Chapter 12
Dissemination of Internet of Things Streams in a Real-time Linked Dataspace

Yongrui Qin, Quan Z. Sheng, and Edward Curry

Keywords Linked data · Event processing · Stream dissemination · Event matching · Dataspaces · Internet of Things

12.1 Introduction

The Internet of Things (IoT) envisions smart objects and intelligent systems collecting and sharing data on a global scale to enable smart environments. One challenging data management issue is how to disseminate data to relevant consumers efficiently. This chapter leverages semantic technologies, such as linked data, which can facilitate machine-to-machine communications to build an efficient stream dissemination system for Semantic IoT. The system integrates linked data streams generated from various collectors and disseminates matched data to relevant consumers based on user queries registered in the system. We design two new data structures to suit the needs of high-performance linked data stream dissemination in the following two scenarios: (1) stream dissemination in point-to-point systems; and (2) stream dissemination in wireless broadcast systems. The evaluation of the approaches using real-world datasets shows that they can disseminate linked data streams more efficiently than existing techniques.

This chapter is structured as follows: Sect. 12.2 introduces the data management challenges of the IoT from the perspective of dataspaces. The design of the stream dissemination services is covered in Sect. 12.3. Section 12.4 discusses point-to-point linked data stream dissemination, and wireless broadcast is discussed in Sect. 12.5. Experimental evaluation of linked stream dissemination in the contexts of both point-to-point and wireless broadcast is discussed in Sect. 12.6. Section 12.7 discusses related work, with Sect. 12.8 summarising and highlighting future work.

E. Curry, *Real-time Linked Dataspaces*,
https://doi.org/10.1007/978-3-030-29665-0_12

12.2 Internet of Things: A Dataspace Perspective

The Internet of Things (IoT) aims to connect everyday objects, such as coats, shoes, watches, ovens, washing machines, bikes, cars, even humans, plants, animals, and changing environments, to the Internet to enable communications/interactions between these objects [288]. The goal of IoT is to enable computers to see, hear, and sense the real world. In the era of IoT, it is envisioned that smart objects and intelligent systems collect and share data on a global scale via the Internet.

In IoT, connecting all the things that people care about in the world becomes possible. All these Internet-connected things will produce vast scales of data in real time. Smart environments, leveraging IoT, are enabling data-driven intelligent systems that are transforming our everyday world, from the digitisation of traditional infrastructure (smart energy, water, and mobility) [289], the revolution of industrial sectors (smart autonomous cyber-physical systems, autonomous vehicles, and Industry 4.0), to changes in how our society operates (smart government and cities) [1]. Some promising IoT applications in future smart cities include resource management issues [290], effective urban street-parking management for reducing traffic congestion and fuel consumption [31, 291], efficient ways to distribute drinking water, assisting tracking and recovering stolen property [288], energy management [16], and so on.

On the Internet, the primary data producers and consumers are human beings. However, in the IoT, the main actors become *Things*, which means things are the main data producers and consumers. Therefore, in the context of the IoT, addressable and interconnected things, instead of humans, act as the primary data producers, as well as the primary data consumers. Intelligent systems will be able to learn and gain information and knowledge to solve real-world problems directly with the data fed from things. As a goal, intelligent systems within a smart environment enabled by IoT technologies will be able to sense and react to the real world for humans.

To make the potential of IoT a reality, we need to manage and process data efficiently and effectively, which will require new approaches to data management. Given the scale of data generated in IoT, topics such as distributed processing, real-time data stream analytics, and event processing are all critical. We may need to revisit these areas to improve upon existing technologies for applications in IoT scale [292, 293].

12.2.1 Real-time Linked Dataspaces

To support the interconnection of intelligent systems in the data ecosystem that surrounds an IoT-based smart environment, there is a need to enable the sharing of data among systems. A data platform can provide a clear framework to support the sharing of data among a group of intelligent systems within a smart environment [1] (see Chap. 2). In this book, we advocate the use of the dataspace paradigm within the design of data platforms to enable data ecosystems for intelligent systems.

A dataspace is an emerging approach to data management that is distinct from current approaches. The dataspace approach recognises that in large-scale integration scenarios, involving thousands of data sources, it is difficult and expensive to obtain an upfront unifying schema across all sources [2]. Within dataspaces, datasets *co-exist* but are not necessarily fully integrated or homogeneous in their schematics and semantics. Instead, data is integrated on an *as-needed* basis with the labour-intensive aspects of data integration postponed until they are required. Dataspaces reduce the initial effort required to set up data integration by relying on automatic matching and mapping generation techniques. This results in a loosely integrated set of data sources. When tighter semantic integration is required, it can be achieved in an incremental *pay-as-you-go* fashion by detailed mappings among the required data sources.

We have created the Real-time Linked Dataspace (RLD) (see Chap. 4) as a data platform for intelligent systems within smart environments. The RLD combines the pay-as-you-go paradigm of dataspaces with linked data/knowledge graphs and real-time stream and event processing capabilities to support a large-scale distributed heterogeneous collection of streams, events, and data sources [4].

