Stream Reasoning for the Internet of Things: Challenges and Gap Analysis

Xiang Su
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
xiang.su@ee.oulu.fi

Ekaterina Gilman
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
ekaterina.gilman@ee.oulu.fi

Peter Wetz
Information & Software Engineering Group
TU Wien
Vienna, Austria
peter.wetz@tuwien.ac.at

Jukka Riekki
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
jukka.riekki@ee.oulu.fi

Yifei Zuo
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
yifei.zuo@ee.oulu.fi

Teemu Leppänen
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
teemu.leppanen@ee.oulu.fi

ABSTRACT

The Internet of Things (IoT) is not only about interconnecting embedded devices to the Internet, but also about providing knowledge on such devices and what they sense from the physical world. One focus of IoT is put on extracting actionable knowledge and providing value-added services by means of reasoning techniques. Stream reasoning techniques offer a promising solution for processing dynamic, heterogeneous, and volume data for IoT. In this article, we identify the challenges for utilizing stream reasoners from the IoT point of view, review the landscape of stream reasoning techniques, and examine their capabilities to meet the challenges of IoT. Moreover, we present an experimental IoT system implementing stream reasoning and perform a gap analysis to evaluate stream reasoners. Finally, based on the analysis, we suggest several recommendations for future development of stream reasoners in order to overcome the identified gaps.

CCS Concepts

•**Information systems** → *Information systems applications;* •**Theory of computation** → *Semantics and reasoning;*

Keywords

stream reasoning; Semantic Web; dynamics; gap analysis

1. INTRODUCTION

The Internet of Things (IoT) enables connecting physical objects and devices, also called "things", to the Internet. However, IoT is not only useful for collecting and publishing data from things, but rather about providing knowledge of such things and what they can sense from the physical environment and personal or social activities. New data is continuously generated from a huge number of sources in IoT, therefore, knowledge is frequently changed and updated.

It is critical to deduce timely, sufficiently accurate, and reliable knowledge from IoT systems to take actions. Knowledge bases and reasoning techniques are promising candidates to manage the knowledge for IoT systems. For example, IoT systems can analyze data about traffic, weather, or pollution for better traffic management and recommendations. However, traditional approaches developed by the Semantic Web community are designed for data updated at a low frequency, such as switching on/off devices or a person entering/leaving a building. Most reasoning techniques are capable of handling computationally intense inference methods at run time. Such solutions, however, do not scale for IoT systems, because of their heterogeneous, ubiquitous and resource-constrained nature [21].

Stream reasoning is defined as a problem of generating a stream of conclusions by means of reasoning over terminological or assertional axioms [31]. Stream reasoning combines reasoning and stream processing techniques. Such a combination enables handling of dynamic and heterogeneous data continuously produced from a large amount of sources, processing several streams on-the-fly, and implementing real-time services. In general, stream reasoning meets the requirements of processing dynamic, heterogeneous, and scalable data for IoT.

In our earlier work, we studied how Semantic Web techniques can be embedded in the data generated by IoT nodes, and specifically, how the benefits of semantics can be utilized without sacrificing the efficiency of IoT in terms of energy consumption and latency [29, 20]. Delivering IoT data with different representations from IoT nodes to Semantic Web reasoners and RDF triple stores were evaluated. We found the average reasoning latency makes up for 45% - 65% of the overall processing time, which is much higher than transmission and other data processing time [20]. If reasoning latency could be reduced by lightweight solutions, poten-

WIMS '16 June 13–15, 2016, Nîmes, France

DOI: 10.475/123_4

tial IoT systems would be able to provide better real-time services. In this paper, we identify challenges of IoT for stream reasoning, survey state-of-the-art stream reasoning techniques and systems, and examine their capabilities to meet the challenges proposed by IoT. Moreover, we present our preliminary stream reasoning experiment with IoT data streams, and then perform a gap analysis to evaluate selected stream reasoners. Based on the analysis, we propose recommendations for future stream reasoner development in order to fill these gaps for IoT applications.

The remainder of this article is organized as follows: Section 2 examines challenges of IoT for stream reasoning. Section 3 presents the review of a representative list of stream reasoners and discusses their capabilities to meet the challenges of IoT. In Section 4, we present our preliminary experiment to demonstrate the feasibility of utilizing stream reasoners for IoT application. We perform a gap analysis and enumerate recommendations for filling the gap in Section 5, and finally conclude the paper with discussing the future plan in Section 6.

2. IOT CHALLENGES FOR STREAM REASONING

In this section, we extract some general challenges from typical IoT systems, such as smart city, intelligent transportation, health care, and home automation. These challenges derive key requirements, which will be examined in different stream reasoning systems.

2.1 Variety, Velocity, and Volume of Data Streams (C1)

The amount of information collected from IoT devices is enormous, and information is continuously generated at high sampling rates. Additionally, integration of multiple sources further amplifies this challenge. Information is represented in different formats and various models. For example, it is a challenge to directly utilize low level data provided by sensors with a well-defined ontological knowledge model. Research is needed to extract higher level information, guide the understanding of complex situations, and finally deduce new knowledge. All these tasks introduce novel challenges to IoT applications.

IoT connects a large amount of multi-vendor, multi-platform, heterogeneous devices and services. While the main challenges of IoT data communication at low level are being solved, handling heterogeneity, velocity, and volume of data at the semantic level is still a key issue. Moreover, semantic reasoning requires the integration of data coming from IoT data streams with background knowledge. The background knowledge describes the application domain, which is critical to implement richer functionality. For instance, traffic related data, such as the position of cars, their speed, information from sensors installed on the road, and information from third party services, need to be combined with domain ontologies and user-defined rules to deduce driving situations. This combination is challenging, because retrieving application-specific knowledge from a large background knowledge base for real-time stream reasoning can be particularly complex and resource-expensive. Moreover, when the background knowledge is updated, propagation of updated knowledge in a system with different reasoning applications introduces additional challenges.

