Matching Over Linked Data Streams in the Internet of Things

A new system increases efficiency by integrating Linked Data streams generated from data collectors and disseminating matched data to relevant data consumers, based on user queries. The data structure, called TP-automata, suits the needs of high-performance Linked Data stream dissemination. Here, the authors use a real-world dataset generated in a Smart Building IoT Project to evaluate the system. Results show that the system can disseminate Linked Data Streams at one million triples per second with 100,000 registered user queries, which is several orders of magnitude faster than existing techniques.

Yongrui Qin and Quan Z. Sheng
University of Adelaide

Edward Curry
National University of Ireland, Galway

The Internet is a global system of networks interconnecting computers using the standard Internet protocol suite. It has significant impact on the world, serving billions of users worldwide. Millions of private, public, academic, business, and government networks – from local to global in scope – all contribute to the Internet's formation. It's a network of networks, and each network connects various computers. Hence, the traditional Internet has a focus on computers and can be called the Internet of Computers. In contrast, the Internet of Things (IoT) aims to connect everyday objects – such as coats, shoes, watches, ovens, washing machines, bikes, cars, humans, plants, animals, and changing environments – to the Internet to enable communications and interactions.[1] IoT's ultimate goal is to enable computers to see, hear, and sense the real world. Ericsson predicts that the number of Internet-connected things will reach 50 billion by 2020.[1]

Connecting all the things that people care about becomes possible in the IoT, and this leads to vast scales of real-time data. By exploiting such data, cities will become smarter and more efficient. Some promising IoT applications in future smart cities include aiding resource-management issues,[2] effectively managing street parking for reducing traffic congestion and fuel consumption,[3] efficiently distributing drinking water, tracking and recovering stolen property,[1] and so on.

Making IoT's potential a reality requires that we manage and process data efficiently and effectively. Given the scale of data generated in IoT, topics

 Published by the IEEE Computer Society 1089-7801/15/$31.00 © 2015 IEEE

Related Work on Linked Data Stream Dissemination

Recent work in data summaries on Linked Data[1] transforms Resource Description Framework (RDF) triples into numerical space. Then data summaries are built upon numerical data instead of strings, as summarizing numbers is more efficient than summarizing strings. To transform triples into numbers, you apply hash functions on the individual components (s, p, o) of triples. Thus, you can consider a derived triple of numbers as a 3D point. In this way, you can map a set of RDF triples into a set of points in a 3D space. To facilitate query processing over data summaries, one group of researchers adopted a spatial index named QTree[1] (evolved from standard R-tree[2]) as the basic index. Data summaries are designed mainly for indexing various Linked Data sources and are used for identifying relevant sources for a given query.

However, data summaries aren't suitable for our Linked Data stream dissemination system. First, techniques on data summaries, such as QTree, don't consider variables in the Basic Graph Patterns (BGPs); these techniques only consider RDF triples with concrete strings. Further, because data summaries are concise and imprecise representations of data sources,[1] they just provide match estimation. Hence, query evaluation on them would return false negative results, which isn't allowed in our system.

Researchers also have studied semantic matching, which aims to match semantically related RDF triples against BGPs. This approach might provide false positive match results, but not false negative. Both approximate event matching[3] and thematic event processing[4] apply semantic matching. These techniques also return false-negative matching results, which aren't allowed in our system.

Moreover, existing work on pattern matching, such as stream reasoning[5] and Linked Data stream processing,[6] doesn't support large-scale query evaluation, instead focusing on evaluation of a single query or a small number of parallel queries over the streaming Linked Data. Therefore, the issue of supporting pattern matching over a large number of BGPs against Linked Data streams remains open.