Within the IoT, semantic technologies such as linked data, which aim to facilitate machine-to-machine communications, are playing an increasingly important role [294]. Linked data is part of a growing trend towards highly distributed systems, with thousands or potentially millions of independent sources providing structured data. These data sources can be managed within a dataspace for a smart environment. Due to a large amount of data produced within an IoT-based smart environment by different things, challenges exist with disseminating relevant data to multiple mobile data consumers in an efficient manner.

12.3 Stream Dissemination Service

The design of the stream dissemination service within the Real-time Linked Dataspace is based on our existing work in this area, including [122, 295, 296], which is brought together in this chapter and contextualised for use within the dataspace paradigm.

Figure 12.1 shows an overview of the RLD in a smart city scenario. We assume that data generated by all kinds of things will be represented in the form of linked data streams using the Resource Description Framework (RDF) and are managed within a dataspace. In Semantic IoT, the Semantic Sensor Network Ontology (SSN, http://www.w3.org/2005/Incubator/ssn/ssnx/ssn) can be used to model sensing data. Our service consists of two components: the *matching component* and the *index construction component*. Data consumers (humans and 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 match triples with user queries efficiently. Finally, the system will disseminate matched data to relevant data consumers for further processing.

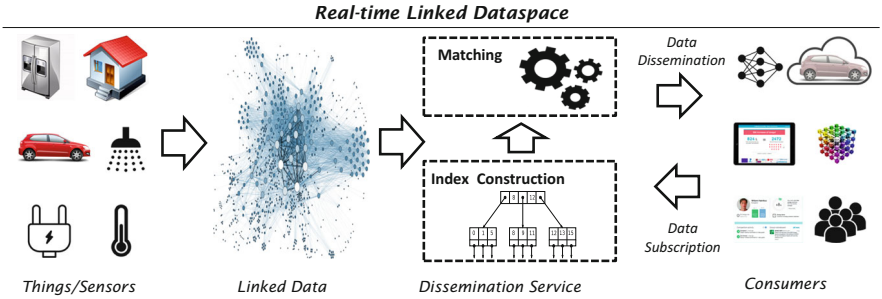

Fig. 12.1 Overview of linked data stream dissemination service [295]

It is important to efficiently disseminate the most relevant data from IoT-based smart environments for the effective utilisation of the information streams. To this end, the Real-time Linked Dataspace supports data dissemination with a dedicated support service. The service uses two efficient data stream dissemination methods for semantic IoT enabled by linked data and semantic technologies.

- *Point-to-Point Stream Dissemination*: Supports the retrieval of relevant data from the deluge of the IoT data, which can then facilitate the extraction of useful information. The system first integrates the data generated from various data collectors. Then it transforms all the data to linked data streams (in RDF format, http://www.w3.org/RDF/). Meanwhile, data consumers can register their interest in the form of Basic Graph Patterns (BGPs, http://www.w3.org/TR/rdf-sparql-query/) in the system. Based on these BGPs, the system disseminates matched linked data to relevant users.
- *Stream Dissemination by Wireless Broadcasting*: Supports the massive number of mobile users in IoT-based smart environments. As the batteries of smart objects are often limited, an efficient way to reduce energy consumption and lower access latency is imperative. This can be achieved by designing effective and efficient air indexes for broadcasting linked data on air. We introduce a novel method by adopting 3D Hilbert curve mappings [297] for all the points converted from RDF triples. These mappings transform the 3D points into a sequence of one-dimensional points, which are suitable for efficient sequential access on air.

12.3.1 Pay-As-You-Go Service Levels

Dataspace support services follow a tiered approach to data management that reduces the initial cost and barriers to joining the dataspace. When tighter integration into the dataspace is required, it can be achieved incrementally by following the service tiers defined. The incremental nature of the support services is a core enabler of the pay-as-you-go paradigm in dataspaces. The functionality of the stream dissemination service follows the 5 Star pay-as-you-go model (detailed in

Chap. 4) of the RLD. It should be noted that currently, the stream dissemination service only supports events in linked data format. The stream dissemination service has the following tiered-levels of support:

1 Star **No Service:** No stream dissemination is supported.
2 Stars **No Service:** No stream dissemination is supported.
3 Stars **Point-to-Point:** Basic dissemination of streams between two points based on a simple matching of event sources using BGP. Services handle the transformation of streams to linked data at this level.
4 Stars **Wireless Broadcast:** Air indexes for broadcasting linked data within the smart environment.
5 Stars **Complex Patterns:** More expressive queries for complex event detection, in collaboration with the CEP service.

After receiving relevant data, users can further make use of the data to extract information for their purposes, such as environment monitoring, event detection, complex event processing, and so on. However, we will not discuss the data processing at the user side as this is the subject of Part IV of this book; in this chapter, we focus on how to match many BGPs against linked data streams efficiently.

