A Semantic Processing Framework for IoT-Enabled Communication Systems

Muhammad Intizar Ali[1(✉)], Naomi Ono[1], Mahedi Kaysar[1], Keith Griffin[2], and Alessandra Mileo[1]

[1] INSIGHT Centre for Data Analytics, National University of Ireland, Galway, Ireland
{ali.intizar,naomi.ono,mahedi.kaysar,alessandra.mileo}@insight-centre.org
[2] Cisco Systems, Galway, Ireland
kegriffi@cisco.com

Abstract. Enterprise Collaboration Systems are designed in such a way to maximise the efficiency of communication and collaboration within the enterprise. With users becoming mobile, the Internet of Things can play a crucial role in this process, but is far from being seamlessly integrated in modern online communications. In this paper, we showcase the use of a solution that goes beyond today's ad-hoc integration and processing of heterogeneous data sources for static and streaming data, providing more flexible and efficient processing techniques that can bridge the gap between IoT and online Enterprise Communication Systems. We document the technologies used for sensor deployment, sensor data acquisition based on the OpenIoT framework, and stream federation. Our main contributions are the following, i) we present a conceptual architecture of IoT-enabled Communication Systems, that builds upon existing frameworks for semantic data acquisition, and tools to enable continuous processing, discovery and federation of dynamic data sources based on Linked Data; ii) we present a semantic information model for representing and linking IoT data, social data and personal data by re-using and extending the existing standard semantic models; iii) we evaluate the performance of virtualisation of IoT sources based on OpenIoT in our testbed and show the impact of transmission, annotation and data storage, as well as initial results on scalability of RDF stream query processing in such a framework, providing guidelines and directions for optimisation.

Keywords: IoT · RDF stream processing · Stream federation · Communication systems · OpenIoT · Linked data

1 Introduction

Enterprise communication systems currently and historically have been primarily aimed at person to person communication. Users of such systems typically

This research is sponsored by Science Foundation Ireland (SFI) grant No. SFI/12/RC/2289 and Cisco Systems.

M. Arenas et al. (Eds.): ISWC 2015, Part II, LNCS 9367, pp. 241–258, 2015.
DOI: 10.1007/978-3-319-25010-6_14

interact with an endpoint such as a phone, video system or unified communications software client capable of multi-modal communications. Communication modes typically consist of instant messaging, voice, video and voicemail to allow individuals or groups to communicate in real time. Such systems have not historically enabled open machine to machine or machine to person communication. The emergence of Internet of Things (IoT) provides the potential to enable communication between sensory devices and communication systems using open interfaces, but this potential is under investigated and few solutions have existed in isolation. As a result, the flexible integration of a large amount of multi-modal data streams from diverse application domains is still one of the key challenges in developing IoT-enabled communication systems.

The lack of interoperability results into the inability for such systems to integrate information from external sources in an easy and cost-effective way. This issue becomes more evident if we consider advances in the IoT space, which demands dynamic and flexible exchange of information between IoT sources. To overcome these interoperability issues in communication systems and across smart enterprise applications towards IoT-enabled solutions, we developed a Linked Data infrastructure for networking, managing and analysing streaming information. In order to ensure high reusability, we leveraged existing semantic models for the annotation of sensor data (e.g. SSN), social web (e.g. FOAF) and personal information (e.g. PIMO), and extended the ontological model to incorporate personal, business and online communication concepts.

In order to set the basis for our evaluation, we identified a usecase scenario in the enterprise communication space, to illustrate the potentials of IoT-enabled Communication Systems. We then designed and developed the processing pipeline from IoT sources to stream processing and reasoning, which is seamlessly integrated in our framework. Our main contributions in this paper include:

- design of a Linked Data framework for IoT-enabled smart enterprise applications that connects physical to virtual sensors and enables scalable stream processing and reasoning;
- interoperable integration of various IoT sources (corresponding to capabilities) in the context of an open source online communication system;
- demonstration of the effectiveness of our proposed framework based on a concrete instance of OpenIoT and Apache Open Meetings;
- experimental validation of the performance and scalability of our IoT-enabled infrastructure and lessons learned.

The remainder of this paper is organised as follows: Section 2 presents our scenario and state of the art, Section 3 details our IoT-enabled Linked Data infrastructure and it's software components, which we evaluate in Section 4 before we conclude with some remarks and lessons learned in Section 5.

2 Motivation and State of the Art

Sensor technologies and sensory devices are nowadays part of our everyday lives. The Internet of Things (IoT) not only provides an infrastructure for sensor deployment, but also a mechanism for better communication among connected sensors. Data generated by these sensors is huge in size and continuously produced at a high rate. This requires mechanisms for continuous analysis in real-time in order to build better applications and services. Data streams produced by various sensors can be classified into three different categories, namely, (i) *Physical (static) Sensors*, (ii) *Mobile & Wearable Sensors*, and (iii) *Virtual Sensors & Social Media Streams*.

Among the above three categories, mobile sensors are harder to integrate within enterprise communication systems. This is not only due to technical integration issues and interoperability, but also due to their dynamic nature and constantly changing context. Mobility and location-based sensory input, for example, result into a higher level of unpredictability and lower level of control over the distributed infrastructure that characterises enterprise communication systems. These challenges are matched by new opportunities for IoT-enabled collaboration and communication systems to be designed in order to sense the context of a mobile user and take decisions according to dynamic sensory input. In the domain of enterprise communication systems, mobile users have the potential to produce a lot of dynamic sensory input that can be used for the next generation of mobile enterprise collaboration, with great potentials for better user experience. In this paper we propose a framework and a set of software component for IoT-enabled online meeting management that combine existing technologies in a scalable infrastracture.

