GRANT AGREEMENT No 609035
FP7-SMARTCITIES-2013

Real-Time IoT Stream Processing and Large-scale Data Analytics for Smart City Applications

Collaborative Project

Semantic Data Stream Annotation for Automated Framework

Document Ref.	D3.1
Document Type	Report
Workpackage	WP3
Lead Contractor	UNIS
Author(s)	Sefki Kolozali (Editor,UNIS), Daniel Puschmann (UNIS), Athanasios Karapntelakis (ERIC), Hongxin Liang (ERIC), Daniel Kuemper (UASO), Thorben Iggena (UASO), Muhammad Intizar Ali (NUIG), Feng Gao (NUIG)
Contributing Partners	UNIS, ERIC, UASO
Planned Delivery Date	M12
Actual Delivery Date	22/09/14
Dissemination Level	Confidential
Status	Completed
Version	V1.3
Reviewed by	Alessandra Mileo, Stefan Bischof

Executive Summary

Smart City is a generic term that refers to interconnection of real-world services, which are provided by smart objects, and sensors that enable interaction with the physical world. Cities are evolving into large interconnected ecosystems in an effort to improve sustainability and operational efficiency of the city services and infrastructure. However, it is often difficult to perform real-time analysis of large amount of heterogeneous data and sensory information that are provided by various sources. This report describes a framework that is a solution of CityPulse for real-time semantic annotation of streaming IoT and Social media data to support dynamic integration into the Web using a middleware. This will enable delivery of large volume of data that can influence the performance of the smart city systems that use IoT data. We present an information model to represent summarisation and reliability of stream data. The framework is evaluated with the data size and average exchanged message time using summarised and raw sensor data. Based on a statistical analysis, a detailed comparison between various sensor points is made to investigate the memory and computational cost for the stream annotation framework.

Table of Contents

Table of Figures

1. Introduction

Recent advances in Information Communication Technologies (ICT) have led to a technological turn in an increasing number of cities by enabling a new type of embedded spatial intelligence that advances the information and knowledge capabilities of communities. This has resulted in emergence of a new concept called Smart City that aims to provide a set of new generation services and infrastructure in an effort to improve living standards and quality of life for citizens by tackling common urban challenges, such as reducing energy consumption, traffic congestion and environmental pollution. To achieve this objective, cities require an interconnection of diverse objects or things, also known as Internet of Things (IoT) that allows monitoring the physical world and providing their virtual representations on the Internet. The availability of such connected objects and sensors open opportunities to observe the status of the physical world in real time, and process this information to improve the operational efficiency of the city services and infrastructure. Thus, combining this technology with smart cities paves the way for intelligent, real-time decision making enabling services such as intelligent traffic management, route optimisation, bus or train arrival prediction, and parking space management.

The integration of a large amount of multi-modal data streams from diverse application domains (e.g. traffic information, parking spaces, bus timetables, waiting times at events, event calendars, and environmental sensors for pollution or weather warnings, GIS databases) is one of the key challenges in developing smart city frameworks. Therefore, information management is a primary concern for smart cities, and until recently most of the solutions that have been created for various scenarios and applications existed in isolation. For instance, traffic or governmental data, encyclopaedic sources such as Wikipedia[1], and forecast data collection (e.g. the UK Met Office[2]) do not interact with each other, even though they can deal with similar kind of data. As a result, information managed by one of these resources may not benefit from information held by the other. This issue becomes much more noticeable once we consider the exchange of information between IoT resources. For this reason, linkage of the IoT data sources with other data sources to enable them to be meaningful is as essential as providing access to IoT data stream through web services.

In parallel, the growing popularity of social networks (e.g. Facebook [3], Twitter[4], Foursquare[5]) made it possible to obtain people-sensed data via participatory sensing, where information stemming from both social and sensor networks can be combined to extract meaningful information to improve context awareness in cities. Citizen sensing (Sheth, 2009) component that may provide complementary or corroborative information is often ignored in the current state-of-the-art analytics for Smart Cities (Filipponi et al., 2010) (Lu et al., 2010).

Given that cities are dynamic and evolving eco-systems, there is also a need to continuously link, interpret and share dynamic information across city stakeholders and citizens in order to utilise information before it is out-dated. Real-time stream annotation is, therefore, another critical endeavour that ought to be dealt with in smart city frameworks.

The quality of information (QoI) and quality of services (QoS) are another challenge that can influence the usability of a data source in heterogeneous and distributed environments. In smart cities, data sources that are emerged from diverse application domains can have different qualities, modalities, and trust and reputation, which need to be identified and associated to a set of criteria that represent

[1] https://www.wikipedia.com
[2] http://www.metoffice.gov.uk/
[3] https://www.facebook.com/
[4] https://www.twitter.com/
[5] https://www.foursquare.com

the quality of information and service. However, although there exists a variety of study in QoS, QoI and data provenance in services area, little attention has been paid to the investigation on QoI trustworthiness and provenance in IoT environments. Therefore, there still remains a need for a method for reliability testing that combines the semantic web languages and quality of information research to overcome the dynamicity and heterogeneity in the IoT systems.

To overcome these issues within the domain of smart cities, we developed a framework in the scope of the CityPulse project[6] for real-time IoT stream annotation that employs a knowledge-based approach to represent data streams and to support mashups. We also show that social data streams in the context of Smart Cities can provide a comprehensive view of events in a city (complementing other modalities such as observations from city authorities). This empowers semantic expressiveness which enables the use of web based technologies and techniques for accessing and linking both sensor data and metadata.

To deal with large amount of data, we use Advanced Message Queuing Protocol (AMPQ) as proposed in (Vinoski, 2006) to increase the communication performance of the system. We also present an information model to provide a representation for summarisation and reliability of IoT stream data. In order to investigate the performance of the framework, a traffic dataset is collected from a city environment. The framework is evaluated with the data size and average exchanged message time using summarised and raw sensor data to investigate the memory and computational cost for the stream annotation framework. This report is based on our work published in (Sefki Kolozali, 2014) (Payam Barnaghi, 2013) that offers a way to represent data stream, and enriches it with semantic annotations.

The remainder of the report is organised as follows. Section 2 describes the requirement analysis and the-state-of-the-art, and the overall functional components of the developed semantic annotation framework. Section 3 demonstrates the virtualisation component of the framework, and presents data wrappers and resource management components to facilitate access to heterogeneous data sources. Section 4 provides an information model for semantic annotation of streams in order to express summarisation, reliability, and complex events regarding the data streams. Section 5 demonstrates the Reliable Information Processing component to measure and process accuracy and trust in data acquisition, federation and aggregation. Section 6 describes a middleware to enable delivery of large volume of data that can influence the performance of the smart city systems that use IoT data. Section 7 details the realisation of semantic annotation framework, and Section 8 concludes the report and described the achievements.

2. Overview of Real-time Stream Annotation Framework

This section is organised in two parts: (i) the state-of-the-art on the key themes addressed by the report, and (ii) the proposed framework for real-time stream annotation for smart cities.

2.1. Requirements and State-of-The-Art

Smart Cities aims the connectivity and interoperability of physical devices. A typical example is a traffic sensor; its current state is sensed by a sensor and changed or reported by an actuator. By making this "thing" available to other things, new use-cases such as energy reduction in transportation happens by avoiding traffic congestion. However, to accomplish such a connectivity and interoperability in a smart city environment, some of the requirements, which have been tackled in this report, are as follows:

[6] http://www.ict-citypulse.eu/page/

i) To provide information in a manner that can be automatically interpreted and exchanged meaningfully among devices and services;

ii) To efficiently deliver information among heterogeneous devices and services through a middleware;

iii) To assess the quality of information with a method of reliability testing.

We will examine these requirements by providing a detailed discussion on the related work that has been carried out to date.

2.1.1. Federation of heterogeneous data streams using semantic description and annotation.

Internet of Things research in recent years has focused on modelling domain knowledge of sensor networks and services. The SSN ontology (Michael Compton, 2012) is one of the most significant efforts in development of an information model for sensory data. The SSN Ontology provides a vocabulary for describing concepts, such as sensors, outputs, observation value and feature of interests. Most notable extensions include ontologies for coastal features, services and roles for emergency response. However, although the SSN ontology defines a high-level scheme of sensor systems, it does not include representation of aggregation/summarisation of observation and measurement data. IoT-A model (Suparna De, 2012) and IoT.est semantic representations (Wei Wang, 2012) describe how to enhance the utilisation and representation of domain knowledge in sensor networks: the former provided an architectural base for further IoT projects, and the latter enhanced the IoT-A model with some service and test concepts.

The Observation and Measurement (O&M) descriptions for sensory data are also described as a part of the Sensor Web Enablement (SWE) standards (Mike Botts, 2008) from the Open Geospatial Consortium (OGC). While it provides several important syntactic descriptions, due to the fact that it is based on XML, it has a weak semantic structure for expressing knowledge, and lacks some important features to describe ontology in more detail. There has been a recent study to improve the semantic richness of the O&M ontology where authors transformed it into Ontology Web Language (OWL) representation (Cory A. Henson, 2009). However, the O&M ontology continues to not only lack temporal features to represent time-series observations in detail, but also semantics for our purposes due to the straightforward approach that has been used in the process.

Another very similar approach has been carried out in (Cory A. Henson, 2009), in which all the XML tags of O&M ontology have been mapped into OWL concepts. However, although the authors present how to access the annotated data through SPARQL queries, these queries are not efficient for the applications that need to access sensor data in real time, as the SPARQL queries generates significant traffic if the sampling rate is high Consequently, there still remains a need for a framework to handle real-time semantic annotation as well as efficient knowledge representation of sensory data in dynamic environments such as smart cities.

To establish a level of interoperability, we will provide modular IoT resource/data and stream representation using modular semantic models that can be adopted and extended according to specific needs of different applications and domains. The concepts and relations in the CityPulse models are adopted from existing common ontologies and semantic models to support interoperability between existing systems and different providers. It will also provide a semantic models and knowledge representation framework in the form of linked data for IoT streams. This will enable linking multiple related IoT data streams in Smart City environments to each other and associating the descriptions to the domain knowledge.

2.1.2.IoT Stream Processing via an Efficient Middleware

Efficient interaction through communication technologies between service consumers and providers, system components, or devices is another significant challenge that needs to be addressed within Smart Cities. The growing quantity of devices that are connected to the Web makes it more difficult to deal with the communication issues among diverse applications and devices. Middleware enables the connection between functionality of the devices or services (e.g. between sensor nodes and higher applications, or among software components) and is therefore located in the application layer. Due to heterogeneity of the devices and platforms, however, the middleware should be lightweight, and offer energy-efficient communication protocols including effective networking capabilities and communication mechanisms to work with or between the high- level applications and networks. There have been numerous studies and developed many applications to deal with real-time data delivery. We will review some of the related studies within this Section.

