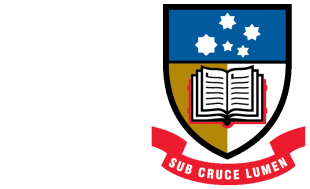
Managing Data Dynamics, Streams and Sharing in the Internet of Things



THE UNIVERSITY *of* ADELAIDE

Yongrui (Louie) Qin

School of Computer Science

The University of Adelaide

This dissertation is submitted for the degree of

Doctor of Philosophy

Supervisors: A/Prof. Michael Sheng, Dr. Nickolas J.G. Falkner and Prof. Hua Wang

August 2015

To my mother and father,

my wife and my little princess,

my brother,

who made all of this possible,

for their endless encouragement and patience.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Yongrui (Louie) Qin

August 2015

Acknowledgements

This thesis would not have been possible without the support and help from some important people in my life. I would like to take the opportunity to thank all those who have helped me during my PhD journey.

First of all, my sincere thanks will go to my principal supervisor Prof. Michael Sheng, who has taught me how to do good research as well as how to be a person with better personality. He has always been patient, passionate, encouraging throughout the whole journey of my PhD study. His many insightful suggestions and comments on my research have significantly improved the work in this thesis.

Second, I am very thankful to my co-supervisors, Dr. Nickolas J.G. Falkner and Prof. Hua Wang. I want to thank Dr. Falkner for his insightful suggestions on improving my research work and drafts of my papers. I want to thank Prof. Wang for his guidance on my research and for his encouragement to do high-quality research.

I would also like to express my gratitude to Dr. Edward Curry at the Insight Centre for Data Analytics at the National University of Ireland, Galway for providing me the opportunity to work with him and his team as a research intern. My internship experience there has significantly contributed to the work in this thesis.

It has been a great pleasure working with the faculty, staff, students at the University of Adelaide, during my PhD study, and I would like to thank them all for such a great graduate school experience.

I express my sincere appreciation to the University of Adelaide, who provided the Adelaide Scholarship International (ASI), to financially support my work in this dissertation.

No need to mention the great love and support from my family. I am thankful to my parents, who have given up so much and worked so hard to earn money for my education and better future. I am forever indebted to them. I would like to thank my lovely younger brother for his company with me during my childhood and primary and secondary education. I sincerely wish that he will be able to escape from schizophrenia soon and live a better life in future. I am also deeply grateful to my wife and little princess for their constant love and support.

Abstract

Recently, the Internet of Things (IoT) has gained momentum in connecting everyday objects to the Internet and facilitating machine-to-human and machine-to-machine communication with the physical world. IoT offers the capability to connect and integrate both digital and physical entities, enabling a whole new class of applications and services.

This thesis firstly reviews the state-of-the-art research efforts in IoT from data-centric perspectives, including data stream processing, data storage models, complex event processing, and searching in IoT by identifying an IoT data taxonomy, which includes ten key data elements of IoT data under three categorizations. In this thesis, we focus ourselves on three aspects of data management in IoT: data dynamics, data velocity, and data incompleteness. More specifically, we study data dynamics in dynamic graphs, handle data velocity in streams, and tackle data incompleteness via sharing.

In IoT, connections and relations between things are universal and highly dynamic. It is natural to model these connections and relations using dynamic graphs. Meanwhile, shortest path computation is one of the most fundamental operations for managing and analyzing graphs. In this thesis, we focus on the problem of computing the shortest path distance in graphs subject to edge failures. We propose SIEF, a Supplemental Index for Edge Failures in a dynamic graph, which is based on distance labeling, to support distance queries in dynamic graphs with edge failures efficiently.

In IoT, one challenging issue is how to disseminate streaming data to relevant consumers efficiently. Semantic technologies aim to facilitate machine-to-machine (M2M) communication and are attracting more and more interest from both academia and industry, especially in the emerging IoT. This thesis leverages semantic technologies, such as Linked Data, which can facilitate M2M communications to build an efficient information dissemination system for semantic IoT. The system integrates Linked Data streams generated from various data collectors and disseminates matched data to relevant data consumers based on triple pattern queries registered in the system by the consumers. We also design new data structures, *TP-automata* and *CTP-automata*, to meet the high performance needs of Linked Data dissemination.

To tackle data incompleteness, we consider large-scale information sharing scenarios among mobile objects in IoT. By leveraging semantic techniques, we propose broadcasting Linked Data on-air to allow simultaneous access to the information and to achieve better scalability. We introduce a novel air indexing method to reduce the information access latency and energy consumption. We also study the data placement problem of periodic XML data broadcast in IoT environments to facilitate data sharing in IoT. Taking advantage of the structured characteristics of XML data, we present a theoretical analysis on the XML data placement on a wireless channel, which forms the basis of our novel data placement algorithm.

This thesis also discusses on-going and emerging IoT applications, and open research issues for processing and managing IoT data. Several representative domains where IoT can make profound changes are explored, and some key directions for future research and development from a data-centric perspective are identified.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

The Internet is a global system of networks interconnecting computers using the standard Internet protocol suite. It has significant impact on the world as it can serve billions of users worldwide. Millions of private, public, academic, business, and government networks, of local to global scope, all contribute to the formation of the Internet. The traditional Internet has a focus on computers and can be called the Internet of Computers. In contrast, evolving from the Internet of Computers, the Internet of Things (IoT) emphasizes things rather than computers [1]. It 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 communication/interactions between these objects. The ultimate goal of IoT is to enable computers to see, hear and sense the real world. It is predicted by Ericsson that the number of Internet-connected things will reach 50 billion by 2020. Electronic devices and systems exist around us providing different services to the people in different situations: at home, at work, in their office, or driving a car on the street [2]. IoT also enables the close relationship between human and opportunistic connection of smart things [3].

"Changes brought about by the Internet will be dwarfed by those prompted by the networking of everyday objects", says a report by the United Nation (UN) [4]. IoT is widely regarded as the number one of top 10 technologies that will change the world in the next 10 years [5]. The National Intelligence Council [6] foresees that "by 2025, Internet nodes may reside in everyday things – food packages, furniture, paper documents, and more. Widespread diffusion of an Internet of Things (IoT) could contribute invaluably to economic development."

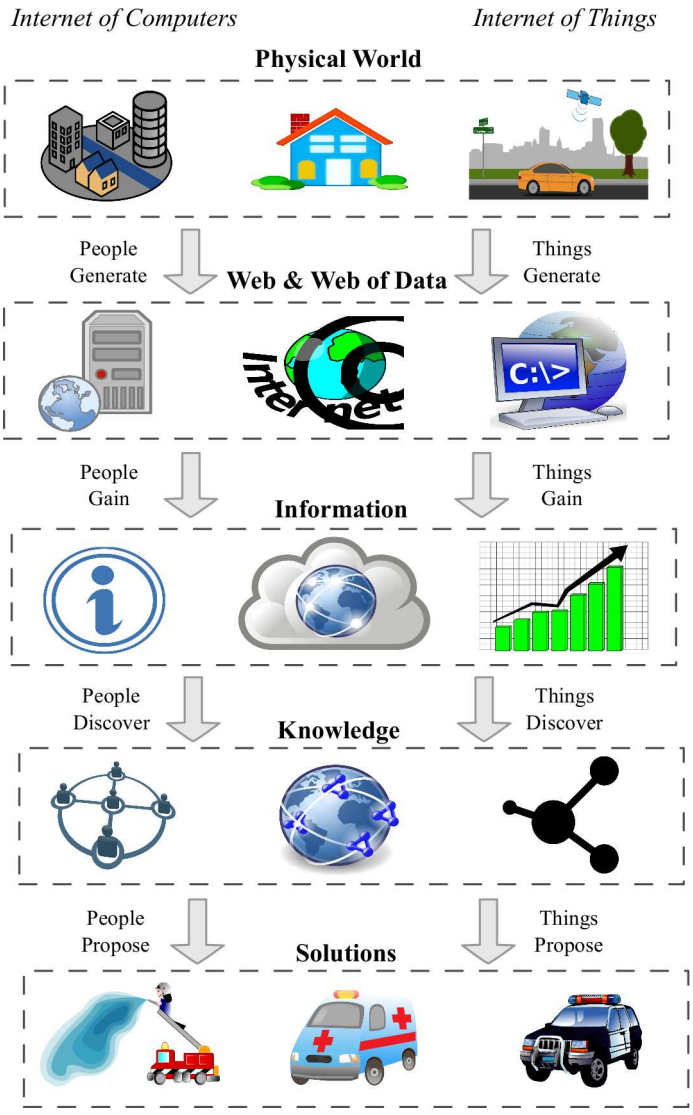


Fig. 1.1 Internet of Computers v.s. Internet of Things

There are several definitions or visions of IoT from different perspectives. From the viewpoint of services provided by things, IoT means “a world where things can automatically communicate to computers and each other providing services to the benefit of the human kind" [7]. From the viewpoint of connectivity, IoT means “from anytime, anyplace connectivity for anyone, we will now have connectivity for anything" [8]. From the viewpoint of communication, IoT refers to “a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols" [9]. Finally, from the viewpoint of networking, IoT is the Internet evolved “from a network of interconnected computers to a network of interconnected objects" [10].

We focus on our study of the Internet of Things from a *data perspective*. As shown in Fig. 1.1, data is processed differently in the Internet of Things and traditional Internet environments (i.e., Internet of Computers). In the Internet of Computers, both the main data producers and consumers are human beings. However, in the Internet of Things, the main actors become *things*, which means things are the majority of data producers and consumers. Therefore, we give our definition of the Internet of Things as follows:

“*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 the Internet of Things technologies will be able to sense and react to the real world for humans.*”

As of 2012, 2.5 quintillion (2.5×10^{18}) bytes of data are created daily[1]. In IoT, connecting all of the things that people care about in the world becomes possible. All these things would be able to produce much more data than nowadays. The volumes of data are vast, the generation speed of data is fast and the data/information space is

[1] http://www-01.ibm.com/software/data/bigdata/

global [2]. Indeed, IoT is one of the major driving forces for *big data analytics*. Given the scale of IoT, topics such as storage, distributed processing, real-time data stream analytics, and event processing are all critical, and we may need to revisit these areas to improve upon existing technologies for applications of this scale.

1.2 Research Scope and Contributions

This thesis presents research on data management in the context of IoT from three aspects, including data dynamics, data velocity, and data incompleteness. More specifically, we study data dynamics in dynamic graphs, handle data velocity in streams, and tackle data incompleteness via sharing. The main contributions of this thesis can be summarized in the following.

Survey on State-of-the-Art Data Techniques for IoT

This thesis reviews the state-of-the-art research efforts in IoT from data-centric perspectives, including data stream processing, data storage models, complex event processing, and searching in IoT by identifying an IoT data taxonomy, which includes ten key data elements of IoT data under three categorizations. This thesis also discusses on-going and emerging IoT applications, and open research issues for processing and managing IoT data. Several representative domains where IoT can make profound changes are explored, and some key directions for future research and development from a data-centric perspective are identified.

Efficiently Answering Distance Queries in a Graph with Edge Failures

In IoT, connections and relations between things are universal and highly dynamic. It is natural to model these connections and relations using dynamic graphs. Meanwhile, shortest path computation is one of the most fundamental operations for managing

and analyzing graphs. In this thesis, we focus on the problem of computing the shortest path distance in graphs subject to edge failures. We propose SIEF, a Supplemental Index for Edge Failures in a dynamic graph, which is based on distance labeling, to support distance queries in dynamic graphs with edge failures efficiently.

Disseminating Linked Data Streams in the Context of IoT

This thesis leverages semantic technologies, such as Linked Data, which can facilitate machine-to-machine (M2M) communications to build an efficient information dissemination system for semantic IoT. The system integrates Linked Data streams generated from various data collectors and disseminates matched data to relevant data consumers based on triple pattern queries registered in the system by the consumers. In thesis, we also designs new data structures, *TP-automata* and *CTP-automata*, to meet the high performance needs of Linked Data dissemination.

Broadcasting Linked Data in IoT to Achieve Data Sharing

To tackle data incompleteness, this thesis studies large-scale information sharing scenarios among mobile objects in IoT. By leveraging semantic techniques, this thesis proposes to broadcast Linked Data on-air to allow simultaneous access to the information and to achieve better scalability. Specifically, this thesis introduces a novel air indexing method to reduce the information access latency and energy consumption.

Broadcasting XML Data in IoT to Achieve Data Sharing

In the past decades, XML has become prevalent in ubiquitous and mobile computing devices and applications around the world. The sharing and dissemination of XML data is more and more important in the IoT era where the number of smart devices is unprecedented. This thesis investigates the data placement problem of periodic XML data broadcast in IoT environments to facilitate data sharing in IoT. Taking advantage

of the structured characteristics of XML data, this thesis presents a theoretical analysis on the XML data placement on a wireless channel, which forms the basis of the proposed novel data placement algorithm.

1.3 List of Publications

During my PhD study, I have published/produced 20 publications. The following eight publications are the core publications resulting from this thesis. I am very grateful to all of the people who collaborated with me for these publications. Their comments and suggestions were very helpful and insightful. I would also like to thank anonymous reviewers for their valuable comments on earlier drafts of these publications.

1. **Yongrui Qin**, Quan Z. Sheng, Muntazir Mehdi, Hua Wang, Dong Xie, "Effectively Delivering XML Information in Periodic Broadcast Environments", In *Proceedings of the 24th International Conference on Database and Expert Systems Applications (DEXA)*, Pages 165–179, 2013.

2. **Yongrui Qin**, Quan Z. Sheng, Nickolas J.G. Falkner, Ali Shemshadi, Edward Curry, "Towards Efficient Dissemination of Linked Data in the Internet of Things", In *Proceedings of the 23rd ACM Conference on Information and Knowledge Management (CIKM)*, pages 1779–1782, 2014.

3. **Yongrui Qin**, Quan Z. Sheng, Nickolas J.G. Falkner, Wei Emma Zhang, Hua Wang, "Indexing Linked Data in a Wireless Broadcast System with 3D Hilbert Space-Filling Curves", In *Proceedings of the 23rd ACM Conference on Information and Knowledge Management (CIKM)*, pages 1775–1778, 2014.

4. **Yongrui Qin**, Quan Z. Sheng, Wei Emma Zhang, "SIEF: Efficiently Answering Distance Queries for Failure Prone Graphs", In *Proceedings of the 18th*

International Conference on Extending Database Technology (EDBT), pages 145–156, 2015.

5. **Yongrui Qin**, Quan Z. Sheng, Nickolas J.G. Falkner, Ali Shemshadi, Edward Curry, "Batch Matching of Conjunctive Triple Patterns over Linked Data Streams in the Internet of Things", In *Proceedings of the 27th International Conference on Scientific and Statistical Database Management (SSDBM)*, Article No. 41, 6 pages, 2015.

6. **Yongrui Qin**, Quan Z. Sheng, Edward Curry, "Matching Over Linked Data Streams in the Internet of Things", *IEEE Internet Computing 19(3): 21-27 (2015)*.

7. **Yongrui Qin**, Quan Z. Sheng, Nickolas J.G. Falkner, Schahram Dustdar, Hua Wang, Athanasios V. Vasilakos, "When Things Matter: A Data-Centric View of the Internet of Things", submitted to *Journal of Network and Computer Applications (JNCA)* (currently under revision).

8. **Yongrui Qin**, Quan Z. Sheng, Hua Wang, Nickolas J.G. Falkner, "Organizing XML Data in a Wireless Broadcast System by Exploiting Structural Similarity", submitted to *Journal of Network and Computer Applications (JNCA)* (currently under revision).

1.4 Thesis Organization

The remainder of the thesis is organized as follows.

Chapter 2 systematically reviews the key technologies related to the development of IoT and its applications, particularly from a data-centric perspective. The aim of this chapter is to provide a better understanding of the current research activities related to data management in IoT. Specifically, this chapter reviews and compares

technologies including data streams, data storage models, searching, and event processing technologies, which play a vital role in enabling the vision of IoT.

Chapter 3 focuses on the problem of computing the shortest path distance in graphs subject to edge failures. SIEF is proposed, which is a Supplemental Index based on distance labeling for Edge Failures in a dynamic graph. This chapter firstly introduces concepts related to 2-hop distance labeling and then exploits properties of distance labeling in static graphs to compute very compact distance labeling for answering distance queries in dynamic graphs.