12.4 Point-to-Point Linked Data Stream Dissemination

To disseminate high-quality information and provide high-performance matching services to data consumers (or subscribers) within a dataspace, we aim to design a system that will not return false-negative matched results. Therefore, we investigate pattern matching in this chapter. Pattern matching performs individual component matching between RDF triples and BGPs. It does not consider semantic relatedness between an RDF triple and a BGP (see Chap. 13 for semantic matching approaches). It may return false-positive matching results, but not false-negative ones. Recent works on pattern matching include linked data stream processing [298] and stream reasoning [299]. However, since these solutions are designed for optimisations of individual query evaluations, they are not suitable for processing many concurrent queries.

User Queries Basic Graph Patterns (BGPs) are adopted as user queries in our system. BGPs are sets of triple patterns. The possible triple patterns in a BGP are: (1) (#s, #p, #o), (2) (?s, #p, #o), (3) (#s, ?p, #o), (4) (#s, #p, ?o), (5) (?s, ?p, #o), (6) (?s, #p, ?o), (7) (#s, ?p, ?o), and (8) (?s, ?p, ?o). Here, ? denotes a variable, while # denotes a constant. Similar to data summaries [300], we apply hash functions (there are many different hash functions that are suitable for this purpose. For more details, please refer to [300]) to map these patterns into numerical values.

An example of pattern matching is that pattern (?s, :is, :Student) will match triple (:James, :is, :Student) but will not match (:James, :is, :PhDStudent). Other types of matching include match estimation and semantic matching, both of which may

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 summarisation techniques among multiple datasets [300] to avoid querying all known datasets directly. In contrast, semantic matching will match semantically related triples compared to a specified pattern [151]. For example, pattern (?s, :is, :Student) may match (:James, :is, :PhDStudent) since the term:Student in the pattern is semantically related to :PhDStudent in the triple.

Representations of Queries and Triples In our linked data stream dissemination system, when the user queries (in the form of BGPs) are registered, all queries will be transformed into numerical values. The reason for this is that the comparisons between numbers are faster than strings. Note that we will have three numbers for the three components in a query as described above. Then a suitable index will be constructed for efficient evaluation between linked data streams and user queries. Before matching starts, RDF triples in the data streams will be mapped into numerical values. Then, these numerically represented triples will be matched with BGPs represented as numerical values in the constructed indexes.

12.4.1 TP-Automata for Pattern Matching

Automata techniques have been adopted to process XML-based data streams [301]. They are based 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).

Firstly, 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 "?". This indicator will be mapped into a fixed and unique numerical value but not the whole range of a specific coordinate axis. This unique numerical value will be treated differently as well later in the triple evaluation process.

Figure 12.2 depicts the construction process of TP-automata. Firstly, user queries will be transformed into triple pattern state machines, as shown in the middle of Fig. 12.2. As can be seen from the figure, 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 [301], these state machines can be combined into one machine by exploiting shared

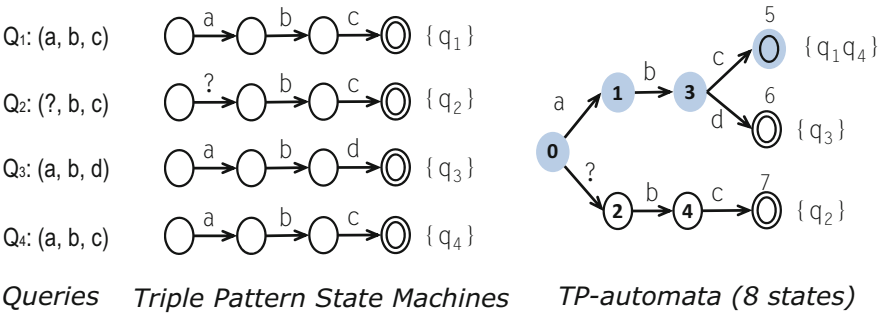

Fig. 12.2 Query index structure: TP-automata [295]

common states with the same transitions. The combined machine, TP-automata, is shown on the right of Fig. 12.2. The shaded circles represent combined states.

To perform pattern matching over TP-automata, triples in the linked data stream will be first mapped 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. It first checks the initial state of TP-automata and looks for state transitions with the condition a or condition ?. Following the state transitions, state 1 and state 2 become the currently active states at the same time. It then looks for state transitions with condition b or ? from state 1 and state 2. Following the transitions, state 3 and state 4 become active states. Finally, following transitions with condition c or ? from state 3 and state 4, two final states, state 5 and state 7, are reached. By checking both final states, the system returns $\{q_1, q_2, q_4\}$ as the matching results. It should be noted that q_3: (a, b, d) will not match the input triple (a, b, c) as its object component's pattern is d, which does not match with c. The matching process stops if and only if all current active states are final states or states with no satisfied transition.