Even though there are several technologies to establish middleware (SOAP, REST or XML-RPC among others), SOAP and REST are among the most widely used methods for interfacing to the Web. SOAP is a standard protocol proposed by the W3C to interface Web Services, and extends the Remote Procedure Call (XML-RPC) (Gudgin et al., 2007). It is capable of performing procedure calls to distributed services along with access to objects and methods that are residing in remote services. On the other hand, Rest is a style of software architecture for distributed hypermedia systems such as the World Wide Web; and its architecture consists of clients and servers, in which clients initiate requests to servers; servers process requests and return appropriate responses (Fielding, 2000). Compare to the SOAP, REST services allow straightforward and simple way of achieving a global network of smart things due to seamless integration to the Web. On the other hand, SOAP is heavier than RESTful Web services, due to the fact that utilisation of XML increases the time taken to parse the messages (Castillo et al., 2011) (Guinard et al., 2011) (Vinoski, 2002).

However, while most of the existing Web-based approaches use HTTP protocol to transport data between devices, HTTP is half duplex, and functions as a request-response protocol on the client-server computing model. Therefore, HTTP can simulate real-time communication only with a high price - increased latency and high network traffic. As a result, HTML5 has introduced new connectivity methods, such as WebSockets and WebRTC, which provide full duplex network enabling the services and devices to simultaneously send and receive messages (Cha & Yun, 2012). WebSocket is a web technology providing such bi-directional, full-duplex communications channels, over a single TCP socket giving clients and servers a simple way to communicate over a persistent stream without the need for third-party plugins or hacks. Additionally, it serves a purpose for Web applications that require real-time bi-directional communication (Hämäläinen, 2012) (Wang et al., 2013). Many researchers have tested and continue to test Web-Socket usage for real-time applications. While the WebSocket API is currently a W3C draft, the protocol is estimated to provide a three-to-one reduction in latency against half-duplex HTTP polling applications (Pimentel & Nickerson, 2012) (Mandyam & Ehsan, 2012).

Parallel to real-time data delivery over an Internet connection in a peer-to-peer fashion (e.g. WebSockets) that significantly reduces latency, a broker-based messaging is another approach that is introduced in order to reduce the complexity between a number of peers. Utilizing message queues in a broker architecture enables to employ advanced routing patterns. Message Oriented Middleware is an infrastructure -- which involves the two of these approaches, namely peer-to-peer and broker-based messaging -- for loosely coupled inter process communication and aims to handle scalability, flexibility, and reliability in enterprise and cloud environments. It provides several features such as i) asynchronous and synchronous communication mechanisms; ii) data format transformation; iii) loose coupling among applications; iv) parallel processing of messages; v) several levels of Quality of Service support (Curry, 2004).

Today's most notable open standards for messaging are the Advance Message Queuing Protocol (AMQP) and Message Queue Telemetry Transport (MQTT) (Lampesberger, 2014). AMQP allows communication through a session that has been established with TCP connection once peers negotiate with a handshake. Peers exchange so called frames that includes header fields and binary content. In terms of interoperable representation of messages, AMQP provides a self-contained type system, which consists of a set of primitive data types, descriptors for specifying custom types, restricted data types, and composite types for structured information. This enables self-describing annotated content when interaction between heterogeneous platforms occurs. On the other hand, MQTT is a lightweight messaging protocol on top of TCP and low bandwidths as encountered in Internet of Things ecosystems. Although it provides a binary message format and a small fixed-size header of only two bytes, which results in a little overhead, it has a limited bandwidth and has no notion of content type (Fernandes et al., 2013). Therefore, it needs to use custom protocol. For peer-to-peer approach, the Extensible Messaging and Presence Protocol (XMPP) and the text-based Streaming Text Oriented Messaging Protocol (STOMP) are the most well-known interoperable messaging protocols that have resemblance to HTTP which can operate through a bidirectional byte-stream based transport protocol such as TCP or WebSockets (Frecon, 2013).

The Smart City environments are based on real-time as well as event-driven communication models, which suggest that the solely synchronous (i.e. request/response) approach is not adequate for every scenario. For that reason, we use a MOM based approach that consist of XMPP or STOMP to reduce the latency among the components of the system for real-time communication; AMQP to perform transmission when it is triggered by an event occurs of the sink note generates a query. Such dissemination is well supported by the publish/subscribe paradigm where a publisher sends available data to subscribers in an asynchronous way. The more detail regarding these approaches are provided in Section 6.

2.1.3. Reliable Information Processing, QoI, testing and monitoring

Reliability in information processing is one of the most important aspects in the IoT. Quality of Service (QoS) has been widely studied in sensor networks, and has well defined and measurable properties, such as throughput, jitter or packet loss, inherited from the field of network communications. With ontologies like QoSOnt (Dobson et al., 2005) efforts to categorize these quality attributes have been undertaken. In the area of WebServices QoS attributes are subclasses of the ServiceParameter class in the OWL-S ontology. However, although it has been spotted as one crucial item in data networks, the Quality of Information (QoI) is still not well defined and sometimes difficult to measure. Wang and Strong point out the negative impacts of faulty information and their social and economic influences (Richard Y. Wang, 1996); and indicate that not only the information itself is of importance, but the QoI is important too. They identify the QoI through a hierarchical framework which is developed based on a wide list of parameters. This study initiated the initial idea about the categorisation of the QoI parameters into intrinsic, contextual, representational and accessibility data quality.

In (Besiki Stvilia, 2007), a framework is described for the assessment of QoI attributes, based on the work of Wang and Strong. Their work focuses on issues and circumstances compromising the exploitation of annotated QoI values. They identified three possible causes reducing the utilisation of the proposed QoI values: mapping of the information and QoI values, context changes of information consumers and changes to the data sources, e.g. reducing the update frequency to protect the network. A disadvantage of the proposed system is that it mainly depends on the concrete usage, respectively the context, of the information consumer. Parameters like Informativeness or Relevance dependent on a concrete use case and the application processing this information. Additionally it lacks of parameters to describe cost attributes.

To avoid the context dependent usage of QoI, the authors in (Chatschik Bisdikian, 2009) developed a framework, which splits the information quality attributes on data streams into QoI and Value of Information (VoI). QoI attributes represent objective quality attributes where VoI represents the usability from an application point of view. A later improved version (Chatschik Bisdikian, 2013) could be used as a basis for a smart city QoI framework as it contains additional information about cost and provenance.

The provenance of an information can be seen as a QoI attribute (Amotz Bar-Noy, 2011) (Chatschik Bisdikian, 2013). Provenance is the information about the origin of data and thus linked to the reputation and trustworthiness, but has to be examined separately. Typically entities grade each other individually by comparing the performance with their own expectations. Contrary to trust the reputation is aggregated and mainly calculated on the past behaviour of other sensors. For instance, a reputation-based framework developed by Ganeriwal and Srivasta in (Saurabh Ganeriwal, 2004) uses a reputation metric to evaluate the trustworthiness of sensors in a sensor network. To calculate the reputation each sensor maintains a list of reputations for other sensor nodes. Especially in large smart city environments this approach has its limitations. The sensor nodes need the information about neighbouring sensor nodes and require the computational power to grade their neighbours. Furthermore an evaluation of different stream types (e.g. physical sensor and social media stream) cannot be realised on such low level. CityPulse uses a higher-level approach. Reputation is the ratio between the promised/expected quality and the observed quality over all QoI attributes and may decrease if the current quality lies below the promised one. Here, the correctness is just one of many QoI attributes, determined through correlating data sources. In 2012 the W3C published PROV-O[7] an ontology to “form assessments about quality, reliability or trustworthiness” of a piece of data. An alternative to the PROV-O could be the Open Provenance Model[8] (OPM) ontology as it allows a similar description of provenance. CityPulse utilize the PROV-O ontology to link information entities and their sources.

However, current testing approaches in smart city environments are not able to rely on collected test data and require energy consuming procedures to ensure application availability. Beyond distinct information if applications function is working or services QoS parameters are satisfying there is no output information about the information on quality sensors that deliver comparison to their neighbours.

2.2. Real-time Stream Annotation Framework

The real-time annotation framework aims to semantic annotation of IoT stream data by taking into account dimensionality reduction and reliability concepts that have been presented as requirements in the previous section. The framework involves four main units: *a)* virtualisation, *b)* middleware, *c)* Reliable Information Processing and *d)* semantic annotation. Figure 1 depicts the architecture of the annotation framework. In the remainder of the Section each block will be briefly introduced and that each of the blocks will be detailed separately in Sections 3-6.

[7] http://www.w3.org/TR/prov-overview/
[8] http://openprovenance.org/model/opmo

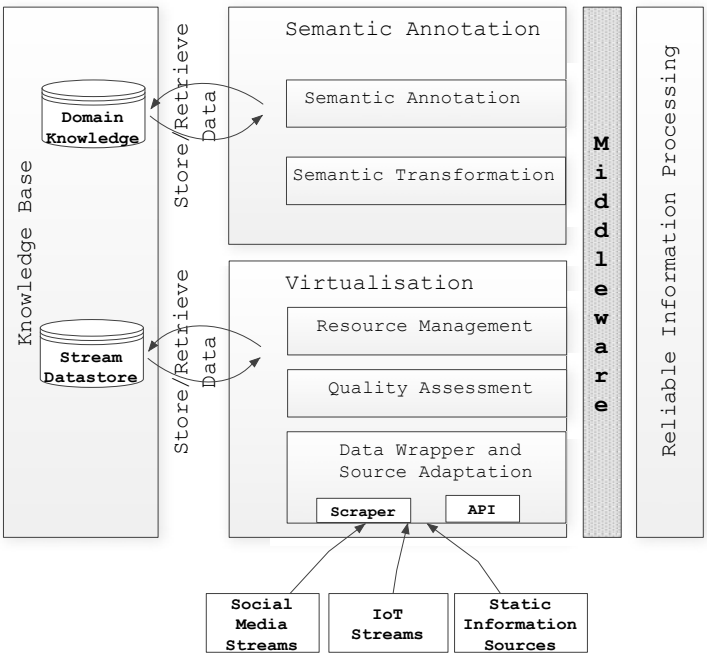


Figure 1 - Framework for real-time stream annotation in smart city applications

2.2.1. Virtualisation

The virtualisation component facilitates access to heterogeneous data sources and infrastructure concealing the technical facets of data streams such as location, storage structure, access format, and streaming technology. The system designates various wrappers to encompass a large number of input formats, while it provides a unified format as output (i.e. RDF or Turtle), which is defined in the system.

In the context of smart cities and real-time information processing, the IoT streams represent the city sensors and actuators as well as data repositories, which collect the information relevant to the operation of the city. The IoT stream virtualisation allows modelling of the resources (e.g. sensors, actuators, data repositories, citizens) in a manner, which enables a device, such as a parking application, to access these resources systematically in interoperable way. The citizen communication devices (e.g. smart phones) can be also used as virtual sensors.