2.1 Motivating Scenario: IoT-Enabled Meeting Management System

Alice is hosting an online meeting for her company *FictionDynamic*. The meeting is planned to hold in Meeting Room B at 11:00 am. Bob and Charlie attending the meeting while they are on the move, thus their availability and ability to participate to the meeting in various ways is dynamically changing. The IoT-enabled Meeting Management System (IoT-MMS) enables i) automatic on-the-fly semantic enrichment of IoT information related to the meeting attendees, ii) communication of such richer information to the participants via their IoT-MMS client through a panel showing IoT values and related user capabilities (e.g. ability to hear properly, share a screen, type, talk), iii) use of such rich information to improve user experience and optimise meeting management on-the-fly. The integration of a web-based MMS with sensory input and enterprise data such as attendees details, calendars and agenda items makes it possible to characterise and manage the following aspects in a flexible and interoperable way:

- updating (enabling or disabling) users capabilities based on IoT input (via sensors abstraction and interpretation, semantic integration and stream query processing);

- managing agenda items, including users involved and capability requirements via business logic rules;
- dynamically verifying privacy-constraints on agenda items based on location and context.

In Sections 3 and 4, we illustrate the design and implementation of our IoT-MMS framework enabling characterisation and management of the above mentioned aspects.

Enabling and disabling user capabilities has the potential of improving user experience: acting on microphones and speakers of attendees based on their participation and the level of noise can avoid unpleasant feedback loops, and guidance for the meeting host on changing capabilities of attendees on the move would promote more effective management of online meetings. In the same way as capabilities are enabled or disabled, additional functionalities can be characterised by adding specific semantic queries and action triggers. For example, the IoT-MMS can support the meeting host in dynamically re-assigning agenda slots to participants, based on users involved and their changing capabilities. Also, queries over the attendees calendars and presence status [10] for availability would make it possible to suggest alternative meeting slots if key attendees become unavailable or if their capabilities become compromised.

2.2 State of the Art

Internet of Things (IoT) research in recent years has focused on modelling domain knowledge of sensor networks and services [3,4,12,16]. The Semantic Sensor Network (SSN) ontology is one of the most significant efforts in the development of an information model for sensory data [6]. The SSN Ontology provides a vocabulary for expressive representation of the sensors, their observations and knowledge of the surrounding environment[1]. SSN is being widely adopted by many IoT-based applications for the representation of sensor data. SSN ontology defines only a high-level scheme of sensor systems, therefore SSN alone cannot represent an information model for a richer IoT infrastructure and needs to be aligned with the existing ontologies or with new concepts from application domains. Consider our scenario in Section 2.1, the SSN ontology needs to be aligned with existing semantic models for the representation of the meeting/calendar information and personal/social data.

Data acquisition from distributed heterogeneous sensors is another essential aspect of IoT-enabled applications. The Global Sensor Network (GSN) middleware facilitates flexible integration and discovery of sensor networks and sensor data [1], enabling fast deployment and addition of new IoT platforms by supporting dynamic adaptation[2]. X-GSN [5] is an extension of GSN and therefore supports all virtual sensors and wrapper developed for the GSN middleware. X-GSN is deployed as a web server which continuously listens for sensor data over

[1] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[2] http://sourceforge.net/projects/gsn/

a pre-configured port (default port = 22001), and it contains various wrappers built as subclasses of the GSN wrappers, each acting as a thread in the GSN.

OpenIoT [2] is an open source middleware for collecting information from sensor clouds. OpenIoT can collect and process data from virtually any sensor in the world, including physical devices, sensor processing algorithms and social media processing algorithms (http://openiot.eu). OpenIoT combines and enhances results from leading edge middleware projects, such as the Global Sensor Networks - GSN and the Linked Sensor Middleware[3](LSM) [1,9].

However, IoT-enabled applications not only require to gather sensor data from distributed sensors network, but also demand to provide adaptive applications which can query data streams generated by sensors and can take smart decisions accordingly. Furthermore, IoT-enabled applications need to provide robustness because of the autonomous and distributed nature of the underlying architecture. OpenIoT in its current state does not support stream query processing over data streams generated by various sensors, hence lacking the ability to facilitate realtime decisions. We used the OpenIoT framework for sensor data acquisition and semantic annotation, creating additional wrappers that are needed for streaming IoT data, and we extended it by introducing stream query processing [8] and stream reasoning capabilities based on rules [11].

3 IoT-enabled Communication Systems

In this section, we introduce the layered architecture of the IoT-enabled Communication System and briefly describe each of the layers involved in the processing pipeline.

3.1 Application Architecture

Figure 1 illustrates our conceptual architecture for IoT-Enabled Communication System. OpenIoT acts as a core component for data acquisition and semantic annotation of the data produced by various sensors. We extended the functionalities of the OpenIoT platform by introducing HTTP Listener wrapper for capturing streaming data, and semantic querying and reasoning layer, which allows IoT-enabled communication systems to include semantically annotated data streams produced by sensors as an additional source information. IoT-enabled Communication Systems can perform real-time continuous queries over data streams and consume the results of these queries to take context-aware and user-centric decisions in real-time. As shown in Figure 1, there are three main layers involved in our IoT-enabled Enterprise Communication System architecture, namely (i) *Data Acquisition and Semantic Annotation Layer*, (ii) *Stream Processing and Reasoning Layer*, and (iii) *Application Layer*. Below, we further elaborate on each of these layers and their components.

