Smart Environment: a call for interoperability

SOFIA project and Smart M3 architecture
- pros: interoperability at information layer
- cons: hard to extend with new core features

Plug-in Interface
- dynamically extend Smart M3

Profiling service
- performance indicators based on Plug-in and regular KPs
Smart Environment & Information Interoperability

- From Personal/Ubiquitous Computing to **Smart Spaces**

- Smart Environment paradigm: “anywhere, anytime, anything”
  - *cooperation and data sharing* to enhance *information availability* and enable new services and features
  - need to overcome *standardization issues* related to the physical world
**SOFIA project: Motivations and Principles**

- **SOFIA**: Smart Objects For Intelligent Applications
  - http://www.sofia-project.eu/

- **Mission**: InterOperability Platform (IOP) to overcome standardization issues
  - information interoperability
  - cooperation between application and smart services

- IOP enables a **seamless access to the distributed content** from heterogeneous devices, ranging from smartphone to desktops
Smart M3: Multi-vendor, Multi-device, Multi-domain
- Linux-based reference implementation developed by Nokia

Communication based on blackboard approach: entities do not know each other

Smart M3 IOP: interoperability based on semantic consensus, but in a localized manner
- no globally accepted semantic or ontology
- devices share and access information based on locally agreed semantics

Smart M3 does not depend on the underlying communication
Smart M3 Architecture

- **Smart Space**: scope of interesting information
- **Semantic Information Broker (SIB)**: maintains shared data stored as RDF triples
- **Knowledge Processors (KPs)**: external entities interacting each other by publishing/reading data to/from the SIB
- **Smart Space Access Protocol (SSAP)**: lightweight communication protocol with simple and efficient operations, e.g., join, leave, insert, remove, update, query, subscribe, unsubscribe
Smart M3: Pros

- Information published through shared Semantic Information Brokers (SIBs): *decoupled interaction*
- Information based on *common ontology* models and common data formats (RDF)

- **Smart-M3 is device, domain, and vendor independent**: maximum flexibility, simple availability
  - *user*: freedom of choice (multi-vendor)
  - *device manufacturer*: seamless operations with every devices (multi-device)
  - *application developer and service company*: focus on consumer interests gaining competitive edge (multi-domain)
Smart M3: Cons

- Smart M3 is still growing and under development
  - http://sourceforge.net/projects/smart-m3/

- Ongoing work
  - access control and *security management* (SSAP secure implementation)
  - service discovery and *composition*

- Open issues
  - SIB distribution protocol to create a *distributed shared repository*
  - *context-awareness* support: sustaining scalability and efficient management of available resources/services
  - *features statically defined* at compile-time
Enhancing M3 Flexibility

- Need to clearly separate aspects related to application logic from general purpose features
  - full interoperability and maximum re-usability
  - only KPs can offer services, always using SSAP
  - SIB features are defined at compile-time, no dynamic addition is possible at run-time

- Goal: supporting the dynamic addition of management core features
  - depending on the context and the capability of the hosting node
  - different scope and operating layer compared to KP
  - possible useful features: node characterization, information management, garbage collecting

- Joint contribution by University of Bologna and Nokia
Plug-in Interface Solution to Dynamically Add Features

- General purpose API to **dynamically register and execute third-party services (plug-ins)** to provide additional features
  - exclusive and privileged access to stored data
  - no communication overhead: directly interact with the RDF datastore

- Services and extensions developed according to a well-defined programming discipline in compliance with a **standard template**
  - `evaluateState`: checking executing conditions
  - `run`: starting plug-in execution
  - `stop`: ending plug-in execution, depending on execution time and/or scheduling needs
Dynamic architecture: plug-ins implemented through *dynamic linked library* (Shared Objects)

- a .so can be dynamically loaded/unloaded and linked at *SIB execution time*
- *additional features separated from SIB executable*, reducing its size and used disk space
- *SIB customization*: provide additional features/plug-ins only when required

**Efficient check of plug-in template compliance** through proper functions provided by the OS
Plug-in Interface: Components and Behavior

- **Plug-in Entry Point**
  - un/register plug-in extensions
  - check template compliance

- **Plug-in Manager**
  - periodically activates registered extensions

- **Plug-in Timer**
  - fairness enforcement in terms of plug-in execution time
Profiling service
- dynamically evaluate **SIB performance**
- useful to compare SIBs

Two alternative approaches to gather performance
- Profiling KP
  - based on a regular KP
  - SSAP-based access to RDF store in competition with other KPs
- Profiling Plug-in
  - implemented as a plug-in
  - direct and exclusive access to the RDF store
**Profiling Plug-in**: best possible performance without any interference, such as traffic overhead or concurrent KPs
- *ideal upper bound* KPs are not able to exceed

**Profiling KP**: performance affected by SSAP overhead and current load
- achieved value *closer to what is actually possible* for a regular KP