References

1. A. Harth et al., "Data Summaries for On-Demand Queries over Linked Data," *Proc. Conf. World Wide Web*, 2010, pp. 411–420.
2. A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. Sigmod,* 1984, pp. 47–57.
3. S. Hasan and E. Curry, "Approximate Semantic Matching of Events for the Internet of Things," *ACM Trans. Internet Technology,* vol. 14, no. 1, 2014, article no. 2.
4. S. Hasan and E. Curry, "Thematic Event Processing," *Proc. 15th Int'l Conf. Middleware,* 2014, pp. 109–120.
5. D. Anicic et al., "EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning," *Proc. Conf. World Wide Web,* 2011, pp. 635–644.
6. D.L. Phuoc et al., "A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data," *Proc. 10th Int'l Conf. Semantic Web,* 2011, pp. 370–388.

such as distributed processing, real-time data stream analytics, and event processing are critical. We might need to revisit these areas to improve existing technologies for applications at the IoT scale.[4,5] In this context, semantic technologies such as Linked Data, which aim to facilitate machine-to-machine communications, play an increasingly important role.[6] Linked Data is part of a growing trend toward highly distributed systems, with thousands or potentially millions of independent sources providing structured data. In collecting all of this data, one challenge is how to efficiently disseminate the data to relevant data consumers.

Thus, here we focus on the study of IoT from a *data perspective*. As Figure 1 shows, data are processed differently in IoT than in traditional Internet environments (such as the Internet of Computers). In the Internet of Computers, the main data producers and consumers are human beings. However, in the IoT, the main actors become *things*, where things are the majority of data producers and consumers. Therefore, in the context of the Internet, addressable and interconnected things (instead of humans) act as the main data producers, as well as the main data consumers. Computers will be able to learn and gain information and knowledge to solve real-world problems directly with the data fed from things. As an ultimate goal, computers enabled by IoT technologies will be able to sense and react to the real world for humans.

To move toward this goal, we must efficiently retrieve the most-relevant data from IoT environments (see, for example, the process of converting *data* into *information* in Figure 1). Hence, we propose an efficient data stream dissemination system for semantic IoT by leveraging semantic technologies, such as Linked Data. Our system efficiently retrieves relevant data from the deluge of IoT data, which can then facilitate the extraction of useful information (for others' work in this area, see the related sidebar). The system first integrates data generated from various data collectors. Then it transforms all the data to Linked Data streams in Resource Description Framework (RDF) format (see www.w3.org/RDF). Meanwhile, data consumers can register their interest in the

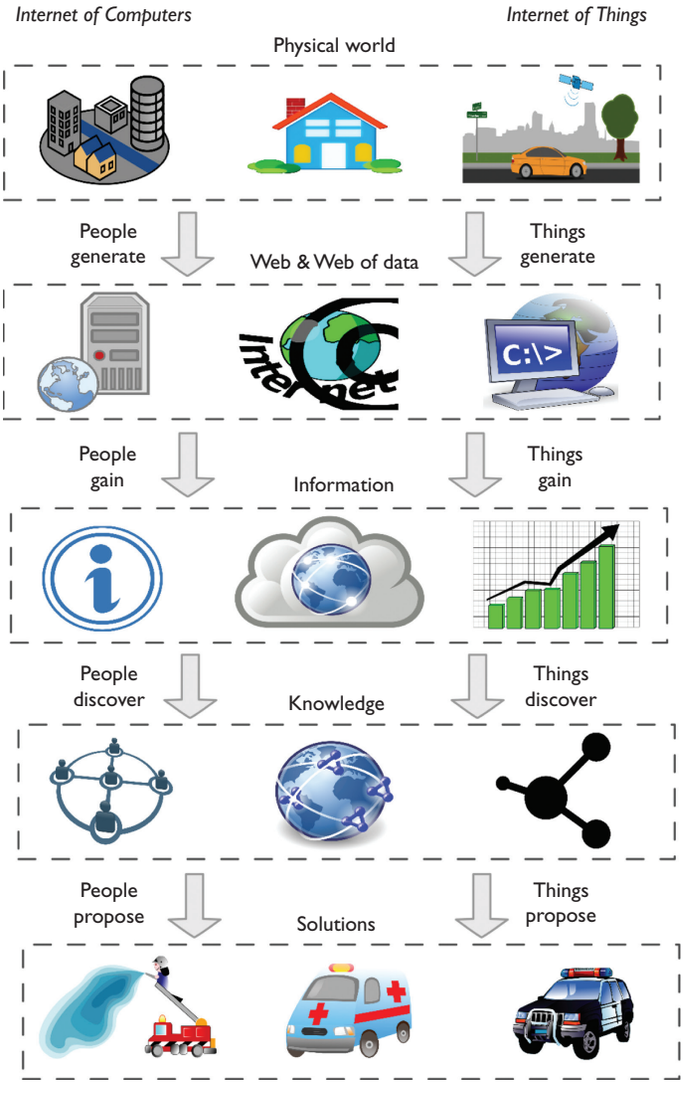

Figure 1. Internet of Computers versus the Internet of Things (IoT). In the Internet of Computers, the main data producers and consumers are human beings. However, in the IoT, the main actors become things, where things are the majority of data producers and consumers.