[3] http://lsm.deri.ie

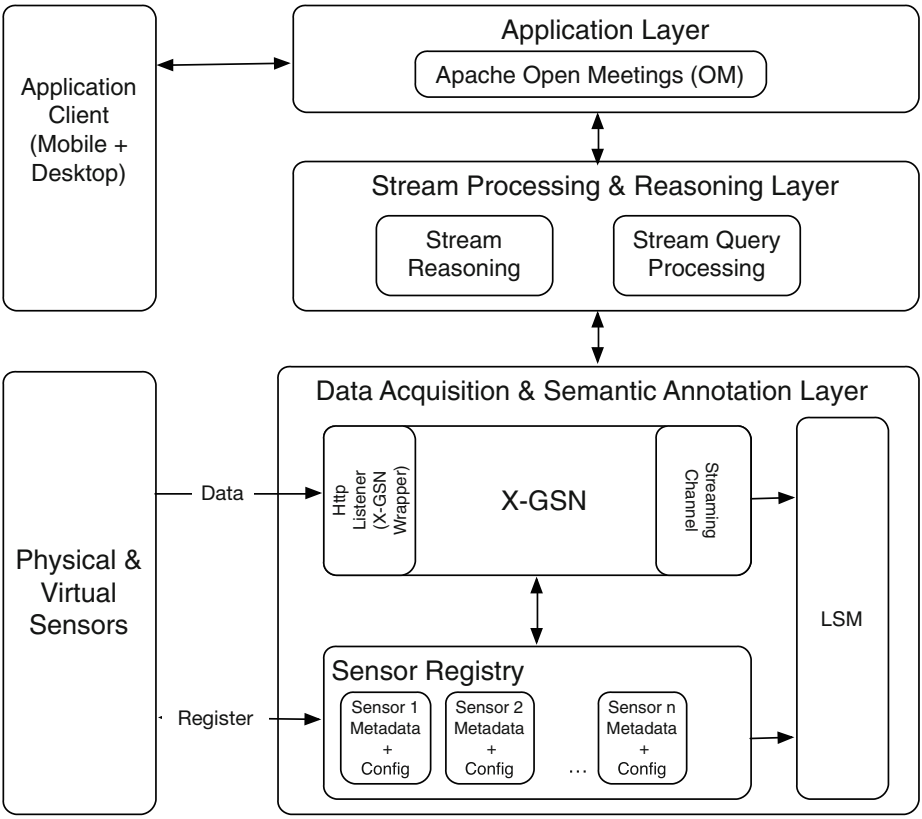

Fig. 1. IoT-Enabled Communication System Architecture

3.2 Data Acquisition and Semantic Annotation Layer

This layer is mainly responsible for acquiring sensor data from mobile devices and performing semantic annotation of the acquired data using our information model. We briefly discuss each of the components and their functionalities.

Data Acquisition

Our proposed architecture can acquire data from any type of sensor, whether it is physical sensor deployed at a fixed location, a mobile sensor or even a virtual sensor representing virtual data streams (e.g. social media data streams). However, considering the IoT-MMS scenario presented in Section 2.1, we focus on data acquisition from mobile sensors only.

Mobile Application for Data Acquisition: In order to receive data from various mobile sensors, we developed an android base application which can continuously sense the information from a mobile device. Once, the application is launched, a registered user can choose the sensors for which he/she wants to

share the data. Data produced by the selected sensors is continuously sent to the OpenIoT server.

Sensor Registration: A sensor is considered as a basic entity in the OpenIoT platform. Each and every sensor participating within the framework should be registered in the OpenIoT Platform before sending any observation. Sensors are uniquely identified within the OpenIoT platform by assigning a unique id. Mobile devices are registered as platforms (ssn:platform[4]), which can host multiple sensors. During the sensor registration process, some meta information (e.g. type of sensor, owner of the device, sensor observation unit etc.) is acquired. Sensors can be either registered individually and associated to an individual user or they can be registered in bulk if multiple sensors have the same meta information attached to them.

Sensor Observations Transmission: Whenever a sensor is successfully registered in the OpenIoT platform and the mobile application for data acquisition is launched using any mobile device, all the selected sensors on that particular device start transmitting their observations to the OpenIoT platform. Our processing pipeline makes it possible to select and de-select sensors dynamically without re-launching the application. We developed an X-GSN wrapper for mobile data acquisition, which is deployed over the X-GSN Server. As shown in Figure 1, the *Http Listener* is an integral part of the *X-GSN Wrapper* which continuously listens for the sensor observations. As soon as any observation is received, this layer starts processing the data accordingly using meta information of that particular sensor from which the observation is acquired. *X-GSN* also includes a *Streaming Channel* component which publishes semantically annotated RDF streams.

Semantic Annotation

We reused and integrated different semantic models for the representation of all acquired information in our IoT-MMS scenario, including sensor metadata, sensor observation, meeting/event data, meeting attendees and their capabilities. Linked Data representation allows for easy integration of semantic information collected from heterogeneous data streams as well as integration with static knowledge to perform querying and reasoning.

