**: SIB architecture provides a modular way of enhancing the functionality.

- **Dependability**: SIB operation is resilient. SIB runs continuously for lengthy time periods. In case of failures, SIB recovers its working state.

- **Portability**: Host devices for SIB are diverse. Traditional Linux and Windows based systems as well as embedded systems (e.g., OpenWrt on routers).
Redesigning: Our Approach

- Based on RedSIB
- Eliminated D-BUS
- Plug-in approach: dynamic libraries
- Modular architecture
- Qt framework
Renewed SIB Architecture
Smart Space Access Protocols

SSAP: join, leave, insert, remove, update, (un)subscribe

KSP differences:

- compact binary format;
- transactions are based on the SPARQL 1.1 (and SPARQL UPDATE) only;
- no join and leave operations;
- possibility to define the maximum size for SIB response;
- additional persistent operations, which continuously change the smart space content.
## Spectrum of Supported Access Operations

<table>
<thead>
<tr>
<th>Type</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic operations (SSAP)</td>
<td></td>
</tr>
<tr>
<td>Session management</td>
<td>Join, Leave</td>
</tr>
<tr>
<td>Content access and management</td>
<td>Instant Query, Insert, Remove, Update</td>
</tr>
<tr>
<td>Persistent operations</td>
<td>Subscribe, Unsubscribe</td>
</tr>
<tr>
<td>Extended operations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Persistent Insert, Remove, Update</td>
</tr>
<tr>
<td></td>
<td>SIB configuration rules</td>
</tr>
<tr>
<td>SPARQL operations</td>
<td></td>
</tr>
<tr>
<td>SPARQL</td>
<td>SELECT, CONSTRUCT, ASK, DESCRIBE</td>
</tr>
<tr>
<td>SPARQL Update</td>
<td>INSERT, DELETE, INSERT DATA, DELETE DATA</td>
</tr>
</tbody>
</table>
Subscription Mechanism (as in RedSIB)
Properties of renewed SIB

- **Simplicity**: functional allocation into modules, D-BUS is eliminated.

- **Extensibility**: modular architecture allows to extend SIB functionality (new protocols, operations, rules).

- **Dependability**: SIB implementation takes into account problems of other SIBs. Code is based on Qt framework which contributes dependability.

- **Portability**: D-BUS removal and cross-platform Qt framework allows to run SIB on Windows and Linux machines as well as various embedded devices.
Conclusion

- Renewed SIB design for the smart space applications development
- Simplicity, extensibility, dependability and portability of SIB
- Compatibility with previous Smart-M3 applications for Smart-M3 SIB and RedSIB
- New opportunities for application development due to advanced smart space access operations

Thank you for attention

E-mail: galov@cs.karelia.ru