form of Basic Graph Patterns (BGPs; see www.w3.org/TR/rdf-sparql-query) in the system. Based on these BGPs, the system disseminates matched Linked Data to relevant users. After receiving relevant data, these users can further make use of the data to extract information for their own purposes, such as environment monitoring, event detection, complex event processing, and so on. However, we won't discuss data processing from the user side; instead, we focus on how to efficiently match a large number of BGPs against Linked Data streams.

We believe that this research aligns well with the vision of physical-cyber-social (PCS) computing (see http://wiki.knoesis.org/index.php/PCS). It deals with data from both the physical and cyber worlds. After being disseminated to relevant data consumers, these consumers can integrate such data with information and knowledge from the social world to provide better understanding, correlation, and contextually relevant abstractions to humans.

Linked Data Stream Dissemination System

To disseminate high-quality information and provide high-performance matching services to data consumers (or subscribers), we aim to design a system that won't return false-negative match results. Therefore, we investigate pattern matching here. Pattern matching performs individual component matching between RDF triples and BGPs. It doesn't consider semantic relatedness between an RDF triple and a BGP. It might return false-positive matching results but not false-negative ones. Recent work on pattern matching includes Linked Data stream processing[7] and stream reasoning.[8] However, because these solutions are mainly designed for optimizations of individual query evaluations, they aren't quite suitable for processing a large number of concurrent queries.

An example of pattern matching is that pattern (?s, :is, :Student) will match triple (:James, :is, :Student) but won't match (:James, :is, :PhDStudent). Other types of matching include match estimation and semantic matching, both of which might return false-negative results. Again, take pattern (?s, :is, :Student) as an example. In match estimation, the main task is to estimate which dataset matches pattern (?s, :is, :Student) the best by using some summarization techniques among multiple datasets[9] to avoid querying all known datasets directly. In contrast, semantic matching will match semantically related triples compared to a specified pattern.[10] For example, pattern (?s, :is, :Student) might match (:James, :is, :PhDStudent) because the term :Student in the pattern is semantically related to :PhDStudent in the triple.

System Overview

Figure 2a shows an overview of our system in the smart city scenario. We assume that data generated by all types of things will be represented in the form of Linked Data streams using RDF. In Semantic IoT, we can use the Semantic Sensor Network