Recent efforts in the area of Semantic Web, such as Semantic Sensor Network Ontology[1], have been introduced to enable semantic interoperability. However, reasoning for heterogeneous big IoT data streams, as a key enabler of producing knowledge, still needs more research.

2.2 Efficiency (C2)

Efficiency is considered with respect to (1) low latency reasoning, i.e. timely generation of new knowledge before it becomes outdated, and (2) reasoning on resource-constrained devices and systems, i.e. delivery of reliable solutions considering computing, storage, communication, and energy limitations of IoT devices and protocols and provide global, scalable, and reliable solutions. Reasoning on resource-constrained devices enables processing data closer to data sources and users, which reduces communication costs, provides better data availability and quality, supports real-time response, and preserves privacy.

To cope with big IoT data, most IoT systems demand high processing throughput. It is common that the incoming data rate is higher than the rate considered by traditional reasoners for updating their inference results [33]. Time plays an important role for real time IoT systems. Time models, data aggregation strategies, as well as efficient and distributed reasoning algorithms are considered important to achieve low latency reasoning.

IoT demands reasoning solutions able to cope with constraints posed by available resources. Standardized semantic technologies are mainly designed for Web applications. Hence, they offer expressive representations and complex reasoning techniques, which require a considerable amount of resources unavailable in IoT systems. This conflict represents a novel challenge.

2.3 Semantic Expressive Power (C3)

IoT nodes produce low level data. It is a challenge to extract higher level knowledge, which would facilitate the understanding of complex situations and allow to address them properly. Suitable knowledge models and processing approaches are required to make it possible to deduce new knowledge for IoT applications in an efficient manner. The utilization of knowledge models, such as RDF and OWL ontologies, would allow semantic integration of information from multiple sources. Combining RDF with other ontology languages enables the inference of new knowledge from already existing data. RDF has been widely accepted as the data model for semantically representing static information. However, it needs to be extended to consider data that changes frequently over time, e.g. integrate the possibility to annotate the data items with the time of occurrence and validity information. Finally, extracting high level knowledge often requires transformation of various data formats, which differ in their expressive power.

2.4 Robustness (C4)

IoT data are often unreliable, incomplete, arrive in wrong order or even get lost. This means in practice that the input of stream reasoning systems and the availability of data at different resources are not always deterministic. Hence, additional challenges are introduced for processing solutions utilizing IoT data. In addition, different IoT applications could present different requirements for the same data streams.

[1]http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

Another challenge is inability to guarantee complete reasoning results. Because a stream is observed only through a window of finite (and often pre-determined) size, it is possible that the processing component does not consider the entire input and thus the reasoning result might be incomplete. Therefore, suitable uncertainty management models are required for stream reasoning.

3. A REVIEW OF STREAM REASONING SOLUTIONS FROM IOT PERSPECTIVE

In this section, we introduce stream reasoning technologies in general and then present a review of semantic stream reasoning solutions. Moreover, we examine these stream reasoning solutions from an IoT perspective.

3.1 Stream Reasoning for IoT

Stream reasoning is one approach for querying and reasoning over continuous distributed data streams. With a streaming query engine, simultaneous queries can be passed to a reasoner as an input. Thus, new knowledge can be inferred and RDF graphs can be updated on-the-fly. By continuously querying data stream that are observed trough time windows, the engine gives advantage on query processing performance and data storage overhead. These features could facilitate distributed reasoning from continuous IoT data streams. Stream reasoning techniques offer a time-based data model where data items can be annotated with timestamps, either occurrence time or validity time period. Defining a suitable time processing model is probably one of the main contributions of stream reasoning technologies to IoT.

Figure 1 presents a general diagram of utilizing stream reasoning techniques in typical IoT applications. Applications and IoT devices generate IoT data streams, which are unbounded sequences of time-varying data elements [11]. Background knowledge, such as ontologies, constitute knowledge sources essential to deduce new knowledge. Query and reasoning components are core components of stream reasoners, and they can be combined into a single unifying processing component. The query model represents the interface for gathering information from the system, while reasoning is utilized to enrich query results.

Stream reasoning engines typically make use of extended SPARQL-based query languages over continuous data streams. Further, we review a selection of currently available stream reasoners [21] and examine their capabilities to meet the challenges of IoT.

3.2 A review of Semantic Stream Reasoning Systems

C-SPARQL (Continuous SPARQL) [4, 5] is one of the first contributions in the area of stream reasoning. It is a language for continuous queries over streams of RDF data with the goal of bridging the gap between stream processing systems and SPARQL. C-SPARQL extends SPARQL for querying RDF streams. An RDF stream is defined as an ordered sequence of pairs, where each pair is constituted by an RDF triple and its timestamp t:

($<\text{Subject}_i, \text{Predicate}_i, \text{Object}_i > t_i$)

C-SPARQL supports (1) timestamped RDF triples, (2) continuous queries over RDF streams, and (3) integration

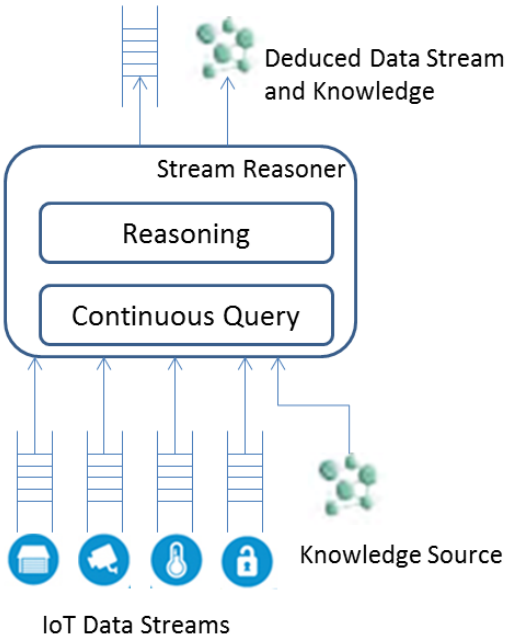

Figure 1: A General Diagram of Stream Reasoning in an IoT system.

and processing both background data and streams. The results of C-SPARQL queries are continuously updated as new streaming data flows into the system.

