Quality of Observation within Sensor Web systems: from theory to practice

Qualité des Observations pour les systèmes Sensor Webs : de la théorie à la pratique

Antoine Auger

March 6, 2018

TéSA seminar
1. Context and Motivation

2. A generic framework for QASWS

3. A functional QASWS prototype: the iQAS platform

4. A deployment scenario: QoO for challenging Internets

5. Conclusions and Perspectives
Context and Motivation
A World of sensors

Temperature, humidity, wind speed, water salinity, GPS, accelerometers, etc.

How to discover, access and retrieve observations from all these sensors in an unified manner?
A World of sensors

Temperature, humidity, wind speed, water salinity, GPS, accelerometers, etc.

How to discover, access and retrieve observations from all these sensors in an unified manner?
⇒ Short answer: by using sensor middlewares, a.k.a **Sensor Webs**!
What was a Sensor Web?

**NASA JPL (1999)**

Developmental collections of sensor pods that could be scattered over land or water areas or other regions of interest to gather data on spatial and temporal patterns of relatively slowly changing physical, chemical, or biological phenomena in those regions.

Back then, consumers:

- had basic needs in terms of **Quality of Service (QoS)**
- were mainly interested in **physical** sensors
- had access to **dedicated** sensors
Motivation ► Required Background

- New paradigms have emerged:

  - '80s Wireless Sensor Networks
  - 1997 Sensor Webs
  - 2000 Internet of Things (IoT)
  - 2007 Semantic Web / Web of Things (WoT)
  - 2007 Virtual sensors
  - 2010 IoT platforms / Cloud of Things
  - 2014 Internet of Everything (IoE)

- Sensors, consumers and uses are changing
- 50+ billion Things will be connected to the Internet by 2020
Motivation ▶ Required Background

- New paradigms have emerged:
  - '80s: Wireless Sensor Networks
  - 1997: Internet of Things (IoT)
  - 2000: Semantic Web / Web of Things (WoT)
  - 2007: Virtual sensors
  - 2010: IoT platforms / Cloud of Things
  - 2014: Internet of Everything (IoE)

- Sensors, consumers and uses are changing
- 50+ billion Things will be connected to the Internet by 2020

⇒ Sensors Webs need to evolve as well to cope with new issues
Motivation ▶ Required Background

What *is* a Sensor Web? (update)

OGC SWE (2011)
Sensor Web is to sensor resources what the WWW is to general information sources - an infrastructure allowing users to easily share their sensor resources in a well-defined way. [Brö+11]

Guest Editors for a Sensor Web journal (2016)
Sensor Web can be defined as the paradigm that enables the integration of sensors/sensor networks and Web-based platforms.
Observations should be of “good quality” for each consumer

- Quality of Service (QoS) but…
- Quality of Experience (QoE) but…

Still insufficient to characterize all consumer needs!
Observations should be of “good quality” for each consumer

- Quality of Service (QoS) but…
- Quality of Experience (QoE) but…

Still insufficient to characterize all consumer needs!

**Quality of Information (QoI)**

QoI is the collective effect of information characteristics (or attributes) that determine the degree by which the information is (or perceived to be) fit-to-use for a purpose. [Bis+09]

Within sensor-based networks, information ≡ observations

⇒ **Quality of Observation (QoO)** to remain coherent
New challenges for modern Sensor Webs:

**Integration**
How to bridge the gap between sensor capabilities and consumer needs while reducing the complexity of end applications?
New challenges for modern Sensor Webs:

**Integration**
How to bridge the gap between sensor capabilities and consumer needs while reducing the complexity of end applications?

**Quality of Observation (QoO)**
How to provide fit-for-use observations in a consumer-specific fashion?
New challenges for modern Sensor Webs:

<table>
<thead>
<tr>
<th>Integration</th>
<th>How to bridge the gap between sensor capabilities and consumer needs while reducing the complexity of end applications?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality of Observation (QoO)</td>
<td>How to provide fit-for-use observations in a consumer-specific fashion?</td>
</tr>
<tr>
<td>System adaptation</td>
<td>How to take into account context changeability over time?</td>
</tr>
</tbody>
</table>
We envision a new generation of Sensor Webs:

**QoO-aware A(daptive S)ensor W(eb) S(systems) (QASWS)**

We proposed two contributions:

1. A generic framework for QASWS
2. A functional QASWS prototype
A generic framework for QASWS
Framework for researchers and developers who may want to conceive their own QASWS:

- Defined from **ISO/IEC/IEEE 42010 standard** (terminology and concepts)
- Platform-Independent Model (PIM), no technology or software is specified
- Should be **instantiated** to a specific use case (see iQAS)
Our framework is composed of:

1. A Reference **Model** (composed of several sub-models)
2. A Reference **Architecture** (composed of several views)
3. Reference **Guidelines**
Generic framework for QASWS ➤ Reference Model

Functional model ➤ Adaptation model ➤ Domain model ➤ Observation model

Observation consumers

Semantic layer
\( f_{\text{sem}}(\ldots) \)

Information layer
\( f_{\text{charac}}(\ldots) \)

Raw Data layer
\( f_{\text{digit}}(\ldots) \)

Observation producers

Management & Adaptation layer

\( f_{\text{sem}}(\text{Information, OntoModel}) = \text{Knowledge} \)

\( f_{\text{charac}}(\text{Raw Data, Context}) = \text{Information} \)

\( f_{\text{digit}}(\text{Sensor outputs}) = \text{Raw Data} \)
Generic framework for QASWS ➤ Reference Model

- **Functional model**
- **Adaptation model**
- **Domain model**
- **Observation model**

**Filtering**
- Observation stream
- Time interval
- Observations

**QoO Pipeline**
- in QoO mech.
- #2
- out QoO mech.
- #n-1
- QoO mech.
- #3

**Parameter(s) required by the mechanism**
- Threshold(s)
- RD, I, K

Best applied on
Generic framework for QASWS ➤ Reference Model

- **Functional model**
- **Adaptation model**
- **Domain model**
- **Observation model**

- Observations
- Time interval
- Observation stream
- Filtering
- RD, I, K
- Threshold(s)
- Parameter(s) required by the mechanism
- QoO Pipeline
- in
- QoO mech. #2
- QoO mech. #3
- QoO mech. #n-1
- out
Use of **ontologies** to make the link between all concepts introduced by the different sub-models
Ontology

A formal explicit description of concepts, properties and restrictions in a domain of discourse.
Why should we use ontologies?

- To share knowledge among people or software agents
- To enable reuse of domain knowledge
- To separate domain from operational knowledge
- To analyze domain knowledge

[Source: Protégé]
By reusing existing standards (W3C SSN), we propose the **QoOnto ontology** to describe sensors, mechanisms and pipelines:
Generic framework for QASWS ➤ Reference Architecture

- **Functional model**
- **Adaptation model**
- **Domain model**
- **Observation model**

**Sensor layer**
- **Semantic layer**
- **Information layer**
- **Raw Data layer**
- **Application layer**

- **Ontology base model**
- **Context**

- **Raw Data**
  - **Network QoS guarantees**

- **Sensor outputs**

**Semantic layer**
- **Knowledge**
  - **Observation annotation**
  - **Sensor description**

**Information layer**
- **Context annotation**
- **QoI computation**

**Application layer**
- **SLAs**
- **Feedback**

**Adaptation API**

- **AM #1**
- **AM #2**
- **AM #3**

- (*) SANETs only

- **QoS guarantees**

(*) SANETs only
MAPE-K loop for enabling resource-based (sensors, pipelines) and QoO-based adaptation

✓ Specific adaptation per sensor/request/consumer
✓ Lazy adaptation strategy
A functional QASWS prototype: the iQAS platform
The iQAS platform ➤ Instantiation process

- **System of Interest**
  - General Requirements
    - Reference Model
    - Reference Architecture
    - Reference Guidelines
    - Generic Framework for QASWS

- **Concerns**
  - Use Cases
    - Specific Requirements

- **Stakeholders**
  - Implementation choices

- **Observation Storage**
  - Monitor QoO level if QoO level not satisfied:
    - adapt_qoo_level
  - If QoO level not reachable:
    - cancel_request

- **Define QoO Pipelines**
  - Find a suitable QoO Pipelines
  - Submit observation request if QoO constraints exist:
    - enforce_sla
    - monitor_qoo_level

- **Adapt QoO level**
  - Mediation
  - «includes»
  - «includes»
  - «includes»

- **Monitor QoO level**
  - Subscribe to specific observations
  - Cancel observation request
  - Retrieve info about the iQAS platform
  - Update QoOnto ontology

- **Publish observations**
  - Check available sensors
  - Adapt QoO level
  - Enforce SLA
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»

- **Submission**
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»

- **Implementation choices**
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»

- **VirtualApp**
  - Application
  - «includes»
  - «includes»
  - «includes»

- **VirtualSensor**
  - Sensor
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
  - «includes»
The iQAS platform ➤ Specific requirements

- i-F1 Observation requests (SLAs)
- i-F2 Observation retrieval
- i-F3 Feedback provided by the platform
- i-F4 Sensor “plug-and-play”
- i-F5 Definition of new QoO Pipelines
- i-F6 Definition of new QoO attributes
- i-F7 Characterization of QoO Pipelines
- i-NF1 Adaptability
- i-NF2 Transparency
- i-NF3 Scalability
- i-NF4 Extensibility
- i-NF5 Interoperability
The iQAS platform ▶ Implementation choices

- Apache Jena
- mongoDB
- akka

Component

Reactive Streams

Actor model
The iQAS platform ♦ High-level architecture

- **Physical, Logical or Virtual sensors**
  - Temperature, visibility, humidity, etc.