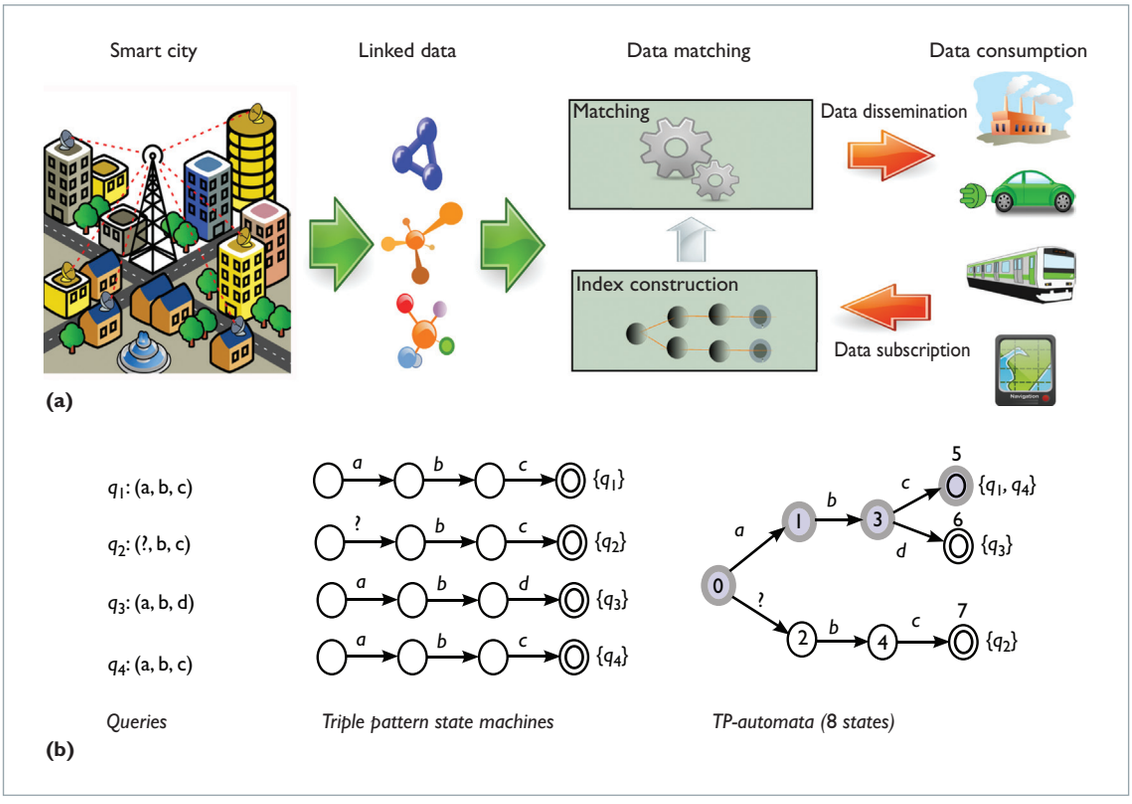

Figure 2. System overview and index structure. (a) Overview of Linked Data stream dissemination system. (b) Query index: Triple Pattern (TP)-automata.

Ontology (SSN; see www.w3.org/2005/Incubator/ssn/ssnx/ssn) to model sensing data. Our system mainly consists of two components: the *matching component* and the *index construction component*. Data consumers (humans and/or smart things in the city) can register their interests as user queries in the system. Then the index construction component will construct an index for all the user queries. The matching component will then evaluate incoming Linked Data streams against the constructed index to efficiently match triples with user queries. Finally, the system will disseminate matched data to relevant data consumers for further processing.

User queries. Our system adopts BGPs as user queries. BGPs are sets of triple patterns. The possible triple patterns in a BGP are (#s, #p, #o); (?s, #p, #o); (#s, ?p, #o); (#s, #p, ?o); (?s, ?p, #o); (?s, #p, ?o); (#s, ?p, ?o); and (?s, ?p, ?o). Here, ? denotes a variable while # denotes a constant. Similar to data summaries,[9] we apply hash functions (there are many different hash functions that are suitable for this purpose. Please refer to other work[9] for a discussion on mapping these patterns into numerical values.

Representations of queries and triples. In our Linked Data stream dissemination system, when the system registers the user queries (in the form of BGPs), it transforms all queries into numerical values. The reason for this is that the comparisons between numbers are faster than strings. Note that we have three numbers for the three components in a query. Then we construct a suitable index for efficient evaluation between Linked Data streams and user queries. Before matching starts, we map RDF triples in the data streams into numerical values. Then, the system matches these numerical represented triples with BGPs represented as numerical values in the constructed indexes.

TP-automata for Pattern Matching

Researchers have adopted automata techniques to process XML-based data streams.[11] They mainly base these techniques on languages with SQL-like syntaxes, and relational database execution models adapted to process streaming data. In our system, to support pattern matching, we apply automata to match each individual component of a triple with its counterparts of a BGP efficiently, which we call Triple Pattern automata (TP-automata).

```
@prefix do: <http://energy.deri.ie/ontology#>
@prefix dr: <http://../deri/deri rooms#>
:event1026fd7b0e5a  a                        events:PowerConsumptionEvent.
:event1026fd7b0e5a  do:consumer              do:platform.
:event1026fd7b0e5a  do:consumerType          dr:Room01.
:event1026fd7b0e5a  do:consumerLocation      dr:building01.
:event1026fd7b0e5a  do:powerUsage            :usage9739ccddc76d.
:event1026fd7b0e5a  do:consumerDepartment    "facilities".
:event1026fd7b0e5a  do:atTime                :timedb2c06100b33.
:usage9739ccddc76d  a                        dul:Amount.
:usage9739ccddc76d  do:hasDataValue          171.87.
:usage9739ccddc76d  do:isClassifiedBy        dr:watt.
:timedb2c06100b33   a                        do:Instant.
:timedb2c06100b33   do:inDDateTime           "2014-08-12T18:17:18".
```