Chapter 4 leverages semantic technologies, such as Linked Data, to build an efficient information dissemination system for semantic IoT. The system integrates Linked Data streams generated from various data collectors and disseminates matched data to relevant data consumers based on triple pattern queries registered in the system by the consumers. Two new data structures are designed, namely *TP-automata* and *CTP-automata*, to meet the high performance needs of Linked Data dissemination.

Chapter 5 describes how to broadcast Linked Data on-air using RDF format to allow simultaneous access to the information and to achieve better scalability. A novel air indexing method based on 3D Hilbert curves is introduced to reduce the information access latency and energy consumption. A novel search algorithm is also designed to efficiently evaluate queries against the air indexes.

Chapter 6 takes advantage of the structured characteristics of XML data and proposes algorithms to effectively broadcast XML data in a periodic broadcast environment. This chapter provides theoretical analysis on the XML data placement on a wireless channel, which forms the basis of the novel data placement algorithm.

Chapter 7 describes potential IoT applications from several representative application domains. Open research issues are also discussed in this chapter.

Chapter 8 concludes this thesis and discusses several possible directions for future work.

Chapter 2

Literature Review

This chapter focuses on reviews of the main techniques and state-of-the-art research efforts in IoT from data-centric perspectives, including data stream processing, data storage models, complex event processing, and searching in IoT. It is organized as follows. Section 2.1 identifies a taxonomy that contains ten key elements of IoT data. Section 2.2 reviews the data streaming techniques and Section 2.3 focuses on the data models and storage technologies for IoT. Search and event processing technologies are discussed in Sections 2.4 and 2.5, respectively.

2.1 IoT Data Taxonomy

In this section, we identify the intrinsic characteristics of IoT data and classify them into three categories, including *Data Generation*, *Data Quality*, and *Data Interoperability*. We also identify specific characteristics of each category, and the overall IoT data taxonomy is shown in Fig. 2.1.

2.1.1 Data Generation

- **Velocity**. In IoT, data can be generated at different rates. For example, for GPS-enabled moving vehicles in road networks, the GPS signal sampling frequency

could be every few seconds, every few minutes, or even every half an hour. But some sensors can scan at a rate up to 1,000,000 sensing elements per second[1]. On one hand, it is challenging to handle very high sampling rates, which require efficient processing of the fast generated data. On the other hand, it is challenging to deal with low sampling rates, due to the fact that some important information may be lost for data processing and decision making.

- **Scalability**. Since things are able to continuously generate data together with the foreseeable excessively large number of things, the IoT data is expected to be at an extremely large scale. It is easy to imagine that, in IoT data processing systems, scalability will be a long standing issue, aligning with the current Big Data trend.

- **Dynamics**. There are many dynamic elements within IoT data. Firstly, many things are mobile, which will lead to different locations at different times. Since they will move to different environments, the sensing results of things will be changing to reflect the real world. Secondly, many things are fragile. This means the generated data will change over time due to the failure of things. Thirdly, the connections between things could be intermittent. This also creates dynamics in any IoT data processing system.

- **Heterogeneity**. There will be many kinds of things potentially connecting to the Internet in the future, ranging from cars, robots, fridges, mobile phones, to shoes, plants, watches, and so on. These kinds of things will generate data in different formats using different vocabularies. In addition, there will be assorted IoT data processing systems, which will provide data in customized formats to tailor different data needs.

[1] https://www.tekscan.com/support/faqs/what-are-sensors-sampling-rates

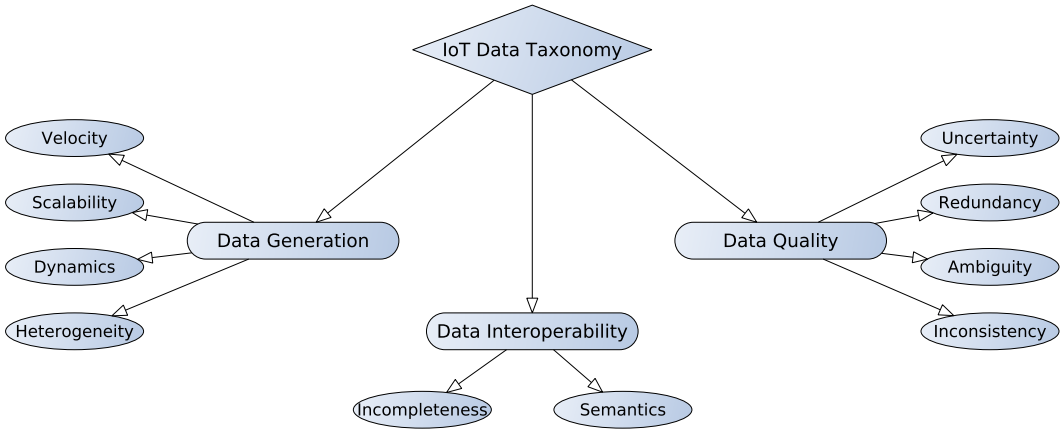


Fig. 2.1 IoT Data Taxonomy

2.1.2 Data Quality

- **Uncertainty**. In IoT, uncertainty may come from different sources. In RFID data, the uncertainty can refer to missing readings, readings of non-existing IDs, etc. In wireless sensor networks, uncertainty can refer to sensing precision (the degree of reproducibility of a measurement), or accuracy (the maximum difference that will exist between the actual value and the indicated value), etc.

- **Redundancy**. Redundancy can be easily observable in IoT. For example, in RFID data, the same tags can be read multiple times at the same location (because multiple RFID readers exist at the spot or tags are read multiple times at in the same spot) or at different locations. In wireless sensor networks, a group of sensors of the same type may be deployed in a nearby area, which can produce similar sensing results of that area. For the same sensor, due to the possible high sampling rates, redundant sensing data can be produced.

- **Ambiguity**. Dealing with a large amount of ambiguity in IoT data is inevitable. The data produced by assorted things can be interpreted in different ways due to different data needs from different things or other data consumers. Such data can be useful and important to any other kind of things, which brings about

the challenges of proper interpretation of the produced data to different data consumers.

- **Inconsistency**. Inconsistency is prevalent in IoT data. For example, in RFID data, inconsistency can occur due to missing readings of tags at some locations along the supply chain. It is easy to observe inconsistency in sensing data as when multiple sensors are monitoring the same environment and reporting sensing results. Due to the precision and accuracy of the sensing process and other problems including packet loss during transmission, data inconsistency is also an intrinsic characteristics in sensing data.

2.1.3 Data Interoperability

- **Incompleteness**. In order to process IoT data, being able to detect and react to events in real-time, it is important to combine data from different types of data sources to build a big and complete picture of relevant backgrounds of the real world. However, as this process relies on the cooperation of mobile and distributed things who are generating relevant background data, incompleteness is easily observable in IoT data. Suppose there are a large number of available data sources. it is of great importance to determine which data sources can best address the incompleteness of data for a given data processing task.

- **Semantics**. To address the challenges posed by the deluge of IoT data, things, or machines acting as the major data consumers should be a promising trend for data processing in the IoT era. Inspired by Semantic Web technologies, in order to enable machines to understand data for human beings, injecting semantics into data could be an initial step. Therefore, semantics within IoT data will play an important role in the process of enabling things/machines to understand and process IoT data by themselves.

2.2 Data Streams

A data stream is a sequence of data objects, of which the number is potentially *unbounded*. A data stream may be continuously generated at a rapid rate. In the data stream, each data object can be described by a multi-dimensional attribute vector within a continuous, categorical, or mixed attribute space [11]. There are some typical characteristics of data streams:

- Continuous arrival of data objects
- Disordered arrival of data objects
- Potentially unbounded size of a stream
- Normally no persistence of data objects after being processed
- Changing probability distributions of the unknown data generation process

Due to the excessive amount of data produced by all kinds of things in the era of IoT, data streams play an important role in data processing and analysis. This section will focus on related data stream research efforts that can help handle IoT data. Our discussions include general data stream processing, RFID data stream processing, and RDF triple stream processing.

2.2.1 General Data Stream Processing

Data streams can be generated in various scenarios, including a network of sensor nodes, a stock market or a network monitoring system and so on. In many scenarios such as the sensor network scenario, sensor nodes are normally powered by batteries or solar panels. Therefore, in typical a sensor data processing system, one of the challenging issues is power constraints. In most applications, communication across sensor networks or with a centralized server requires the largest amount of energy as

sensing consumes less energy [12]. If sensor nodes send their raw sensing data to a server without consideration of the amount of energy needed to communicate, the battery life of the sensor nodes could be drastically reduced. Consequently, sensor data processing techniques, including data aggregation, data compression, modeling and online querying, should be performed *on-site* or *in-network* to reduce communication cost [12]. Furthermore, numerous demands on efficient data processing algorithms for sensor systems arise due to the limitations of computational power of sensor nodes as well as the existence of inaccuracy and bias in the sensor readings. In other scenarios, such as stock market and network monitoring systems, there also exist challenges in processing high-rate data streams.

Query Processing

There are several important queries to be considered [12]:

- *Aggregate Queries*. Aggregate queries are an important class of queries in sensor systems, including `MIN`, `COUNT` and `AVG` operators. Various techniques have been proposed to efficiently process these aggregate operators in sensor systems, which can help to effectively reduce power consumption. Considering the properties of the aggregate functions, the in-network partial data could be preprocessed first, which can then be utilized to produce the final results for the issued queries.

- *Join Queries*. An example of join queries is *"Return the objects that were detected in both regions R1 and R2"* [12]. To evaluate the query, stream readings from the sensors in regions R1 and R2 should be joined first before we can determine whether an object was detected in the two designated regions. Join queries are useful in many applications, such as monitoring an environment where multiple sensors are deployed, tracking moving objects that are monitored by several types of sensors, etc.

- *Top-k Monitoring*. The general problem of monitoring top-*k* values from distributed data streams is investigated in [13]. A technique is proposed to ensure the validity of the most recently communicated top-*k* answers by maintaining some specified arithmetic constraints at the stream sources. User specified error tolerance is also considered in order to provide high-quality answers. This technique can help reduce the overall communication cost between different sources.

- *Continuous Queries*. To monitor designated changes in an environment, sensors are typically required to answer queries in continuous manner. For instance, motion or sound sensors might be used to evaluate some continuous queries, such as “Turn lights off if no motion is detected in area A in the past 10 minutes". When the query constraints are satisfied, the action of turning lights off could be automatically triggered by these sensors. If there is more than one continuous query evaluated over the same sensor readings, the storage and computation can be optimized by exploiting the fact that the sources of the queries and their partial results could overlap [12].

In IoT, query processing over streaming data will need to focus more on IoT related aspects of the streaming data, such as uncertainty, ambiguity and inconsistency. Furthermore, it is also imperative to address issues related to velocity and heterogeneity. We list a few examples of issues in the following.

- How can one efficiently aggregate the information stream from more than one thing with a large amount of ambiguity and inconsistency?

- How can one perform accurate join queries over uncertain IoT data streams with ambiguous and incomplete information?

- How can one identify top-*k* values from millions of heterogeneous IoT data streams efficiently and effectively?

- How can one monitor changes based on continuous queries from a large number of dynamic, fast, heterogeneous and incomplete IoT data streams, etc?

- New types of queries may also need to be considered, such as source selection queries for overcoming data incompleteness, and so on.

Stream Mining

Stream mining can extract useful rules/information from data streams. Some typical tasks for stream mining are listed in the following:

- *Clustering*. Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). Clustering techniques for data streams typically continuously cluster objects on memory constrained devices with some time limitations. Due to these restrictions, there are some requirements to consider when designing algorithms for clustering data streams [14]: (i) providing clustering results via fast and incremental processing of data objects; (ii) rapidly detecting new clusters or changes of existing clusters; (iii) scaling to the potentially unbounded number of objects in data streams; (iv) providing a model representation that is consistently compact regardless the number of data objects; (v) rapidly detecting the presence of outliers and acting accordingly; and (vi) dealing with different data types, such as XML trees, DNA sequences, GPS temporal and spatial information.

- *Classification*. Classification uses prior knowledge to guide the partitioning process to construct a set of classifiers to represent the possible distribution of patterns [15]. Basically, compared with clustering, classification is a supervised learning process whereas clustering is an unsupervised learning process. More formally, a typical classification algorithm can be defined as follows [15]: given

a predefined classifier and two sets of data, labeled data and unlabeled data, the labeled data is used to train the classifier and the unlabeled data can then be classified by the trained classifier.

- *Outlier and Anomaly Detection*. In outlier and anomaly detection, the main task is to find data points that are most different from the remaining points in a given data set. Most existing outlier detection algorithms are based on the distance between every pair of points. The points that are most distant from all other points will be marked as outliers [16]. To be more specific, an object O in a dataset T is a $DB(p,D)$-outlier (DB here refers to distance-based) if at least fraction p of the objects in T lies greater than distance D from O. This kind of algorithms suffer from the same performance issue as they all run in $O(n^2)$ time. Hence, it is difficult to extend such approaches to distributed streaming data sets because points in those data sets normally arrive at multiple distributed end-points and must be processed incrementally.

- *Frequent Itemset Mining*. Frequent itemset mining is to find sets of items or values that co-occur frequently, or in other words, to find co-occurrence relationships in a transactional data set. Here a transactional data set refers to a data set where a set of items appear together in some specified context. Given a predefined support s, the goal in frequent itemset mining is to find all subsets of items that occur at least s number of times, or in other words, that appear in at least s transactional data sets at hand. Frequent itemset mining is both CPU and I/O intensive. Therefore, it is costly to completely re-mine a dynamic data set, which will be a typical case in IoT.

In IoT, multiple data streams processing would be more preferable as data streams can be generated at anywhere around the world and can be accessed globally via the Internet if being made public. For example, SmartSantander[2] proposes a city-scale ex-

[2]http://www.smartsantander.eu/

perimental research facility in support of typical applications and services for a smart city. Around 20,000 sensors have been deployed to provide a variety of services, such as static environmental monitoring, mobile environmental monitoring, parks and gardens irrigation, outdoor parking area management, guidance to free parking lots and traffic intensity monitoring. A large number of data streams have to be processed efficiently to provide real-time monitoring of a smart city. Furthermore, how to efficiently and effectively mine IoT data streams that are highly dynamic, heterogeneous, uncertain, ambiguous, inconsistent, and incomplete will also require a revisit of the existing streaming mining techniques.

2.2.2 RFID Data Stream Processing

In 2003, a nonprofit open forum called the Ubiquitous ID Center[3] was established. So far, more than 500 companies and organizations worldwide have contributed to it, publishing uID standards and industrial open standard specifications. uID standards are based on the uID architecture [17], which identifies real-world entities via Radio-Frequency Identification (RFID) tags or barcodes, determines contextual information such as environmental parameters from networked sensors, and adapts information services according to the data it obtains.

RFID systems consist of radio frequency (RF) tags (also called transponders) and RF tag readers (also called transceivers). Readers may be able to both read data from and write data to a transponder. RFID is an established electronic identification technology that enables real-time monitoring and tracking applications in a variety of domains. Object identification information is stored on an RFID tag. This could be an Electronic Product Code (EPC)[4]. EPC is a unique item identification code, which normally contains information about the manufacturer, the type of item and the serial number of the item (the tag ID). Streams of RFID reading data, whose basic form is

[3]uID Center: www.uidcenter.org
[4]http://www.epc-rfid.info/

a triplet $< tag\_id; reader\_id; timestamp >$, raise new challenges since the data may be insufficient, incomplete, and voluminous [18].

In the past decade, the Auto-ID Center, now which is called the Auto-ID Labs[5], has attracted industrial interests from companies and government initiatives to advance new developments and interests in RFID technology. One of the important advances is the so-called “Networked RFID" [19]. Networked RFID aims at connecting isolated RFID systems and software via the Internet. The EPcglobal Network, initially designed by the Auto-ID Labs and then further developed by EPCglobal at GS1[6], is one of the notable efforts for Networked RFID.

In the following, we review some major RFID data stream processing techniques and summarize them in Table 2.1.

RFID Data Cleaning (Uncertainty and Unreliability)

SMURF (Statistical sMoothing for Unreliable RFid data) [20] is the first declarative, adaptive smoothing filter for cleaning raw RFID data streams. Unlike conventional techniques which expose the smoothing window parameter to the application, SMURF adapts the window size automatically and continuously over the lifetime of the system based on observed readings.

Periods of dropped readings and periods when a tag has moved are difficult to distinguish, which poses some challenges for the design of SMURF. To overcome such difficulty, a statistical sampling-based approach is put forward in SMURF. The main motivation is that RFID data streams can be modeled as a random sample of the tags in a reader’s detection range. This sample-based view of observed RFID readings enables SMURF to develop algorithms based on statistical sampling theory to adapt the window size effectively.