Semantic Annotation of Sensor Data Streams: We used the SSN ontology for representing sensors, their observations, and their platform (mobile device). The OpenIoT platform carries out annotation of the virtualised data streams that have been provided by the X-GSN data wrappers. We used an information model to explicitly define semantics of sensory data such as sensor types, models, methods of operation and common measurement definitions. As a result, sensor capabilities can be defined in accordance with existing conditions and be integrated as Linked Data. This helps bridging the gap between real-time information generated from various independent data sources and a huge amount of

[4] http://purl.oclc.org/NET/ssnx/ssn$#$Platform

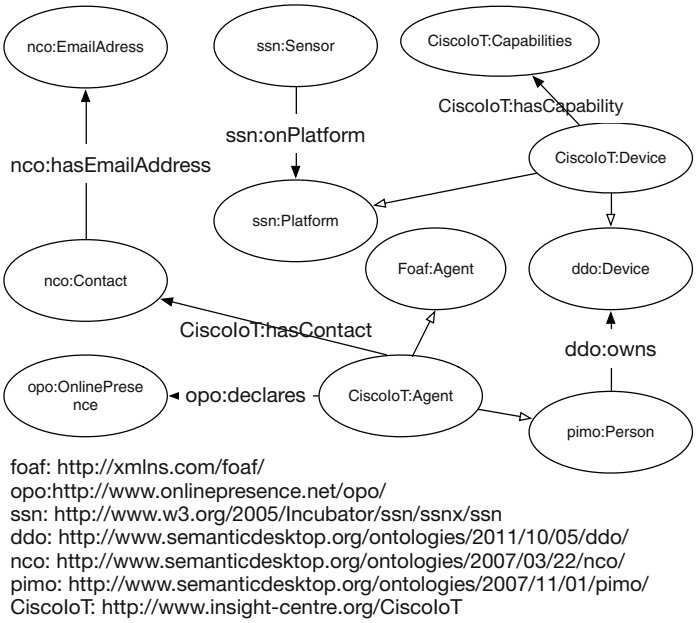

Fig. 2. Information Model - Device and Contact

interconnected information already available over the Web. For example, sensors and their data can be linked to geographic data (e.g. correlated natural phenomena), user-generated data (e.g. Meeting Data), and some implicit information (e.g. user profiles, calendar) through our semantic driven approach.

Semantic Annotation of Application Users and Mobile Devices: SSN is a de-facto standard for semantic annotation of sensor data streams. However, it still lacks the information to associate data generated from the sensors with any particular owner or user of that particular sensor. Keeping the usecase scenario of the IoT-MMS in mind, we represent the mobile client user as an owner of sensors embedded in the particular mobile device the user has used to log-in to the IoT-MMS mobile client. We used the NEPOMUK Contact Ontology (nco) [7] to represent a user and his/her contact information, while we used Digital.Me Device Ontology (ddo) to associate a device with any particular user [15]. As described earlier, multiple sensors embedded within a single mobile device can be easily represented using ssn:platform concept. Figure 2, depicts the information model for integration/linkage of sensor data with the contact information of the user as well as the device hosting that particular sensor. Each device can have multiple capabilities (e.g. noise, light, proximity) depending on the available sensors embedded within the device.

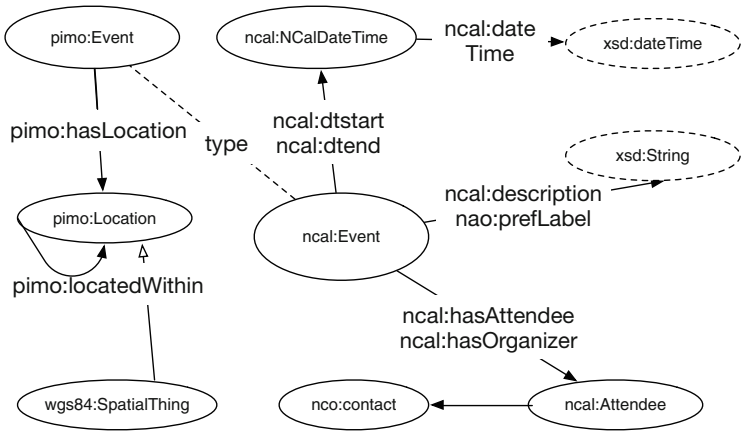

Fig. 3. Information Model - Meeting Management

Semantic Annotation of Meeting Data: We used the NEPOMUK[5] and related semantic desktop ontologies[6] for semantic representation of meetings, their description, organiser, list of attendees, location, starting time and duration [14]. NEPOMUK Calendar Ontology (ncal) is used for semantic annotation of the meetings created by any user of the IoT-MMS. Figure 3 gives an overview of the information model for the semantic annotation of meeting data.

3.3 Stream Processing and Reasoning Layer

One of the most important factors for IoT-enabled applications is their ability to detect events within minimal time delay. The *Stream Query Processing* component -shown in Figure 1- enables to continuously query sensor data streams and detect events in realtime, while the *Stream Reasoning* component contains application logic to make smart decisions customised to the particular requirements and context of the user.