Figure 3. Listing 1: an event example. This event is a power consumption event, showing the real-time power consumption in Room01 of building01. As shown in the event, the power consumption in Room01 at the moment of "2014 08 12T18:17:18" was 171.87 watts.

As mentioned, operating on numbers is more efficient than operating on strings. Note that when we map BGPs into numerical values, we treat variables in a BGP as a universal match indicator, represented by ?. The system maps this indicator into a fixed and unique numerical value, but not the whole range of a specific coordinate axis. The system treats this unique numerical value differently, as well, later in the triple evaluation process.

Figure 2b depicts the construction process of TP-automata. First, user queries will be transformed into triple-pattern state machines (see the middle of Figure 2b). As you can see, each triple-state machine contains an initial state, two internal states, one final state, and three transitions. In the figure, the first circle of a state machine represents the initial state, the next two circles represent the two internal states, and the doubled circle represents the final state. The three arrows associated with conditions are three transitions between different states. Similar to Yanlei Diao and her colleagues' approach,[11] we combine these state machines into one machine by exploiting shared common states with the same transitions. Figure 2b shows the combined machine, TP-automata, on the right. The shaded circles represent combined states.

To perform pattern matching over TP-automata, first we map triples in the Linked Data stream into numerical values. For example, suppose a triple (s, p, o) is mapped into a 3D point (a, b, c). The system will match it against TP-automata in the following process: First, it checks the initial state of TP-automata and looks for state transitions with condition a or condition ?. Following the state transitions, state 1 and state 2 become the current active states at the same time. It then looks for state transitions with condition b or ? from states 1 and 2. Following the transitions, states 3 and 4 become active states. Finally, following transitions with condition c or ? from states 3 and 4, two final states, states 5 and 7, are reached. By checking both final states, the system returns $\{q_1, q_2, q_4\}$ as the matching results. We should note that q_3: (a, b, d) won't match the input triple (a, b, c) because its object component's pattern is d, which doesn't match with c. The match process stops if and only if all current active states are final states or states with no satisfied transition.

Experimental Evaluation

To evaluate the system's capabilities and efficiency, we tested it using a real-world dataset.

Experimental Setup

The dataset used in our experiments was generated in a Smart Building Energy Project.[12] The energy readings were collected from 4–19 August 2014. In total, there are around 6.2 million triples in the dataset. Listing 1 (see Figure 3) depicts an event example.

As an initial work, we used simple BGPs (single triple patterns) as queries in the experiment. We can simulate the join queries by letting data subscribers issue multiple simple BGPs. But we leave extending our system to support complex BGPs or join queries as our future work. We randomly generated BGPs using the aforementioned seven patterns, based on our dataset. We didn't consider (?s, ?p, ?o) in our experiment, because it requires every triple in the Linked Data stream. In this case, no query index is needed. We generated from 10,000 queries to 100,000 queries for different runs in the experiment.