In C-SPARQL, a time window always selects the most recent items from a stream, and can be either count-based (a fixed number of items) or time-based (a variable number of items occurring in a fixed time interval). Each query has a pre-defined and fixed evaluation frequency, which can be specified by the consumers. C-SPARQL does not include any explicit operator to capture temporal patterns over input elements. Although it allows defining queries that consider time by directly accessing the timestamp of each information item, this is possible only inside the boundaries of windows.

IMaRS (Incremental Materialization for RDF Streams) is developed on top of C-SPARQL and it focuses on materialization [9]. The authors present an algorithm for maintaining the materialization of ontological entailment in the presence of streaming information. Materialization is making all possible conclusions from a given input explicit, meaning no further reasoning is needed when the user queries the knowledge base. IMaRS utilizes an incremental reasoning approach and the specific data and processing models of C-SPARQL to compute the expiration time of streaming RDF triples based on the windows of deployed queries.

By annotating each RDF triple with its expiration time and utilizing a hash table to index triples by their expiration time, IMaRS reduces the amount of computation that needs to be performed to update the results of reasoning. However, IMaRS relies on the strong assumption that the expiration time of each triple can be pre-computed, which limits its applicability.

TrOWL. TrOWL [25, 14] is another notable reasoner for incremental reasoning. Compared with IMaRS, TrOWL

presents two distinctive features: (1) it supports more complex ontology language, covering the expressive power of OWL2 DL; (2) it utilizes syntactic approximation to reduce reasoning complexity. Syntactic approximation ensures that all derived knowledge is correct, i.e., it ensures soundness, but not completeness of reasoning.

TrOWL keeps traceability relations between deriving facts and derived facts. It provides experimental evidence of the benefits of their approach while modifying increasing portions of the knowledge base. Moreover, TrOWL is designed for more dynamic knowledge. The design of TrOWL considers updates from 20% to 80% of the ontology and the authors measure a processing time that varies from 20% to 70% with respect to the naive approach of recomputing all derivations. In comparison to TrOWL, IMaRS focuses on much smaller updates,considering less than 20% of the ontology. IMaRS presents large advantages when updating less than 10% of the ontology, but TrOWL may benefit for highly dynamic IoT applications.

Deductive and inductive stream reasoning is a noteworthy extension to the C-SPARQL model [3]. This work proposes to combine inductive with deductive reasoning to increase result accuracy. The authors apply this technique in a real world scenario deriving knowledge from social media analysis and showing the accuracy of the reasoning algorithm in this context. Their performance evaluation results show that the proposed approach exhibits low processing delays which remarkably outperforms SPARQL query engines. Inductive stream reasoning deals with mining of large portions of data and applying statistical and machine learning techniques to extract new knowledge. Inductive stream reasoning supports heterogeneous data integration and possible materialization. However, the processing efficiency still needs to be improved.

C-SPARQL on S4 implements partial RDFS reasoning and part of the C-SPARQL query language on the S4 streaming platform [13]. This is an important effort to improve scalability and efficiency, especially useful in IoT related scenarios. C-SPARQL on S4 enables splitting the processing load over multiple machines. Operators for triple selection, filtering, join, projection, and aggregation are implemented. However, it is noted that only time-based windows can be used, while count-based windows are not available.

CQELS [18] is another comparable system with C-SPARQL. CQELS and its accompanying query engine are designed for combining static and streaming linked data. Similar to C-SPARQL, CQELS adopts the processing model of DSMSs, providing windowing and relational operators together with ad-hoc operators for generating new streams from the computed results. Different from C-SPARQL, CQELS offers a processing model in which query evaluation is not periodic, but triggered by the arrival of new triples. The distinctive difference of CQELS and C-SPARQL lies in their processing engines. CQELS strictly integrates the evaluation of background and streaming data, without delegating them to external components. This makes it possible to apply query rewriting techniques and optimizations well studied in the field of relational databases.

Streaming SPARQL. Streaming SPARQL is another extension of SPARQL for processing RDF streams [7]. The authors mainly focus on the specification of the semantics of Streaming SPARQL using temporal relational algebra and provide an algorithm to automatically transform SPARQL queries into the new extended algebra. The main contributions of this work are theoretical. Moreover, the authors present a small test scenario based on transitive properties.

Streaming Knowledge Bases is built on top of the TelegraphCQ DSMS and provides reasoning using a subset of RDFS and OWL over streaming RDF triples [34]. This approach allows to pre-compute and store inferences to reduce the overall computational effort, and consequently, the delay, during the evaluation of the queries.

Sparkwave [15] is a system designed for high performance and on-the-fly reasoning over RDF data streams. It trades complexity for performance. However, Sparkwave poses several limitations to the size of the background knowledge, which has to fit into the main memory of a single machine. Moreover, it operates with a pre-loaded RDF schema and provides limited reasoning functionality. Sparkwave implements a variant of the RETE algorithm, in which a preprocessing phase is used to materialize derived information before performing pattern matching. The portions of data considered during the processing are isolated through traditional windowing mechanisms, similar to those used by DSMSs and C-SPARQL.