2.2.2. Semantic Annotation

Describing the obtained sensor data stream for interoperability or facilitated search is the core objective of the semantic annotation component, as many information management tools. However, the amount of traffic generated by Smart Cities applications can be voluminous, particularly for real time applications in environments with resource constrained devices, for example sensors with limited bandwidth, memory or power. Therefore, the information model that is being used by the system not only needs to explicitly represent the meaning and relationships of terms in vocabularies but also should be lightweight in order to reduce the traffic and processing time. We use a lightweight information model to represent sensory data, which will be detailed in Section 4.2.

2.2.3. Reliable Information Processing

The dynamicity and heterogeneity of IoT environments involves changes and noise in the data, especially when dealing with data created by people. The methods for information extraction and data

processing, however, require accuracy and trust issues to be taken into account. This module measures and processes accuracy and reputation in data acquisition, federation and aggregation. It integrates techniques for monitoring and testing, ensuring reliable information processing. For example, it provides fault tolerance mechanisms when malfunctioning or disappearing sensor are detected, or providing conflict resolution strategies when data analysis result in conflicting information.

Provenance also plays an important role for Smart cities applications. These applications acquire data from heterogeneous sources, some of them more reliable (e.g. government data), and some of them less reliable (e.g. crowd sourced data). Based on user or application preferences, the application provider could choose to use less reliable data in various cases, e.g. it needs more up to date information. The Reliable Information Processing module performs quality analysis to assert the reliability of the data. To enable the process of annotating data with QoI a list of QoI parameters is defined in Section 5.2 and an ontology is developed that is described in Section 5.3.

2.2.4. Middleware

As IoT applications are distributed systems there is a need for communication between the individual components. On one hand the data coming from the data wrappers has to be passed through to the data processing components such as the Reliable Information Processing and the Semantic Annotation. Since the components can be developed on different platforms there is a need for a unified, platform neutral format for the messages to enable the communication.

On the other hand the components need a possibility to pass signalling messages to each other e.g. to initiate processes or keep the other components informed about recognised events. In order to make sure that these possibly critical signalling messages are delivered in a timely fashion and are not blocked by the continuously published data from the wrappers, different channels are used for the tasks. This will be discussed in more detail in Section 6.

3. Virtualisation

Data wrappers are responsible from the interaction with particular type of external data, for example one may solely obtain data about traffic sensors, another one may retrieve data regarding the weather conditions in a city, and another one that obtains data from social media. This Section presents the data wrappers we use within the CityPulse project. In Section 3.2, we will present the data wrappers that are used to obtain real world data streams from IoT, social media and city information sources in order to create their virtual representation. In Section 3.3, we describe the resource management component that is used as an interface to add description and ID regarding the corresponding data source. In Section 3.4, we describe the quality assessment process that is utilised in the framework.

3.1. Data Wrappers for IoT Resources

IoT data comes from a lot of different sources and in many different facets. The data sources can be completely different from each other, as is the case with data from social media streams versus numerical data from physical sensors. But even when comparing two different numerical sensor sources there exists lot of variety in terms of structure of the data as well as accessibility. Therefore Data Wrappers are needed which are specifically designed to cope with the underlying data source, can fetch the data and send it to the CityPulse framework via the middleware. New data sources can then easily be added in the future just by providing an appropriate data wrapper.

3.1.1. CKAN Wrapper

Open Data Aarhus (ODAA) is built on the Drupal CMS system, where CKAN is comparable to a CMS system, but only for open data. CKAN's target audience is not the end user, but developers who can use the data to develop, for example apps for devices. CKAN is built with Python on the backend and JavaScript on the frontend, and uses the Python web framework and SQL Alchemy as its Object-relational Mapping (ORM). Its database engine is PostgreSQL and its search is powered by SOLR. CKAN has a modular architecture that allows extensions to be developed to provide additional features such as harvesting or data upload. CKAN uses its internal data model to store metadata about different records, and presents it on a web interface that allows users to browse and search metadata. It also offers a powerful API that allows third-party applications and services to be built around it.

For the CityPulse project we have implemented a data wrapper in python, which pulls in the data from the CKAN API in real-time corresponding to the update rate of the data set (e.g. every 5 minutes for the real-time traffic data). For each data sample corresponding raw values and metadata are fetched. Then the messages are transformed into the format defined in Section 4.1 and Section 4.2 and then published via the middleware to all subscribing components.

3.1.2. IoT Framework Data Wrapper

Data is provided from Ericsson's IoT framework [9]. The IoT framework supports both "raw" and "virtual" streams. Raw streams contain unprocessed sensor readings, while virtual streams are results of computational processes of raw streams (e.g. minimum, maximum, average values). The IoT Framework provides a RESTful API[10] for managing data streams and associated data points where CRUD (create, read, update, delete) operations on data streams and data points are supported using HTTP requests.

[9] https://labs.ericsson.com/sites/default/files/blogs/final_product_report.pdf

[10] https://github.com/projectcs13/sensor-cloud/wiki/API

A wrapper for IoT framework will support the aforementioned API, so that higher layers of the architecture can create, subscribe to, and receive data from arbitrary data streams.

3.1.3. Data Wrapper for City of Brasov

The data wrapper is used for fetching data from the City of Brasov information platform and publish via a middleware to all subscribing components of the CityPulse framework. The information platform has a bundle of web services, which provides information about the incidents reported by the citizens, the status of ski slopes and skiing infrastructure from Poiana Brasov resort. The web services deliver the data in JSON format. Utilising the platform allows to benefit from having access to several traffic video streams from important/critical areas of the city.

The wrappers are responsible also for data format adaptation and the messages published by them are compliant with the format defined in Section 4.1 and 4.2. The data wrapper is based on Mosami API for processing the stream of images, which enables to determine the traffic congestion of the specific area.

3.1.4. OpenIoT and LSM/X-GSN Wrappers

OpenIoT is an open source middleware for getting information from sensor clouds, without worrying what exact sensors are used. OpenIoT can collect and process data from virtually any sensor in the world, including physical devices, sensor processing algorithms, social media processing algorithms and more[8]. OpenIoT combines and enhances results from leading edge middleware projects, such as the Global Sensor Networks (GSN) [9] and the Linked Sensor Middleware (LSM) [10]. Wrappers developed by LSM as well as GSN are supported by OpenIoT middleware, in what follows, we give brief description of the wrappers developed within both LSM and GSN middleware.

There are two basic ways to import stream data into LSM: pull-based and push-based. In the pull-based approach, LSM periodically polls a data source similar to a data feed. In contrast, the push-based approach enables data sources to actively send data to LSM. Both mechanisms are used within the OpenIoT infrastructure.

X-GSN is an extension of GSN and therefore supports all virtual sensors and wrapper developed for the GSN middleware[11]. X-GSN is deployed as web server which will be running and listening on default port 22001. All wrappers are built as subclasses of GSN wrappers and each wrapper acts as a thread in the GSN. X-GSN provides three mechanisms to semantically annotate sensor data, i) sensor metadata document, ii) RDF metadata description, and iii) a REST Service. Detailed documentation about OpenIoT and X-GSN including instructions for usage can be found at: https://github.com/OpenIotOrg/openiot/wiki

3.2. Data Wrappers for Social Media

Typically, a city has many departments such as public safety, urban planning, energy and water, environmental, transportation, social programs, and education. Some of the services offered by these departments are dynamic, e.g., transportation services may vary in response to sporting and music events, accidents, and weather conditions. Timely understanding of the situation is important for city authorities. Figure 2 depicts real-world city events reported directly by citizens in near real-time on twitter. They relate to power outages, poor water quality, a procession, and a delay experienced in public transport system. This information complements sensor data or textual data from conventional sources such as city departments. For example, sensors deployed on a road may report reduced speed of vehicles, which can be explained by the procession obstructing traffic.

[11] http://sourceforge.net/apps/trac/gsn/wiki/Documentation

Figure 2 - Tweets reporting various concerns about a city spanning power supply, water quality, traffic jams, and public transport delays

Twitter (a microblogging platform) has developed into a near real-time source of in- formation spanning heterogeneous topics of varying importance. With over 500 million users worldwide, twitter generates 500 million tweets a day. Increasingly, tweets do provide interesting and vital information such as status of public transport, traffic and environmental conditions, public safety, and general events in a city. For collecting twitter data, we use the twitter streaming API with location bounding box as City of Aarhus. We leverage the open domain knowledge available for a city, for instance, vocabulary related to traffic from OSM for city locations. For those domains not covered by the available sources, we aim to develop a vocabulary for event terms.

3.3. Resource Management

The resource management component interface performs run-time management of data wrappers. As there are diverse data sources and domains in city environment, it manages and adds metadata information about the data sources currently feeding information to semantic annotation component. An example of resource management would be providing information regarding the application domain and the data provider such as stream #5 is a traffic data and provided by City of Aarhus. Providing descriptive metadata regarding the data source will not only be helpful for semantic annotation of the sensory data but also discovery and identification, which will be discussed in Deliverable 3.2, as such information can be used to search and locate an object using location and application domain.

3.4. Quality Assessment

To obtain scalability whilst using information of incoming data streams for the Reliable Information Processing (WP4), we attached atomic monitoring components directly at the virtualisation level. This enables to assess incoming data streams based on annotated quality parameters and a value dependent time series analysis to model and evaluate expected stream behaviour. An example use case for the atomic monitoring could be an outdoor temperature sensor, which should deliver values in an expected range for outdoor sensors. If a sensor reports a temperature of -100°C it can be assumed that the sensor is not working correctly. With utilisation of time series analysis it is additionally possible to compare the expected value with the measured value automatically detect potential false temperature values. In both cases an event is created to inform the QoI Evaluation System about the faulty information extracted from the monitored data stream.

The integration of the atomic monitoring enables resource efficient analysis of stream information accounting for resource restrictions. It prevents proliferation of information and safes networking resources since information is aggregated before events are delivered to the Reliable Information Processing component. These events are used to update quality parameters and to enable monitoring the correlation in order to assess the quality of associated streams which no need of storing information to the Reliable Information Processing component. As a result the atomic monitoring can be seen as a kind of pre-processing before the complex annotation of data streams is handled within the following QoI Evaluation System component.

4. Semantic Annotation and Data Modelling

This Section details the process of annotation of sensory data with semantic descriptions specifying spatial, temporal and thematic attributes. Section 4.1 explains the semantic transformation and annotation process in which the delivered IoT data is being transformed from middleware message form into a bag of metadata; and annotated to provide a basis for interoperation among different components using machine interpretable language (i.e. RDF and Turtle). Section 4.2 presents an information model, for semantic modelling of the streaming data, as an extension of the SSN ontology which allows representing not only sensor observations but also aggregated values and temporal entities such as segments and data points involved in a data stream. Section 4.3 details the complex event service ontology which has been developed to describe event services and requests.