Stream Query Processing: We integrated the CQELS (Continuous Query Evaluation over Linked Streams) query engine for the execution of continuous queries over semantically annotated data streams of mobile sensors [8]. CQELS is a state of the art stream query processing engine for RDF data streams, which allows to register queries over sensor data streams. Once a query is registered, CQELS continuously monitors sensor data streams and produces a stream of

[5] http://nepomuk.semanticdesktop.org
[6] http://www.dime-project.eu

results matching the query patterns. Listing 1, shows a CQELS query to monitor the noise level of a certain user of the IoT-MMS.

Stream Reasoning: This components consumes the stream generated as a result of the CQELS queries and facilitates smart decisions by associating patterns of events to actions. This is modelled using event-condition-action (ECA) rules in AnsProlog, where the events are triggers for actions to be executed. For example, results of the CQELS query in Listing 1 can be used by the stream reasoning component to trigger a rule that, based on the noise level, mutes a single or multiple users whenever noise level surpasses the specified threshold, and different thresholds can be dynamically selected based on indoor or outdoor user location. Similarly, rules can be used to suggest changes to the agenda when the associated attendee is late or temporarily disconnected, or to warn attendees on certain privacy threats when they are in a public place like an airport lounge or a train.

```
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
prefix lsm: <http://lsm.deri.ie/ont/lsm.owl#>
select ?noiseValue
WHERE {
STREAM <http://lsm.deri.ie/resource/1409752298064700000> [RANGE 5s]
{
?ob_5 rdf:type ssn:Observation.
?value_5 ssn:observedProperty <http://lsm.deri.ie/resource/1409752298163783000>.
?ob_5 ssn:featureOfInterest ?foi.
?value_5 lsm:isObservedPropertyOf ?ob_5.
?value_5 lsm:value ?noiseValue.}
}
```

Listing 1. A Sample CQELS Query to Monitor Noise Level

3.4 Application Layer

This layer represents the class of enterprise applications that can benefit from IoT intelligence which we showcase using our IoT-enabled Communication System based on Apache OpenMeetings. We extended the OpenMeetings server to generate semantically annotated data and to communicate with the reasoning component of our framework by continuously observing the status of relevant sensors generated by the stream processing layer and take appropriate actions at the application layer.

In what follows, we describe the process flow of our IoT-enabled OpenMeetings extension and illustrate the concepts and implementation of our online Meeting Management solution.

OpenMeetings (OM) is an open source software used for web conferencing, collaborative white board drawing, document editing etc. It uses OpenLaszlo RIA framework for client side generation and Red5 streaming server for remoting and streaming. In a physical conference room it can use Real-Time Messaging Protocol (RTMP) for high performance transmission of video, audio and data between Flash and the server. RTMP is a TCP based protocol which keeps the persistence connection and allows low latency communication.

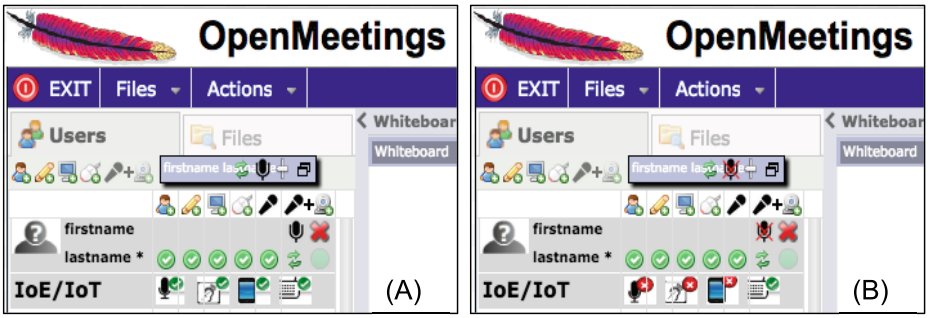

Fig. 4. Snapshots of IoT-enabled OpenMeetings Client

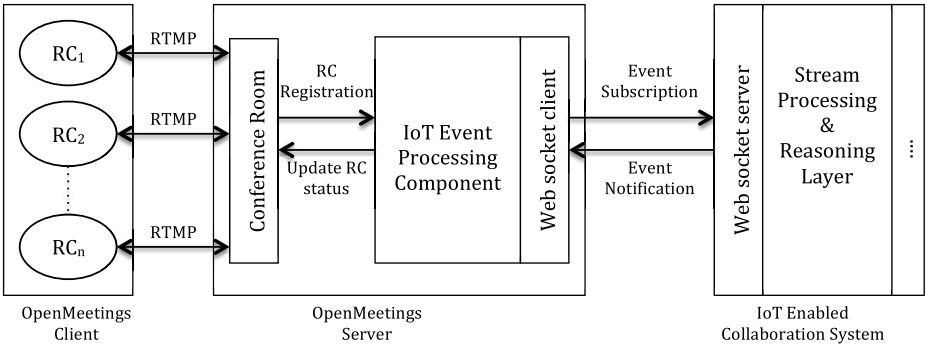

Fig. 5. Process Flow of IoT-enabled OpenMeetings System

Meeting Management in online communication and collaboration systems like OpenMeetings is enhanced with IoT input using our framework. In order to do that, the status of sensors registered on the IoT platform is monitored by the *Stream Reasoning* component, which identifies their status and determines appropriate actions based on rules. The sensors we considered include noise, proximity, light and location, while the actions are related to changing the status of a set of capabilities. Capabilities illustrate real-time availability of the participants to perform certain actions including talking, listening, reading a display or typing, and they are represented in the application control panel as a new set of IoT-related icons. Based on thresholds on the value of readings from specific IoT sources, the status of these icons is automatically updated from green to red or vice versa, indicating whether a participant can perform the corresponding activity or not. This provides the meeting host with updated information on the capability of the attendees, and can further be used to act on specific actuators (e.g. muting a microphone). For example, the ability to read the screen is not active if a user is connected via phone and answers a phone call. Figure 4(A) shows a client with all capabilities active, while Figure 4(B) shows the configuration for a mobile user surrounded by a lot of noise and

talking on the phone. Location can also be used to trigger privacy related rules (e.g. prompt a warning, if a user is in a call with a customer in public spaces like airport lounges).