EP-SPARQL [1] [2] is another language designed for semantic event processing and reasoning. Similar to other solutions, EP-SPARQL also provides windowing operators for isolating portions of the streams. However, EP-SPARQL is a solution that inherits the language constructs and processing model of Complex Event Processing (CEP) systems. EP-SPARQL focuses more on detection of RDF triples in a specific temporal order. Another important difference of EPSPARQL with respect to other languages such as C-SPARQL, is in the data model and consists in the way time is associated to RDF triples. While CSPARQL associates one single timestamp to each triple.

STARQL [24] [17] is designed as an expressive and flexible stream query framework that offers the possibility to embed different temporal description logics as filter query languages over ontologies. STARQL can embed various query languages that can be combined with more expressive DL ontology languages. STARQL's flexibility enables implementing orthogonal approaches such as ABox modularization, a technique that supports accessing big data over very expressive ontologies. This framework utilizes a First Order Logic (FOL) fragment for temporal reasoning over ABox sequences constructed within the query. STARQL sets up sequencing operators at every time point, therefore, a finite sequence of ABoxes on which temporal reasoning can be applied. This sequencing strategy distinguishes STARQL from other approaches.

Below, we introduce time-aware reasoning approaches, including TA-RDF, Temporal RDF, and stRDF.

Time Annotated RDF(TA-RDF) [26] is a semantic content management middleware to provide transparent integration among heterogeneous data streams and their metadata. TA-RDF provides basic functionality for the representation and querying of time-related RDF data. Resources are annotated with timestamps, similar to the idea adopted by C-SPARQL. Then, they introduce the TA-SPARQL query language, which extends SPARQL to support queries that refer to the past. However, the current proposed model only allows one-time queries. Although the authors recognize the importance of continuous queries in streaming scenarios, they do not currently support them in their language.

Table 1: Analysis of Stream Reasoners for IoT (Note: ST stands for Single timestamp, SO for Squencing operation, QA for Query answering, and M for Materialization)

Reasoners	C-SPARQL	IMaRS	TrOWL	Ded. Ind. Stream Reasoning	C-SPARQL on S4	CQELS	Streaming-SPARQL	Streaming Knowledge Bases	Sparkwave	EP-SPARQL	STARQL	TA-RDF	Temporal RDF	stRDF
Heterogeneous data	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Efficiency	Low	Low	Low	Low	High	Low	Low	Low	High	Low	Low	Low	Low	Low
Scalability	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Stream combination	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Background knowledge	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Reasoning Capability	RDFS	Transitive Property	OWL2 DL/EL + Approximation	✗	RDFS Subset	✗	✗	RDFS/ OWL2 Subset	RDFS Subset	RDFS	First Order Logic	✗	✗	✗
Continuous query	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Time model	ST	ST	ST	ST	ST	ST	ST	ST	ST	Interval	SO	ST	ST	ST
Typical Reasoning Tasks	QA	M	M	M	QA	QA	QA	QA	M	QA	QA	QA	QA	QA
Open source	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗

Temporal RDF. Gutierrez et al. presents a similar approach [12] to TA-RDF. Temporal RDF is an extension to RDF to capture the notion of time, enabling reasoning over time enriched RDF data. The authors provide semantics for temporal RDF graphs and show how reasoning over temporal RDF does not add extra asymptotic complexity with respect to non-temporal RDF.

stRDF provides a constraint data model that extends RDF with the ability to represent spatial and temporal data [16]. An extension of SPARQL is developed for querying stRDF data. This work follows the main idea of constraint databases and represents spatial and temporal objects as quantifier-free formulas in a first-order logic of linear constraints.

3.3 Support for IoT Challenges

In Table 1, we list representative stream querying and reasoning techniques, and examine them with respect to the expectations of IoT. This table provides visual information for selecting feasible solutions when developing IoT systems. We emphasize some requirements stemming from IoT challenges presented in Section 2. Further, we discuss some practical issues for deploying stream reasoners in IoT.

Supporting Heterogeneous Data: This requirement is derived from C1 (Challenge 1). All reasoners support RDF based data models and some reasoners support different syntaxes of RDF. There is a possibility to transfer other formats to RDF as input data for reasoners. On the other hand, most stream reasoners are only evaluated with relatively small data and simple scenarios.

Efficiency: This requirement is straightforwardly derived from the C2. Most reasoners do not consider the efficiency issue in their design. However, C-SPARQL on S4 and TA-RDF demonstrate high throughput.

Scalable Data Processing: This requirement is derived from the C1 and C2. It requires reasoners to support, e.g. parallel and distributed processing, in order to improve scalability. Only two systems are considering parallel processing (C-SPARQL on S4 and CQELS).

Data Stream Combination: This requirement is derived from the C1. It requires reasoners to combine and query multiple data streams. Most surveyed reasoners support this feature.

Background Knowledge Integration: This requirement is derived from the C1. Integration of background knowledge will enable richer functions for reasoning. Most systems we analyzed enable data stream integration with static background knowledge. However, none of the current system addresses the merging of large background data with streams [10].

Reasoning Capability: This requirement is derived from C3. Deducing new relevant knowledge from streaming data and background knowledge facilitates intelligent behavior of IoT systems. However, semantic reasoning is computationally expensive, therefore, some existing solutions implement only a reduced set of reasoning entailments. For example, some reasoners focus on single properties, while others consider subsets of RDF Schema (RDFS) or Web Ontology Language (OWL) 2 RL. TrOWL seems to be able to manage the most sophisticated vocabulary, as it supports OWL 2 DL/EL and approximation.