12.5 Linked Data Stream Dissemination via Wireless Broadcast

In a wireless data broadcast system, there is a base station that pre-processes data before it broadcasts the data on the wireless channel. If mobile clients have registered an interest in some data on the server, they can listen to the wireless channel and download the data. All mobile clients can share the wireless channel. In this way, the broadcast system could be able to serve an arbitrary number of mobile clients simultaneously.

For clients to efficiently locate data of interest, air indexing techniques are used to facilitate the searching of data on air. Air indexes usually are lightweight and concise summaries of the data to be broadcast. Based on air indexes, mobile clients within

the communication range of the base station can evaluate their queries directly and then locate requested data on the wireless channel.

Similar to the existing work in data broadcast, we use access latency and tuning time as the primary performance metrics [302]. Access latency refers to the time elapsed between the moment when a query is formed, and the client starts listening to the server, to the moment when all requested data has been received. Tuning time refers to the period that a client must stay active to complete a query.

12.5.1 The Mapping Between Triples and 3D Points

Existing lightweight data summaries (e.g. [300, 303]) have proven to be effective to index linked data. However, they are not suitable in a wireless broadcast system. To develop a new index structure for broadcasting linked data, similar to data summaries, we choose to use hash functions to map RDF triples into numerical values. These numerical values can be regarded as coordinates in a 3D space. Precisely, given a hash function F, a triple (s, p, o) can be mapped into a 3D point (F (s), F (p), F (o)). We call such a point mapped from a triple a data point to differentiate it from other points in the 3D space. Using this approach, a set of RDF triples can be mapped into a set of 3D data points.

BGPs are again used [303] to represent queries in our system. Like RDF triple mappings, a single BGP containing only one RDF triple pattern can be mapped into a point, a line, or a plane in a 3D space, or even the whole 3D space, depending on the number of variables in the triple pattern. The possible triple patterns in a BGP are: (1) (#s, #p, #o), (2) (?s, #p, #o), (3) (#s, ?p, #o), (4) (#s, #p, ?o), (5) (?s, ?p, #o), (6) (?s, #p, ?o), (7) (#s, ?p, ?o), and (8) (?s, ?p, ?o). Here, ? denotes a variable while # denotes a constant. Clearly, pattern 1 can be mapped into a 3D data point. Patterns 2 to 4 can be mapped into lines in the 3D space and patterns 5 to 7 can be mapped into planes. It should be noted that we do not consider pattern 8 in this approach, as it will be mapped into the whole 3D space and requires a traversal of all the data points in the whole 3D space, where air indexing is not required.

12.5.2 3D Hilbert Curve Index

A space-filling curve in D dimensions is a continuous, surjective mapping between one-dimensional space and D-dimensional space. A Hilbert curve is an example of a space-filling curve. It generally has good locality properties [297] and can efficiently support matching against BGPs with variables that can be mapped into lines or planes. Hence, the Hilbert curve is adopted as the foundation of our indexing method.

Mapping 3D Points to One-Dimensional Points To simplify our discussion, we use a 2D Hilbert curve to illustrate our ideas, which can then be generalised to 3D Hilbert curves. Figure 12.3 shows 2D Hilbert curves for order 1 and 2, that is, H1 and H2, respectively. Note that, a K order Hilbert curve, denoted as HK, passes all centre points of 2KD subdividing squares (or hypercubes) in a D-dimensional space. In Fig. 12.3a, each centre point of a subdividing square in 2D space is assigned a Hilbert value, which can be regarded as a one-dimensional point. Note that, the mapping between centre points and Hilbert values are bijective, which means for a given Hilbert curve, we can freely convert between centre points and Hilbert values in constant time.

From Fig. 12.3b, we can see that high-order Hilbert curves can be easily derived using transformation from low-order Hilbert curves like the one shown in Fig. 12.3b. To derive H2 from H1, in Fig. 12.3b, the 2D space is divided into 2D ($D = 2$ in this case) sub-regions, where each sub-region contains an H1. After rotating the lower two H1 curves, an H2 Hilbert curve is derived (see Fig. 12.3c). Since H1 has 22 subdividing squares, H2 has entirely subdividing squares.

Figure 12.4 presents an example of a 3D Hilbert curve of order 1. Higher-order 3D Hilbert curves can be derived using a similar process. To accommodate a larger 3D data space, that is, the hashing space for RDF triples, we need to utilise higher-order 3D Hilbert curves. We can quickly check that a K order 3D Hilbert curve can have up to $23 \times K$ data points. In other words, when mapping to a K order 3D Hilbert curve, all RDF triples will be mapped into at most $23 \times K$ data points (also centre points of hypercubes) in a 3D space.