4.1. Semantic Annotation

4.1.1. Stream Transformation

This component aims to transform a message delivered from the virtualisation component into a bag of metadata to be used in the semantic annotation process. In the context of the Semantic Web, syntactically defined messages that are provided by IoT applications need to be conceptually reengineered in order to extract the semantics (i.e. precise meaning) of the intended actions and the underlying IoT domain concepts. As IoT data can be acquired in various file format, the component functions as a transition unit between semantic annotation and virtualisation component. However, this requires not only structural transformation but also semantic transformation. This conceptual reengineering can be referred to as semantic transformation. The ultimate result of semantic transformation is a set of semantic models, which are used for annotation, based on the provided information model, which contains corresponding references to extracted semantics.

4.1.2. Semantic Annotation of IoT and Social Media Resources

While the Internet of Things (IoT is an extension of the internet into the physical world that allows interaction with physical entities in their environment, considering the fact these entities could be almost anything (e.g. human, robot, car, or house), there is a need to use ontologies to represent the domain knowledge in order to deal with various forms of heterogeneity, including data modality (e.g. sensed, textual, and graphical), data model (e.g. terminology), and data representation (e.g. XML, RDF). Utilisation of semantic technologies for IoT advances interoperability among IoT resources, information models, data providers and consumers. The Semantic Web technologies for knowledge representation, information sharing and interlinking heterogeneous resources provide some remarkable solutions to the stated problem scenario.

4.1.2.1. Semantic Annotation of IoT Data Stream

The semantic annotation component carries out annotation of the virtualised data streams that have been provided by data wrappers given in Section 2.1. In an effort to perform the semantic annotation we use 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 it allows sensor capabilities to be defined in accordance with existing conditions as well as to be integrated with Linked Data, which aims at making information freely available on the Semantic Web in order to create bridges between independent data sources to have a large machine-processable data web. For example, sensors and their data can be linked to geographic data (e.g. correlated natural phenomena), user-generated data (e.g. social feedback), and government data (e.g. public transportation) through our knowledge-based approach

The semantic annotation component supports a communication protocol that allows receiving

information from other components in order to further update the knowledge base, where all annotated data is kept. This can especially be crucial in smart city environments where each component might need different time period to accomplish a particular process. In our case, for instance, the semantic annotation component receives data from the Reliable Information Processing component regarding the trustworthiness and provenance and updates the knowledge base. Since sometimes this process may take a month to be completed, it is very important to handle semantic annotation task asynchronously.

4.1.2.2. Semantic Annotation of Social Media Data

Current social media annotation techniques use event specific patterns based on event types (Grishman et al., 2002)(Tanev et al., 2008). The text is expected to have some structure (e.g., news documents). Such a technique does not scale for city events from twitter text due to the informal nature of tweets. Further, the aggregation is done at a cluster level, which is too coarse grained for city related events. Important city events may be reported by few citizens given the wide scope of topics on twitter (Kwak et al., 2010). In such situations, clustering based techniques (Naughton et al., 2006) may fail to separate minority tweets related to the city infrastructure. Twitter messages may be noisy and convoluted with little context in which the message was generated. Such a characteristic of tweets challenges a dictionary-based approach for spotting location and event terms. For example, Figure 3 shows one of many instances where a single event term 'accident' is used in different contexts.

(a) Accident in the context of a dream (b) Accident in the context of a road accident

Figure 3 - Addressing ambiguity: challenge for event extraction - tweets reporting very different events using the same event term "accident"

The tweet Figure 3(a) refers to the dream a person had while the tweet Figure 3(b) actually refers to an accident. The tweet Figure 3(a) does not contain any location information while the tweet Figure 3(b) has "Golden Highway" as the location. It is clear that relying solely on a dictionary based spotting of event terms cannot capture these nuances due to context-sensitive dependencies between words. It is possible to use a purely dictionary based approach for spotting event terms but would require human inspection for accurate tagging. Such manual inspection of the results of event spotting is infeasible because of the volume and velocity of the tweets, and the need for quick action. In order to automate this process, we formalized this problem of spotting event terms and location names as a sequence-labelling problem. We will evaluate the performance of dictionary based spotting of event and location terms for a relative comparison with sequence labelling models. Additionally, we will provide some insights on using dictionary-based approaches for creating training data instead of directly using the dictionary for entity spotting. In practice, the training data may require some cleaning depending on the required accuracy of spotting and availability of resources. Our approach is motivated by the open domain event extraction from twitter (Ritter et al., 2012). More detail regarding the event detection from social media data will be provided in Deliverable 3.4.

4.2. Stream Annotation Ontology

Smart city applications use data from different stream sources. Therefore, the amount of traffic generated by these applications can be voluminous, particularly for real time applications in environments with resource constrain devices, for example sensors with limited bandwidth, memory or power. While the proposed data model should be lightweight in order to reduce the traffic and processing time, it should explicitly represent the meaning and relationships of terms in vocabularies. In this project, we propose a lightweight data model, which uses well-known models to represent IoT.

The CityPulse information model contains 3 main modules, namely Stream Annotation Ontology (SAO), Quality, and Complex Event information model that will be given in Deliverable 3.2. Figure 4 shows an overview of the proposed information model. Some of these modules have been adapted from the IoT.est model (Wei Wang, 2012).

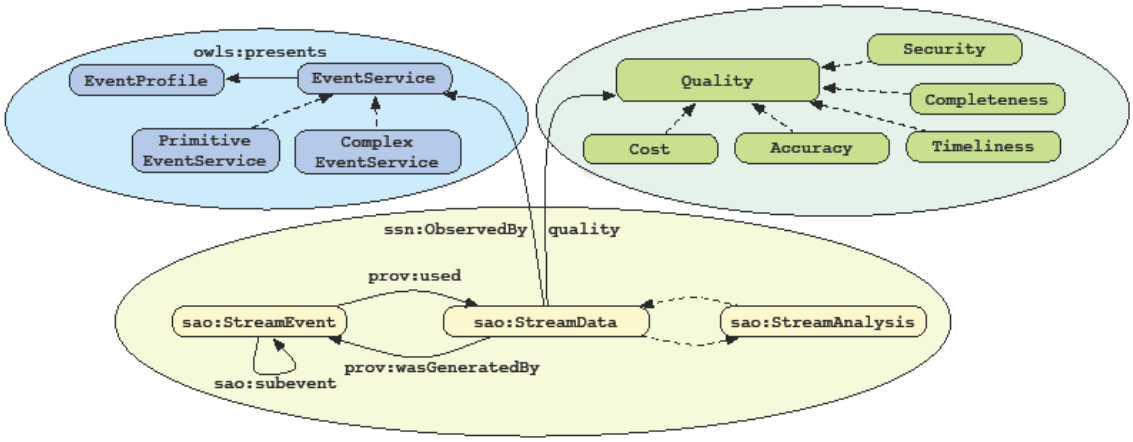


Figure 4 - Describing a stream annotation workflow using the CityPulse Information Model

Representing IoT stream data is an important requirement in semantic stream data applications, as well as in knowledge-based environments for Smart Cities. The SAO can be used to express the features of a stream data. It allows publishing content-derived data about IoT streams and provides concepts such as sao:StreamData, sao:Segment, sao:SegmentAnalysis on top of the TimeLine[12] and IoTest models. The SAO uses the broad definition of the StreamEvent concept in order to express an artificial classification of a time region, corresponding to a particular stream data. It also extends the sensor observations described in SSN Ontology (ssn:Observations) through a concept, sao:StreamData, that allows to describe sao:Segment or sao:Point linked to time intervals or time instants. Figure 5 shows the basic structure of the ontology.

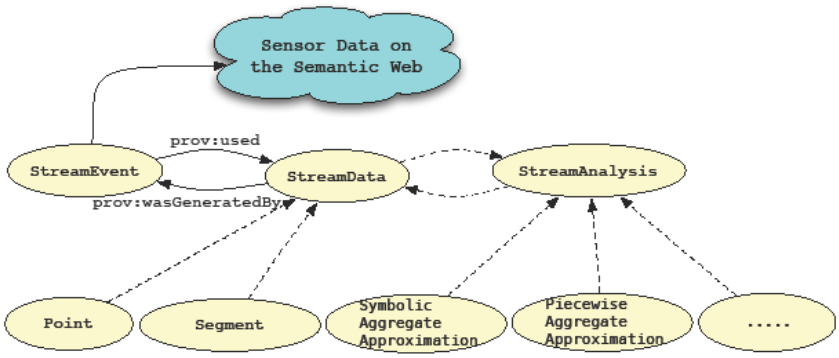


Figure 5 - Depiction of the main concepts and relationships in the Stream Annotation Ontology

In the context of smart cities, dimensionality reduction of data stream or stream transformations obtained through shifted (overlapping) windows can result in a data rate, different from the sample rate of the original sensor observation. Using the SAO Ontology, we can describe a data stream and a timeline instance to link the segment description with the time extent of a temporal entity representing the data stream. Thus, we can express a stream data as a time interval on the universal timeline, and also relate such an interval with the corresponding interval on the discrete timeline along with its discrete sampling rate. With regards to the previous conceptualisations of sensory data, the

[12] Timeline Ontology extends OWL-Time with various timelines (e.g. universal or discrete), temporal concepts, such as instants and intervals, and interval relationships. Available at: http://motools.sourceforge.net/timeline/ timeline.html

SAO ontology deals with representation of aggregated stream data and temporal characteristics. It is free from deep taxonomical organisation, and does not attempt to describe the deep interrelationships or computation of stream data.

4.3. Complex Event Service Ontology

A Complex Event Service (CES) ontology is developed to describe event services and requests. CES ontology is an extension of OWL-S. OWL-S is a standardized ontology to describe, discover and compose semantic web services. Figure 6 illustrates the overview of CES ontology. An event service is described with a *Grounding* and an *EventProfile*. The concept of Grounding in OWL-S tells an event consumer how to access the event service by providing information on service protocol, message formats etc. An *EventProfile* is comparable to the *ServiceProfile* in OWL-S, which describes the events transmitted by the service

An *Event Profile* describes a type of event with a Pattern and Non-Functional Properties (NFP). A *Pattern* describes the correlations between a set of member events involved in the pattern. An event pattern may have other patterns or (primitive) event services as sub-components, making it a tree structure. An event profile without a *Pattern* describes a primitive event service; otherwise it describes a complex event service. *NFP* refers to the QoI and/or QoS metrics (e.g., precision, reliability, cost), they are modelled as sub-classes of *ServiceParameter* in OWL-S.

An *EventRequest* is specified as an incomplete *EventService* description, without specific bindings to the set of federated event services used by the requested complex event. *Constraints* can be specified by users to declare their requirements on the event pattern and NFPs in *EventRequests*. Preferences can be used to specify a weight between 0 to 1 over different quality metrics representing users' preferences on QoS metrics: higher weight indicate the user cares more on the particular QoS metric. More detailed information regarding the complex event service will be provided in deliverable report 3.2.