Continuous Query: This requirement is derived from the C1. It is considered as one of the fundamental requirements for stream reasoners. It requires querying and producing new or updated results when new input data is available. All analyzed systems support continuous queries.

Time Model: This requirement is derived from the C3. Considering the importance of dynamic and time-changing data in IoT applications, approaches of defining time with suitable models is one of the key issues for building stream reasoning techniques. Moreover, a proper choice of the time model is needed to satisfy the requirements set by IoT applications. Most stream reasoners annotate a single timestamp to each piece of data. This might be a proper approach for expressing the occurrence of events. However, an interval based time model is utilized in some systems, such as EP-SPARQL, where two timestamps are utilized for indicating the interval. STARQL's approach is sequencing operators that set up at every time point a finite sequence of ABoxes on which temporal reasoning can be applied. It is noted that, the time models of TA-RDF, Temporal RDF, and stRDF are significantly different from the others.

Typical Reasoning Tasks: This requirement is derived from the C3. Here, we consider reasoning tasks, including query answering and materialization. Most reasoners

support query answering, and some recent efforts start to develop materialization, such as IMaRS and TrOWL.

Open Source: Open-source reasoners are considered to be more promising than proprietary alternatives. This is because, on the one hand, open source reasoners are expected to enable faster integration of new solutions across the application domains. On the other hand, the use of open source solutions has been reported to speed up the adoption of software technology in a bottom-up fashion. We noticed that some analyzed reasoners are open-source, while others are proprietary.

4. EXPERIMENTS

To demonstrate the feasibility of utilizing stream reasoning for IoT applications, we have implemented a small smart office IoT system. This system owns sensing capabilities based on events and delivers alerts to client applications on mobile devices, such as waste of electricity if the user left the room with lights on. The behavior of IoT devices could also be changed according to the reasoning results. For example, some data streams could be turned off and become unavailable. Figure 2 presents the architecture of this system. The system includes three kinds of ambient environment sensors, including a light sensor to detect office usage, a Passive Infrared Sensor (PIR) to detect movement, and a tilt sensor to detect change in the office door position, i.e. open or close. The location of a user is determined in relation to selected WiFi access point signal strength, i.e. inside or outside the room, and door position sensor. Sensor data are sent at a frequency of three seconds.

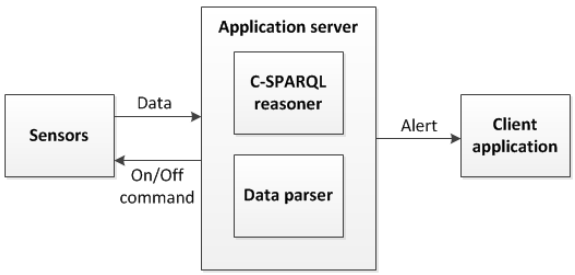

Figure 2: The architecture of our experimental system.

Sensors send data in JSON format which is then transformed to RDF statements. Data streams are processed on-the-fly and do not require a considerable amount of resources to make decisions. Compared to traditional reasoning approaches, which require possessing of an ontological model and updating the model with sensor data, we noticed that stream reasoners process data in an efficient manner. Therefore, our experiment shows stream reasoners are potential solutions to process variety, velocity, and volume of data streams (C1) in an efficient manner (C2). Moreover, combining on-the-fly several data streams simultaneously would enable much more interesting scenarios, and we will examine stream reasoners' semantic expressive power and robustness in our future experiments.

We utilize C-SPARQL to processing sensor data streams in this IoT system. Example querys in C-SPARQL are listed as follows: The first query is: When light is on and PIR sensor detects no movement, we deduce wasting energy.

```
REGISTER QUERY WasteEnergyLight AS
   PREFIX ee1: <http://myexample1.org/>
   PREFIX ee2: <http://myexample2.org/>
   SELECT ?sensor1 ?sensor2
   FROM STREAM <http://ee.oulu.fi/V1
         #/lightSensorStream> [RANGE 120s STEP 10s]
   FROM STREAM <http://ee.oulu.fi/V1
         #/pirSensorStream> [RANGE 120s STEP 10s]
   WHERE {
   ?sensor1 ee1:status ?st1 .
   ?sensor2 ee2:status ?st2 .
   FILTER NOT EXISTS { ?sensor1 ee1:status
         ee1:off }
   FILTER NOT EXISTS { ?sensor2 ee2:status
         ee2:moving }
   };
```

Listing 1: Query 1 in C-SPARQL

And the second query is: When the door is closed and no Wi-Fi or no movement for two minutes, we turn off the Wi-Fi sensor, PIR sensor and light sensor.

```
REGISTER QUERY WasteEnergyRoom AS
   PREFIX ee2: <http://myexample2.org/>
   PREFIX ee3: <http://myexample3.org/>
   PREFIX ee4: <http://myexample4.org/>
   SELECT ?sensor3 ?sensor2 ?sensor4
   FROM STREAM <http://ee.oulu.fi/V1
         #/pirSensorStream> [RANGE 120s STEP 10s]
   FROM STREAM <http://ee.oulu.fi/V1
         #/tiltSensorStream> [RANGE 120s STEP 10s]
   FROM STREAM <http://ee.oulu.fi/V1
         #/wifiSensorStream> [RANGE 120s STEP 10s]
   WHERE {
   ?sensor2 ee2:status ?st2 .
   ?sensor3 ee3:status ?st3 .
   ?sensor4 ee4:status ?st4 .
   FILTER NOT EXISTS { ?sensor2 ee2:status
         ee2:moving }
   FILTER NOT EXISTS { ?sensor3 ee3:status
         ee3:opem }
   FILTER NOT EXISTS { ?sensor4 ee4:status
         ee4:on }
   };
```

Listing 2: Query 3 in C-SPARQL

This experiment is our first step for demonstrating the feasibility of utilizing stream reasoning in IoT context. To generalize to other engines, we will implement and compare different engines in the same IoT system with their resource usage and reasoning latency. Moreover, stream reasoners will be developed for complex IoT scenarios with more IoT devices.