Indexing One-Dimensional Points on Air After mapping 3D points into one-dimensional points using 3D Hilbert curves, we can utilise B+-trees to index one-dimensional points on a 3D Hilbert curve. An example is depicted in Fig. 12.5. Each one-dimensional point in the leaf nodes contains a pointer to a real triple that will be broadcast on the wireless channel. Such B+-trees can be serialised and broadcasted on the linear wireless channel as air indexes for the linked data on the air in the form of data packets. We adjust the fan-out of a B+-tree according to the packet capacity of the wireless channel so that a complete node of a B+-tree can fit in a packet. After downloading a part (e.g. a few packets) of an air index, mobile clients

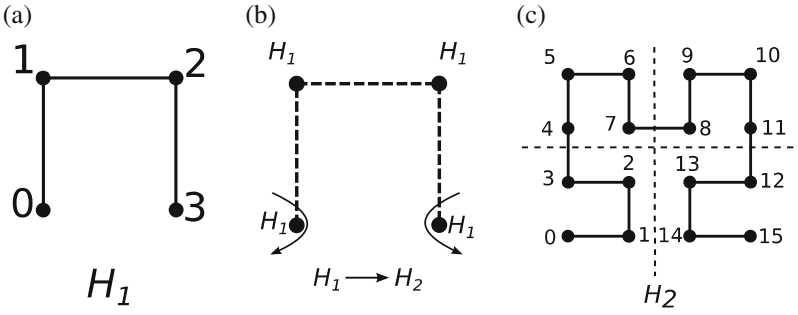

Fig. 12.3 2D Hilbert curves of order 1 and 2. (**a**) H1, (**b**) H1 to H2, (**c**) H2

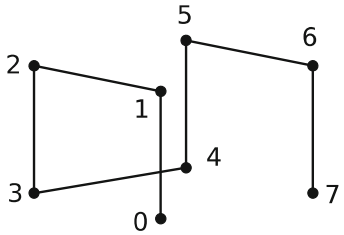

Fig. 12.4 3D Hilbert curve of order 1

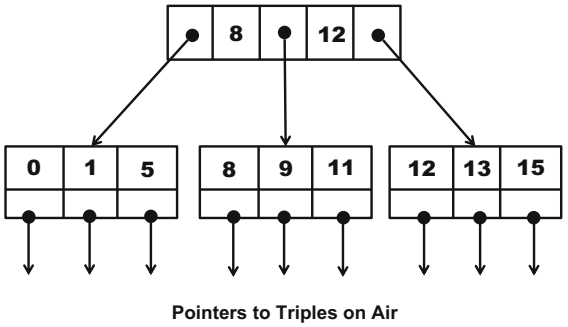

Fig. 12.5 B+-tree for some points on a Hilbert curve

can then evaluate their queries (i.e. BGPs) against the partial index, determine which remaining index packets should be further retrieved, and finally compute the broadcast time of matched triples after all necessary index information has been acquired.

Evaluating Queries Against an Air Index In the query evaluation process, one challenging issue is how to match a one-dimensional point against BGPs. As mentioned previously, BGPs could be mapped into a point, a line, or a plane in a 3D space. In order to match BGPs with data points in the 3D space, we need to transform one-dimensional points in B+-tree based air indexes into 3D points. We then examine whether such 3D points fall into the subspace defined by a BGP. If yes, RDF triples pointed to by these points match the BGP. Otherwise, they do not match.

Query Evaluation Example Suppose a triple (a, b, c) can be hashed as (112, 31, 92) in a 3D space and its Hilbert value is 1137. Now suppose a mobile client issues a BGP (a, b, ?o). This BGP can be converted into a 3D line (112, 31, ?). After receiving Hilbert value 1137 from the air index, the mobile client firstly converts it back into a 3D point (112, 31, 92) and then it finds that this point falls on the 3D line defined by its BGP. Then the client knows the triple pointed to by Hilbert value 1137 is of its interest. As mentioned earlier, the conversion between a Hilbert value and a 3D point can be calculated in a constant time given a Hilbert curve of order K (here, K is a constant).

Reducing Search Space One issue needs to be addressed in the above query evaluation process: how to reduce the search space of Hilbert values indexed by B +-trees, thereby leading to fewer index packets required to download for query

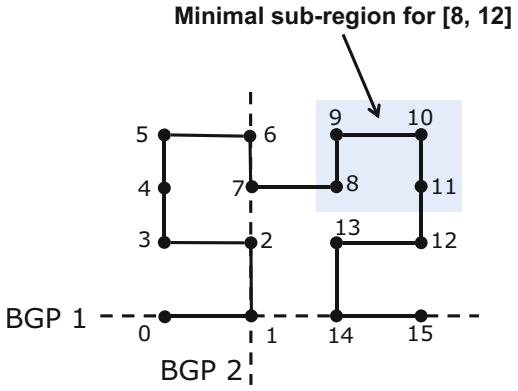

Fig. 12.6 Minimal sub-region

evaluation. Given an air index like the one shown in Fig. 12.5, the root node has three child nodes. Based on the Hilbert values in the root node, we need to determine which child node would contain triples that may match a given BGP. We observe that each child node contains multiple Hilbert values, and the range of these values can be easily computed from the root node. For example, the value ranges of the three child nodes are [0, 8], [8, 12], and [12, HMAX] (here HMAX refers to the maximum Hilbert value of a Hilbert curve). For each value range, we have two bounding Hilbert values.