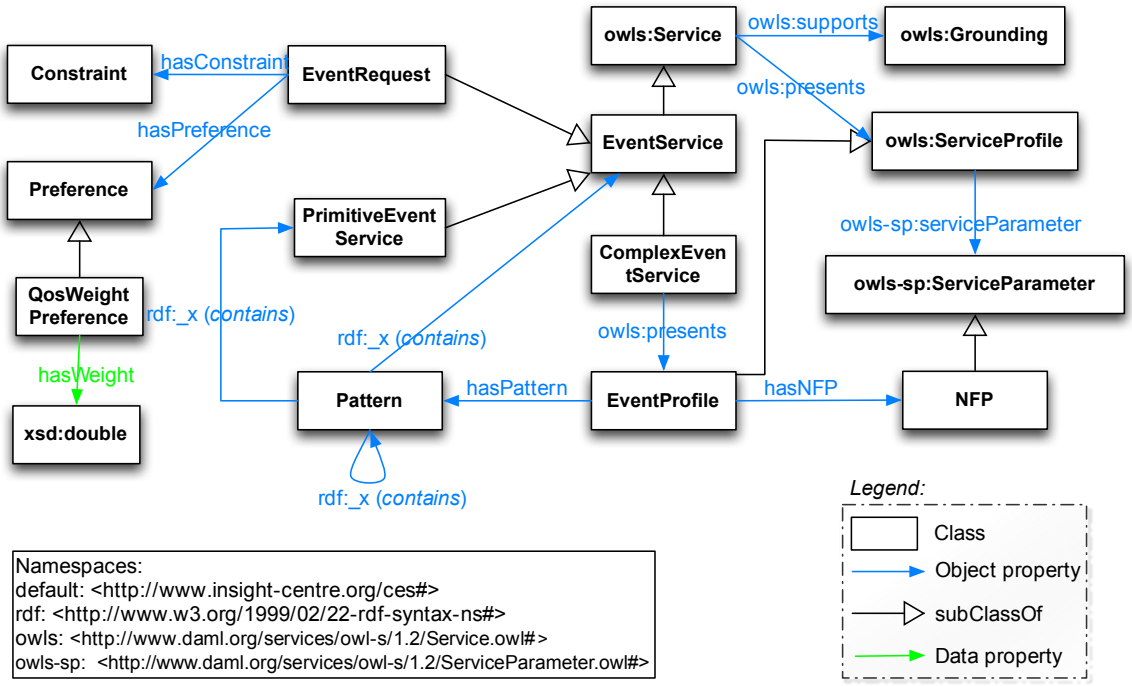


Figure 6 - Overview of Complex Event Service Ontology

5. Reliable Information Processing

The process of QoI and provenance annotation of raw stream data, which is used within the Reliable Information Processing component in the CityPulse framework, is described in detail in the following sections. In Section 5.1 the general idea of QoI and provenance annotation is explained and first parameters for the annotation are defined or extracted from literature. In Section 5.2 we highlight the parameters that are used to measure quality in detail. Finally we present the information model that is developed to represent an application independent view of quality and provenance information for data streams in Section 5.3.

While Quality of Service (QoS) has been widely studied in sensor networks and has well defined measurable properties (e.g. throughput, jitter or packet loss), which are inherited from the field of network communications, the Quality of Information (QoI) is not well defined and sometimes difficult to measure. In this section, we present quality and provenance annotation process of data streams, and highlight the parameters that are used to obtain corresponding measurements. In order to explicitly represent the reliability of the information and the adequacy of the data in the Semantic Web environment, we also develop an information model, which uses an existing ontology to describe the provenance of the processed information.

5.1. Annotation of Quality and Provenance

A key aspect for choosing an appropriate data source is the quality of the provided source. Determining the complete quality requires identification of the quality of the source (e.g. a sensor or a social stream), which is made available through a wrapper service, as well as the quality of the provided data (e.g. temperature values, traffic conditions). Hence, in the CityPulse project quality is considered as the combination of QoS and QoI metrics. A clear distinction of metrics in each category is not always possible, since there can be a close relationship between their meanings. For example, the Precision of a sensor and the Accuracy of a sensor value may describe the identical metric.

An observation made in the real world is announced as an instance of StreamData. CityPulse considers a StreamData instance as a composition of one or more Segments provided by an Entity, e.g. a sensor device or a social media platform. For example a weather sensor may provide temperature, humidity, luminance and wind speed values. Each Segment has its own quality independent from the other Segments. The quality of the StreamData is computed out of all qualities from each Segment. The provenance of a StreamData is called “Agent” which possesses a Reputation as an indicator of its performance over time. The reputation is determined by a Reputation System, which is a sub component of the Reliable Information Processing (see deliverable 2.2). One task of the Reputation System is the observance of data-streams and to check if the delivered values and information are correct, for example through a comparison with neighbouring sensors. Based on this analysis the reputation can be decreased or increased. The Reputation is the key attribute to decide whether information is believable and trustworthy. An Agent can be either a Person carrying a sensor or an Organization deploying a sensor network or a piece of software. The Agent and Entity classes as well as their subclasses are defined in more detail in the PROV-O ontology[13].

The concepts that are used to assess the quality of a StreamData or a Segment are as follows: Cost, Timeliness, Communication, Security and Accuracy. The following list defines the categories that are involved in Quality concept. The definitions are based on the studies published in (Irfan Rafique, 2012) (Amotz Bar-Noy, 2011) (Natale, 2011).

[13] http://www.w3.org/TR/2013/REC-prov-dm-20130430/

- Accuracy: *"The degree to which delivered information is correct, precise, credible, traceable and current in a specific context of use"*
  - Correctness: *"The extent to which information is reliable in the sense of being free of errors."*
  - Precision: *"Extent of detail"*
  - Completeness*: "The ratio of attribute values compared to expected parameters"*
- Communication: *"Describing the technical realisation of the communication used to get the information"*
  - Packet Loss: *"The probability that a set of data / a packet will not be transported correctly from the source to its sink"*
  - Bitrate: *"Min/Avrg/Max amount of bandwidth that is required to transport the stream"*
  - Jitter: *"Deviation from true periodicity of an assumed periodic signal"*
  - Latency: *"measure of the time delay between the stream is sent and received in the virtualisation layer"*
  - Queuing: *"Order of streaming information. Could information measured at time T occur in the stream after Information of T+1 was received"*
- Cost
  - Energy Consumption: *"The amount of energy used to access the steam"*
  - Monetary Consumption: *"Is the usage of the stream free of charge or how much does it cost"*
  - Network Consumption: *"How much traffic is caused by usage of the data source"*
- Security
  - Confidentiality: *"The degree to which information has attributes that ensure that it is only accessible and interpretable by authorized users in a specific context of use."*
  - Encryption: *"Utilised encryption/signing methods to protect the stream in terms of: reading and altering"*
- Timeliness
  - Age: *"The time an information was created/measured/sensed"*
  - Volatility: *"The amount of time the information remains valid in the context of a particular activity"*

5.2. Quality Parameters

Table 1 shows a summarization of all Quality parameters with their measurement units and value ranges. The example column shows what values can be expected after a successful annotation for the specific parameter. The last two columns indicate if the parameter should be annotated during the registration process and if it is updated within the QoI calculation process.

Parameter-name	subcategory	abstraction level	Measurement unit	Range	Example	annotated during registration	updated during QoI calculation
Accuracy							
Correctness		information	Probability that information is within the range of precision and completeness	0-1	0.88	should	must
Precision							
	Resolution	information	absolute value in sensing unit	(0;∞]	0.1°C	should	should not
	Deviation (max)	information	maximum deviation percentage	0-1	+-0.1	should	should not
Completeness		information	Probability that all stream data sets contain the defined values and are updated in their defined frequency	0-1	0.97	should not	should
Communication							
Network Performance							
	Error rate (Packet Loss)	technical	Ratio	0-1	1/10^6	should	may
	Bandwith (Bitrate)	technical	Bits per second	0-∞	100bps	should	may
	Latency	technical	(mili, micro)seconds	0-∞	30ms	should	may
	Jitter	technical	(Milli)Seconds	0-∞	10ms or 2500ms	should	should
	Throughput	technical	Bits per second	0-∞	100bps	should	may
Queuing							
	Queuing Type	technical	Queue Type	FIFO|LIFO|unordered|...	FIFO	should	must not
	Ordered	technical	Probability that data sets arrive in the defined queuing order	0-1	0.92	should	should
Cost							
Energy Consumption		technical	Defined per information or per operating time	0-∞	1W	should	must not
Monetary Consumption		operational	Defined per information or per operating time	0-∞	3ct or 3ct/100 requests	should	must not
Network Consumption		technical	Defined per information or per operating time	0-∞	10bps	should	must not
Security							
Confidentiality (reuse of rights ontology, e.g. http://creativecommons.org/ns)							
	licence defintion	operational	Reference to Licence class, e.g. http://creativecommons.org/ns#Licence	reference	-	should	must not
	may be used...	operational	Reference to Permission class, e.g. http://creativecommons.org/ns#Permission	reference	-	should	must not
	may be published	operational	Reference to Permission class, e.g. http://creativecommons.org/ns#Permission	reference	-	should	must not
Encryption		operational	Encryption method, authority for key management	as defined in RFC4253 p. 9	aes256-cbc	should	must not
Signing							
	authority	technical	Certificate authority	text	cacert.org	should	must not
	public key	technical	Key to decrypt signatures	binary	-	should	must not
Timeliness							
Age		information	maximum time between measurement and publication	0-∞	180s	should	should
Volatility		information	Average Duration how long the information is usable, measured in seconds	0-∞	3600s	should	should
Frequency		technical	Maximum timespan between two data sets	0-∞	1update per 60s	should	should

Table 1 - Summary of Quality parameters

5.3. Quality and Provenance Ontology

Figure 7 shows the quality ontology used for the Reliable Information Processing. It contains the parameters described in previous Section. The quality ontology uses the PROV-O ontology to describe the provenance and reputation of data sources. An alternative to the PROV-O could be the

Open Provenance Model[14] (OPM) ontology as it allows a similar description of provenance. Since the PROV-O as it is used in the SOA ontology (Section 4.2) to describe the relation between StreamData and StreamEvent, it is also reused for the provenance annotation.

Figure 7 - Quality Ontology

The quality ontology uses the same concept with StreamData as the Stream Annotation Ontology described in Section 4.2. To specify the provenance of the StreamData a property “hasProvenance” is defined. Destination of this relation is an “Agent” of the PROV-O ontology who could be an organization, software agent or a person. Every Agent has a Reputation via the “hasReputation” property to describe the trustworthiness. As a result it is possible to get a reputation for every single StreamData.

Main aspect of the quality ontology is the description of data streams with quality attributes. This quality can be used in the further processing of the streams, for example to find a stream with a high precision that is enough for a specific use case. The Quality class has the five subclasses Cost, Security, Accuracy, Communication and Timeliness.