5. GAP ANALYSIS

Based on the analysis of existing solutions for stream reasoning, we identify the following main gaps and suggest future research in Table 2. The identified gaps include support of heterogeneous and scalable data sources, reasoning, managing uncertainty, time model, reasoning tasks, and benchmarking.

Table 2: A Gap Analysis Summary

	Current Status	Expectations	Gaps	Recommendations
Heterogeneous and scalable Data Streams	- Limited stream reasoners support scalable data processing. - Stream reasoners accept limited selection of data model and formats.	- Handling a large amount of heterogeneous, ubiquitous, and dynamic IoT data. - Resource-constrained IoT nodes need to be considered.	- Support of resource constrained IoT nodes. - Ease of data integration with different models.	- Developing data models. - Applying parallel and distributed reasoning approaches.
Reasoning	- Focusing on retrieval and search. - Trading expressive power for efficiency.	- Support application scenarios with user defined rules. - Lightweight reasoning mechanisms.	- Balance between expressive power and efficiency	- Incremental algorithms. - Limiting materialized knowledge. - Reasoning with user defined rules.
Managing Uncertainty	- limited support.	- Processing imprecise, incomplete, inconsistent, and incorrect IoT data.	- Suitable models are required, considering different levels of uncertainty.	- Introduce uncertainty modelling approaches.
Time model	- Simple time models.	- Flexible time models for diverse IoT applications	- Expressive time models for representing various timing and ordering relations.	- Developing a simple, but flexible time model.
Reasoning Tasks	- Support query answering and materialization	- Perform RDFS and OWL 2 reasoning to compute full materialization of dynamic IoT data.	- Flexible software architecture for different IoT applications.	- Stream Reasoning oriented software architecture
Benchmarking	- Existing stream systems are evaluated on small-scale scenarios.	- Stream systems should process heterogeneous large volumes of IoT data.	- Large scale evaluations.	- Development of benchmarks for concrete measurements of existing solutions.

To take into account the inherent disparity of IoT data stream sources, unifying data models need to be developed. These data models should be able to capture different types and domains of data while ensuring compatibility with different kinds of stream reasoners. An underlying question here is also, if future developments move towards a single solution overcoming the discussed challenges or if the aim is to move towards interoperability of different solution proposals. The latter approach could, for instance, be realized via an *Actor Model* allowing to compose networks of stream reasoning engines as has been recently proposed [8]. Moreover, it will be essential to have distributed reasoning approaches at hand allowing to cope with the big data nature of IoT. Some research efforts have been made [19] [32], but specific research challenges in IoT context need to be addressed.

Concerning reasoning, we detect a big gap when comparing current approaches to the state of the art of the traditional Semantic Web. At best, simple RDFS entailment is supported. In fact, most current stream reasoning engines do not support any entailment at all posing a big shortcoming in its current state. To fulfill the requirements of the IoT domain, research needs to be conducted towards balancing expressive power of rules and efficiency for providing low latency applications. Current solutions mostly focus on simple data retrieval, whereas reasoning would represent a key advantage in comparison to non-semantic streaming engines.

In the IoT domain, uncertainty is an essential parameter to be dealt with. Issues such as data arrival out of order, or sensors providing incorrect readings should be addressed by stream reasoning engines in order to provide practical results. Currently, limited research effort [23] [28] [30] is devoted to this area, but is recently attracting more research interests. Bayesian models or fuzzy logic approaches can be used to introduce uncertainty models into the stream processing pipeline.

The notion of time and timing of events is crucial in the IoT domain and a suitable time model is required to establish temporal relations between streamed data items. Possible time models can be interval-based (two timestamps establish lower and upper bounds of an interval), single timestamp-based (single timestamps represent a single point in time), or implicit (no explicit timestamps, sequence order only). Currently, there is no unifying time model or well-defined transition between different time models available. However, there is ongoing work in the W3C RDF Stream Processing Community Group[2] dealing, among other things, with the definition of proper semantics, including a time model. Intervals are useful for representing events which are valid over a period of time, whereas single timestamps can be used to depict events which can be associated via one explicit point in time. Single timestamps can be seen as a special case of an interval of zero duration. However, an interval can also be modelled as two separate timestamps. There is a pressing need to get a common understanding of which modelling approach is most suitable for IoT use cases.

Windows are a practical means to divide unbounded data streams into finite subsets to enable proper processing of contained data items. Typically, a window utilizes process-

[2]https://www.w3.org/community/rsp/, Accessed 2015-12-15

ing time to define its boundaries. Processing time is the time at which events are observed in a system. However, in real world scenarios, the actual event occurrence time, i.e. event time, differs from the processing time. The fact that most windows use processing time which possibly is highly skewed when compared to actual event time leads to missing data and ultimately to incorrect results. In order to provide complete and correct results, windows have to be based on event occurrence time. This is a significant gap which needs to be addressed in future developments.

We observe two typical reasoning tasks for stream reasoners, i.e. query answering and materialization. IoT applications demand different tasks based on their requirements. Some applications may require to perform RDFS and OWL 2 reasoning to compute the full materialization of semantic IoT data. And other applications may focus on querying IoT data. Therefore, stream reasoning oriented software architecture should be considered when IoT platforms and systems are developed.

