The Semantic Event Broker

Francesco Morandi
What are we doing and what future for Smart M3?

• Is it possible to consider today Smart M3 still a «triplestore» or an «endpoint»
  alternative? Modern SPARQL Endpoint (like Jena, Sesame, Virtuoso...) have:

  > Experience (in some cases almost 20 years)
  > Improved database distributed management
  > Multithread architecture
  > Big communities

  (... the only «smart» answer is no.)

• Smart M3 should be a layer over existing triplestore or endpoint, providing new
  capabilities for Smart Environments (or CPS).
• New algorithms for adding new functionalities and for improving performances.
• Avoiding the «stupid hat» on top of a perfectly working platform.
The legacy of the SOFIA project and the particularities of our Semantic Web

• The SIB can be considerate a dynamic real time context handler, so generally has to store a “limited” amount of triples coming from the real scenarios. We can consider the Smart Environment and the SIB related by a bijective function of entities.

• The RDF content changes continuously with relatively “small” triples updates (“update” will be referred to insert, remove or insert-remove operations) for these reasons:

  > Many of commercial sensors produce periodical updates of a limited number of basic parameter (e.g. temperature, pollution, hearth rate, weight, pressure...)

  > Aggregator KPs generically send few triples per time as result of some elaborations, or just some triples for triggering actuators.

  > Actuator KPs generally are notified and consequently update the Smart Space with some (generally few) results coming from the feedback of the operation.
The legacy of the SOFIA project and the particularities of our Semantic Web

• The SIB store context information is structured for retrieving almost real time data from the ontology knowledge with no historical purpose. Every historical service must be done externally (KPs based) for not dramatically affecting the performances. This problem generally does NOT depends from the specific application.

• The SIB must avoid synchronization between KPs who share multiple access to the same resources.

• One of the most powerful feature in Smart M3 is the publish-subscribe implementation. This allows the KP to be automatically waken (without consuming resources) whenever the required parts of graph change, allowing logical composition of sub-graphs and filtering operations.
Overcoming the SIB

- **SPARQL 1.1** (query and update) is now the *official* W3C language for RDF. All *other methods* for querying, subscribing and updating the SIB (RDF operations, other query methods...) can be considered *obsolete*. Particularly all the RDF operations can be considered specific SPARQL sub-cases.

- **SPARQL applied to publish subscribe** can become extremely powerful, similar to rules or *event language* taking all the advantages of the *semantic*. For allowing this a particular algorithm has been implemented for improving performances and exploiting the multithreading functionalities.

- **A Time Management** mechanism is absolutely necessary in the SIB for allowing synchronization and watchdogs (the latter *extremely* difficult to implement in Smart M3).
To the Semantic Event Broker: The programming model

The reference behaviour: Event Languages.

E.g. TESLA* event based (Running on powerful parallel HW, like CUDA):
Events are more powerful due to the time integration and performances.

```
define Fire (Val)
from Smoke () and
each Temp(Val > 45) within 5 min from Smoke
where
Val = Temp.Val
```  

Possible to have the same behaviour with the added value of the semantic?
(Related Article C-SPARQL, EP-SPARQL, ...)

Extremely **difficult** to implement a rule like this in Smart M3 ordinary KP.

A KP for having this feature can be complex.
> Thread with Sleeps, counter thread with reset..
> Synchronizations

THE KP MODEL
SHOULD BE SIMPLER
AND BETTER DEFINED
To the Semantic Event Broker: The programming model

We can imagine to introduce some new and specialized class of KPs, event-oriented and time-based: the Semantic Event Processor (SEP)

A SEP can interact only with the SEB:

- Aggregator SEP

Or even with the Physical World:

- Producer SEP
- Consumer SEP
To the Semantic Event Broker: The programming model

- Every Producer SEP is provided with a Legacy Layer (able to read data from the physical world) and performs SPARQL Updates on changes.

- Consumer KP holds a SPARQL Subscribe and send data to the legacy layer on every variation.
To the Semantic Event Broker: The programming model

Every aggregator SEP is based on:

- **Context Subscribe** (Sparql Subscribe)
- **Callback**
  - Processing (Combinatorial, Causal) [Optional]
  - **Sparql Update** (Scheduled[Optional])

And on time functionalities.
To the Semantic Event Broker: Example

The scenario includes a presence sensor and a lamp.
(the example is almost identical to the fire alarm of the events)

Behavior:
• If a presence is detected the lamp must be turned on.
• When the presence is no more detected for 10 seconds the lamp must be switched off.
To the Semantic Event Broker: Example

Producer and consumer KP:
To the Semantic Event Broker: Example

Aggregator:

**Context subscribe**

**ASK**

**WHERE**

  ns:PresenceSensor_123 ns#hasValue “True” }

**CALLBACK (On variation)**

**Processing SEP**

**Agent TurnLampOn**

**SELECT** ?timestamp

**WHERE**

  ns:PresenceSensor_123 ns#hasValue “False” .
  ns:PresenceSensor_123 ns#hasSIBTimestamp ?timestamp }  

**timestamp** = 1378737852.770930

**time_to_switch** = timestamp + 10

**CALLBACK (On variation)**

**Processing SEP**

**Agent TurnLampOffDelayed**

**INSERT**

{ ns:LampActuator_456 ns#hasValue “True” .
  ns:LampActuator_456 ns:hasSIBTimestamp ?currentsibtime}  

**DELETE**

{ ns:LampActuator_456 ns#hasValue “False” .
  ns:LampActuator_456 ns:hasSIBTimestamp ?OLDsibtime}  

**SCHEDULED AT (1378737862.770930)**

**INSERT**

{ ns:LampActuator_456 ns#hasValue “False” .
  ns:LampActuator_456 ns:hasSIBTimestamp ?currentsibtime}

**DELETE**

{ ns:LampActuator_456 ns#hasValue “True” .
  ns:LampActuator_456 ns:hasSIBTimestamp ?OLDsibtime}

**WHERE**

{ ns:PresenceSensor_123 ns#hasValue “False” .
  ns:PresenceSensor_123 ns#hasSIBTimestamp “1378737852.770930”.
  ns:LampActuator_456 ns#hasValue “True” .
  ns:LampActuator_456 ns:hasSIBTimestamp ?OLDsibtime .
  BIND ( get_sib_tme() AS ?currentsibtime ) }
To the Semantic Event Broker: Example

Time behaviour:

- **Light**
  - True
  - False
- **Presence**
  - True
  - False

- **TurnLampOn**
- **TurnLampOff**
- **Status Changed!**
- **WAIT 10; Run TurnLampOff**
  - SPARQL UPDATE
- **Unsuccessful Update**
- **Successful Update**
To the Semantic Event Broker: The Internal Architecture

- Multithread for any active subscribe
- All the update operations are forwarded in to the threads through async queues
- Every subscription thread is provided by
  - LUTT
  - Internal Sub-Triplestore
  - Sparql Accellerator * (or Sparql Engine)
  - Dispatcher for Indications
Conclusions:

As simple triplestore, Smart M3 can be considered inadequate compared to most SPARQL Endpoints.

If we agree to consider Smart M3 a layer over existing TS the aim can be:

Create a convergence with the world of events by:

- A KP implementation philosophy based on only on SPARQL SUBSCRIBE, SPARQL UPDATE and time functionalities. (e.g. queries will become obsolete)

- An internal architecture based on:
  - SPARQL optimization for the subscriptions
  - Multithreading for subscriptions
  - Time management managed as sleeps in theads (low resource)