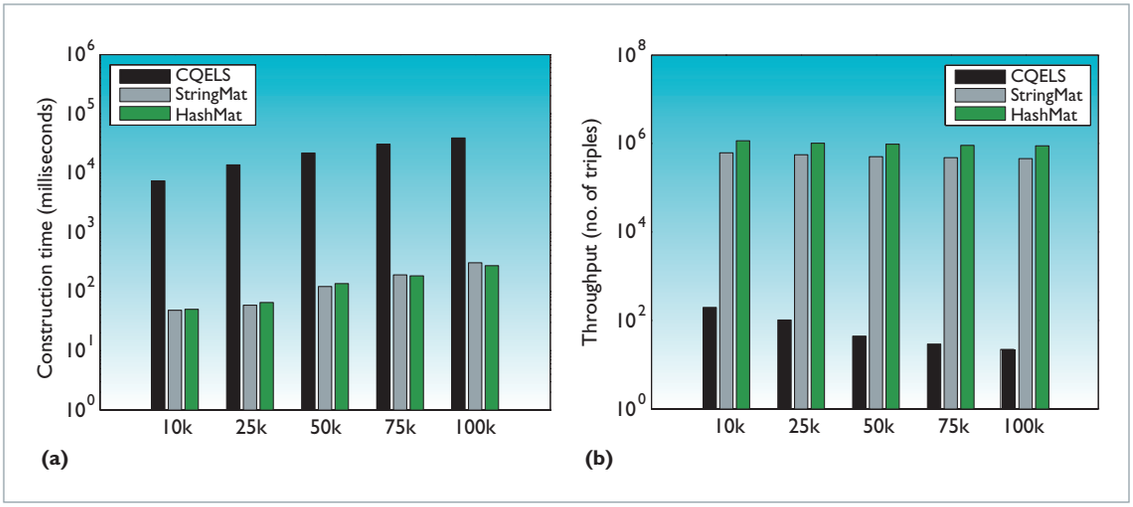

Figure 4. Performance on pattern matching. (a) Average construction time. (b) Average throughput.

We evaluated our approach's performance in terms of average construction time (in milliseconds) of the indexes and average throughput (in number of triples per second). We implemented hash-based TP-automata (where we map triples and queries into numerical values and denote this method as *HashMat* in the following figures) and string-based TP-automata (where we use triples and queries "as-is" and denote this method as *StringMat* in the following figures). We compared HashMat and StringMat with the state-of-the-art pattern-matching technique, Continuous Query Evaluation over Linked Stream (CQELS; see https://code.google.com/p/cqels/),[7] which is also designed for Linked Data streams. We examined the matching quality of the hash-based TP-automata as well. We implemented all of the methods on the Java Platform Standard Edition 7 running on Linux (Ubuntu 12.10, 64-bit operating system), with a quad-core CPU at 2.20 gigahertz (GHz) and 4 Gbytes of memory. We ran each experiment 10 times and reported their average experimental results.

Performance Study

Figure 4 presents the TP-automata's pattern-matching performance. We compare average construction time in Figure 4a. The construction times for both hash-based and string-based TP-automata are similar to each other in most settings. For larger numbers of queries, such as 75,000 and 100,000 queries, the construction of string-based indexes takes a slightly longer time. Normally, we can complete the construction within a few hundred milliseconds. However, the construction time of CQELS takes much longer, normally requiring around 10,000 ms.

Figure 4b depicts the throughput performance of pattern matching. It shows some large differences between the CQELS and TP-automata approaches (HashMat and StringMat). Generally, HashMat and StringMat can achieve throughput at the speed of nearly a million triples per second and are about four orders of magnitude faster than CQELS. The main reason for this is that CQELS is a much more comprehensive system, focusing on optimizing evaluation of queries with complex operators and semantics but not on evaluation of a large set of concurrent and simple queries over Linked Data streams. In this regard, we can also adapt our approach to complement CQELS for dealing with our Linked Data stream dissemination scenario. Regarding HashMat and StringMat, in most cases, HashMat is about twice the throughput speed compared to StringMat.

Finally, we investigated the matching quality of hash-based TP-automata (HashMat) via precision, recall, and F_1 score. This is because collisions are difficult to avoid in any hash-based approaches, and false positives exist in hash-based TP-automata, which affects matching quality. Specifically, we looked into precision and F_1 score when the recall is 100 percent, because we observe that the matching quality of HashMat is already excellent in such cases. As Table 1 shows, the precision and F_1 score are 100 percent when the number of queries is 10,000 or 25,000. For larger numbers of queries (for example, 50,000, 75,000, and 100,000), both the precision and F_1 score are still greater than 99.99950 percent. This demonstrates that HashMat provides an extremely high matching quality.

Table 1. Matching quality of HashMat (when the recall is 100 percent).

Queries (in thousands)	Recall (%)	Precision (%)	F_1 Score (%)
10	100	100	100
25	100	100	100
50	100	99.99975	99.99987
75	100	99.99982	99.99991
100	100	99.99960	99.99980

Our evaluation shows that TP-automata can disseminate Linked Data at the speed of nearly 1 million triples per second with 100,000 registered user queries and is several orders of magnitude faster in terms of both index construction time and throughput compared with the state-of-the-art technique. Further, using hash-based TP-automata, the throughput is doubled compared with string-based TP-automata with high matching quality.