The quality ontology is an important part of the Reliable Information Processing component. Via the ontology it is possible to annotate data with additional information regarding its quality. In addition data sources can be annotated with reputation to determine their trustworthiness. Both, the QoI and the reputation, ensure that applications for the CityPulse framework can operate accurately with the best available information. An example for handling information with QoI and reputation could be a simple bus planning application. The planning application needs a data stream with the times of bus departures for a bus stop. There are different data streams available. On the one hand an official data stream of the municipality with the departure times. As this is the official timetable it has a high reputation with a good QoI. On the other hand there is an aggregated data stream of two persons who are reporting that the bus is delayed. As the aggregated stream only contains two reporting persons, who are unknown to the CityPulse framework, this data stream has a lower reputation as the official stream. Furthermore the aggregated stream has a lower QoI than the official data stream because the delays are reported manually and are less precise. Hence the official stream is chosen. But if more persons report a delayed bus the reputation of the aggregated stream increases. Additionally the QoI, for example the precision, could increase if different persons report similar delays. If the reputation and QoI are high enough the Reputation and QoI Evaluation System will notify stream selecting components about a more suitable data stream. Now the application should also consider the data from the second stream and show a delay to the user. For further planning the delay of the bus should be included in the computations to avoid missing of a connected bus.

[14] Summary of Quality parameters

6. Middleware

While virtualisation component is introduced to facilitate seamless access and management of sensor observation and measurement data based on semantic web technologies, there is a need for a middleware to exchange messages among different components. In this Section, we describe the technologies that we have previously discussed in Section x. We present a peer-to-peer, WebSocket-enabled middleware for real-time message delivery, and Broker-messaging communication for event-driven message delivery, which has been used with apache thrift technology for cross-language development of software components. The proposed approaches are illustrated in Figure 8.

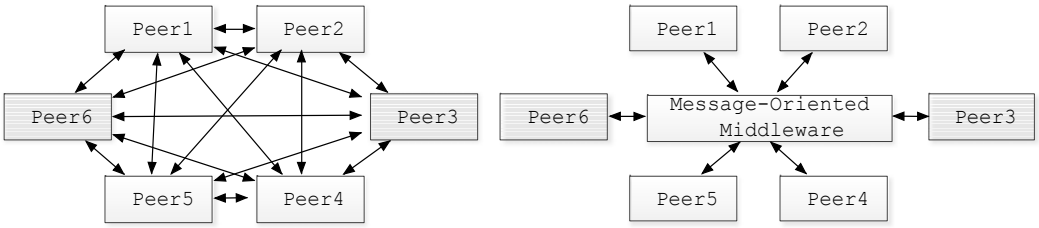


Figure 8 – The architecture of peer-to-peer (left) and broker-messaging communication (right). A peer-to-peer communication can provide a robust peer-to-peer real-time communication, while a message broker can better deal with event-driven communication and reduce the communication complexity in crowded networks where a peer can participate as a client, service or both.

6.1. WebSocket-Enabled Peer-to-Peer Communication

In real-time communication systems latency is very important issue. Therefore, middleware should be capable of delivering messages on time among different services or sensors. Traditional HTTP based approaches use HTTP polling or HTP long-polling techniques in order to communicate among clients and servers. The polling approach is a straightforward way to obtain new real-time communication on the Web with no need of utilisation of non-standard mechanism. The basic idea for polling is that the client makes periodical request to the server through the standard HTTP(S) ports, and the server sends data within the response in case there is data. For long polling, on the other hand, a client establishes a connection to the server and the connection is retained open until new data is available or a timeout occurs. Once the client receives a response to the request a new connection to the server is established. However, while with utilisation of these techniques an application must repeat HTTP headers in each request and response between client and server; this may lead to greater communication cost depending on the application.

WebSockets are amongst the technologies which have been introduced by HTML5, and cuts down on latency simply because it doesn't rely on request and response mechanism as in Restful architectures. WebSockets enables to have full-duplex communication via a single socket between a client and a remote server. Thus, this technology minimizes the quantity of opening ports on server sides compare with the traditional means of sending or retrieving messages (Pimentel & Nickerson, 2012) (Mandyam & Ehsan, 2012) (Cha & Yun, 2012). Figure 9 illustrates the differences in the communication architectures of polling, long polling and WebSocket models.

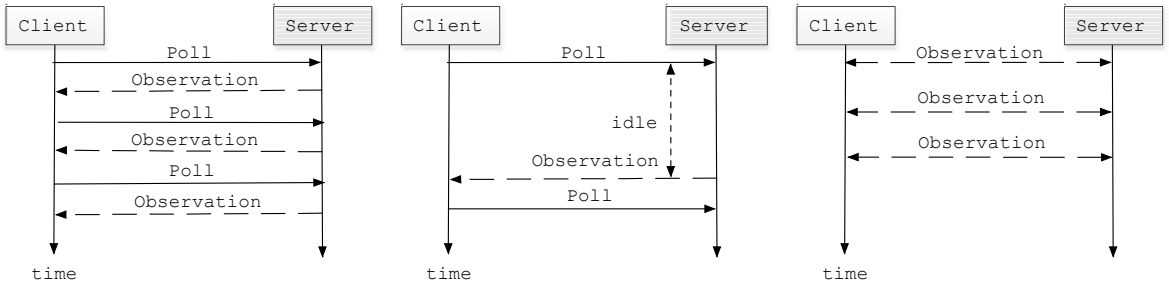


Figure 9 – The message exchange models of HTTP Polling (left), HTTP Long Polling (centre), and WebSocket protocols

The emergence of WebSocket protocol has brought new alternatives to support on the Web, such as XMPP and STOMP. These are application level protocols for exchanging information between any network entities in near real-time; and their main purpose is to address the issues with instant messaging (IM) and presence applications of that time. XMPP exchanges messages in form of XML stanzas and bodies are restricted to text. It allows client to server and also service to server communication in case of federated services, and can be extended to publish/subscribe scenarios. STOMP enables a client and a server to establish a session and asynchronously exchange messages. It supports both bilateral messaging and broker-based publish-subscribe (Frecon, 2013). Even though, both of the technologies are compatible with WebSocket based communication, we'll further investigate to develop the WebSocket based communication, and provide details in deliverable 3.2.

6.2. Broker-Messaging Communication

There are many solutions that offer communication in distributed systems. The shortcomings of the alternatives are combinations of coupling of space (i.e. sender and receiver need to hold references about each other), time (i.e. the components need to interact at the same time) and synchronisation (i.e. the individual components block their activity while waiting for other processes to finish). To solve these issues we use a publish/subscribe mechanism, which decouples time, space and synchronisation. Furthermore the message delivery logic lies with the message broker, decoupling it from the application layer.

In particular we are using RabbitMQ[15] as a message broker. I natively support the Advanced Message Queuing Protocol (AMQP), which has been introduced in (Vinoski, 2006) as an open standard for message oriented middleware. The protocol adds another advantage to the table by further dividing the message brokering task into exchanges and message queues, whereby the exchange decides which messages will be pushed into which queue. Furthermore the message broker supports the use of Message Queuing Telemetry Transport (MQTT) (Urs Hunkeler, 2008), a lightweight publish/subscribe protocol designed specifically for Wireless Sensor Networks(WSN)

This leads to enhanced flexibility for developers and avoids the need for static implementations. In order to handle scalability issues, which arise in the context of IoT and Smart City data, we propose aggregating the data before passing it through the middleware. In (Frieder Ganz, 2014) we examine how these data abstractions can help to reduce data traffic and ultimately even energy consumption on the sensory level.

[15] http://www.rabbitmq.com/

The messages in our system have three fields. The first field is called ”message types”: it defines the type of message. The second field is called “metadata”: it contains location and time information as well as information about the data source. The third field is called ”data”: it consists of the raw values and identifier. We have defined three types of messages: transform, store and forward. Generally the messages include all the three types. In our case, the subscriber will initially perform some computations on the data, store it for later evaluations and then publish the transformed data. This approach allows different components to work asynchronously on stream data.

Apache Thrift

Apache Thrift allows for cross-language development of software components, by providing a platform for rapidly prototyping Remote Procedure Call (RPC) clients and servers. This –for example- means that a method developed in Java can be called by a function developed in Python. Thrift does that by serializing RPC request/responses between software components.

In principle, it allows for different software providers to create the components in their own language and expose a generic “service” or “services” as a Thrift file to other components, without having to consider implementing an interface to the programming language the other components are using.

Since the communication over the message bus as described in the previous Section will be carried out between different components which are developed on different platforms we need a unified format for (de-)serialising the messages. For that we will use Apache Thrift [3] which natively supports code generation for a large variety of programming languages, including C++, Java and Python. Figure 10 shows how the (de-)serialisation fits into the component communication.

Figure 10 - Communication and (de-)serialisation between two components

Apart from providing cross-platform access, the serialisation will improve the performance of the message passing by compressing the messages into binary format. An Interface Description Language is used to define the cross-platform objects. The preliminary format for our messages is as follows:

```
enum MType {
        transform,
        forward,
        store
}
struct Message {
        1: list<MType> messageTypes
        2: map<string,string> data
        3: map<string,string> metadata
}
```

Table 2 - Depiction of the message format that is used by middleware. Depending on the platform on the producer/consumer, the corresponding code generator has to be used in order to generate the message model

7. Realisation of Semantic Annotation Framework

In this section we present examples of data that can be collected from cities, discuss a use case scenario around this data and put forward some preliminary evaluations to investigate the performance of the semantic annotation framework through the middleware that will be used to deliver messages among some of the components of the system.

7.1. Description of Smart City Data

While smart city data sources offer a variety of data to be processed, we are currently building a linked-data model for semantic annotation of data streams in smart city environments (see Section 4.2 and 5.3), which will also provide a set of smart city data access and processing scenarios. This can help to identify a set of common properties among smart city data that can be used for semantic modeling and description of multi-modal data in smart city applications and services.

Table 3 shows an example of the type of data that can be collected from cities. This data is currently collected from cities and in companies participating in the project[16]. The sampling column relates to the periodicity of the incoming data. Static indicates that the data is never updated and the dataset is used as reference (any updates are manual). Semi-dynamic sampling means that the data is updated periodically, whereas dynamic means continuous updating.

Table 3 - Information that could be potentially retrieved about cities

Data Directory	Owner (Data Publisher)	Data Description	Sampling
Transport	Traffic Authority	Maps of Cities (Roads, Street Names, POIs, subway and bus stations, etc.)	Static
	Municipality	Public Transportation Schedules	Semi-Dynamic
	Traffic Authority	Transport Authority Updates(Roadwork, traffic status, etc.)	Dynamic
Air Quality	Environmental Agency	Particle concentration	Dynamic
Traffic	Traffic Authority	Number of vehicles passing between two points, speed	Dynamic
City Events	Cultural Groups	Entertainment (movie/theatre plays)	Semi-Dynamic
Municipality Services	Municipality	Library Data Waste Collection Data	Dynamic
Citizen Data	Private Individuals	Social Media Information: Tweets, Status updates and	Semi-Dynamic

[16] http://www.ict-citypulse.eu/page/

		blog posts, popular places (“check-ins") Household Energy Consumption	 Semi-Dynamic

7.2. Use Case Scenario

One of the key issues in heterogeneous ecosystem of smart cities is real-time traffic data analysis. Enabling smart cities to efficiently manage traffic data and provide alternative routes will not only help in reducing transportation cost but also pollution that has been caused by traffic congestion. As a use case scenario, we use public traffic data[17] that has been obtained from the city of Aarhus in Denmark. The database consists of traffic data that has been measured among various sensors in different cities providing information regarding the geographical location, timestamp, and traffic intensity such as average speed and vehicle count. The data is taken from 135 sensors and samples every 5 minutes.