[5]http://www.autoidlabs.org
[6]http://www.gs1.org/

Basically, the false reads in RFID streams can be classified into two categories [21]:

- *Missing-Reads*. Though an RFID tag is located in the range of a reader, it might not be read at all, thereby leading to a false prediction that the tag is not present. This may be caused by the weakness of RF signal, shortage of power, shield of signal between the tag and the reader, and the collision between tags. This type of errors is also referred as false negative.

- *Cross-Reads*. When an RFID tag locates outside the range of a reader, but it might be captured by this reader which leads another false prediction that the object is present in the scope of this reader (sometimes called *ghost reading*). Cross-reads may be arisen by the reflection of metal items, the abrupt strengthen of RF, and the change of antenna directions. This type of errors is also called false positive.

SMURF cannot eliminate the cross-reads generated by physical factors. A kernel density-based probability cleaning method, called KLEAP, can be used to filter the cross-reads in RFID data streams [21]. KLEAP considers cross-reads as outliers, thus, the determination of cross-reads is transformed into the issue of detecting outliers on data streams. The density-based methods often perform better than the distance-based one, so KLEAP applies the density-based methods to detect cross-reads. It detects the exact positions of tags over the RFID data streams through examining the kernel densities of each tag captured by multiple readers.

The knowledge on the map of the real world and on the motility characteristics (such as the maximum speed) of the monitored objects is exploited by [22]. From this knowledge of the domain, constraints can be naturally derived on the connectivity between pairs of locations (direct unreachability constraints) and/or on the time needed for reaching a location starting from another one (traveling-time constraints). These constraints can be used to discard interpretations of the data corresponding to incon-

sistent trajectories. Then a graph is built in the following way: its nodes correspond to pairs $< location, timestamp >$ and inside the graph, paths from source to target nodes one-to-one correspond to the valid trajectories in real word. Each node or edge is assigned a probability obtained by revising the a priori probability of the corresponding pair $< location, timestamp >$, so that the overall probability of a source-to-target path is the conditioned probability of the corresponding trajectory. In this way, trajectories of RFID-monitored objects can be cleaned.

RFID Data Inference and Compression

RFID data inference techniques are closely related to RFID data cleaning techniques because inference techniques will need to clean RFID data first and then they can infer to the high level information about the tagged objects, i.e., location and containment relationships. Since raw RFID data contains a large amount of redundancies, RFID data compression is also applied to reduce space requirements after inference results have been obtained. RFID data compression is a further step beyond inference, where compression is performed based on the results of inference to remove the redundant data.

Noisy, raw data streams from mobile RFID readers are considered and a probabilistic approach to translate these streams into clean, rich event streams with location information is empoyed by [23]. Their probabilistic model is built based on the mobility of the reader, object dynamics, and noisy readings. *Particle filtering* is used to infer clean information about object locations from raw streams captured from mobile RFID readers.

The aforementioned data cleaning and inference techniques focus on smoothing over time, where containment relationships are not considered. Containment refers to *inter-object* relationships, e.g., containment between objects, cases, and pallets. Containment queries can be useful for enforcing packaging and shipping regulations.

Table 2.1 Comparisons of RFID Streaming Techniques.

Approach	Vel	Sca	Dyn	Het	Unc	Red	Amb	Incs	Incp	Sem
SMURF [20]			√		√					
KLEAP [21]			√		√			√		
Mobility-Monitoring [22]			√		√	√			√	√
Mobile-RFID-Data-Cleaning [23]			√		√	√			√	
RFID-Data-Cleaning (Containment) [24, 25]			√		√	√			√	√

Note: Key elements that have been considered.

Vel: Velocity	Sca: Scalability	Dyn: Dynamics	Het: Heterogeneity
Unc: Uncertainty	Red: Redundancy	Amb: Ambiguity	Incs: Inconsistency
	Incp: Incompleteness	Sem: Semantics	

Some examples of containment queries have been provided by [24], such as “raise an alert if a flammable item is not packed in a fireproof case" or “verify that the food containing peanuts is never exposed to other food cases for more than an hour". They also observe that some known containment relations can be used to determine object locations by smoothing over these facts. For example, suppose that we can infer that a specific set of objects have been packed in the same container. According to such knowledge, if one object in the container is read, all of the other objects must be in the same place. However, the fact is that the containment relationships are not known in advance. Therefore, a graphic model is proposed to infer containment relationships and to detect changes in containment relationships [24, 25].

2.2.3 RDF Triple Stream Processing

Linked Data is a method for publishing structured data and interlink such data to make it more useful. It builds upon standard Web technologies such as HTTP, RDF and URIs and extends these technologies to share information. Linked Data is format that can be used to encode data. Computers can understand the encoded data generated

by things. Therefore, data from different sources can be connected and queried in the form of Linked Data. Basically, Linked Data refers to a set of best practices to be followed in order to publish and link data on the Web, using the following basic principles[7]:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using appropriate standards (RDF, SPARQL).
- Include links to other URIs, so that more things can be discovered.

The concept of *Linked Stream Data* applies the Linked Data principles to streaming data, so that data streams can be published as part of the Web of Linked Data. Stream reasoning can provide the abstractions, foundations, methods and tools required to integrate data streams, the Semantic Web and reasoning systems. Substantial research efforts have been put forward, focusing on how to apply reasoning on streaming data, how to publish raw streaming data and connect them to the existing data sets on the Semantic Web, and how to extend the SPARQL query language to process streaming data [26]. These research efforts lay some foundations of semantic IoT technologies, facilitating machine-to-machine communication in IoT.

Linked Stream Processing and Reasoning

Efforts to apply the linked data principles to stream (sensor) data have been initiated and this wealth of information could be easily included in the Linked Data cloud[8].

There are three typical streaming RDF/SPARQLS engines, including Streaming SPARQL [27], SPARQLStream [28], C-SPARQL [29], and EP-SPARQL [30]. Each

[7] http://www.w3.org/DesignIssues/LinkedData.html
[8] http://linkeddata.org/

Table 2.2 Comparisons of Linked Stream Processing and Reasoning.

Approach	Native	Aggregation Support	Reasoning Support	SPARQL 1.1 Support
Streaming SPARQL [27]	No	Limited	Limited	Limited
SPARQLStream [28]	No	Limited	Limited	Limited
C-SPARQL [29]	No	Rich	Limited	Limited
EP-SPARQL [30]	No	Limited	Rich	Limited
CQELS [31]	Yes	Limited	Limited	Limited

of these systems also proposes its own SPARQL extension for streaming data processing. In these studies, SPARQL has been extended to have sliding window operators for RDF stream processing.

For example, Streaming SPARQL extends SPARQL to support window operators. But it does not consider performance issues, specially when designing the data structures. Further, it does not consider the sharing of computing states for continuous execution. Another example is SPARQLStream, which aims at enabling ontology-based access to streaming data. It defines a SPARQLStream language, which can be translated into another relational stream language based on mapping rules.

C-SPARQL (Continuous SPARQL) [29] attempts to facilitate reasoning upon rapidly changing information. In C-SPARQL, continuous queries are divided into static and dynamic parts and streaming data is transformed into non-streaming data within a specified window in order to apply standard algebraic operations, such as aggregate functions like `COUNT`, `COUNT DISTINCT`, `MAX`, `MIN` and `AVG`. The static parts will be loaded into relations, and the continuous queries are executed by processing the stream data against these relations. Event Processing SPARQL (EP-SPARQL), a language to describe event processing and stream reasoning, can be translated to ETALIS [30], a Prolog-based complex event processing framework.

Different from the above approaches, CQELS [31] is a native streaming RDF/SPARQL system built from scratch. CQELS defines and implements a native processing model

in the query engine. Its query execution framework can also dynamically adapt the query processor to changes in the input data. By using data encoding and caching of intermediate query results, CQELS reduces external disk access on large Linked Data collections. Some indexing techniques are also adopted to enable faster data access. Table 2.2 compares all these systems from various aspects.

Extracting RDF Triples from Unstructured Data Streams

Although the current Linked Open Data (LOD) cloud has tremendously grown over the last few years, it delivers mostly encyclopedic information (such as albums, places, kings, etc.) and fails to provide up-to-date information [32]. Based on such observation, they develop RdfLiveNews, an approach that allows extracting RDF from unstructured (i.e., textual) data streams in a fashion similar to the live versions of the DBpedia[9] and LinkedGeoData[10] datasets. RdfLiveNews takes unstructured data streams as its input. It firstly removes duplicates in the streams. Then it uses the cleaned streams as a basis to extract patterns for relations between known resources. Next, the patterns will be clustered to labeled relations and finally will be used as a basis for generating RDF triples.

2.3 Data Storage Models

The nature of data produced by the Internet of Things calls for a revisit of data storage techniques, which will be further discussed in this section.

2.3.1 New Architecture

Traditional Database Management Systems (DBMSs) employ record-oriented (i.e., a record is represented by a row in a relational table) storage systems. With this row

[9] http://live.dbpedia.org/sparql
[10] http://live.linkedgeodata.org/sparql

store architecture, a single disk write is able to store a single record with multiple attributes to disk. Records writes and updates are normally of high performance in these systems. Therefore, a DBMS with a row store architecture can be called a *write-optimized* system. In contrast, some systems need to deal with ad-hoc querying of large amounts of data, where read performance is of more importance. For such systems, *read-optimized* is the major design factor. Take data warehouses as an example. They represent one class of read-optimized system. In these read-optimized systems, a column-store architecture is a better choice. This is because in a column-store system, the values for each single column (or attribute) are stored contiguously, which can be easily optimized for high-performance querying.

C-Store, a column-store architecture that supports the standard relational logical data model, has been designed by [33]. Compared with the traditional DMBS architecture, the major differences are: (i) data in C-Store is not physically stored using its related relational logical data model; and (ii) whereas most row stores implement physical tables directly and then add various indexes to speed access, C-Store implements only projections. Here, projections are sorted subsets of the attributes of a table. Furthermore, superior performance of column store based systems has been shown over the major RDBMS (relational DBMS) system [34]. It is experimentally demonstrated that specialized engines in the data warehouse, stream processing, text, and scientific database markets can speed up the querying performance by 1-2 orders of magnitude using the column-store architecture. They also suggest that the DBMS vendors (and the research community) should start from scratch and design novel systems for requirements to be fulfilled in the near future, rather than just adapting current systems for those new requirements.

2.3.2 Large-Scale Storage in Distributed Environments

Storage issues in large scale systems have arisen due to the arrival of the big data era. For example, users of websites such as Facebook, Ebay and Yahoo! usually demand fast response times. One solution for this is to replicate data across globally distributed datacenters. However, it is discovered that to replicate all data to all locations may waste huge amounts of resources since users from different locations may have different data consumption needs [35]. For example, a European server may not need to maintain a replica of some rare accessed records in an Asian server. By exploiting such observations, [35] propose a selective replica strategy which supports replica of tables in the Web databases at record level to alleviate the overly-replicated issue. In the selective replica strategy, each replica location stores a full or partial copy of the replicated table depending on the data needs. Specifically, in each location, a given record is stored either as a full replica or as a stub. A full replica is a normal copy of the record and possibly some metadata for supporting the selective replica strategy while a stub contains only the record's primary key and metadata. In this way, since large-scale Web databases are selectively replicated on a record-by-record basis, bandwidth and disk costs can be saved.

Therefore, to meet the exceptional demands of data storage in IoT, developments of large-scale, distributed storage systems are of essential. There are three factors or requirements to be considered when designing a distributed storage system [36]:

- *Consistency:* Consistency means ensuring that multiple copies of the same data are identical since server failures and parallel storage may cause inconsistency.

- *Availability:* Availability refers to the requirement that the entire distributed storage system (which contains multiple servers) should not be seriously affected by some extent of server failures and should be able to provide satisfactory reading and writing performance.

Table 2.3 Comparisons of three types of distributed storage systems.

Type	Pros	Cons	Representatives
CA	Single copy of data; Consistency is easily ensured; Availability is assured by the excellent design of databases	Could not handle network failures	Traditional small-scale relational databases
CP	Maintain several copies of the same data; A certain level of fault tolerance is ensured; Consistency is ensured by guaranteeing multiple copies of data to be identical	Could not ensure sound availability due to the high cost for consistency assurance	BigTable [38]; Hbase [39]
AP	Maintain several copies of the same data; A certain level of fault tolerance is ensured; Availability is assured by the design of distributed storage systems	Strong consistency is not ensured; May cause a certain amount of data errors	Dynamo [40]; Cassandra [41]

- *Partition Tolerance:* Since multiple servers are interconnected by a network and the data is partitioned across the network, the distributed storage system should have a certain level of tolerance to problems caused by network failures. This refers to partition tolerance requirement.

Interestingly, it has been proven by [37] that a distributed storage system could not simultaneously meet the requirements on consistency, availability, and partition tolerance, and at most *two of the three requirements* can be satisfied at the same time. On top of this theory, there are three types of distributed storage systems: (1) a CA system, which ignores partition tolerance; (2) a CP system, which ignores availability; and (3) an AP system, which ignores consistency. The comparisons of these systems and some of their representative works are summarized in Table 2.3.

2.3.3 Storage on Resource-Constrained Devices

Storage issues also arise in resource-constrained scenarios in IoT. For example, in sensor networks, communication activity normally plays a more important role than storage. But it is argued that for batch data collection, delay-tolerant mobile applications, and disconnected operations in static networks, the storage-centric paradigm becomes more critical [42]. It is favored by decreasing costs and increasing capacity of storage hardware. SQUIRREL is also proposed in the same work, which is a lightweight run-time layer allocating data to different storage areas, based on data size versus energy trade-offs.

SolarStore, a power storage service for solar-powered storage-centric sensor networks has been developed by [43]. The main goal of SolarStore is to improve the total amount of data that can be eventually retrieved from the network. It adaptively balances data reliability against data sensing since solar energy is renewable and dynamic. For example, it chooses to replicate data in the network until the next opportunity to upload data to the server. The degree of data replication also varies dynamically depending on the availability of solar energy and sensor storage.

Early database systems for sensor networks such as TinyDB and Cougar only act as filters for data collection networks and not as databases, i.e., no data is stored in, or retrieved from, any database. A database management system for resource-constrained sensors named Antelope is presented by [44]. Antelope supports run-time creation and deletion of databases and indexes and hence is a dynamic database system. It is the first DBMS for resource-constrained sensor devices, which enables a class of sensor network systems where every sensor holds a database. It is envisioned that database techniques would become increasingly important in the progress of sensor network applications and energy-efficient storage. Further, indexing and querying would play important roles in emerging storage-centric applications.

Besides, flash storage has been used for logging data on a sensor node, which is called *amnesic storage* systems [45]. An amnesic storage system archives streaming data using two key techniques: (i) data is compressed (usually with lossy compression methods) in an online fashion before being archived; and (ii) an amnesic storage system uses aging archived data by reducing the fidelity of older data to make space for newer data.

2.4 Search Techniques

Searching and finding relevant objects from billions of things is one of the major challenges for the future Internet of Things and can bring about huge potential impact to humans. Supporting technologies for searching things in the IoT are very different from those used in searching Web documents because things are tightly bound to contextual information (e.g., location) and have no easily indexable properties (e.g., human readable text in the case of Web documents). In addition, the state information of things is dynamic and rapidly changing. Things discovery calls for innovative ways of managing and searching from dynamic data, which makes it different from traditional Web searching. This section overviews the relevant areas such as the Deep Web, Semantic Web and then discusses state-of-the-art techniques in searching things in the IoT environments. We also summarize these techniques in Table 2.4.

2.4.1 Deep Web and Semantic Web

Deep Web refers to the portion of content on the World Wide Web that is not indexed by standard search engines. Deep Web data is not directly being seen from Web pages but is accessible typically via HTML forms from the Web pages. The size of the Deep Web is estimated to be several orders of magnitude larger than that of the so-called *Surface Web* (the Web that is accessible and indexable by text search engines).

Table 2.4 Comparisons of Search Techniques.

Approach	Vel	Sca	Dyn	Het	Unc	Red	Amb	Incs	Incp	Sem
Twitter-Sensing [46]	√		√		√					
Real-time-Micro-Blogging-Search [47]	√		√		√				√	
MIDAS-RDF [48]	√	√								
Web-Search-from-Structured-Databases [49]		√		√			√		√	√
Similarity-based-Entity-Search [50]		√		√			√		√	√
Snoogle [51]	√	√								
MAX [52]	√	√								
Microsearch [53]	√									
Dyser [54]	√				√				√	
Collaborative-Mobile-Object-Sensing [55]	√	√			√					

Note: Key elements that have been considered.