The IoT-enabled OpenMeetings Process Flow is illustrated in Figure 5. When a remote client (RC) connects to an online conference in OpenMeetings, the server creates a Real Time Messaging Protocol (RTMP) connection and it registers as a web-socket client endpoint. Semantic information related to the client and the IoT sources is retrieved, and semantic queries are automatically generated to monitor updates of sensory input from that client. The server also subscribes to these queries, and when sensory updates are detected, the *Stream Reasoning* component processes them by consuming events as they are produced and applies the rules to determine which action should be executed in the client application, returning results as a JSON object to the corresponding web socket clients. Based on these results, the web socket client calls a remote method of its remote client and changes the status of the relevant IoT icons accordingly, prompting a warning message if required.

4 Evaluation

We evaluated our proposed architecture mainly by measuring performance and scalability of OpenIoT and query processing within our framework. We believe this is key for the applicability of our approach, since it demonstrates that semantic technologies embedded in OpenIoT can be used in this practical system without hindering feasibility and user experience, and enabling enhanced IoT-intelligence capabilities and business logic to be deployed by leveraging semantic representations. In the current paper we focus on the system and the software tools. Next steps would aim at a full in-use application in an industry setting. As a result, we aim at a realistic set-up and study that will provide more in-depth evaluation of usability and user experience.

Performance and scalability are two critical aspects for applications which are designed to adapt and react to changes in near real-time. In this section, we present the results of our feasibility tests conducted to evaluate the performance and scalability of our proposed solution. With this evaluation we also aim at demonstrating how semantic technologies in IoT can be applied to real scenarios and create a new market for IoT-enabled solutions like in the collaboration and communication systems space, highlight what are the main drawbacks and limitations of state-of-the-art technologies such as X-GSN and OpenIoT in this setting, and provide some suggestions on what key aspects should be tackled by the research community to make the technology deployable on a larger scale.

Experimental Setup (Testbed). We deployed our OpenIoT Server over a machine running Debian GNU/Linux 6.0.10, with 8-cores of 2.13 GHz processor and 64 GB RAM. Apache OpenMeetings server is installed over a machine with Ubuntu 12.04.5 LTS, 1 core of 2.30GHz and 1 GB RAM, while Android App for sensor data transmission was installed over a mobile device running on Android

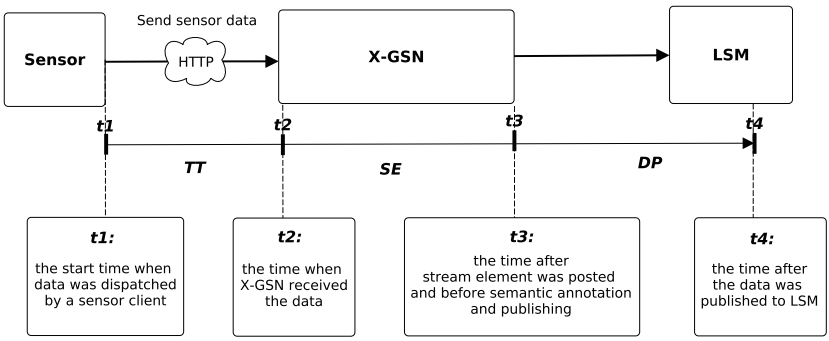

Fig. 6. Different Points for Processing Time Measurements

OS v4.3 (Jelly Bean), with Dual-core 1.5 GHz Krait and 1GB RAM, supporting Android SDK version 8 to the SDK version 21.

Performance. In order to evaluate the performance of our proposed solution, we observed the processing time required by various components of our infrastructure with varying size of the sensory stream. We identified various time points of observation to examine the average time delay for each time point of measurement. Figure 6, illustrates four time points of measurement, where:

- $t1$ is the start time when data generated by a sensor is sent.
- $t2$ is the time when X-GSN receives the data, hence we refer to $TT = t2 - t1$ as to the the time required by the network to send the data from the sensor to the server (transmission time).
- $t3$ indicates the time when the raw data has been processed and stream elements have been created with time stamp allocation to each sensor observation, hence $SE = t3 - t2$ is the time needed to create a semantically annotated stream element.
- $t4$ is the time when semantically annotated stream elements are are successfully published to LSM, hence $DP = t4 - t3$ is the time required for publishing the semantically annotated triples into LSM.