To reduce the search space, we compute the minimal sub-region defined by a lower-order Hilbert curve that covers the range defined by both bounding Hilbert values (see Fig. 12.6, where two dash lines represent two BGPs). If the minimal sub-region intersects with the sub-space (i.e. a line) defined by a BGP, the child node with the value range may contain triples that match that BGP. Otherwise, no triples in the child node will match that BGP. The example shown in Fig. 12.6 illustrates a minimal sub-region for the value range [8, 12]. We can see that two BGPs that are represented as two dash lines have no intersections with it. So, we can infer that no triples pointed to by the second child node (triples whose Hilbert values are 8, 9, and 11) in Fig. 12.5 will match the two BGPs shown as two dash lines in Fig. 12.6.

12.6 Experimental Evaluation

12.6.1 Evaluation of Point-to-Point Linked Stream Dissemination

The dataset used in this experiment was generated in a smart office pilot, as discussed in Chap. 14 [100]. The energy readings were collected from August 4, 2014, to August 19, 2014. In total, there are around 6.2 million triples in the dataset. An event example is depicted in Table 12.1.

Table 12.1 An event example [295]

@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" .

As an initial work, we used simple BGPs (i.e., single triple patterns) as queries in the experiment. We can simulate the join queries by letting data subscribers issue multiple simple BGPs. However, we leave extending our system to support complex BGPs or join queries as our future work. We randomly generated BGPs using the seven patterns mentioned in Sect. 12.4 based on our dataset. We did not consider the pattern (?s, ?p, ?o) in our experiment as it requires every triple in the linked data stream. In such a case, no query index is needed. We generated 10,000 queries to 100,000 queries.

We evaluated the performance of our approach in terms of Average Construction Time (in milliseconds) of the indexes and Average Throughput (number of triples per second). We implemented hash-based TP-automata (i.e. we map triples and queries into numerical values and denote such method as *HashMat* in the following figures) and string-based TP-automata (i.e. we use triples and queries as is and denote this method as *StringMat* in the following figures). We compared HashMat and StringMat with state-of-the-art pattern matching technique, CQELS [298] (https://code.google.com/p/cqels/), which is also designed for linked data streams. We examined the matching quality of the hash-based TP-automata as well. All methods were implemented on Java Platform Standard Edition 7 running on Linux (Ubuntu 12.10, 64-bit Operating System), with quad-core CPU@2.20GHz and 4 GB memory. We ran each experiment 10 times and reported their average experimental results.

The performance of pattern matching on TP-automata is presented in Fig. 12.7. Average construction time is compared in Fig. 12.7a. The construction times for both hash-based TP-automata and string-based TP-automata are similar to each other in most settings. For more significant numbers of queries, such as 75k and 100k queries, the construction of string-based indexes takes a slightly longer time. Usually, the construction can be completed within a few hundred milliseconds. However, the construction time of CQELS takes much longer, which usually requires around ten thousand milliseconds.

Throughput performance of pattern matching is depicted in Fig. 12.7b. It shows some substantial differences between CQELS and TP-automata based 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 optimising evaluation of queries with complex operators and semantics but not on the evaluation of a large set of concurrent and straightforward queries over linked data streams. In this regard, our approach can also be adapted to complement CQELS for dealing with our linked data stream dissemination scenario. Regarding HashMat and StringMat, in most cases, HashMat achieves about twice the throughput speed compared with 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 investigated Precision and F_1 Score when the Recall is 100% since we observe that the matching quality of HashMat is already excellent in such cases. As can be seen in Table 12.2, the Precision and F_1 Score are 100% when the number of queries is 10k or 25k. For more significant numbers of queries (e.g. 50k, 75k, and 100k), both Precision and F_1 Score are still higher than 99.99950%. This demonstrates that HashMat provides a very high matching quality.

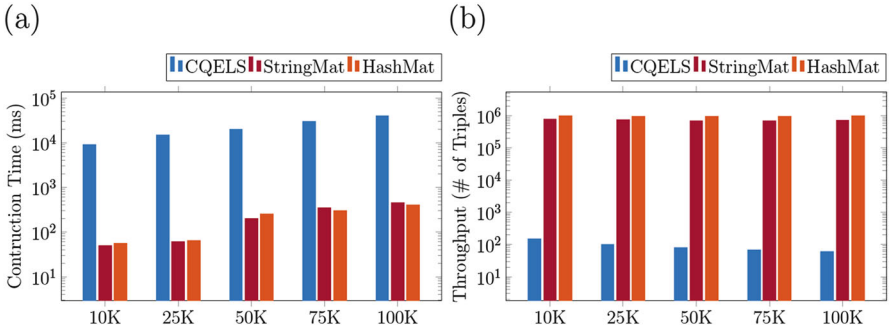

Fig. 12.7 Performance on pattern matching. (**a**) Average construction time. (**b**) Average throughput [295]

Table 12.2 Matching quality of HashMat when the recall is 100%

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

12.6.2 Evaluation on Linked Stream Dissemination via Wireless Broadcast

The dataset used in this experiment is a subset of the current version of the English DBpedia.[1] It contains resources of type dbpedia-owl:Event. Each event is a triple of the form <eventURI, rdf:type, dbpedia-owl:Event>. An example of an event URI is http://dbpedia.org/resource/Battle_of_Brentford_(1642). There are approximately 400,100 triples in the dataset.