Vel: Velocity	Sca: Scalability	Dyn: Dynamics	Het: Heterogeneity
Unc: Uncertainty	Red: Redundancy	Amb: Ambiguity	Incs: Inconsistency
	Incp: Incompleteness	Sem: Semantics	

The Deep Web provides a wealth of hidden data in semi-structured form, accessible through Web forms and Web services. Since the data is hidden, to reach the whole content of the World Wide Web by just following hyperlinks is impossible. Regarding such issues, on top of XML, the Semantic Web grows as a common structured data source. With the W3C standards Resource Description Framework (RDF) and Web Ontology Language (OWL), the Semantic Web aims to unify the way semantic information is stored and exchanged. The Semantic Web makes it possible for machines themselves to not just read, but also "understand" the data from data sources, which enables machine to machine communication. In particular, languages such as as Microformats[11] and schema.org, can be used to add semantics to the descriptions of Web resources (including things).

[11] http://www.microformats.org

2.4.2 Web Search

The frequent changes and the unprecedented scale of the Web together pose enormous challenges to Web search engines, making it challenging to provide the most up-to-date and highly relevant information to their users. In IoT, this may become even more challenging as things would scale up the Web further and make the Web change more rapidly. For example, Tsubuyaku Sensor[12] is a new wireless device from Japanese Ubiquitous Computing Technology. It can monitor conditions such as temperature, humidity and radiation levels. It then automatically tweets the resulting data via Twitter. In this way, a sensor becomes a virtual Twitter user, which can actively post tweets on the Web.

Real-time Web Search

Real-time web search refers to the retrieval of very latest content which is in high demand. It is reported that Twitter handled more than 50 million tweets per day in 2010 [56]. Providing real-time search service is indeed very challenging in such large-scale microblogging systems because thousands of new updates need to be processed per second.

Twitter is real-time micro-blogging and the real-time interaction of events such as earthquakes in Twitter is investigated in [46]. They consider each Twitter user as a virtual sensor and apply Kalman filtering and particle filtering for estimating the centers of earthquakes and the trajectories of typhoons. Similarly, two challenges not encountered in non-real-time web search when supporting real-time web search have also been identified by [47], which are (i) quickly crawling relevant content and (ii) ranking documents with link and click information. Then they propose to use the micro-blogging data stream to detect fresh URLs and to compute novel and effective features for ranking fresh URLs based on micro-blogging data.

[12] http://ts.uctec.com/tsensor/index-e.php

Searching information over RDF Data

Searching information from RDF data is important as more and more information is published in the form of RDF (e.g., via Linked Open Data Cloud). Efficient management of RDF data is also an important factor in realizing the Semantic Web vision. Performance and scalability issues need to be addressed as the Semantic Web technology is applied to real-world applications. Unlike the relational database community, the Semantic Web community uses a very different data model, which is RDF.

MIDAS-RDF, a distributed P2P RDF/S repository that is built on top of a distributed multi-dimensional index structure, has been presented by [48]. It features fast retrieval of RDF triples satisfying various pattern queries by translating them into multi-dimensional range queries, which can be processed by the underlying index in hops logarithmic to the number of peers.

Collaborative Web Search

Web search engines often answer user queries based on data and information in relevant structured databases, which will be searched in isolation. Since a single database may not contain sufficient information to answer the query, the search often produces empty or incomplete results. Motivated by this observation, web search results and the items in structured databases have been exploited together to produce more complete answers to a wide range of queries that traditional web search cannot support well [49]. Take the query "light-weight gaming laptop" as an example. Dell XPS M1330 should be considered a match to such query as it is a light-weight laptop and suitable for gaming. But if searching only for the query keywords {`light-weight, gaming`} on the Web, Dell XPS M1330 may not appear in the search results. Therefore, the web search results (e.g. a set of relevant web documents) can be utilized to help identify relevant information in some structured databases [49]. Then the user queries could be better answered.

Similarly, web search engines have also been exploited to define new similarity functions for recognizing named entities such as products, people names, or locations from documents, such as “X61" and the entity “Lenovo ThinkPad X61 Notebook" [50]. The proposed new similarity functions are more accurate than existing string-based similarity functions because they aggregate evidence from multiple documents, and exploit web search engines to measure similarity.

2.4.3 Search of Things in IoT

In IoT, connecting things enabled by RFID, embedded sensors and sensor networks to the Internet and publishing their output on the Web would become a reality. Real-world objects would have their own Web presence. Considering the potential and profound impact of IoT technologies, search of things in IoT will become as important as today’s document search on the Web.

Key Words based Search of Things

Unlike search engines such as Google, searching for information in the physical world is more difficult because the physical objects do not have (reliable) connections to virtual space. For example, online books can be easily discovered by searching but physical books at home may be more difficult to find. This observation motivates the design of Snoogle [51], a search engine for the physical world. The basic idea behind Snoogle is that sensor nodes carry a textual description of the object they will be attached to. Such description forms the keywords for search of things. Then the key words information of the whole sensor network is indexed using a two-tiered hierarchy. The lower tier contains many mediators, which are also called *index points*. Each index point maintains an aggregate view of all sensors in a local area (e.g. a room) and every sensor in the same area will be assigned to the same index point. In

the top tier, there is a single mediator called the *key index point*. The key index point will maintain an aggregate view of the whole network.

MAX, a system that users can easily locate objects, is also designed [52]. The main assumption is that tags are attached to everyday objects and each tag stores a descriptor of the object it is attached to (e.g., the book of `Harry Potter`). Multiple descriptor words are allowed in each tag, enabling users to label the object with richer information, so that others can locate the object based on the label. A three-tiered hierarchy of mediators is used. In the lowest tier, substations represent immobile objects such as tables or shelves, on which mobile tagged objects can be placed. In the middle tier, base stations represent a geographical space such as a room containing multiple substations. In the top tier, the MAX server represents the entire space covered by the system. When searching for an object left behind, it is easy to locate where the object has been left by exploiting the knowledge of which substation and base station it belongs to.

Microsearch is a system that runs on resource constrained small devices capable of being embedded into everyday objects [53]. It allows users to do textual search in the local storage of a stand-alone small device, without support from a backend server. The challenge is that Microsearch runs in a resource constrained platform, where conventional search engine design and algorithms cannot be used. Information retrieval (IR) techniques for query resolution can answer top-k queries in a space-efficient manner [53].

Another search engine mainly designed for searching things, called Dyser, is proposed by [54]. Dyser allows users to search for real-world entities with a given state, such as “hot” or “cold”. However, this approach imposes two strong conditions: (i) to perform a query, end users have to know the vocabulary used by sensors (how states are named); and (ii) an entity must be represented by all the sensors that compose it. In order to estimate the probability of a sensor matching a query with sufficient accuracy and to rank sensor matching results, prediction models are adopted. The key

idea of sensor ranking is to exploit the periodic nature of people-centric sensors by using appropriate prediction models.

Collaborative Search of Things

A comprehensive system for managing and finding everyday objects relying on the collaboration of mobile phones in an urban area as object-sensing devices has been presented by [55]. For such tasks, the authors argue that the necessary infrastructures for such system include a sensing infrastructure, a communication infrastructure and a commercial infrastructure. Because of these requirements, the modern mobile phone system, which contains mobile sensors, provides a unique opportunity to realize collaborative search of everyday objects. The sensing model of the proposed system associates a probability with locations, meaning that the object currently has a certain probability of being at a certain location, thereby accelerating the search speed and reducing communication cost. Mobility provided by mobile sensors increases spatial coverage and hence the probability of finding a sought object.

2.5 Complex Event Processing

Data streaming techniques typically process incoming data through a sequence of transformations based on common SQL operators, such as `selection`, `aggregate`, `join`, and these operators are defined in general by relational algebra. By contrast, the *complex event processing* (CEP) model views the information in the streams as events in the physical world. These events must be filtered, combined and transformed into higher-level events for better understanding by computers and humans. Similar to traditional publish-subscribe systems, CEP systems allow subscribers to express their interest in composite events. The focus of the CEP model is on detecting occurrences of particular patterns of (low-level) events indicating some higher-level events, which may be interesting to some particular event subscribers. In the era of IoT, CEP

techniques lay part of the foundation of supporting computers to sense and react to events in the physical world. In the following, we survey some major CEP techniques related to IoT and summarize these techniques in Table 2.5.

2.5.1 Complex Event Processing

Systems for event processing and in particular event recognition (*event pattern matching*) accept a stream of time-stamped, simple or low-level events as input. A low-level event is the result of applying a computational derivation process to some other event, such as an event coming from a sensor. Using low-level events as input, (complex) event processing systems identify composite or high-level events of interest [57]. They are also collections of events that satisfy certain patterns.

SASE is a complex event processing system designed for monitoring queries over streams of RFID readings [58]. The SASE defines its own declarative event language that combines filtering, correlation, and transformation of events. The overall structure of the SASE language contains the `EVENT` clause specifying event patterns, the `WHERE` clause specifying qualifications and the `WITHIN` clause specifying window sizes. To meet the needs of RFID-enabled monitoring applications, several operators are also defined, including the `ANY` operator, the `SEQ_` operator, the `SEQ_WITHOUT` operator, the `Selection` operator and the `WITHIN_` operator. In order to process SASE queries, a query plan in SASE adopts a subset of six operators: `sequence scan`, `sequence construction`, `selection`, `window`, `negation`, and `transformation`. Pipelined execution of the above operators is used. More specifically, if a query matches a current event and some previous events, these events will be emitted from sequence scan and sequence construction immediately and form an event sequence. This event sequence is then pipelined through the subsequence operators, and added to the final output. To realize sequence scan, the basis of the whole process, Non-deterministic Finite Automata (NFA) are used.

Pattern matching over streams has been studied by [59]. It presents two new challenges: (i) compared with languages for regular expression matching, languages for pattern matching over streams are significantly richer; and (ii) the conventional techniques for stream query processing are inadequate for efficient evaluation of pattern queries over streams. In order to represent each pattern query, a new query evaluation model is designed for processing pattern matching over RFID streams, employing a new type of automaton that comprises a nondeterministic finite automaton (NFA) and a match buffer, named NFA^b [59]. Because of the powerful expressiveness of NFA, the semantics for the complete set of event pattern queries can be captured by the NFA^b model. Optimizations and query evaluation plans can also be produced and applied based on this model over event streams.

Nested CEP language called NEEL is proposed to support the flexible nesting of `AND`, `OR`, `Negation` and `SEQ` operators at any level [60]. One NEEL query example is given in Fig. 2.2, which expresses “a critical condition that after being recycled and washed, a surgery tool is being put back into use without first being sharpened, disinfected and then checked for quality assurance" [60]. Several techniques are also proposed to accelerate the evaluation of nested queries. Firstly, nested event expressions will be converted into normal forms by a normalization procedure. Secondly, a group of similar sub-expressions will be processed using prefix caching, suffix clustering methods and a customized physical execution strategy. Thirdly, an optimizer for optimal shared execution method is also designed based on the idea of iterative improvement. Compared with the traditional iterative nested execution, the optimized NEEL execution is up to two orders of magnitude faster.

Recent efforts have also been put on other aspects of complex event processing. For example, complex event processing in a distributed environment is studied and FUGU, an elastic allocator for Complex Event Processing systems, is proposed [61]. FUGU can dynamically allocate and de-allocate both stateless and stateful queries in order to meet the utilization goals. To that end, FUGU relies on bin packing to allo-


```
PATTERN   SEQ(Recyle r, Washing w, !SEQ(Sharpening s,
              Disinfection d, Checking c, s.id=d.id=c.id=p.id),
              Operating o, r.id=w.id and o.ins-type="surgery")
WITHIN    1 hour
```

Fig. 2.2 Nested CEP Query Example
(Adapted from the work by [60])

cate queries to hosts. Very recently, load shedding techniques have been investigated for complex event processing under various resource constraints [62]. Like other stream systems, CEP systems often face bursty input data. Since over-provisioning the system to the point where it can handle any such burst may be uneconomical or impossible, during peak loads a CEP system may need to "shed" portions of the load. The key technical challenge is to selectively shed work in order to eliminate the less important query results, thereby preserving the more useful query results defined by some utility function. Motivated by this, several load shedding algorithms are designed, including CPU-bound load shedding, memory-bound load shedding, and dual-bound load shedding (with both CPU- and memory-bound), depending on which resource is constrained.

2.5.2 Semantic Complex Event Processing

The combination of event processing and knowledge representation can lead to novel semantic-rich event processing engines [63, 64]. These intelligent event processing engines can (i) help to understand what is happening in terms of events, (ii) state and know what reactions and processes it can invoke, and furthermore (iii) decide what new events it can signal. The identification of critical events and situations requires processing vast amounts of data and meta-data within and outside the systems.

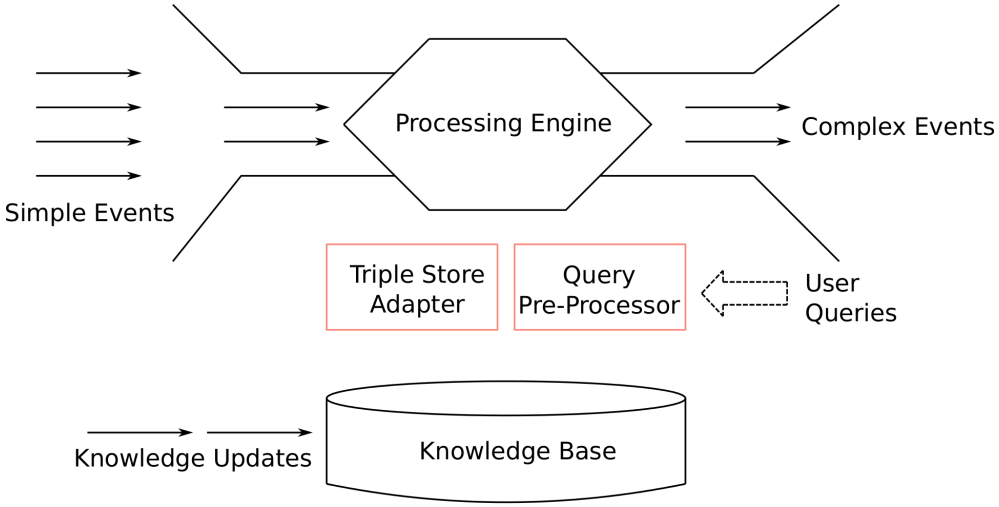


Fig. 2.3 Semantic Complex Event Processing System Overview
(Adapted from the work by [64])

Semantic CEP System

A semantic CEP system is shown in Fig. 2.3. Semantic models of events can improve event processing quality by using event meta-data in combination with ontologies and rules (i.e., *knowledge bases*). The fusion of background knowledge with data from an event stream can help the event processing engine to know more about incoming events and their relationships to other related concepts. A Knowledge Base (KB) can be used to provide background knowledge about the events and other non-event resources [64]. This means that events can be detected based on reasoning on their type hierarchy, temporal/spatial relationships, or their relationship to other objects in the application domain.

The benefits of using background knowledge in complex event processing can be seen as two major advantages over state-of-the-art CEP systems. The first benefit is its higher expressiveness and the second one is its flexibility. Expressiveness means that an event processing system can precisely express complex event patterns and reactions to events which can be directly translated into business operations. Flexibility means that a CEP system is able to integrate new business changes into the systems in a fraction of time rather than changing the whole event processing rules. Com-

Table 2.5 Comparisons of CEP Techniques.

Approach	Vel	Sca	Dyn	Het	Unc	Red	Amb	Incs	Incp	Sem
SASE [20]	√		√							
RFID-Streams-Pattern-Matching [21]	√		√							√
NEEL [22]	√		√							√
FUGU [23]	√		√							
Resource-Constrained-CEP [24, 25]	√		√		√				√	
KB-Fusion [64]									√	√
Semantic-Event-Enrichment [65]									√	√
Approximate-Semantic-Matching [63]					√					√
Heterogeneous-Approximate-Semantic-Matching [66]				√	√					√

Note: Key elements that have been considered.

Vel: Velocity	Sca: Scalability	Dyn: Dynamics	Het: Heterogeneity
Unc: Uncertainty	Red: Redundancy	Amb: Ambiguity	Incs: Inconsistency
	Incp: Incompleteness	Sem: Semantics	

plex event patterns are independent of current businesses and are defined in a higher level of abstraction based on business strategies. When something is changed in the business environment, it can be considered simply as an update in the background knowledge and the complex event detection patterns which are defined based on the business plans should not be changed.