Figure 7 depicts the average processing time required to perform the three main steps of the OpenIoT data processing pipeline, namely, (i) *Transmission Time (TT)*, (ii) *Stream Element Creation Time (SE)*, and (iii) *Data Publishing Time (DP)*. We specified an average delay of 0.5 seconds before sending each sensor observation to avoid overloading the system by concurrent requests (the impact of concurrent requests are investigated in the scalability analysis). All execution times shown in Figure 7 are the average of 5 executions. It is evident from the results that there is no significant delay in the *Stream Element Creation* and *Data Publishing* times, despite the increase in number of sensors from 10 to 10000. Similarly, it is no surprise to see the increase in the *Transmission Time* corresponding to an increase in the network traffic.

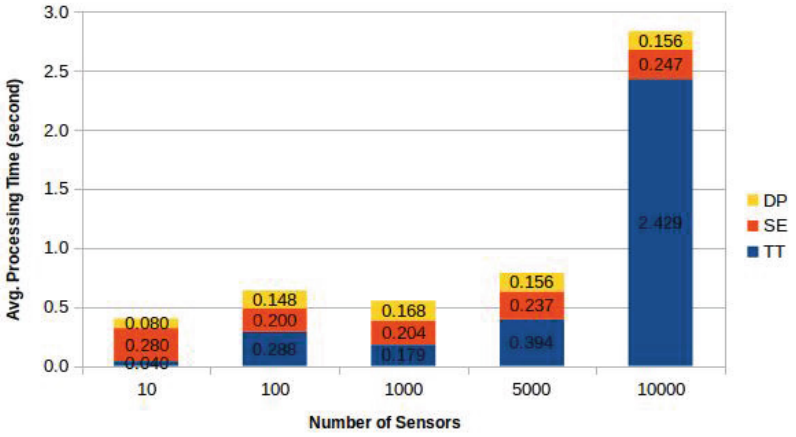

Fig. 7. Average Processing Time with Varying Number of Sensors

Scalability. We conducted our scalability test of the OpenIoT framework by sending concurrent requests with increasing number of users and observed the throughput (ability to deal with the concurrent users/requests per second) for each of the three phases of the OpenIoT data processing pipeline, namely (i) *Data Acquisition*, i.e. the ability of OpenIoT to receive data from sensors, (ii) *Stream Element Creation*, i.e. the ability of OpenIoT to process raw data and assign stream time stamps to each observation, and (iii) *Stream Data Publication*, i.e. the ability of OpenIoT to semantically annotate and publish/store the semantically annotated data within the LSM framework.

We used Apache JMeter [7] for conducting the scalability tests, which is a well known tool to perform stress tests over distributed web applications. We observed the throughput of OpenIoT with increasing number of concurrent requests (10,100,1000,5000 and 10000) sent by Apache JMeter with a ramp-up time of (5,50,500,2500 and 5000) accordingly. We allowed the execution time of 10 minutes after the completion of ramp-up time. As shown by our results in Figure 8, the throughput for the *Data Acquisition* phase remains higher than 200 requests/sec when the input size is 100 concurrent sensor requests or below, and it is reduced to around 76 requests/sec with concurrent requests sent from 10000 sensors. Similar throughput was achieved for *Stream Elements Creation* phase. However, the throughput of the *Stream Data Publication* phase, which is the ability of OpenIoT to publish the semantically annotated sensor data streams either to a streaming channel or storing within a data store, seems to be a bottleneck. Increase in the throughput of the *Stream Data Publication* phase from 1000 sensor inputs onwards, as shown in Figure 8, is a false positive. In fact, a deeper investigation into the results of Apach JMeter logs revealed that

[7] http://jmeter.apache.org/

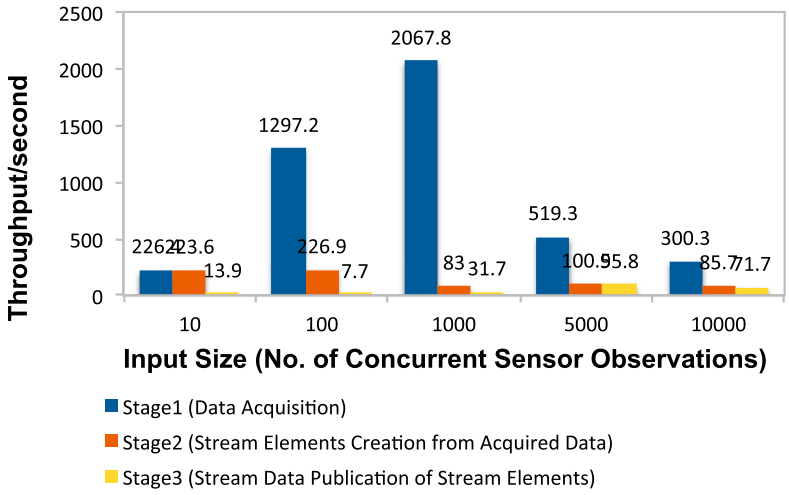

Fig. 8. Throughput of the Various Components of OpenIoT

this higher throughput was achieved because of significant increase in error rate caused by the fact that the LSM server starts refusing connection requests when the number of concurrent users increases beyond 1000. Further investigation is required in this respect to perform experiments where the noise generated by refused connections is filtered out.

5 Discussion and Future Deployment

In this paper, we showcase the applicability of semantic technologies in the IoT space for enterprise communication on the move. We focused on the advantages and feasibility of using the OpenIoT framework (extended with continuous query processing and IoT intelligence) in the Apache OpenMeetings collaboration and communication systems. We characterised requirements that can produce scalable solutions and issues to be investigated more carefully.