While most of the systems constantly create and transmit raw sensor data, we need to be able to express a narrower, more specific workflow such as representation of aggregated and summarised data for individual sensor recordings and the smart city workflow. This will pave the way towards having scalable systems, and reduce memory and computational cost of massive amount of real-time data produced by sensors. For this reason, the semantic representation of the summarised data is as important as annotation of raw stream data. In this Section, we exemplify the use of the proposed information model, describing the outcome of a pair of sensor recordings, and its representation in a road traffic environment.

Figure 11 - A visual representation of geographical coordinates on Google Map for a pair of road traffic sensors provided by city of Aarhus, Denmark

[17] http://www.odaa.dk/dataset/realtids-trafikdata

Figure 11 illustrates a sample location from city of Aarhus in Denmark on Google Maps showing a pair of traffic sensor points, that have been virtualised in our system, and aggregated and semantically annotated based on Stream Annotation Ontology. The sensor points refer to exact geographical coordinates (i.e. latitude, longitude), and linked to resources such as DBPedia[18] and GeoNames[19] that are publicly available as a part of the Linked Open Data cloud. In aggregation process, the streaming data that has been obtained from these sensors, divided into segments and patterns is created for each segment. These patterns represent an aggregation of a set of raw sensor data during a period of time. The pattern construction is performed using the Symbolic Aggregate Approximation (SAX) technique (Jessica Lin, 2003). SAX is used in data mining and time series data for dimensionality reduction and creation of symbolic patterns. It divides a time series data into equal segments and then creates a string representation for each segment. Figure 12 depicts the data captured for average speed via the corresponding sensor points and illustrate SAX patterns created from the raw data.

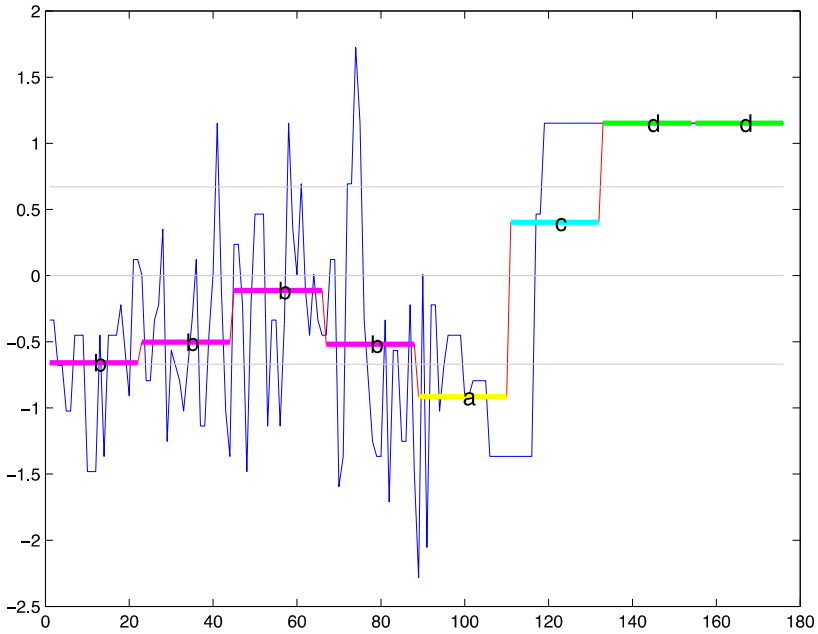


Figure 12 - A real time average speed data obtained from a pair of sensor points is mapped into SAX word, "bbbbacdd", with the segment size of ``8" and alphabet size of ``4" for 176 samples

In Table 4, we describe a set of sensor recordings obtained from the sensor platforms, given in Figure 11,and represent summarised data, shown in Figure 12 as well as temporal entities using the Stream Annotation Ontology. As the proposed semantic model is directly connected to the PROV-O Ontology, we can track the provenance of the information. For instance, in this case the raw data is coming from a public provider, and it has been processed with the stream analysis algorithm SAX, then it has been stored as a stream observation in SAO ontology. This provenance tracking can be used to measure the reliability of the information. With the reliability results, the application developer or the user can make the decision to trust the information or not. We can also annotate QoI concepts, such as *freshness* of the data, taken from the database field *timestamp*; *availability*, taken from the database field *status; granularity*, taken from the database field *VehicleCount*.

[18] http://dbpedia.org/
[19] http://www.geonames.org/ontology/

Table 4 - An excerpt from an RDF data annotated for a set of sensor recordings given in Figure 6 and 7 based on Stream Annotation Ontology

```
@prefix sao: <http://example.com#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix qoi: <http://example.com/QoSQoI.owl#> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .

:government a foaf:Organisation, prov: Agent .
:sefki a foaf:Person, prov:Agent ;
        foaf:givenName "Sefki" ;
        foaf:mbox        <mailto:s.kolozali@surrey.ac.uk>
        prov:actedonBehalfOf :ccsrSurrey ; .
:sensorRec1 a sao:StreamData, ssn:SensorObservation ;
        prov: wasAttributedTo :government .
:sensorRec2 a sao:StreamData, ssn:SensorObservation ;
        prov: wasAttributedTo :government  .
:traffic-sensor-recording-619 a sao:StreamEvent ;
    prov:used [ a sensorRec1; sensorRec2] ;
    sao:time [a tl:Interval;
        tl:at "2014-02-13T08:25:00"^^xsd:dateTime;
        tl:duration "PT15H30M"^^xsd:duration;
    ] ;
    prov:wasAsscoatedWith :sefki ;  .
:freshness-traffic-619 a qoi:Freshness ;
        qoi:value "2014-02-13T08:25:00"^^xsd:dateTime .
:sax_AverageSpeedSample a SymbolicAggregateApproximation;
        rdfs:label "The sax representation of the traffic sensor
        recording obtained from Aarhus City.";
        sao:value "bbbbacdd";
        sao:alphabetsize "4"^^xsd:int ;
        sao:segmentsize "8"^^xsd:int ;
```

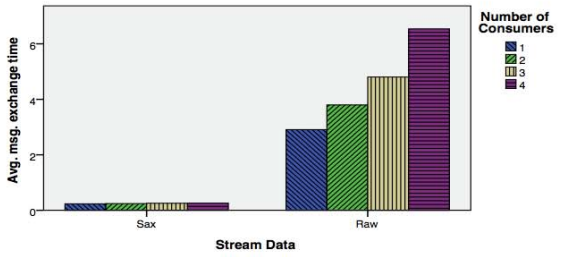
prov:wasGeneratedBy traffic-sensor-recording-619;

qoi:hasQoI freshness-traffic-619 .

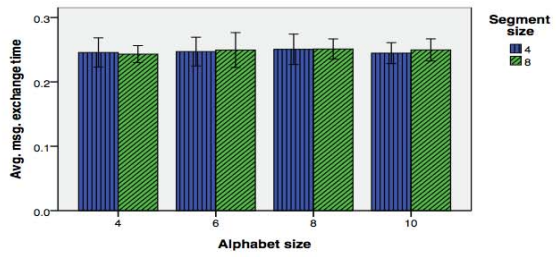
7.3. Evaluation

In this Section, the performance of the stream annotation framework is evaluated using data size and average message exchange time. The evaluations were performed using a RabbitMQ server, that is based on AMQP, on a Personal Computer (PC) running Windows 7 Professional operating system with an Intel Core i7-2670QM 2.2GHz processor and 4GB RAM memory. The aim of this experiment is to send the raw and summarised data as messages through middleware with a different number of consumers to read the messages.

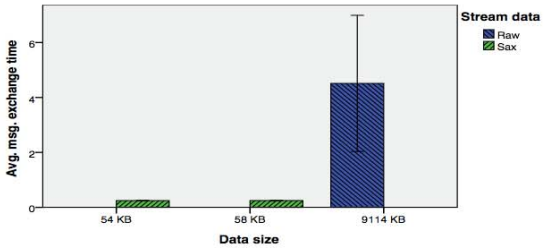
Our experimental dataset consists of two sets of stream samples: (*i*) raw dataset that contains 72240 samples, and (*ii*) summarised dataset that contains 444 samples of the sensor stream data obtained from the road traffic of the city of Aarhus. The overall results of average message delivery time are obtained by averaging the results obtained in the 10 experimental runs. The level of accuracy estimated by these metrics were analysed utilising a two-way Multivariate Analysis of Variance (MANOVA). The independent variables were the data dimension (i.e. raw and sax stream data), and number of consumers (i.e. 1, 2, 3, 4). The dependent variables were the following metrics: the size of data in KiloBytes (KB), and the average message exchange time in second. The Holm-Sidak procedure (Holm, 1989) and a risk α of .05 were used in the MANOVA tests. In addition, we have used the following definitions in our interpretations of the effect sizes: small effect size η^2 ≤.01), medium effect size (.01 ≤ η^2 ≤.06) and large effect size (.06 ≤ η^2 ≤.14). MANOVA level of significance is reported using the F-statistics, F, and probability, p.



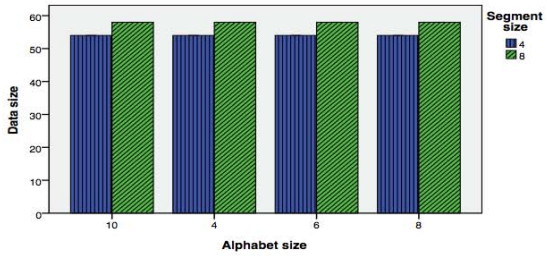
(a) The average message delivery time for each of the number of consumers and data dimensions.



(b) The average message delivery time for each alphabet and segment size of reduced data produced with SAX algorithm.



(c) The average message delivery time and data size for each of the data dimensions.



(d) The data size for each alphabet and segment size of reduced data produced with SAX algorithm.