Semantic Event Enrichment

The usage of background knowledge about events and their relations to other concepts in the application domain can improve the expressiveness and flexibility of CEP systems. Huge amounts of domain background knowledge stored in external knowledge bases can be used in combination with event processing in order to achieve more knowledgeable complex event processing.

An information completeness problem in semantic event processing contexts has been identified by [65] from a different angle. For example, while the basic information item in an event-based system is an event, normal users often require the system to handle information that is not encoded in the event. Such information typically comes from legacy databases or web data sources. This requires some degrees of information completeness or incompleteness for events to be sufficient for tasks such as subscription matching. The process of reducing information incompleteness is called *event enrichment*. Several challenges are identified for event enrichment, including determination of the enrichment source, retrieval of information items from the enrichment source, finding complementary information for an event in the enrichment source and fusion of complementary information with the event. To address these challenges, a model based on unifying enrichment within the event consumer logic and a native enricher that tackles incompleteness before matching are proposed [65].

Approximate Semantic Matching

Approximate semantic matching is first studied by [63]. To achieve approximate matching, semantic selection and inexact selection are used. More specifically, the semantic selection evaluates pattern constraints based on the semantic equivalence of attribute meanings captured by the event ontology instead of syntactic identical attribute values, while the inexact selection selects events and allows a limited number of mismatches to detect relevant patterns. A similarity function is associated with the inexact selection to evaluate relevance between matching patterns and target patterns.

Approximate semantic matching of heterogeneous events is also studied by [66]. The motivation is that heterogeneous events are difficult to match in a distributed computing environment as similar or closely related events may not be described using the same words but in a semantically related form. To match all interesting events, users may have to write many slightly different subscriptions and have to know

the exact format of all the heterogeneous events. Based on such observation, semantic decoupling of events and user's subscriptions becomes necessary. However, after such decoupling, the subscriptions would hardly exactly match the descriptions of events. This indicates that approximate matching and processing of events are inevitable. A model for approximate semantic matching that addresses event semantic decoupling is proposed. The model is evaluated using a hybrid matching approach based on both thesauri, semantic similarity and relatedness measures. After adopting this technique, the number of event subscriptions to achieve sufficiently precise matching results can be greatly reduced because of the decoupling between events and user subscriptions.

2.6 Summary

It is predicted that the next generation of the Internet will be comprised of trillions of connected computing nodes at a global scale. Through these nodes, everyday objects in the world can be identified, connected to the Internet and take decisions independently. In this context, Internet of Things (IoT) is considered a new revolution of the Internet. In IoT, the possibility of seamlessly merging the real and the virtual worlds, through the massive deployment of embedded devices, opens up many new and exciting directions for both research and development. In this chapter, we have provided an overview of some key research areas of IoT, specifically from a data-centric perspective. We have identified an IoT data taxonomy with ten key data elements of IoT data under three categories, including data generation, data quality, and data interoperability. We have also covered detailed discussions on data models, data storage, stream processing, search and event processing.

This chapter provides general background and context for the research in this thesis. For the specific research problems this thesis tries to solve, we will have additional related work and motivations presented in each technical chapter, including Chapter 3, Chapter 4, Chapter 5, and Chapter 6.

In the next chapter (Chapter 3), we will focus on addressing one of the key data elements, data dynamics, in IoT. More specifically, we will study data dynamics in dynamic graphs, by modeling relationships/connections between things with graphs.

Chapter 3

Shortest Path Computation in Dynamic Graphs

In IoT, connections between things are expected to be intermittent due to short battery life and mobility. In addition, things are normally small and fragile. So it is natural to model relationships/connections between things using dynamic graphs. In this chapter, we focus on the problem of computing the shortest path distance in graphs subject to edge failures. We propose SIEF, a Supplemental Index for Edge Failures in a graph, which is based on distance labeling. Together with the original index created for the original graph, SIEF can support distance queries with edge failures efficiently. By exploiting properties of distance labeling in static graphs, we are able to compute very compact distance labeling for dynamic graphs with edge failures. We extensively evaluate our algorithms using six real-world graphs and confirm the effectiveness and efficiency of our approach. The research presented in this chapter was published in [67].

3.1 Overview

Recent years have witnessed the fast emergence of massive graph data in many application domains, such as the World Wide Web, Linked Data technology, online social networks, and Web of Things [68–71]. In a graph, one of the most fundamental challenges centers on the efficient computation of the shortest path or distance between any given pair of vertices. For instance, distances or the number of links between web pages in a web graph can be considered a robust measure of web page relevancy, especially in relevance feedback analysis in web search [68]. In RDF graphs of Linked Data, the shortest path distance from one entity to another is important for ranking entity relationships and keyword querying [69, 72]. For online social networks, the shortest path distance can be used to measure the closeness centrality between users [70].

Shortest path and distance computation plays an important role in smart city computing. Urbanization's rapid progress has led to many big and intelligent cities, which have brought better lives to inhabitants in cities. However, many new challenges have also emerged, such as urban planning, transportation, and control of traffic congestion [73]. Take traffic control as an example, ridesharing is a promising way to save energy consumption and reduce traffic congestion without overlooking people's needs in commute. Specially, taxi ridesharing is considered a potential approach to alleviate taxi service waiting time and traffic congestion during the rush hours. To realize efficient and effective taxi ridesharing, shortest path computation in road networks is one of the key techniques [74]. Moreover, the analysis of a city's road network can help better formulate city planning for the future [75, 73], which also requires intensive shortest path and distance computation.

Because of the importance of shortest path and distance computation, a large body of indexing techniques have been recently proposed to process exact shortest path distance queries in a variety of graphs [76–82]. Among them, a significant portion

of indexes are based on *2-hop* distance labeling, originally proposed by Cohen et al. [83]. The 2-hop distance labeling techniques pre-compute a label for each vertex so that the shortest path distance between any two vertices can be computed given only their labels. These labeling indexes, such as [76, 79, 80, 82], have been proved to be efficient, i.e., being able to answer a distance query within microseconds.

Motivation. The above mentioned approaches generally make the assumption that graphs are static. However, in reality, many graphs are subject to edge failures. In this chapter, we refer to graphs that are not subject to edge failures as *stable graphs*, i.e., static graphs. Similarly, we refer to graphs that are subject to edge failures as *unstable graphs*. For example, the emerging social Web of Things calls for graph data management with edge failures because smart things are normally moving and their connectivity could be intermittent, leading to frequent and unpredictable changes in the corresponding graph models [84, 71]. Also, road networks in a city may have failed road segments due to traffic congestions or road works. Another example is web graphs. It is not uncommon that some web links become invalid as the web evolves. All these are examples of unstable graphs, which are common in the real-world, calling for efficient graph computations by considering link failures. We believe that it is imperative to design novel algorithms that can compute shortest path indexes for fast response on distance queries avoiding any failed edges. Some real-world applications/scenarios that require the computation of shortest path distance avoiding a failed edge are described in the following.

Scenario 1 *In the* ***sensitivity analysis*** *and in many analytical applications of transportation networks, government agencies need to evaluate different road segments (i.e., to find how much a road segment is worth) through Vickrey pricing [85], such that maintenance budget can be allocated accordingly, or the amount of tolls can be adjusted reasonably [75]. For example, if tolls are not charged appropriately and avoiding an expensive toll point causes only a small detour, then it is more likely that*

most drivers would take the detour, rather than pay for the toll. In a smart city, such scenarios may appear in road network planning for setting up toll road segments.

Scenario 2 *The* ***most vital arc*** *problem [86, 87] aims to identify the edge on a given shortest path and the removal of this edge results in the longest replacement path. Here, a replacement path means a shortest path from a source vertex to a destination vertex in a graph that avoids a specified edge. To find the most vital arc in a graph, we need to compute the shortest path distances efficiently when we are given an arc (i.e., an edge) to avoid. In smart city computing, the most vital arc computation could be critical and fundamental when recommending alternative shortest paths to travelers in the city road networks.*

Scenario 3 *In order to develop* ***game-theoretic and price-based mechanisms*** *to share bandwidth and other network resources, a natural economic question is [85]: how much is an edge in a network worth to a user who wants to send data between two nodes along a shortest path? Or in other words, what is the penalty of avoiding an edge in the given network? In the future smart cities, Web of Things could bring fundamental changes to city development and city intelligence. To enhance data availability and data sharing efficiency among different things in a smart city, we need to efficiently estimate the penalty of avoiding data transmission between two things and try to optimize data transmission paths according to the analytical results of the penalties.*

These application scenarios reveal a desire for handling shortest path computations in a graph with single-edge failure and in smart city computing. Here, single-edge failure refers to graph failures with only one failed edge at a time [88]. Note that, other types of edge failure, such as dual-failure in [89], may allow multiple failed edges at a time. But they are considered much harder than single-failure [89]. To shed light on these challenging issues, we focus on exact distance queries answering for

single-edge failure first. We then discuss how to apply our approach to approximately answer distance queries for other types of edge failures.

Contributions. Since 2-hop labeling has shown its power to support instant responses to shortest path distance queries in stable graphs, our work aims at extending this technique to support unstable graphs. Existing shortest path indexing techniques based on 2-hop labeling can be used to pre-compute the whole shortest path index for a graph. The resulted indexes can normally answer distance queries fast using moderate storage space [80, 82]. However, applying indexing techniques designed for static/stable graphs directly to evolving/unstable graphs may lead to inefficiency. When considering every single-edge failure case and constructing a corresponding index for each case, the size of all these indexes will become too big to manage. For instance, a snapshot of the Gnutella peer-to-peer (P2P) file sharing network in August 2002 contains more than 6,000 vertices[1] and 20,000 edges. Using the state-of-the-art method, Pruned Landmark Labeling (PLL) [80], the index size is slightly more than 5 MB. However, suppose we want to construct such index for each single-edge failure case, the total index size would be more than 5 MB $\times 20,000 = 10^5$ MB.

To address the deficiency of existing shortest path indexing techniques, in this chapter we propose a generic framework named SIEF, a Supplemental Index for Edge Failures in a graph, to construct compact shortest path indexes efficiently for unstable graphs where single-edge failure may exist. As an initial attempt on this challenging issue, we focus on unweighted, undirected graphs. Similar to other distance labeling based indexing methods [80, 82], our method can be extended to weighted and/or directed graphs. We highlight our main contributions in the following.

- We present the concept of *well-ordering 2-hop distance labeling* and identify its important properties that can be utilized to design algorithms for shortest path indexes in graphs with edge failures.

[1] http://snap.stanford.edu/

- We analyze shortest path index constructions in graphs with edge failures theoretically. We develop the corresponding theorems as well as novel algorithms to enable constructions of compact indexes for all the single-edge failure cases of the entire graph. By applying our approach to the aforementioned Gnutella P2P dataset, the size of the generated SIEF index together with the original index created for the original graph is merely 14 MB, which is much more compact than 10^5 MB by directly using PLL method [80] to construct indexes for each single-edge failure case.

- We apply the SIEF framework to multi-edge failure cases and design two querying strategies, namely *aggressive strategy* and *conservative strategy*, to approximately answer distance queries with multi-edge failure constraints.

- We conduct extensive experiments using six real-world graphs to verify the efficiency and effectiveness of our method. The results show that our method can efficiently answer shortest path distance queries avoiding a failed edge with very compact labeling indexes.

A preliminary version of this chapter appeared in [67], where we focused on the introduction of the SIEF framework for answering distance queries in failure-prone graphs with only single-edge failure. In this chapter, we further extend our work to multi-edge failure. For this purpose, aggressive strategy and conservative strategy have been proposed to provide approximate answers to distance queries with multi-edge failure constraints. More extensive experiments have also been done to validate the accuracy and fast response time of answering distance queries in multi-edge failure graphs using SIEF.

The rest of this chapter is organized as follows. In Section 3.2, we review the related work. In Section 3.3, we present some preliminaries on 2-hop distance labeling. We then present the SIEF framework and the details of our approach in Section 3.4. We also discussion some variants and extensions of SIEF in the same section. In

Section 3.5, we report the results of an extensive experimental study using six graphs from real-world. Finally, we present some concluding remarks in Section 3.6.

3.2 Related Work

In this section, we review the major techniques that are most closely related to our work, mainly including distance labeling for static graphs, 2-hop reachability labeling in dynamic graphs, and maintenance of 2-hop distance labeling in dynamic graphs.

Distance labeling has been an active research area in recent years. In [76], Cheng and Yu exploit the strongly connected components property and graph partitioning to pre-compute 2-hop distance cover. However, the graph partitioning process introduces high cost because it has to find vertex separators recursively. Hierarchical hub labeling (HHL) proposed by Abraham et al. [90] is based on the partial order of vertices. Smaller labeling results can be obtained by computing labeling for different partial order of vertices. In [91], Jin et al. propose a highway-centric labeling (HCL) that uses a spanning tree as a highway. Based on the highway, a 2-hop labeling is generated for fast distance computation.

Very recently, the pruned landmark labeling (PLL) [80] is proposed by Akiba et al. to pre-compute 2-hop distance labels for vertices by performing a breadth-first search from every vertex. The key idea is to prune vertices that have obtained correct distance information during breadth-first searches, which helps reduce the search space and sizes of labels significantly. Further, query performance is also improved as the number of label entries per vertex is reduced. IS-Label (or ISL) is developed by Fu et al. in [82] to pre-compute 2-hop distance label for large graphs in memory constrained environments. ISL is based on the idea of independent set of vertices in a large graph. By recursively removing an independent set of vertices from the original graph, and by augmenting edges that preserve distance information after the removal of vertices in the independent set, the remaining graph keeps the distance

information for all remaining vertices in the graph. Apart from the 2-hop distance labeling technique, a multi-hop distance labeling approach [79] is also studied, which can reduce the overall size of labels at the cost of increased distance querying time.

The Tree decomposition approach has been recently investigated [77, 92] for answering distance queries in graphs. Wei proposes TEDI [77], which first decomposes a graph into a tree and then constructs a tree decomposition for the graph. A tree decomposition of a graph is a tree with each vertex associated with a set of vertices in the graph, which is also called a *bag*. The shortest paths among vertices in the same bag are pre-computed and stored in bags. For any given source and target vertices, a bottom-up operation along the tree can be executed to find the shortest path. An improved TEDI index is proposed by Akiba et al. in [92] that exploits a core-fringe structure to improve index performance. However, due to the large size of some bags in the decomposed tree, the construction time for a large graph is costly and thus such indexing approaches cannot scale well.

The above studies focus on point to point shortest path distance processing. Some studies also investigate other types of distance queries, such as single-source shortest path (SSSP) distance queries. For example, Cheng et al. propose VC-index [78], a disk-based method for processing SSSP distance queries based on the concept of vertex cover. Highways-on-Disk (HoD) is also proposed in [81] to support SSSP distance queries for directed graphs. HoD reduces I/O and computation costs for query processing by augmenting a set of auxiliary edges or shortcuts in the original graphs.

The maintenance of 2-hop reachability labeling has also been studied. For example, HOPI (2-HOP-cover-based Index) introduces some maintenance techniques for the constructed index. HOPI is developed by Schenkel et al. in [93] and is designed to speed up connection or reachability tests in XML documents based on the idea of 2-hop cover. HOPI is able to update indexes for insertions and deletions of nodes, edges or even XML documents. To the best of our knowledge, HOPI is the first work

on the maintenance of 2-hop labeling. Recently, maintenance of 2-hop labeling for large graphs has also been studied by Bramandia et al. in [94]. However, all these studies focus on reachability queries and are based on 2-hop labeling but not on 2-hop distance labeling.

Incremental maintenance of 2-hop distance labeling is also studied very recently by Akiba et al. in [95]. In that work, incremental updates (i.e., edge insertions) of 2-hop labeling indexes are investigated. To support fast incremental updates, outdated distance labels are kept, which will not affect the distance computation in the updated graphs in the incremental case. However, for the decremental case (i.e., edge deletions), this approach will not work, as outdated distance labels must be removed first and then some necessary labels of the 2-hop labeling index need to be recomputed. Hence, their update algorithms cannot be applied on edge deletions (i.e., edge failures), which will be discussed in this chapter.