We hope this article sheds light on the research of Linked Data stream dissemination to a large scale of data consumers. Future work includes supporting more complex user queries, such as join queries; and supporting semantic matching in a hashing space with the use of Locality-Sensitive Hashing (LSH) techniques[13] that help to map semantically related data together. Both directions will enable the Linked Data stream dissemination system to provide better semantics richness and to support data consumption needs more accurately, which is a critical issue in the IoT.

References

1. Y. Qin et al., "When Things Matter: A Data-Centric View of the Internet of Things," CoRR abs/1407.2704, 2014; http://arxiv.org/pdf/1407.2704v2.pdf.
2. J. Gao et al., "Distributed Resource Management and Matching in Sensor Networks," *Proc. IEEE Int'l Conf. Information Processing in Sensor Networks*, 2009, pp. 97–108.
3. S. Mathur et al., "ParkNet: Drive-By Sensing of Road-Side Parking Statistics," *Proceedings of the 8th Int'l Conf. Mobile Systems, Applications, and Services*, 2010, pp. 123–136.
4. A.E. James et al., "Research Directions in Database Architectures for the Internet of Things: A Communication of the First International Workshop on Database Architectures for the Internet of Things (DAIT 2009)," LNCS 5588, Springer, 2009, pp. 225–233.
5. P.M. Barnaghi, A.P. Sheth, and C.A. Henson, "From Data to Actionable Knowledge: Big Data Challenges in the Web of Things," *IEEE Intelligent Systems*, vol. 28, no. 6, 2013, pp. 6–11.
6. P.M. Barnaghi et al., "Semantics for the Internet of Things: Early Progress and Back to the Future," *Int'l J. Semantic Web Information Systems*, vol. 8, no. 1, 2012, pp. 1–21.
7. D.L. Phuoc et al., "A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data," *Proc. 10th Int'l Conf. Semantic Web*, 2011, pp. 370–388.
8. D. Anicic et al., "EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning," *Proc. Conf. World Wide Web*, 2011, pp. 635–644.
9. A. Harth et al., "Data Summaries for On-Demand Queries over Linked Data," *Proc. Conf. World Wide Web*, 2010, pp. 411–420.
10. S. Hasan and E. Curry, "Approximate Semantic Matching of Events for the Internet of Things," *ACM Trans. Internet Technology*, vol. 14, no. 1, 2014, article no. 2.
11. Y. Diao et al., "Path Sharing and Predicate Evaluation for High-Performance XML Filtering," *ACM Trans. Database Systems*, vol. 28, no. 4, 2003, pp. 467–516.
12. E. Curry, S. Hasan, and S. O'Riáin, "Enterprise Energy Management Using a Linked Dataspace for Energy Intelligence," *Proc. 2nd IFIP Conf. Sustainable Internet and ICT for Sustainability*, 2012, pp. 1–6.
13. S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming First Story Detection with Application to Twitter," *Human Language Technologies: The 2010 Ann. Conf. North American Chapter of the Assoc. for Computational Linguistics*, 2010, pp. 181–189.

Yongrui Qin is a PhD student in the School of Computer Science at the University of Adelaide. His research interests include the Internet of Things, data management, and mobile computing. Qin has an MSc in computer science from Fudan University. Contact him at yongrui.qin@adelaide.edu.au.

Quan Z. Sheng is an associate professor and head of the Advanced Web Technologies Research Group in the School of Computer Science at the University of Adelaide. His research interests include Web technologies, Big Data analytics, and the Web of Things. Sheng has a PhD in computer science from the University of New South Wales. Contact him at qsheng@cs.adelaide.edu.au.

Edward Curry is a research leader and lecturer at the Insight Centre for Data Analytics at the National University of Ireland, Galway. His research interests include IoT, enterprise-linked data, energy informatics, semantic information management, and community-based data curation. Curry has a PhD in computer science from the National University of Ireland, Galway. Contact him at edward.curry@insight-centre.org.