- **Ingest pipeline #1**
  - MAPE-K loop
  - QoO report
  - Obs. rate report

- **Output pipeline #1**
  - appli1_58d39df

- **Ingest pipeline #2**
  - Heal Pipeline #3
  - Output pipeline #3
  - appli2_46d69df

- **Ingest pipeline #3**
  - healer_pipeline #3
  - Output pipeline #4
  - appli3_95d39df

- **Output pipeline #2**
  - appli4_4ad39df

- **iQAS storage**
  - Ontology triple store

- **GUI**
  - API

---

22
The iQAS platform ➤ Graphical User Interface
Software and its configuration may impact QoO...

⇒ Due to our implementation choices, iQAS performances are greatly impacted by **Apache Kafka** and its configuration (broker and clients)

Use of Kafka without parallelism or replication (obs. streams)
A deployment scenario: QoO for challenging Internets
Delay-Tolerant Network (DTN)
A network that may lack continuous network connectivity.

Opportunistic Networks further consider **human social characteristics** to perform routing and data sharing.
Architecture can **fail!**

⇒ DTNs and OppNets as an alternative for challenging Internets
Our goals:

- Imagine a deployment scenario where QoO is of interest
- Study a QoO metric that can be impacted by both network QoS and iQAS processing time
- Show that network QoS and QoO are closely tied

Our means:

- We investigated it from an "Opportunistic Networking" perspective
- We focused on the "observation freshness" attribute
- We reused the HINT emulator from the DGAME project
Deployment scenario ► Experimental setup

- Monitoring & Tuning
- Core Emulator
- Message Broker
- Database

HINT network emulator

Real world

Virtual Sensor Container (VSC)

appli1
Deployment scenario ➤ Experimental setup

Gateway
Virtual Sensor Container (VSC)

HINT network emulator

Observation freshness measurement

appli1

QAS
Deployment scenario ➤ Experimental setup

- Core Emulator
- Message Broker
- Monitoring & Tuning
- Database
- Real world
- HINT emulator
- Gateway
- Virtual Sensor Container (VSC)
- HINT viewpoint
- iQAS viewpoint
- appli1
Deployment scenario ➔ Experimental results

- Obs. freshness from HINT viewpoint
- Obs. freshness from iQAS viewpoint
• iQAS processing time is negligible compared to recollection time

• Some QoO constraints may be partially translated into network QoS constraints

• Network QoS guarantees should be ensured first
Conclusions and Perspectives
Conclusions

- The **Sensor Web** paradigm is in constant evolution
- **QoO** is one of the most important challenges that new Sensor Webs should cope with

- We proposed 2 contributions regarding the design and development of QoO-aware Adaptive Sensor Web Systems (QASWS)

- QoO is a complex but critical notion for **data-centric** systems
- QoO may be **impacted** by software and its configuration (Warning!)
- Network QoS and QoO may be used together to meet **consumer needs**
Keep studying the relationships between the different quality dimensions

How to describe the capabilities of a virtual sensor?

Improve and promote the iQAS platform (internship proposal at ISAE)

How can Sensor Webs take advantage of other paradigms regarding QoO? (Edge Computing, Blockchain, ML, etc.)
Publications

Thank you for your attention.

Question time!

We evaluated the iQAS footprint by defining **Key Primary Indicators (KPIs)**:

- iQAS overhead
- iQAS throughput
- iQAS response time

3-step methodology:

1. Identification of relevant parameters for Kafka configuration
2. Parameter tuning
3. Experiments
The iQAS platform ➤ Evaluation

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_config</td>
<td>Moderate batching, low limit for polling records, no linger time</td>
</tr>
<tr>
<td>highthroughput_config</td>
<td>Extreme batching, high limit for polling records, linger time</td>
</tr>
</tbody>
</table>
The iQAS platform  ➤  Evaluation

**iQAS overhead** (observation freshness)

- **initial_config**: 4 ms for E2E delay
- **highthroughput_config**: 800 ms for E2E delay
The iQAS platform ► Evaluation

**iQAS throughput** (observation rate)

- **Initial config**
  - 10,000 obs. / second

- **High throughput config**
  - 33,000 obs. / second

⇒ Tradeoffs between observation size, latency and throughput (see Queuing Theory)
For similar iQAS requests, only the first one is “costly”