3.3 Preliminaries

3.3.1 2-Hop Distance Labeling

The technique of 2-hop cover can be used to solve reachability problems (using reachability labels) and shortest path distance querying problems (using distance labels) in graphs [83]. Since our work focuses on the shortest path distance querying problems, we adopt distance labels with the 2-hop cover technique. We specifically refer to it as *2-hop distance labeling* or *2-hop distance cover*.

Assume a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. For each vertex $v \in V$, there is a pre-computed label $L(v)$, which is a set of vertex and distance pairs (u, δ_{uv}). Here u is a vertex and δ_{uv} is the shortest path distance between u and v. Given such a labeling for all vertices in G, denoted by L, for any pair of vertices s and t in G, we have

$$dist(s,t,L) = min\{\delta_{vs} + \delta_{vt} | (v,\delta_{vs}) \in L(s) \text{ and } (v,\delta_{vt}) \in L(t)\} \tag{3.1}$$

If $L(s)$ and $L(t)$ do not share any vertices, we have $dist(s,t,L) = \infty$. The distance between any given vertices s and t in G is denoted by $d_G(s,t)$. If we have $d_G(s,t) = dist(s,t,L)$ for all s and t in G, we call the labeling result L a 2-hop distance cover.

3.3.2 Well-Ordering 2-Hop Distance Labeling

For a connected graph G, there exists a sequence of vertices $\sigma = < v_0, v_1, v_2, \ldots, v_{n-1} >$. We denote the order of any vertex v_i as $\sigma[v_i]$ and we have $\sigma[v_i] = i$ for the above given vertex sequence. Based on this, we can define *Well-Ordering 2-Hop Distance Labeling* in the following.

Definition 1 (***Well-Ordering 2-Hop Distance Labeling***). Suppose that (1) each vertex v_i has a distance labeling $L(v_i)$, and the labeling result L of all vertices forms a 2-hop distance cover of G; (2) for any pair of vertices v_i and v_j, given that $\sigma[v_i] < \sigma[v_j]$, then v_j is not in $L(v_i)$ and v_i may be in $L(v_j)$. We call such a 2-hop distance cover a *well-ordering 2-hop distance labeling*. Alternatively we say that a 2-hop distance cover has well-ordering property.

Similar concepts of well-ordering 2-hop distance labeling also appear in recent research efforts such as HHL [90], PLL [80], and ISL [82]. This confirms that well-ordering 2-hop distance labeling is important in the related research area. More importantly, we will show in this chapter that the well-ordering property is also a basic concept in the design of index construction algorithms for distance labeling computation in unstable graphs where edges may fail.

In a graph containing multiple connected components, suppose its 2-hop labeling is L. For any pair of vertices u and v in different connected components, we can

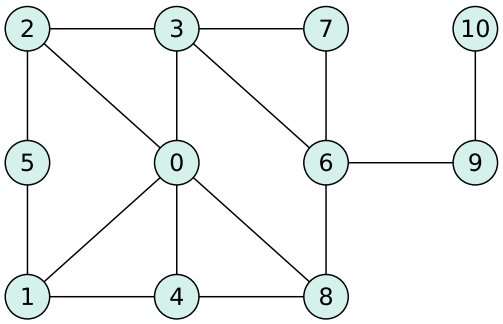


Fig. 3.1 A graph example

Table 3.1 2-Hop Distance Labeling L for Figure 3.1

Label	**Entries**
$L(0)$	(0,0)
$L(1)$	(0,1) (1,0)
$L(2)$	(0,1) (2,0)
$L(3)$	(0,1) (2,1) (3,0)
$L(4)$	(0,1) (1,1) (4,0)
$L(5)$	(0,2) (1,1) (2,1) (5,0)
$L(6)$	(0,2) (2,2) (3,1) (4,2) (6,0)
$L(7)$	(0,2) (2,2) (3,1) (6,1) (7,0)
$L(8)$	(0,1) (4,1) (6,1) (8,0)
$L(9)$	(0,3) (2,3) (3,2) (4,3) (6,1) (9,0)
$L(10)$	(0,4) (2,4) (3,3) (4,4) (6,2) (9,1) (10,0)

assert that $L(u)$ and $L(v)$ do not share any vertex according to the definition of 2-hop cover. Each connected component has its own vertex order. For such a graph, we will have separate vertex orders for each connected component. We denote a connected component containing vertex u as $C(u)$. If u and v belong to the same connected component, we have $C(u) = C(v)$.

Figure 3.1 shows an example graph with 11 vertices and Table 3.1 shows a well-ordering 2-hop distance labeling result L for the graph (L can be constructed by PLL [80] using the same vertex ordering as that specified in the table). In the table, the order of vertices is $< 0,1,2,3,4,5,6,7,8,9,10 >$. Take $L(5)$ as an example to further explain the idea of well-ordering 2-hop distance labeling. $L(5)$ is the label of vertex 5. By the well-ordering property, label entries in $L(5)$ can only contain vertices

$0,1,2,3,4$ and 5. Since label entries containing vertices 3 and 4 are redundant in $L(5)$ (this will be explained in more details later in this section), label entries in $L(5)$ only contain vertices $0,1,2$ and 5.

3.3.3 Properties of Well-Ordering 2-Hop Distance Labeling

Technically speaking, if we index shortest paths for all pairs using a labeling method, we will obtain an index that occupies $O(n^2)$ disk space. This index can be considered as a special 2-hop distance labeling. Obviously, the space complexity of this is too high for large graphs. Constructing a minimal 2-hop distance labeling has been proven to be NP-hard [83]. Therefore, an alternative way to obtain labeling results with reduced sizes is by using heuristic methods [76, 79, 80, 82]. Well-ordering 2-hop distance labeling is one of the techniques that can help to design efficient algorithms for constructing shortest path distance labeling indexes and for index maintenance. We identify its useful properties in the following.

Lemma 1 *Given a well-ordering 2-hop distance labeling L of a connected graph G, suppose $u \in G$ and $\sigma[u]$ is a minimum among all vertices in G, then for any vertex $v \in G$, we must have $(u, \delta_{uv}) \in L(v)$.*

Proof: It is trivial to prove this when $v = u$ since $(u,0) \in L(u)$. We prove the case when $v \neq u$ by contradiction. Suppose there exists a vertex $v \in G$, $(u, \delta_{uv}) \notin L(v)$. By the definition of L, since $\sigma[u]$ is minimum, $L(u)$ will contain only one label entry $(u,0)$. Then it is obvious that $L(u)$ and $L(v)$ do not share any vertex, which leads to $dist(u,v,L) = \infty$. This implies that u and v belong to different connected components, which is false. Therefore, the lemma is proved. □

Lemma 2 *Given a well-ordering 2-hop distance labeling L of a connected graph G, suppose $s,t,u \in G$ and $dist(s,t,L) = dist(s,u,L) + dist(u,t,L)$, then u must be an internal vertex of a certain shortest path between s and t.*

Proof: Since $dist(s,t,L) = dist(s,u,L) + dist(u,t,$
$L)$, there must exist some shortest path that starts from s, passes u, and ends at t. Hence the lemma is proved. □

Take vertices 5, 6 and 2 in Figure 3.1 as an example. From Table 3.1, we have $dist(5,6,L)$=3 and $dist(5,2,L) + dist(2,6,L) = 1+2 = 3$. From Figure 3.1, we can see that vertex 2 is an internal vertex on some shortest path, denoted as p, between vertex 5 and vertex 6. In this case, we have $p =< 5,2,3,6 >$.

Lemma 3 *Given a well-ordering 2-hop distance labeling L of a connected graph G, suppose* $s,t,u \in G$ *and u has minimum vertex order* $\sigma[u]$ *among all shortest paths between s and t. Then we must have* $(u,\delta_{us}) \in L(s)$ *and* $(u,\delta_{ut}) \in L(t)$ *and* $dist(s,t,L) = \delta_{us} + \delta_{ut}$.

Proof: We prove this by contradiction. Without loss of generality, suppose $(u,\delta_{us}) \notin L(s)$. In order to calculate $dist(s,u,L)$, there must exist some vertex v other than u, where $(v,\delta_{vs}) \in L(s)$, $(v,\delta_{vu}) \in L(u)$ and $dist(s,u,L) = \delta_{vs} + \delta_{vu}$. According to Lemma 2, v must be an internal vertex of some shortest path between s and u. Hence v must also be an internal vertex of some shortest path between s and t. Meanwhile, by definition, we must have $\sigma[v] < \sigma[u]$. This contradicts our assumption that u has the minimum vertex order among all shortest paths between s and t. Hence, we must have $(u,\delta_{us}) \in L(s)$. Furthermore, u is an internal vertex of some shortest path between s and t, thus $dist(s,t,L) = \delta_{us} + \delta_{ut}$. Hence the lemma is proved. □

Take vertices 1 and 6 in Figure 3.1 as an example. Paths $p_1 =< 1,0,8,6 >$, $p_2 =<$ $1,0,3,6 >$ and $p_3 =< 1,4,8,6 >$ are all the shortest paths between vertices 1 and 6. Vertex 0 is the one with minimum order along all these paths. From Table 3.1 we can see that both vertices 1 and 6 contain a label entry $(0,\delta)$. We can also easily check that $dist(1,6,L) = \delta_{0,1} + \delta_{0,6} = 1+2 = 3$.

Lemma 4 *Given a well-ordering 2-hop distance labeling L of a connected graph G, suppose* $\sigma[u] < \sigma[v]$. *If there is a label entry* $(u, \delta_{uv}) \in L(v)$, *we must have for any label entry* $(r, \delta_{rv}) \in L(v)$, *(1)* $\delta_{uv} \leq \delta_{rv} + dist(r,u,L)$; *(2) if* $\sigma[r] < \sigma[u]$ *and* $\delta_{uv} = \delta_{rv} + dist(r,u,L)$ *then* $(u, \delta_{uv}) \in L(v)$ *is a redundant label entry.*

Proof: We first prove the first claim that $\delta_{uv} \leq \delta_{rv} + dist(r,u,L)$. By definition and the triangle inequalities we must have $\delta_{uv} = d_G(u,v) = dist(u,v,L) \leq \delta_{rv} + dist(r,u,L)$.

We then prove the second claim. We need to prove that if $\delta_{uv} = \delta_{rv} + dist(r,u,L)$, then for any vertex t, when we calculate $dist(v,t,L)$, (u, δ_{uv}) in $L(v)$ is not required. For t, there are three cases: (1) $(u, \delta_{ut}) \notin L(t)$; (2) $(u, \delta_{ut}) \in L(t)$ but $\delta_{uv} + \delta_{ut} > dist(v,t,L)$; (3) $(u, \delta_{ut}) \in L(t)$ and $\delta_{uv} + \delta_{ut} = dist(v,t,L)$. For Case (1) and Case (2), it is trivial since (u, δ_{uv}) in $L(v)$ is not required to calculate $dist(v,t,L)$. For Case (3), according to Lemma 2, u is an internal vertex of some shortest paths between v and t. Similarly, since $\delta_{uv} = \delta_{rv} + dist(r,u,L)$, r is an internal vertex of some shortest paths between u and v, which means r is also an internal vertex of some shortest paths between v and t. In such case, we prove in the following that there must exist a vertex s other than u and we have $(s, \delta_{sv}) \in L(v)$, $(s, \delta_{st}) \in L(t)$ where $\delta_{st} + \delta_{sv} = dist(v,t,L)$.

Suppose s is the vertex with minimum vertex order among all shortest paths between v and t. According to Lemma 3, we must have $(s, \delta_{sv}) \in L(v)$, $(s, \delta_{st}) \in L(t)$ and $\delta_{st} + \delta_{sv} = dist(v,t,L)$. Since $\sigma[s] \leq \sigma[r] < \sigma[u]$, s is not the same vertex of u. Therefore, (u, δ_{uv}) in $L(v)$ is not required to calculate $dist(v,t,L)$. Hence, the second claim is also proved. □

Take label entries of vertex 5 in Table 3.1 as an example. We have $\sigma(3) < \sigma(5)$ and $\sigma(2) < \sigma(3)$. We also have $\delta_{3,5} = 2 = \delta_{2,5} + \delta_{2,3}$. Therefore $(3,2)$ is a redundant label entry in $L(5)$, which can be removed from $L(5)$.

3.4 The SIEF Approach

In this section, we first provide an overview of our SIEF approach. We then analyze the 2-hop distance labeling computation in graphs with single-edge failure and introduce a set of algorithms to achieve fast and compact index constructions. Finally, new querying strategies are also designed to provide approximate distance estimation in multi-edge failure graphs.

3.4.1 SIEF Overview

After an edge fails in a graph, we observe that distances of a large proportion of shortest paths between any pair of vertices remain unchanged. Therefore, to construct a new index for each single-edge failure case, we only need to compute new labels for those vertices with changed shortest path distances due to the edge failure. Overall, our index construction approach can be divided into two main stages. In the first stage, *IDENTIFY*, we identify affected vertices after an edge fails. In the second stage, *RELABEL*, we relabel all affected vertices with necessary additional label entries for the single-edge failed graph. These new label entries form a new part of the index, which is called a *supplemental index*.

Before the detailed discussions of our algorithms, suppose that the failed edge is (u,v) in G, and the new graph is G', we introduce a concept for the supplemental index construction:

Definition 2 (***Affected vertices*** $AV_{(u,v)}$). For any vertices s and t, if $d_{G'}(s,t) \neq d_G(s,t)$, then $s \in AV_{(u,v)}$ and $t \in AV_{(u,v)}$.

To be specific, $AV_{(u,v)}$ contains all vertices whose distance to some other vertex must have been changed due to the failed edge (u,v). It is quite clear that supplemental indexes should be constructed to maintain all new distances for each single-edge failure case. In other words, supplemental indexes are constructed based on all the

vertices in $AV_{(u,v)}$. Further, in order to be compact, the supplemental indexes should only answer distances that cannot be answered by the original index.

3.4.2 Identification of Affected Vertices

Before we can start to construct supplemental indexes, we need to identify all the affected vertices in $AV_{(u,v)}$ first. A naive method would be to compare distances for any possible pair of affected vertices in the original graph G and the new graph G' with a failed edge (u,v), but that would be very time consuming as it will need to test distances of $O(n^2)$ pairs of vertices. In the following, we will try to identify some important properties for vertices in $AV_{(u,v)}$ for us to identify $AV_{(u,v)}$ more efficiently and accurately.

Lemma 5 *After removing the failed edge (u,v) from graph G, for any vertex s,t in G', we must have $d_{G'}(s,t) \geq dist(s,t,L)$.*

Proof: In the old graph G, there are only two types of shortest paths: (1) shortest paths containing edge (u,v); and (2) shortest paths not containing edge (u,v). For the former, we have $d_{G'}(s,t) \geq d_G(s,t) = dist(s,t,L)$. For the latter, we have $d_{G'}(s,t) = d_G(s,t) = dist(s,t,L)$. Thus the lemma is proved. □

Lemma 6 *After removing the failed edge (u,v) from graph G, for any vertex s,t in G', if $d_{G'}(s,t) > dist(s,t,L)$, and suppose a shortest path between s and t in G is $\pi_G(s,t)$, then we must have $uv \in \pi_G(s,t)$ or $vu \in \pi_G(s,t)$.*

Proof: This can be proved by contradiction. Suppose we have $d_{G'}(s,t) > dist(s,t,L)$ but $uv \notin \pi_G(s,t)$ and $vu \notin \pi_G(s,t)$, which means edge (u,v) does not appear in $\pi_G(s,t)$. In such case, there must exist a path $P_{G'}(s,t)$ in G' where $\pi_G(s,t) = P_{G'}(s,t)$. This means $d_{G'}(s,t)$ must be at most the length of $P_{G'}(s,t)$, i.e., the length of $\pi_G(s,t)$.