As discussed in our introduction and scenario description, semantic-based solutions for IoT in this space can facilitate the deployment of interoperable and flexible IoT-enabled enterprise applications. The ability to semantically integrate and query static and dynamic data makes it easier and more cost-effective to add new external sources and design the business logic (IoT-intelligence) promoting a new market of innovative services that provide significant advantages over ad-hoc IoT deployment. Semantics also helps integrating semantic knowledge about a user independently of the applications producing it (e.g. mobile app for producing sensory input, desktop client for online meeting services, calendar for meeting schedule, etc.) as long as there is a semantic information model that relates the different pieces of knowledge.

Performance. Results are very positive regarding the use of semantics since there is very little and linear impact of semantic-related processing in our IoT-enabled OpenMeetings. The real bottleneck seems to be network traffic, which suggests increasing bandwidth or clever management of queues in order to improve transmission delays.

Scalability. Test results suggests that the actual acquisition and generation of semantic streams can manage up to 200 readings per second if the total concurrent requests by users is not much greater than 100. This could be reasonable in a medium enterprise by constraining the number of concurrent meetings (or participants) that can be scheduled on the OpenMeetings server. If we want the processing pipeline to go as far as the IoT-intelligence goes, we can deal with a much lower throughput of a few (concurrent) sensory input per second, due to the time required to publish the acquired annotated sensor data to the stream processing and reasoning layer.

The X-GSN to LSM communication is performed per-observation by default, this is quite slow and can be very costly when there are a lot of concurrent X-GSN threads (e.g. concurrent sensory input) to be handled. It is worth mentioning that servlet's threads on the server side have been managed without using any optimisation queue, therefore there is easy margin for improvement. Hence, in order to reduce the impact of the bottleneck to publish annotated sensor streams to an application channel via LSM, a possible solution is to either use a cluster version of JBoss hosting X-GSN server, or to configure a queue in the default implementation of X-GSN that makes it possible to buffer the observations.

Deployment Plan. While acting on the application side is entirely dependent on the type of application, acting on the X-GSN implementation is related to improving current semantic solutions and we have already triggerred the discussion to list it as a potential improvement within the OpenIoT developers community. Regarding feasibility for continuous query evaluation and reasoning, we are currently evaluating an initial testbed by mocking up 25 simultaneous meetings using our IoT-MMS. Each meeting consists of 10 attendees including organiser, while 4 type of sensor observations (noise level, proximity, location and light) were monitored for all mobile users. Initial results show that our IoT-MMS is capable of generating simultaneous queries over the proposed test without any substantial performance issues, with similar results as the ones published for the specific stream processing engine evaluation [13]. Following this simulation, we will be setting up a deployment for in-use evaluation of our IoT-MMS system within our partner industry in the next few months, following integration of our solution in a proprietary online collaboration system. This will make it possible to evaluate usability and performances of specific business logic related to online meeting management events and action triggers, and conduct a proper in-use evaluation not only with respect to scalability and performance but also usability and impact.

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proc. of VLDB, pp. 1199–1202. VLDB Endowment (2006)
2. Ait, N.K., Al, J.E.B., Al, A.G.: D2. 3 openiot detailed architecture and proof-of-concept specifications (2013)
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks **54**(15), 2787–2805 (2010)
4. Barnaghi, P., Wang, W., Henson, C., Taylor, K.: Semantics for the internet of things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS) **8**(1), 1–21 (2012)
5. Calbimonte, J.P., Sarni, S., Eberle, J., Aberer, K.: Xgsn: an open-source semantic sensing middleware for the web of things. In: 7th International SSN Workshop (2014)
6. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The {SSN} ontology of the {W3C} semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web **17**, 25–32 (2012)
7. Groza, T., Handschuh, S., Moeller, K.: The nepomuk project-on the way to the social semantic desktop (2007)
8. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
9. Le-Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. Web Semantics: Science, Services and Agents on the World Wide Web **16**, 42–51 (2012)
10. Mehmood, Q., Ali, M.I., Fagan, O., Friel, O., Mileo, A.: Semantically interlinked notification system for ubiquitous presence management. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) ODBASE 2013. LNCS, vol. 8185, pp. 588–605. Springer, Heidelberg (2013)
11. Mileo, A., Abdelrahman, A., Policarpio, S., Hauswirth, M.: StreamRule: a nonmonotonic stream reasoning system for the semantic web. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 247–252. Springer, Heidelberg (2013)
12. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: Vision, applications and research challenges. Ad Hoc Networks **10**(7), 1497–1516 (2012)
13. Phuoc, D.L., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. J. Web Sem. **16**, 42–51 (2012)

14. Sauermann, L., Grimnes, G.A.A., Kiesel, M., Fluit, C., Maus, H., Heim, D., Nadeem, D., Horak, B., Dengel, A.R.: Semantic desktop 2.0: the gnowsis experience. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 887–900. Springer, Heidelberg (2006)
15. Scerri, S., Rivera, I., Debattista, J., Thiel, S., Cortis, K., Attard, J., Knecht, C., Schuller, A., Hermann, F.: di.me: Ontologies for a pervasive information system. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 499–504. Springer, Heidelberg (2014)
16. Wang, W., De, S., Toenjes, R., Reetz, E., Moessner, K.: A comprehensive ontology for knowledge representation in the internet of things. In: IEEE TrustCom, pp. 1793–1798. IEEE (2012)