**Complementary solutions**: comparison of performance achieved by Profiling Plug-in and KP to *estimate current load on the SIB*
Different parameters to quantitatively evaluate SIB performance
- complementary performance indicators useful in relation to KP objective
- periodically computed and locally stored as RDF triples

Current Perf.: \[ CP = \frac{(KP \text{ insert} + 10 \times KP \text{ query} + KP \text{ delete} / 10)}{3} \]
- currently available performance on SIB
- useful for KPs interested in quickly retrieving data
- computed every two hours, to monitor the daily workload

Best Perf.: \[ BP = \frac{(plug-in \text{ ins.} + 10 \times plug-in q. + plug-in del. / 10)}{3} \]
- best performance achievable in ideal conditions, i.e., no concurrent KPs, no communication overhead
- useful for long lasting KPs
- computed once a day (unlikely to vary)

Relative Performance: \[ RP = CP / BP \]
- \[ RP = 1 \] → current performance equal to best performance
- useful to balance workload on available SIBs
**Testbed Details**

- **Two SIBs with different capabilities**: Linux + ad-hoc IEEE 802.11g link
  - High: Intel Core2 Duo P8400 2.26GHz, 3GB RAM
  - Low: Intel Pentium M processor 1,10GHz, 500MB RAM
- **Workload emulation**
  - 8/2/1 inserts/deletes/query for each cycle, RDF triples queried at the end
  - Performance indicators vary in relation to *node capabilities* and *workload*

<table>
<thead>
<tr>
<th>Daily time</th>
<th>Workload Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:00, 3:00, 5:00, 7:00, 23:00</td>
<td>No workload KPs</td>
</tr>
<tr>
<td>9:00</td>
<td>1 workload KP, 18 cycles</td>
</tr>
<tr>
<td>11:00, 13:00</td>
<td>1 workload KP, 100 cycles</td>
</tr>
<tr>
<td>15:00</td>
<td>2 workload KPs from different nodes, 20 cycles</td>
</tr>
<tr>
<td>17:00</td>
<td>2 workload KPs, from different nodes, 40 cycles</td>
</tr>
<tr>
<td>19:00</td>
<td>2 workload KPs, from different nodes, 20 and 40 cycles</td>
</tr>
<tr>
<td>21:00</td>
<td>1 workload KP, 40 cycle</td>
</tr>
</tbody>
</table>

**Graphs**

- **Daily time (High)**: BP = 0.86s
- **Daily time (Low)**: BP = 1.24s

**Note**: The graphs show the performance metrics over a daily cycle, with workload conditions specified at different times. The performance indicators (CP and BP) vary according to the node capabilities and workload conditions.
**Performance Analysis**

- **Two KPs with differentiated requirements**
  - FastKP: few operations with strict delay requirements
  - SlowKP: several operations, without strict delay requirements

- **SIB selection based on CP/BP/RP**
  - FastKP: random vs. lowest CP
    - only FastKP executes
    - *execution time lowers* from 0.63s to 0.35s
  - SlowKP: lowest CP vs. best RP
    - FastKP already executing on SIB with best CP
    - *execution time increases* (not an issue), but *workload fairly distributed* among SIBs

<table>
<thead>
<tr>
<th>Node</th>
<th>High</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both Fast/ SlowKP use CP</td>
<td>4.64%</td>
<td>0.00%</td>
</tr>
<tr>
<td>FastKP uses CP, SlowRP uses RP</td>
<td>1.01%</td>
<td>3.12%</td>
</tr>
</tbody>
</table>
**Overhead due to performance indicators**
- join available SIBs
- query CP/BP/RP
- evaluate SIB

**Joining, gathering, and comparing overhead largely lower than execution time**
- suitable even for FastKP with strict delay requirements
- SIB joining has the greatest impact

<table>
<thead>
<tr>
<th></th>
<th>FastKP</th>
<th>SlowKP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution (s)</td>
<td>0.35</td>
<td>13.62</td>
</tr>
<tr>
<td>Comparison (s)</td>
<td>0.02</td>
<td>0.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>FastKP</th>
<th>SlowKP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution (s)</td>
<td>0.35</td>
<td>23.08</td>
</tr>
<tr>
<td>Comparison (s)</td>
<td>0.02</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Conclusions

- SOFIA project and Smart M3 IOP support interoperability of heterogeneous devices
- Proposed Plug-in API allows **SIB dynamic customization**
  - keeping SIB architecture **very lightweight**
  - supporting domain- and deployment-specific **additional features**
- Profiling service: performance indicator **coupling plug-in and KP approaches**
  - KPs can dynamically select the SIB best fitting their requirements
- Ongoing work
  - proper and well-defined ontology for SIB profiling
  - dynamic federation of distributed SIBs
Any Questions?

Thanks for your attention 😊

Questions time…

Prototype code: http://sourceforge.net/projects/smart-m3/
Additional information: http://www.sofia-project.eu/