Thus, we must have $d_{G'}(s,t) = dist(s,t,L') \leq d_G(s,t)$. This contradicts our assumption $d_{G'}(s,t) > dist(s,t,L)$. □

According to Lemma 6 and the definition of affected vertices, if we have $d_{G'}(s,t) > dist(s,t,L) = d_G(s,t)$, we must have that $s,t \in AV_{(u,v)}$. This further means the shortest path(s) between s and t in the original graph G must contain the failed edge (u,v). Then, after edge (u,v) fails, take any one of these shortest paths (if multiple shortest paths exist; if not, we will have one and only one shortest path containing (u,v)) as an example, denoted as $\pi_G(s,t)$. Then it is easy to imagine that $\pi_G(s,t)$ will become two segments: one segment ends at u, denoted as Seg_u and the other segment ends at v, denoted as Seg_v. Without loss of generality, suppose s falls on the Seg_u and t falls on Seg_v. Since Seg_u and Seg_v must also be shortest paths from s to u and from t to v, respectively, this means we must have $d_G(s,u) = d'_G(s,u)$ and $d_G(t,v) = d'_G(t,v)$. But in the meantime, we must have $d_G(s,v) \neq d'_G(s,v)$ and $d_G(t,u) \neq d'_G(t,u)$ since otherwise we will have $d_{G'}(s,t) = d_G(s,t)$, which is impossible. Based on this observation, we can see that vertices in $AV_{(u,v)}$ form two disjoint sets: one set is $AV_{(u,v)}(u)$ and the other set is $AV_{(u,v)}(v)$, where for $\forall s \in AV_{(u,v)}(u)$ and $\forall t \in AV_{(u,v)}(v)$, we must have $d_{G'}(s,t) > d_G(s,t)$, $d_G(s,u) = d'_G(s,u)$ and $d_G(t,v) = d'_G(t,v)$. Since (u,v) is the failed edge, obviously, we must have $u \in AV_{(u,v)}(u)$ or $v \in AV_{(u,v)}(v)$. Further, it should be noted that, $\forall s,t \in AV_{(u,v)}(u)$, we must have $d_G(s,t) = d'_G(s,t)$. The same conclusion can be made on $\forall s,t \in AV_{(u,v)}(v)$.

Next, we are going to show that all vertices $s \in AV_{(u,v)}(u)$ form a tree rooted at u and similarly, all vertices $t \in AV_{(u,v)}(v)$ also form a tree rooted at v.

Lemma 7 *After removing the failed edge (u,v), for any vertex w in G', suppose w is an affected vertex, i.e. $w \in AV_{(u,v)}(u)$ or $w \in AV_{(u,v)}(v)$. Without loss of generality, we assume $w \in AV_{(u,v)}(u)$. Then we must have $d_G(w,v) = d_G(w,u) + 1$.*

Proof: Since $w \in AV_{(u,v)}(u)$, we must have that $d_G(w,v) \neq d'_G(w,v)$, which means that any shortest path between w and v in G, denoted as p_{wv} must contain the failed

edge (u,v), which must also be a shortest path between w and v. Hence, there must exist a certain shortest path between w and v containing edge (u,v) in the original graph and we can denote it as $p_{wv} = p_{wu} + (u,v)$. Hence, we must have $d_G(w,v) = d_G(w,u) + 1$. □

Lemma 8 *After removing the failed edge (u,v), suppose w in G' is an affected vertex, i.e. $w \in AV_{(u,v)}(u)$ or $w \in AV_{(u,v)}(v)$. Without loss of generality, we assume $w \in AV_{(u,v)}(u)$. Then there must exist a certain shortest path between w and v containing edge (u,v) in the original graph, where each internal vertex is an affected vertex in $AV_{(u,v)}(u)$.*

Proof: Since $w \in AV_{(u,v)}(u)$, then according to Lemma 7, we must have the fact that any shortest path between w and u, denoted as p_{wu}, plus edge (u,v) in the original graph must be a shortest path between w and v. Then, there must exist a certain shortest path between w and v containing edge (u,v) in the original graph and we can denote it as $p_{wv} = p_{wu} + (u,v)$.

It is clear that the internal vertices of p_{wv} must also be on some shortest path p_{wu}. And all shortest paths from these internal vertices to vertex v must contain edge (u,v), which means, their distances to vertex v must have changed in the new graph G'. Therefore, they must also be affected vertices in $AV_{(u,v)}(u)$ like w. □

Note that, according to Lemma 8, $AV_{(u,v)}(u)$ and $AV_{(u,v)}(v)$ can be considered as trees rooted at u and v, respectively. Moreover, we must have $AV_{(u,v)}(u) \bigcap AV_{(u,v)}(v) = \emptyset$. This is because otherwise, any vertex r in $AV_{(u,v)}(u) \bigcap AV_{(u,v)}(v)$ must have $d_G(r,v) = d_G(r,u) + 1$ and $d_G(r,u) = d_G(r,v) + 1$, which is impossible. Lemma 8 forms the basis of Algorithm 1. Note that, in Algorithm 1, we need to calculate distance vectors d_u, d_v, d'_u and d'_v for each single-edge failure case. Here, d_u stores distances from all vertices in G to vertex u while d'_u stores distances from all vertices in G' to vertex u. Distance vectors d_v and d'_v are similar. The calculations can be done efficiently using a BFS algorithm. To reduce the calculation cost, we will fix an end point of failed

Algorithm 1 Identify affected vertices

Input: G, (u,v), distance vectors d_u, d_v, d'_u, d'_v
Output: $AV_{(u,v)}(u)$, $AV_{(u,v)}(v)$

1. Initialize flag $m[t] \leftarrow 0$ for any vertex t in G
2. $m[u] \leftarrow 1$
3. $Q \leftarrow \emptyset$
4. Enqueue u into Q
5. **while** Q is not empty **do**
6. Dequeue t from Q
7. **for all** neighbor vertex r of t **do**
8. **if** $m[r] = 0$ **then**
9. **if** $d_v[r] = d_u[r] + 1$ and $d'_v[r] \neq d_u[r] + 1$ **then**
10. $AV_{(u,v)}(u) \leftarrow AV_{(u,v)}(u) \cup \{r\}$
11. Enqueue r into Q
12. **end if**
13. $m[r] \leftarrow 1$
14. **end if**
15. **end for**
16. **end while**
17. Repeat the above steps by mapping $u \leftarrow v$ and $v \leftarrow u$ to identify $AV_{(u,v)}(v)$

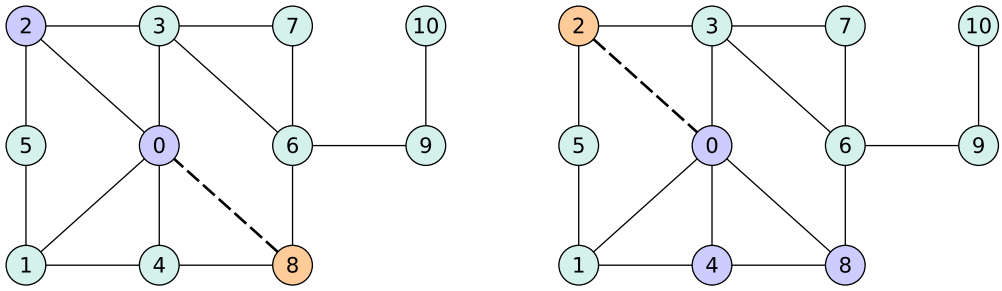


Fig. 3.2 Affected vertices identification

edges, i.e., we will firstly compute affected vertices for all edges attached to u then we move to other vertices for processing the rest single-edge failure cases.

Figure 3.2 shows two examples of identifying affected vertices. It uses the same graph in Figure 3.1. In this figure, the first example is Case (a), where the failed edge is $(0,8)$. The second example is Case (b), where the failed edge is $(0,2)$. In Case (a), starting from vertex 0, we identify the affected vertex set rooted at 0 as $AV_{(0,8)}(0) =$

$\{0,2\}$ since only vertices 0 and 2 have changed their distance to vertex 8. Meanwhile, starting from vertex 8, we identify the affected vertex set rooted at 8 as $AV_{(0,8)}(8) = \{8\}$ since only vertex 8 has changed its distance to vertex 0. Similarly, in Case (b), as can be observed in the figure, we have $AV_{(0,2)}(0) = \{0,4,8\}$ and $AV_{(0,2)}(2) = \{2\}$.

3.4.3 Relabeling: Supplemental Index Construction

After identifying all affected vertices, we can start relabeling the affected vertices in order for fast computation of shortest path distances in the graph with single-edge failure. Only supplemental indexes will be created, i.e., only changed distance information will be captured in supplemental indexes. All the unchanged distance information will be still computed using the original indexes (such as the distance labeling in Table 3.1 for the example graph in Figure 3.1). We develop a novel relabeling algorithm, namely the BFS ALL algorithm, for the supplemental index construction. Detailed descriptions of the algorithm are presented in the following.

BFS ALL algorithm

The BFS ALL algorithm relabels affected vertices based on the traditional BFS algorithm. The detail steps are shown in Algorithm 2. Figure 3.3 also depicts an example of the supplemental index construction process using the BFS ALL algorithm.

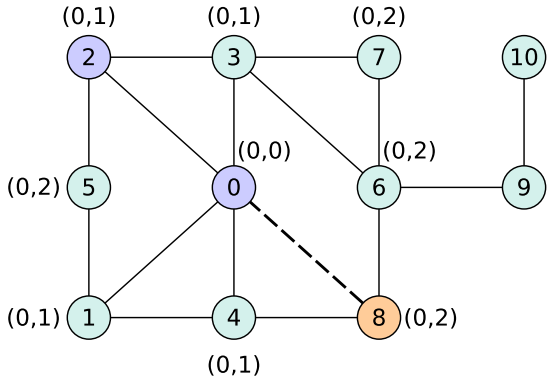
The failed edge is (0,8) in this example and there are three steps in Figure 3.3. Each step relabels one affected vertex. Also the distance information will be kept at each BFS step, using a set of temporary labels stored in TL. This distance information in TL can be used to prune label entries at the later BFS steps of the index construction process for all vertices in the graph and some vertices can be pruned during a BFS process. At Step (1), BFS ALL algorithm performs BFS from vertex 0. The number beside each node is the distance from that node to the BFS root, vertex 0. In this step, vertex 8 is the only affected vertex in $AV_{(0,8)}(8)$ that has larger vertex order than vertex

Algorithm 2 BFS ALL algorithm

Input: G, (u,v), $AV_{(u,v)}(u)$, $AV_{(u,v)}(v)$
Output: The supplemental index SI_u and SI_v for the edge failure case of (u,v)

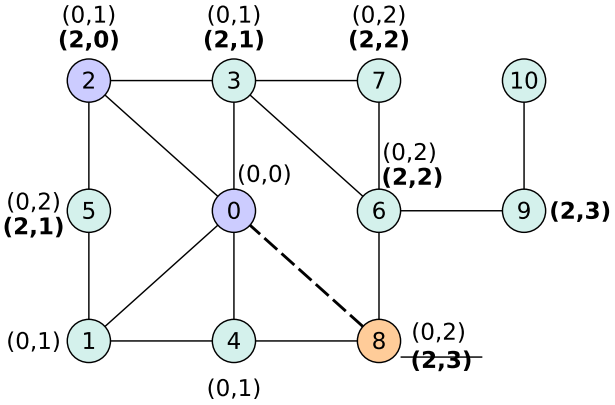
1. $G' \leftarrow G - \{(u,v)\}$
   //Construct SI_u for vertices in $AV_{(u,v)}(u)$
2. $SI_u \leftarrow \emptyset$
3. Initialize temporary labels $TL \leftarrow \emptyset$
4. **for all** vertex $r \in AV_{(u,v)}(u)$ (in ascending vertex order) **do**
5. Initialize supplemental label for r: $SL \leftarrow \emptyset$
6. Start BFS algorithm to compute all the distances from r to any vertices in $AV_{(u,v)}(v)$ that have larger vertex order than $\sigma(r)$ and record all temporary labels for all encountered vertices in TL; during the BFS process, if a new temporary label entry for a vertex w is redundant in TL, all neighbor vertices of w can be ignored by BFS
7. **for all** vertex t in $AV_{(u,v)}(v)$ that has $\sigma(t) > \sigma(r)$ and has been searched by the above BFS process **do**
8. **if** $(t, d_{G'}(t,r))$ is not a redundant label entry in SL **then**
9. $SL \leftarrow SL \cup (t, d_{G'}(t,r))$
10. **end if**
11. **end for**
12. $SI_u \leftarrow SI_u \cup (r, SL)$
13. **end for**//Construct SI_v for vertices in $AV_{(u,v)}(v)$
14. Repeat the above steps by mapping $u \leftarrow v$ and $v \leftarrow u$ to construct SI_v for vertices in $AV_{(u,v)}(v)$

0. Therefore, the BFS process starting from vertex 0 will stop at distance 2 and will not examine vertices 9 and 10. After the BFS process stops, we add a supplemental label entry to the supplemental label of vertex 8, resulting in $SL_{(0,8)}(8) = \{(0,2)\}$. At Step (2) in Figure 3.3, BFS process starts from vertex 2. Similarly, the BFS process starting from vertex 2 will stop at distance 3. In this process, three vertices can be pruned, including vertex 0, vertex 1 and vertex 4, for which we do not add any new labels. This is the *early-pruning strategy*. Then we may want to add another label entry $(2,3)$ into $SL_{(0,8)}(8)$. But based on the original index shown in Table 3.1 and the current supplemental label $SL_{(0,8)}(8) = \{(0,2)\}$, we find that $(2,3)$ is a redundant label entry in $SL_{(0,8)}(8) = \{(0,2)\}$ since the distance between vertex 2 and vertex 8 can be computed based on $SL_{(0,8)}(8) = \{(0,2)\}$ and the original index in Table 3.1. Finally, at Step (3), the BFS process will start from vertex 8. However, since no vertex



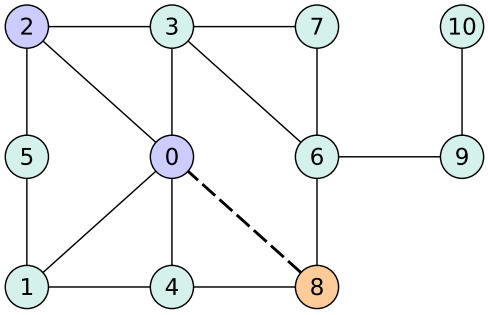
$SL_{(0,8)}(8)=\{(0,2)\}$

Step (1): Relabel affected vertex 8 from vertex 0



$SL_{(0,8)}(8)=\{(0,2)\}$

Step (2): Relabel affected vertex 8 from vertex 2



$SL_{(0,8)}(0)=\{\}$

$SL_{(0,8)}(2)=\{\}$

$SL_{(0,8)}(8)=\{(0,2)\}$

Step (3): Relabel affected vertices 0, 2 from vertex 8

Fig. 3.3 Supplemental index construction: BFS ALL on failed edge $(0,8)$

in in $AV_{(0,8)}(0)$ has smaller vertex order than vertex 8, no label entry will be added to the supplemental index at this step. The final supplemental index that is constructed for the failed edge $(0,8)$ in the graph shown in Figure 3.1 is shown at Step (3). We will show later in Section 3.4.4 that such supplemental index is adequate for distance query evaluation.

3.4.4 Distance Query Evaluation on SIEF

For each single-edge failure case, we classify all possible distance queries into different types. Suppose the graph is G, the original labeling index is L, the failed edge is (u,v), the affected vertices are in $AV_{(u,v)}(u)$ and $AV_{(u,v)}(v)$, and the supplemental index is $SI_{(u,v)}$ (here, $SI_{(u,v)} = SI_u \cup SI_v$). We also denote $G' = G - \{(u,v)\}$. Given any pair of vertices s,t, we would like to compute the distance between s,t in G', denoted as $d_{G'}(s,t)$. Then we have the following different cases:

- Case 1: $s \notin AV_{(u,v)}(u) \cup AV_{(u,v)}(v)$ and $t \notin AV_{(u,v)}(u) \cup AV_{(u,v)}(v)$
- Case 2: $s \notin AV_{(u,v)}(u) \cup AV_{(u,v)}(v)$ and $t \in AV_{(u,v)}(u) \cup AV_{(u,v)}(v)$, or similarly, $s \in AV_{(u,v)}(u) \cup AV_{(u,v)}(v)$ and $t \notin AV_{(u,v)}(u) \cup AV_{(u,v)}(v)$
- Case 3: $s \in AV_{(u,v)}(u)$ and $t \in AV_{(u,v)}(u)$, or similarly, $s \in AV_{(u,v)}(v)$ and $t \in AV_{(u,v)}(v)$
- Case 4: $s \in AV_{(u,v)}(u)$ and $t \in AV_{(u,v)}(v)$, or similarly, $s \in AV_{(u,v)}(v)$ and $t \in AV_{(u,v)}(u)$

Case 1 is trivial and we must have $d_{G'}(s,t) = d_G(s,t) = dist(s,t,L)$.

In Case 2 and in Case 3, according to Lemma 6 and the definition of affected vertices (see analysis in Section 3.4.2), we must also have $d_{G'}(s,t) = d_G(s,t) = dist(s,t,L)$.