Finally, benchmarking is an essential aspect which often is overlooked. Most proposed stream reasoning related approaches are not evaluated in rigorous environments. Since the IoT domain has specific requirements, benchmark frameworks need to be developed taking them into account. These benchmarks have to enable large-scale scenarios of distributed systems integrating heterogeneous data sources with big background knowledge. Key performance indicators which are specific in the context of IoT are required to allow for choosing adequate solutions based on benchmark evaluations. A first step in this direction has already been undertaken [27] [6] [22], however, more specific attempts towards IoT are necessary.

We understand that different IoT domains and applications present different requirements. For example, some applications handle frequently updating data and target low response time, but without complex reasoning. On the other hand, other applications do not require low response time, but demand complete and expressive reasoning. Hence, there will not be a single solution for all IoT applications. Special data models, query languages, reasoning approaches need to be developed to address different requirements accordingly. It remains an open issue to match reasoning ability and expressive power required in different applications.

6. CONCLUSIONS

The IoT vision that every object around us has computing, communicating, and storage capabilities, enables us to build IoT systems accessing heterogeneous, ubiquitous, and dynamic data and deducing actionable knowledge for creating value-added services. Intelligence needs to be brought to the connected and communicating things, so that they are able to work with people, for people and on behalf of people. To achieve this goal, we focus on reasoning voluminous, continuous, and real-time IoT data streams. In this paper, we examined key challenges that stream reasoners should address, studied off-the-shelf stream reasoners, highlighted their strengths and limitations, and examined their capabilities to meet the specific challenges of IoT. We then presented our preliminary experiment and performed a gap analysis to evaluate stream reasoners with respect to scalable data processing, reasoning, managing uncertainty, time modelling, reasoning tasks, and evaluation criteria. Based on this analysis, we proposed recommendations and suggested future research for development of stream reasoners in IoT.

These recommendations include: 1) developing proper data models and applying distributed reasoning approaches for handling heterogeneous and scalable data streams; 2) Applying incremental reasoning algorithms, limiting materialized knowledge, and reasoning with user defined rules for IoT applications; 3) introducing uncertainty modelling approaches, such as probability theory, fuzzy logic, etc; 4) developing flexible time models, which should be simple and expressive, not introducing too much computation; 5) developing stream reasoning oriented software architecture; and 6) developing benchmarks for concrete measurements of existing solutions.

Stream reasoning enables associating reasoning tasks to time windows describing data validity and therefore producing time-varying inferences. Stream reasoning introduces new query and reasoning models, based on time model and continuous queries, enable on-the-fly processing of streaming data for IoT. However, we notice that stream processing technologies are still in their infancy. Research on stream reasoning has mainly focused on query processing, and research on reasoning procedures which involve deducing actionable new knowledge is still very limited. Current developed solutions do not fulfill requirements of IoT identified in Section 2. For example, querying, reasoning and handling uncertainties on big IoT data streams is still an important challenge. How to split data over multiple reasoners, which data structures to adopt, and how to better exploit the limited size of main memory, how to reduce the expensive communication between reasoning nodes are challenges which remain unsolved. Hence, this research area is vastly unexplored, both from a theoretical and from a system development point of view.

To conclude, even though stream reasoning techniques are still in their early stages, we believe that methods for reasoning over heterogeneous, ubiquitous, and dynamic IoT data streams in real time will become increasingly important in the near future. IoT data is often stream data, and their usage is not only restricted to retrieval and search, but also perform tasks such as decision making and deriving new conclusions that may affect the behavior of data producing device and even the whole system. How to combine these tasks and make the best from state-of-the art technologies while optimizing overall resource consumption needs to be explored. Finally, though the main purpose of stream reasoners is for real time reasoning, management of historical data is also an important issue.

In the future, we will study new approaches to deal with streaming data expressed in a format compatible with Semantic Web languages. This approach will be potentially utilized in systems with big data produced by IoT environments where some devices have limited power, memory, processing, and communication capabilities. We will define a suitable model for time as one of the most critical aspects for building a solid theoretical foundation for stream reasoning. Moreover, we will identify all pieces of IoT data and knowledge. By doing so, both real time and historical data will be searchable and reasonable, when needed. Finally, taking into account frequently updated data and time constraints, we will consider incremental techniques operating with data that is influenced by changes. With this technique, a reasoner does not need to handle the entire knowledge base.

7. ACKNOWLEDGMENTS

We would like to thank Bo Li for his effort of developing the IoT system. The first author would like to thank Walter Ahlströmin säätiö, Tekniikan edistämissäätiö, and Tauno Tönningin säätiö for funding this research.

8. REFERENCES

[1] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic. Ep-sparql: A unified language for event processing and stream reasoning. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 635–644, New York, NY, USA, 2011. ACM.

[2] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream reasoning and complex event processing in etalis. *Semantic Web*, 3(4):397–407, October 2012.

[3] D. Barbieri, D. Braga, S. Ceri, E. Valle, Y. Huang, V. Tresp, A. Rettinger, and H. Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 25(6):32–41, November 2010.

[4] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus. Querying rdf streams with c-sparql. *SIGMOD Rec.*, 39(1), September.

[5] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus. C-sparql: A continuous query language for rdf data streams. *International Journal of Semantic Computing*, 04(01):3–25, 2010.

[6] H. Beck, M. Dao-Tran, T. Eiter, and M. Fink. Lars: A logic-based framework for analyzing reasoning over streams. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1431–1438. AAAI Press, 2015.

[7] A. Bolles, M. Grawunder, and J. Jacobi. Streaming sparql - extending sparql to process data streams. In *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 448–462. Springer Berlin Heidelberg, 2008.