As an initial work, we used simple BGPs as queries in the experiment, and we leave extending our system to support complex BGPs or join queries as our future work. We randomly generated BGPs using the seven patterns mentioned in Sect. 12.5.1 based on our dataset. We generated 100,000 queries and reported the average experimental results.

We compared our B+-tree HC (Hilbert Curve) indexes with traditional R-tree indexes, which can be used to index 3D points directly. In the experiment, we varied the packet capacity of the wireless broadcast channel from 128 bytes to 2048 bytes. For each packet capacity setting, we assigned appropriate fan out and leaf order parameters for R-trees and B+-trees to ensure that each packet was able to accommodate a complete node of a tree.

Figure 12.8a shows the comparisons of the access latency and Fig. 12.8b shows the comparisons of the tuning time. From the figure, we can identify that the HC-based index outperforms the R-tree based index. The reason for this is twofold: firstly, by using our novel search algorithm, the search space of the HC-based index is smaller than the R-tree-based index; secondly, the size of each index entry for the HC-based index is smaller than the R-tree-based index. This result confirms the effectiveness and efficiency of our search algorithm and HC-based indexing technique.

The percentage of index tuning time is presented in Fig. 12.9a. Here, we define the percentage of index tuning time as the ratio between the index tuning time (caused by downloading necessary parts of the index on air) and the total tuning time (the sum of index tuning time and content tuning time). This metric is a good indicator of the effectiveness and efficiency of an index. The lower the percentage we get, the better is the effectiveness we can achieve. Figure 12.9a shows that the HC-based index has a much lower percentage of index tuning time when compared with the R-tree-based index. To be specific, the percentage of index tuning time of the HC-based index is below 20% under different packet capacities while that of the R-tree-based index is above 60%.

The index sizes are compared in Fig. 12.9b. We can see that the number of packets required to accommodate the whole index for HC-based index is only about half of that for R-tree-based index. The main reason is that the R-tree index must store 3D points in its nodes while the HC-based index only stores one-dimensional points.

[1] http://downloads.dbpedia.org/3.8/en/

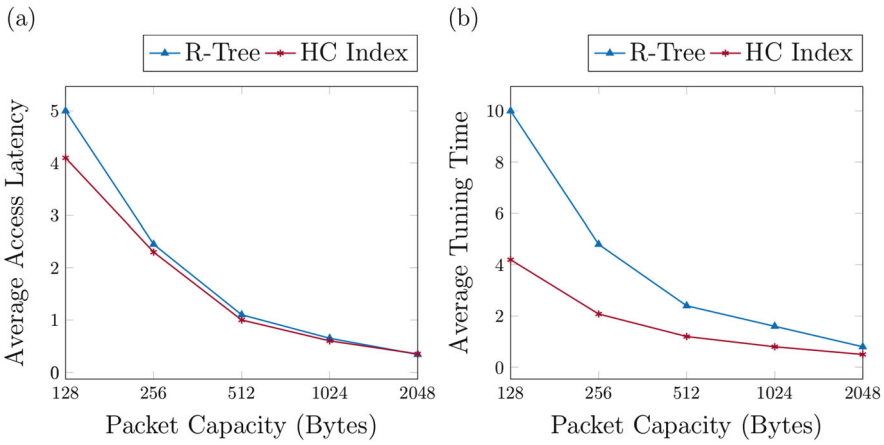

Fig. 12.8 Performance evaluation for (**a**) access latency and (**b**) tuning time

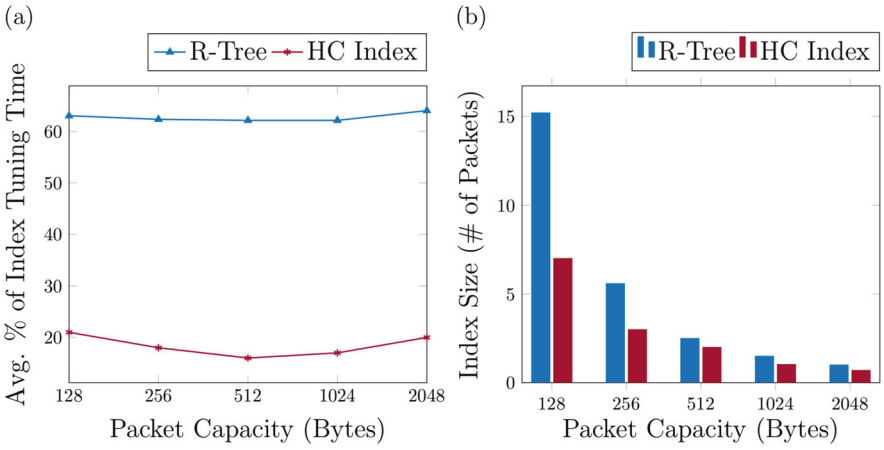

Fig. 12.9 Performance evaluation for (**a**) index tuning time and (**b**) index size