Figure 13 - Summary of the evaluation results for the raw and the sax stream data based on the average message delivery time and data size. The bars refer to the following metrics: number of consumers for each data dimension, namely SAX and raw (Figure 13a); stream data dimension and data size (Figure 13c); the segment and the alphabet size for each SAX stream data (Figure 13b and Figure 13d)

Performance comparison considering the data size and average message exchange time that were examined using the middleware are reported in Figure 13 For the raw data, the data size and average message exchange time were very high: the data size was 9114 KB and the average time that was spent for message delivery was in range of 2.9s to 6.5s varying based on the number of consumers. On the other hand, there was a rapid decrease in message delivery time and data size for the SAX representation of the data stream. For instance, the data size dropped from its initial high value to 54 and 58 KB, which led to a dramatic decrease to 0.25s for the average message delivery time. Intuitively, the difference in message delivery time for various segment and alphabet sizes were very low. For example, the average time difference for the segment size 4 and 8 were 0.01s, and for the alphabet size it was 0.02s. Similarly, there was a very low difference (i.e. 4 KB) in data size for the segment and alphabet size. The overall average differences between the raw and summarised data were 96.2% and 99.4% for the data size and average message delivery time, respectively.

8. Conclusion and Achievements

In this deliverable, we proposed a stream annotation framework as well as an information model for real time IoT stream using a middleware to support delivery of large volumes of data. To represent the summarisation and reliability of data stream, we introduced a new information model that ensures that summarisation techniques can be interpreted as time-based events, even where further semantic associations are unavailable. The framework contributes solutions to three main issues: heterogeneity, quality of information, and delivery of large volumes of data.

Heterogeneity of data sources provided by sensor services and social media is one of the most prominent issues in Internet of Things. To enable seamless connectivity for the heterogeneous data sources, we developed a component that involves virtualisation and semantic annotation based on semantic web technologies. While there are many approaches to establish interoperability and connections among components, considering the fact that a city environment involves continues and asynchronous structure, we used an asynchronous middleware that is based on publish/subscribe mechanism. Additionally, it also supports translation between different software components to call required functions. Thus various components can implement publisher/subscriber mechanism via the middleware to obtain semantically annotated sensor data.

Additionally, Quality of information is another crucial functionality in the IoT domain and sensor networks and is fundamental for smart city frameworks. To efficiently assess the data streams in real world environment, a direct integration of reliability test was ensured, which keeps semantic descriptions in virtualisation stage up-to-date and sends minimum messages to other components in order to reduce the data traffic. On the other hand, for event detection, which will be described and documented in Deliverable 3.3, we propose the middleware connection that can be used at runtime to send messages to multiple components. Therefore, when an event occurs at runtime, an efficient compensation mechanisms has been supplied as a way to notify various components as well and to minimise disruption time.

Deliverable 3.3 will involve incorporation of a wide set of data stream and utilising a computer network, with real-time road traffic data to investigate the performance of middleware with a large number of consumer subscriptions. This work can help to increase community involvement with a vocabulary to cover representation of data analysis features as well as quality and provenance information that are created by state-of-the-art stream analysis and reliable information processing techniques.

9. Abbreviations

SCF	Smart City Framework
ICT	Information Communication Technologies
IoT	Internet of Things
AMPQ	Advanced Message Queuing Protocol
QoI	Quality of Information
QoS	Quality of Service
O&M	Observation and Measurement
SWE	Sensor Web Enablement
OWL	Ontology Web Language
VoI	Value of Information
OPM	Open Provenance Model
RDF	Resource Description Framework
ODAA	Open Data Aarhus
GSN	Global Sensor Networks
LSM	Linked Sensor Middleware
SAO	Stream Annotation Ontology
MOM	Message Oriented Middleware
WSN	Wireless Sensor Networks
MQTT	Message Queuing Telemetry Transport
RPC	Remote Procedure Call
SAX	Symbolic Aggregate Approximation

References

Urs Hunkeler, H.L.T.A.S.-C., 2008. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In *In Communication Systems Software and Middleware and Workshops*., 2008. IEEE.

Wang, V., Salim, F. & Moskovits, P., 2013. *The definitive guide to HTML5 WebSocket*. Apress.

Wei Wang, S.D.R.T.E.R.K.M., 2012. A comprehensive ontology for knowledge representation in the internet of things. In *In 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*., 2012. IEEE.

Vinoski, S., 2002. Putting the "Web" into Web Services: Web Services Interaction Models, Part2. *IEEE Internet Computing*, pp.90-92.

Vinoski, S., 2006. Advanced message queuing protocol. *Internet Computing*, 10(6), pp.87-89.

Amotz Bar-Noy, G.C.R.G.S.V.K.T.F.L.P.M.M.J.N.a.A.Y., 2011. Quality-of-Information Aware Networking for Tactical Military Networks. In *IEEE International Conference on Pervasive Computing and Communications Workshop*., 2011. IEEE.

Besiki Stvilia, L.G.M.B.T.L.C.S., 2007. A framework for information quality assessment. *Journal of the American Society for Information Science and Technology*, 58(12).

Curry, E., 2004. Message-Oriented Middleware. In Q.H. Madmoud, ed. *Middleware for communications*. Wiley. pp.1-28.

Castillo, P.A. et al., 2011. SOAP vs REST: Comparing a master-slave GA implementation. In *Neural and Evolutionary Computing*., 2011.

Cha, S.-H. & Yun, T., 2012. HTML5 Standards and Open API Mashups for the Web of Things. *Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition*, pp.189-94.

Chatschik Bisdikian, L.M.K.M.B.S.D.J.T.D.V.R.I.Y., 2009. Building principles for a quality of information specification for sensor information. In *12th International Conference on Information Fusion, FUSION '09.* Seattle, 2009. IEEE.

Chatschik Bisdikian, L.M.K.M.B.S., 2013. On the Quality and Value of Information in Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 9(4).

Cory A. Henson, H.N.A.P.S.K.T.a.R.B., 2009. An ontological representation of time series observations on the semantic sensor web. In *1st International Workshop on the Semantic Sensor Web (SemSensWeb 2009)*., 2009.

Cory A. Henson, J.K.P.A.P.S.a.K.T., 2009. Semsos: Semantic sensor observation service. In *International Symposium on Collaborative Technologies and Systems*., 2009. IEEE.

De, S., Elsaleh, T., Barnaghi, P. & Meissner, S., 2012. An internet of things platform for real-world and digital objects. In *Scalable Computing: Practice and Experience*., 2012.

Dobson, G., Russell, L. & Sommerville, I., 2005. QoSOnt: a QoS ontology for service-centric systems. In *Software Engineering and Advanced Applications, 2005, 31st EUROMICRO Conference on*. Porto, 2005. IEEE.

Fernandes, J.F., Lopes, I.C., Rodrigues, J.J.P.C. & Ullah, S., 2013. Performance Evaluation of RESTful Web SErvices and AMQP Protocol. *Proceedings of the 5th IEEE International Conference on Ubiquitous and Future Networks (ICUFN)*, pp.810-15.

Fielding, R.T., 2000. Architectural styles and the design of network-based software archictures. *PhD Thesis*.

Filipponi, L. et al., 2010. Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. In *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*., 2010. IEEE.

Frecon, E., 2013. Web(-like) Protocol for the Internet of Things. *Proceedings of the 20th Annual Tcl Conference*, pp.23-27.

Frieder Ganz, P.B.F.C., 2014. Multi-resolution data communication in wireless sensor networks. In *In IEEE World Forum on Internet of Things (WF-IoT)*., 2014. IEEE.

Gudgin, M. et al., 2007. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. [Online] Available at: http://www.w3.org/TR/soap/ [Accessed September 2014].

Guinard, D., Ion, I. & Mayer, S., 2011. In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers' Perspective. *Proceedings of the 8th International ICST Conference*, pp.326-37.

Grishman, R., Huttunen, S. & Yangarber, R., 2002. Real-time event extraction for infectious disease outbreaks. In *Proceedings of the second international conference on Human Language Technology Research*., 2002. Morgan Kaufmann Publishers Inc.

Irfan Rafique, P.L.M.Q.A.Z., 2012. Information Quality Evaluation Framework: Extending ISO 25012 Data Quality Model. *World Academy of Science, Engineering and Technology*, 6, pp.503--508.

Hämäläinen, H., 2012. *HTML5: WebSockets*. Aalto University, Department of Media Technology.

Holm, S., 1989. A simple sequentially rejective multiple test procedure. In *Scandinavian Journal of Statistics*., 1989.

Jessica Lin, E.K.S.L.B.C., 2003. A symbolic representation of time series, with implications for streaming algorithms. In *proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.* San Diego, CA, USA, 2003. ACM Press.

Kwak, H., Lee, C., Park, H. & Moon, S., 2010. What is Twitter, a Social Network or a News Media?. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10). ACM*. New York, NY, USA, 2010.

Lu, J. et al., 2010. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*., 2010. ACM.

Lampesberger, H., 2014. Technologies for Web and Cloud Service Interaction: A Survey. *IEEE International Conference on Service Oriented Computing & Applications*.

Naughton, M., Kushmerick, N. & Carthy, J., 2006. Event extraction from heterogeneous news sources. In *Proceedings of the AAAI Workshop Event Extraction and Synthesis*., 2006.

Natale, D., 2011. Complexity and data quality. In *Conference CHItaly2011*. Alghero, Italy, 2011.

Mandyam, G.D. & Ehsan, N., 2012. HTML5 Connectivity Methods and Mobile Power Consumption.

Michael Compton, P.B.L.B.R.G.-C.O.C.S.C.J.G.M.H.C.H.A.H.e.a., 2012. The ssn ontology of the w3c semantic sensor network incubatorgroup. *Web Semantics: Science, Services and Agents on the World Wide Web*, pp.25-32.

Mike Botts, G.P.C.R.a.J.D., 2008. Sensor web enablement: overview and high level architecture., 2008. Springer.

Payam Barnaghi, W.W.L.D.a.C.W., 2013. A linked-data model for semantic sensor streams. In *International Conference on Internet of Things (iThings2013)*. Beijing, 2013. IEEE.

Pimentel, V. & Nickerson, B.G., 2012. Communicating and Displaying Real-Time Data with WebSocket. *IEEE Computer Society*, pp.45-53.

Suparna De, T.E.P.B.a.S.M., 2012. An internet of things platform for real-world and digital objects. In *Scalable Computing: Practice and Experience*., 2012.

Saurabh Ganeriwal, L.K.B.M.B.S., 2004. Reputation-based framework for high integrity sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks*., 2004. ACM.

Sefki Kolozali, M.B.-E.D.P.F.G.P.B., 2014. A Knowledge-based Approach for Real-Time IoT Data Stream Annotation and Processing. In *IEEE International Conference on Internet of Things 2014*. Taipei, 2014. IEEE.

Sheth, A., 2009. Citizen sensing, social signals, and enriching human experience. In *Internet Computing*., 2009. IEEE.

Richard Y. Wang, D.M.S., 1996. Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems*, 12(4), pp.5-33.

Ritter, A., Etzioni, O., Clark, S. & others, 2012. Open domain event extraction from twitter.. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.*., 2012. ACM.

Tanev, H., Piskorski, J. & Atkinson, M., 2008. Real-time news event extraction for global crisis monitoring. In *Natural Language and Information Systems*., 2008. Springer.