[8] J. Calbimonte and K. Aberer. Reactive processing of RDF streams of events. In *Proceedings of the 4th International Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2015) Co-located with the 12th Extended Semantic Web Conference (ESWC 2015)*, pages 1–11, May 2015.

[9] B. Cuenca Grau, C. Halaschek-Wiener, and Y. Kazakov. History matters: Incremental ontology reasoning using modules. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 183–196. Springer Berlin Heidelberg, 2007.

[10] S. Dehghanzadeh, D. Dell'Aglio, S. Gao, E. Della Valle, A. Mileo, and A. Bernstein. *Approximate Continuous Query Answering over Streams and Dynamic Linked Data Sets*, pages 307–325. Springer International Publishing, Cham, 2015.

[11] J. Domingue, D. Fensel, and H. A. James. *Handbook of Semantic Web Technologies*. Springer, 2011.

[12] C. Gutierrez, C. Hurtado, and A. Vaisman. Introducing time into rdf. *Knowledge and Data Engineering, IEEE Transactions on*, 19(2):207–218, February 2007.

[13] J. Hoeksema and S. Kotoulas. High-performance distributed stream reasoning using s4. In *Proceedings of the Ordering workshop in ISWC 2011*.

[14] A. Hogan, J. Z. Pan, A. Polleres, and Y. Ren. Scalable OWL 2 Reasoning for Linked Data. In *Reasoning Web. Semantic Technologies for the Web of Data. Lecture Notes in Computer Science 6848 Springer, ISBN 978-3-642-23031-8*, 2011.

[15] S. Komazec, D. Cerri, and D. Fensel. Sparkwave: Continuous schema-enhanced pattern matching over rdf data streams. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, pages 58–68, New York, NY, USA, 2012. ACM.

[16] M. Koubarakis and K. Kyzirakos. Modeling and querying metadata in the semantic sensor web: The model strdf and the query language stsparql. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *The Semantic Web: Research and Applications*, volume 6088 of *Lecture Notes in Computer Science*, pages 425–439. Springer Berlin Heidelberg, 2010.

[17] J. Krämer and B. Seeger. Semantics and implementation of continuous sliding window queries over data streams. *ACM Trans. Database Syst.*, 34(1), April.

[18] D. Le-Phuoc, M. Dao-Tran, J. Xavier Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *The Semantic Web - ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 370–388. Springer Berlin Heidelberg, 2011.

[19] D. Le-Phuoc, H. Nguyen Mau Quoc, C. Le Van, and M. Hauswirth. *Elastic and Scalable Processing of Linked Stream Data in the Cloud*, pages 280–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[20] A. Maarala, X. Su, and J. Riekki. Semantic data provisioning and reasoning for the internet of things. In *Internet of Things (IOT), 2014 International Conference on the*, pages 67–72, October 2014.

[21] A. Margara, J. Urbani, F. van Harmelen, and H. Bal. Streaming the web: Reasoning over dynamic data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 25:24 – 44, 2014.

[22] B. Motik, Y. Nenov, R. Piro, and I. Horrocks. Incremental update of datalog materialisation: The backward/forward algorithm. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1560–1568. AAAI Press, 2015.

[23] M. Nickles and A. Mileo. *Web Stream Reasoning Using Probabilistic Answer Set Programming*, pages 197–205. Springer International Publishing, Cham, 2014.

[24] Ö. L. Özçep, R. Möller, and C. Neuenstadt. *KI 2014: Advances in Artificial Intelligence: 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings*, chapter A Stream-Temporal Query Language for Ontology Based Data Access, pages 183–194. Springer International

Publishing, Cham, 2014.

[25] J. Pan, E. Thomas, Y. Ren, and S. Taylor. Exploiting tractable fuzzy and crisp reasoning in ontology applications. *Computational Intelligence Magazine, IEEE*, 7(2):45–53, May 2012.

[26] A. Rodríguez, R. McGrath, Y. Liu, and J. Myers. Semantic management of streaming data. In *International Workshop on Scalable Semantic Web Knowledge Base Systems*, 2008.

[27] T. Scharrenbach, J. Urbani, A. Margara, E. D. Valle, and A. Bernstein. Seven commandments for benchmarking semantic flow processing systems. In *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Proceedings*, pages 305–319, May 2013.

[28] A. Skarlatidis, G. Paliouras, A. Artikis, and G. A. Vouros. Probabilistic event calculus for event recognition. *ACM Trans. Comput. Logic*, 16(2):11:1–11:37, February 2015.

[29] X. Su, J. Riekki, J. K. Nurminen, J. Nieminen, and M. Koskimies. Adding semantics to internet of things. *Concurrency and Computation: Practice and Experience*, 27(8):1844–1860, 2015.

[30] A.-Y. Turhan and E. Zenker. Towards temporal fuzzy query answering on stream-based data. In *HiDeSt@KI*.

[31] G. Unel and D. Roman. Stream reasoning: A survey and further research directions. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, FQAS '09, pages 653–662, Berlin, Heidelberg, 2009. Springer-Verlag.

[32] J. Urbani, A. Margara, C. Jacobs, F. Harmelen, and H. Bal. *DynamiTE: Parallel Materialization of Dynamic RDF Data*, pages 657–672. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[33] E. D. Valle, S. Ceri, F. v. Harmelen, and D. Fensel. It's a streaming world! reasoning upon rapidly changing information. *IEEE Intelligent Systems*, 24(6):83–89, November 2009.

[34] O. Walavalkar, A. Joshi, T. Finin, and Y. Yesha. Streaming knowledge bases. In *International Workshop on Scalable Semantic Web Knowledge Base Systems*, 2008.