12.7 Related Work

12.7.1 Matching

Recent work on data summaries on linked data [300] transform RDF triples into a numerical space. Then data summaries are built upon numerical data instead of strings, as summarising numbers is more efficient than summarising strings. To transform triples into numbers, hash functions are applied to the individual components (s, p, o) of triples. Thus, a derived triple of numbers can be considered as a 3D point. In this way, a set of RDF triples can be mapped into a set of points in a 3D

space. To facilitate query processing over data summaries, a spatial index named QTree [300], which is evolved from standard R-trees [304], is adopted as the basic index. Data summaries are designed for indexing various linked data sources and used for identifying relevant sources for a given query.

However, data summaries are not suitable for our linked data stream dissemination systems. Firstly, techniques on data summaries, such as QTree, do not consider variables in the BGPs but only RDF triples with concrete strings. Further, since data summaries are concise and imprecise representations of data sources [300], they provide match estimation. Hence, query evaluation on them would return false-negative results, which is not allowed in our system.

Semantic matching has also been studied, which aims to match semantically related RDF triples against BGPs. It may provide false-positive match results but not a false-negative. Both approximate event matching [151] and thematic event processing [272] apply semantic matching. Similarly, all these techniques will return false-negative matching results, which are not allowed in our approach.

Moreover, existing work on pattern matching, such as stream reasoning [299] and linked data stream processing [298], does not support large-scale query evaluation but focuses on the 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.

12.7.2 Wireless Broadcast

In order to support efficient query processing among a large number of data sources, relevant sources that can better answer the queries should be identified first, and then the queries are evaluated on them. Lightweight data summaries on linked data [300] have been investigated to determine relevant sources to use during query evaluation. However, these techniques are mainly designed for random access in memory or on disk, and thus they cannot be applied in a wireless broadcast system directly, where only linear access is allowed.

The work of CkNN (Continuous k Nearest Neighbor) query processing on air [305] has introduced indexes and search algorithms for continuous kNN queries in 2D spaces. It uses a similar technique discussed in this chapter to index moving objects modelled in 2D spaces. Their work differs from this work as their main goal is to support continuous kNN queries in 2D spaces, while the main goal of this chapter is to support queries on linked data broadcast in a wireless channel.

Our work differs from the CkNN query processing in the following aspects: (1) CkNN query processing is for CkNN queries while our work aims at processing Basic Graph Patterns (BGPs) on linked data; (2) the search algorithms are different as query evaluation for CkNN queries, and BGPs is different; (3) the indexing space is different as CkNN query processing only considers 2D space but our work considers 3D space; (4) finally, our work can be extended to support more complex

queries on linked data broadcast in a wireless channel while CkNN query processing aims to handle variants of NN queries.

12.8 Summary and Future Work

In this chapter, we have leveraged semantic technologies, such as linked data, to build an efficient stream dissemination system for semantic IoT within smart environments. We present the design of the stream dissemination support service for a dataspace based on two new data structures to suit the needs of high-performance linked data stream dissemination in (1) point-to-point systems; and (2) wireless broadcast systems.

In order to efficiently match a large number of BGPs against linked data streams, we have proposed TP-automata, an automata-based method designed for efficient pattern matching. In our evaluation, we show that TP-automata can disseminate linked data within the dataspace at the speed of nearly one 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 have proposed an effective and efficient air indexing method, HC-Index, for broadcasting linked data on air, which can be used in data sharing among a large number of mobile, smart objects, and intelligent systems in an IoT-based smart environment. Our method is based on 3D Hilbert curve mappings. Firstly, we map RDF triples into points in a 3D space and then adopt 3D Hilbert curve mappings to convert all the 3D points into one-dimensional points. We build B+-trees upon these one-dimensional points and serialise these trees to accommodate them on the linear wireless channels. An efficient search algorithm has been devised to facilitate query processing over the linked data on air. We have conducted experiments and compared our method with the traditional R-tree-based spatial indexing method. Our method has shown better performance over the R-tree-based method in various aspects, including access latency, tuning time, and index size.

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 [306] that help to map semantically related data together. Both directions will enable the linked data stream dissemination system to provide better semantic richness and to support data consumption needs more accurately, which is a critical issue in the IoT. We will also consider the challenges of matching and dissemination of multimedia events [30] within smart environments.