In Case 4, suppose $s \in AV_{(u,v)}(u)$ and $t \in AV_{(u,v)}(v)$ (the other case can be analyzed in the same way). Obviously, distance between s and t changes to a larger value due

to the failed edge. If s and t become disconnected to each other in G', both will not have labels in $SI_{(u,v)}$, then we have $d_{G'}(s,t) = \infty$. If s and t is still connected in G' and without loss of generality, suppose the vertex order is $\sigma(s) < \sigma(t)$, then at least vertex t contains supplemental label entries. This is because in both the BFS ALL algorithm and the BFS BATCH algorithm, the affected vertex with minimum vertex order in $AV_{(u,v)}(u)$ (which is at most $\sigma(s)$) must produce one supplemental label entry for vertex t in $SI_{(u,v)}$ (see Lemma 3 for related details). For vertex s itself, if it does not produce any supplemental label entry for vertex t in $SI_{(u,v)}$, then it must be because the produced label entry is a redundant label. This means, in either case, the label entries of the supplemental label for vertex t in $SI_{(u,v)}$ must already contain adequate distance information for the computation of $d_{G'}(s,t)$. For example, to calculate $d_{G'}(2,8)$ in Figure 3.3, $SL_{(0,8)}(8) = \{(0,2)\}$ combining with $L(2) = \{(0,1)\ (2,0)\}$ in Table 3.1 is adequate and we can see that $d_{G'}(2,8) = 1+2 = 3$.

3.4.5 Some Remarks

Initial Index Construction. Pruned Landmark Labeling (PLL) technique presented in [80] is a state-of-the-art indexing technique for large static graphs. Indexes constructed by PLL [80] already have well-ordering property defined in Section 3.3. Therefore we use indexes constructed by PLL as the initial indexes for all original graphs in our experiments.

Time Complexity. Our algorithms can be directly applied on indexes constructed by PLL. Let w be the tree width [80] of G, n be the number of vertices and m be the number of edges in G. Also let (u,v) be the failed edge. Let $p = |AV_{(u,v)}(u) \cup AV_{(u,v)}(v)|$ for each single-edge failure case (on average). Further, according to analysis of PLL in [80], the number of label entries per vertex is $O(w \log n)$. Then the time complexity of the BFS ALL algorithm is $O(nw \log n + p^2 w \log n)$, where $O(nw \log n)$ is the time upper bound to build temporary labels TL (note that p BFS rounds are enough to

build the TL index that contains at most $nw \log n$ label entries) and $O(p^2 w \log n)$ is the time upper bound for redundancy tests.

Variants. Similar to Pruned Landmark Labeling (PLL) technique presented in [80], our SIEF approach can also be extended to support shortest path queries, weighted graphs, directed graphs and disk-based query answering. For more details, please refer to [80].

3.4.6 Applying SIEF in Multi-Edge Failure Graphs

SIEF can be extended to support dual-edge failure or even other types of multi-edge failure approximately. Take dual-edge failure as an example. Suppose that there is a shortest path from vertex s to vertex t, which contains two failed edges, $(u1, v1)$ and $(u2, v2)$ (as shown in Figure 3.4). In order to estimate the new distance with dual-edge failure using the SIEF framework, we need to divide the original shortest path into two parts: each part of the path contains only one failed edge. For example, in Figure 3.4, the shortest path can be divided into path from s to $v1$ and path from $v1$ to t. It is easy to see that we can use SIEF to calculate the length of these two paths efficiently. The sum of the length of these two paths can be used to estimate the new distance from s to t after $(u1, v1)$ and $(u2, v2)$ have failed. The dividing strategy here is called a *conservative strategy* because we give priority to the nearest vertex along the original path to include exactly one failed edge. The exception is the segment for the inclusion of the last failed edge, e.g., the path from $v1$ to t. We did not choose the path from $v1$ to $v2$ as the segment to include the failed edge $(u2, v2)$. This is because since it is the last failed edge, according to triangularity, we must have $d(v1, v2) + d(v2, t) \geq d(v1, t)$, which indicates the path from $v1$ to t is always a better choice.

Another path dividing strategy is called *aggressive strategy*, where we give priority to the furthest vertex along the path to include exactly one failed edge. For

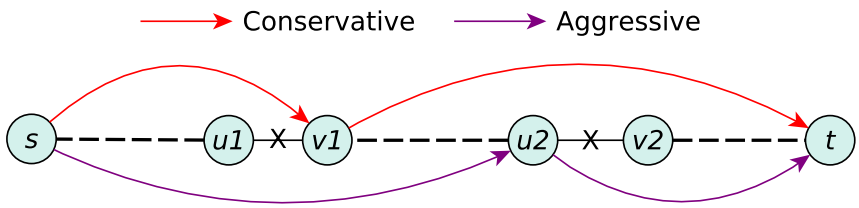


Fig. 3.4 Dual-Edge Failure Distance Estimation

example, we can divide the same path from s to t in Figure 3.4 into two paths: one from s to $u2$ and the other from $u2$ to t. In this way, we will obtain another estimation of the new distance between s and t. We use the smaller estimation as our final estimation of the new distance. Similar strategies can be applied to triple-edge failure or even cases with more than three failed edges.

It is worth mentioning that applying the conservative strategy to divide the original path from s to t is equivalent to applying the aggressive strategy to divide the same path from t to s.

Note that, the *disconnectedness* of a pair of vertices found by the SIEF framework is always correct. That is, if s and t in the above discussion become disconnected after multi-edge failure, the SIEF framework will be able to correctly identify the disconnection each time. This conclusion is due to the following facts: (1) SIEF always provides correct answers to distance queries with single-edge failure; (2) if s and t become disconnected after edge failures, there must exist at least one failed edge leading to the disconnection between s and t. When SIEF is trying to compute the new distance avoiding such failed edge using the above mentioned strategies, it must be able to correctly identify the disconnection caused by this failed edge as well. As will be shown in Section 3.5, disconnectedness has a great impact on the accuracy of SIEF on multi-edge failure.

3.5 Experiments

We evaluated the performance of our proposed SIEF approach and this section reports the results. All experiments were performed under Linux (Ubuntu 10.04) on a server provided by eResearch SA[2]. The server was running on Dell R910 with 32 processing cores (four 8-core Intel Xeon E7-8837 CPUs at 2.67 GHz), 1024 GB main memory and 3 TB local scratch disk. All methods were implemented in C++ (the code of PLL [80] was obtained from the first author's code repository on GitHub[3]) using the same GCC compiler (version 4.4.6) with the optimizer option O3. It is worth mentioning that although we have a large amount of main memory on the server, the memory usage of our approach is in fact quite small and as observed during our experiments, the memory usage was within 12 GB for all datasets.

3.5.1 Datasets

Due to lack of real IoT datasets modeled with graphs, we decided to use some closely related datasets in our work. Table 3.2 lists the six real-world datasets used in our experiments. More details on these datasets can be found at the Stanford Network Analysis Project website[4]. Similar to [95, 80], we treat all graphs as undirected, unweighted graphs.

A brief introduction of our datasets is provided below:

- `Gnutella` is a snapshot of the Gnutella peer-to-peer file sharing network collected in August 2002. Vertices represent hosts in the Gnutella network topology and edges represent connections between the hosts.

[2] http://www.ersa.edu.au/
[3] https://github.com/iwiwi/pruned-landmark-labeling
[4] http://snap.stanford.edu/

- The dataset `Facebook` consists of *circles* (or *friends lists*) from Facebook, which were collected from survey participants using a Facebook app called `Social Circles`.

- `Wiki-Vote` contains all Wikipedia voting data from the inception of Wikipedia till January 2008.

- `Oregon` is a graph of Autonomous Systems (AS) peering information inferred from Oregon route-views on May 26 2001.

- `Ca-HepTh` collaboration network of Arxiv High Energy Physics Theory category (there is an edge if authors coauthored at least one paper). The data covers papers in the period from January 1993 to April 2003 (124 months).

- `Ca-GrQc` collaboration network of Arxiv General Relativity category. Like `Ca-HepTh`, the data covers papers in the period from January 1993 to April 2003 (124 months).

It should be noted that, in Table 3.2, $|V|$ refers to the number of vertices and $|E|$ refers to the number of edges. In addition, IT denotes the indexing time or index construction time (in seconds) and LN denotes the average number of label entries of each vertex. We obtained these IT and LN results by using the Pruned Landmark Labeling (PLL) technique presented in [80]. As mentioned, we applied our index construction algorithms directly on the indexes constructed by PLL in our experiments.

Table 3.2 Real-world Datasets and Their Statistics

Dataset	$\|V\|$	$\|E\|$	**IT** (s)	**LN**
Gnutella	6,301	20,777	0.825	163.647
Facebook	4,039	88,234	0.173	25.887
Wiki-Vote	7,115	103,689	0.525	69.915
Oregon	11,174	23,409	0.080	11.189
Ca-HepTh	9,877	51,971	0.557	75.311
Ca-GrQc	5,242	28,980	0.141	43.828

3.5.2 Performance on Single-Edge Failure

We have conducted extensive experiments to validate our proposed approach on single-edge failure. In the experiments, we compared the average label entry numbers with and without considering edge failures (Section 3.5.2), index size (Section 3.5.2), and the numbers of affected vertices (Section 3.5.2). We performed queries with and without SIEF indexes (Section 3.5.2) and great efficiency improvement was observed if using SIEF indexes. Finally we studied the running time of our approach in terms of identification time, and relabeling time for each dataset (Section 3.5.2 and 3.5.2). Note that, we construct SIEF indexes by computing supplemental indexes for all single-edge failure cases of a given graph.

Supplemental Label Entry Numbers

Figure 3.5 shows the difference between the original label entry number (OLEN) without support of single-edge failure and the supplemental label entry number (SLEN) with support of single-edge failure. SLEN and OLEN of `Wiki-Vote`[5] have the biggest gap, i.e., the ratio of SLEN to OLEN is observed around 80. SLEN and OLEN of `Facebook` have the second biggest gap and the ratio of SLEN to OLEN is around 40. For other datasets, the ratios of SLEN to OLEN are all under 10. This means, compared with the total number of label entries needed for the original graphs without considering edge failures, in the case of edge failures, the total extra number of label entries (in supplemental indexes) is less than 10 times of the number of the label entries in the original index. These results indicate that the SIEF indexes are very compact.

[5]We use the first three letters in the names of each dataset (e.g., `Wik` for `Wiki-vote`) for better illustration in the figure.

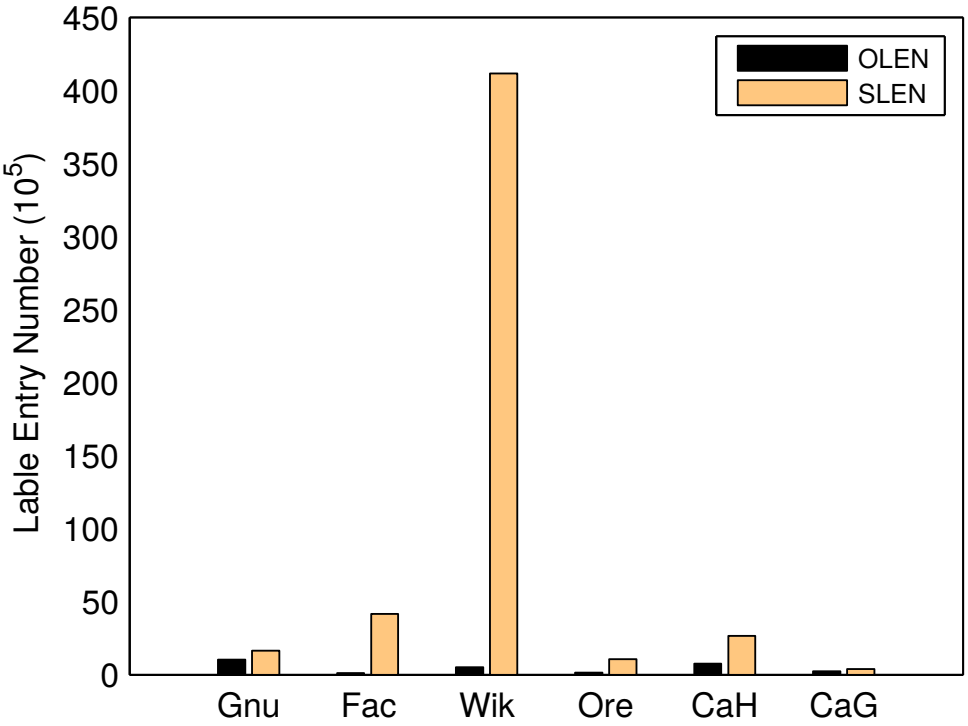


Fig. 3.5 Comparisons between supplemental label entry numbers (SLENs) and original label entry numbers (OLENs)

Index Size

Figure 3.6 shows the original index size for the graphs with no failed edges and the supplemental index size when considering edge failures. The sum of the original index size and the supplemental index size is the total index size for handling shortest path distances in graphs with all single-edge failure cases. From the figure, the `Gnutella` dataset shows comparatively smaller proportion of its supplemental index over its total index size while the `Facebook` dataset shows largest proportion of its supplemental index over the related total index size. The `Wiki-Vote` dataset has the largest supplemental index size due to the fact that each single-edge failure case incurs a large number of affected vertices as well as a relatively large number of supplemental label entries (for more details, please see Table 3.3).

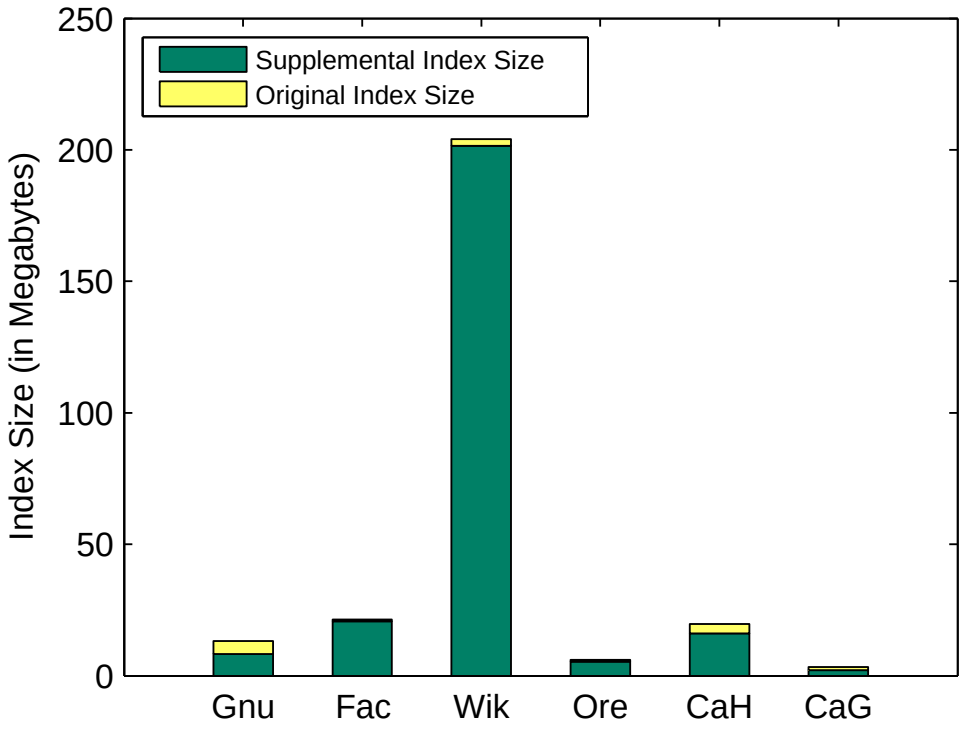


Fig. 3.6 Index Size

Affected Vertices

Table 3.3 presents the relationship between affected vertices and average supplemental label entry number. *Avg* $|AU|/|V|$ represents the average percentage of affected vertices in a single-edge failure case, showing the impact of a single-edge failure in a graph. It is also the average proportion of affected vertices of the original graphs. *Avg* $|AU|$ represents the average number of affected vertices from the graph and *Avg SLEN* denotes the average number of supplemental label entries in a single-edge failure case.

From the table, we can see that the smallest percentage and the smallest average number of affected vertices are both observed in the `Ca-GrQc` dataset, with values of 1.486% and 77.884, respectively. We can also see from the table that the average supplemental label number decreases (or increases) together with the average number of the affected vertices. Also around 36% of vertices are affected in the `Wiki-Vote`

dataset, which is the largest proportion. The largest average number of affected vertices is observed in the `Oregon` dataset, which is around 2,861 affected vertices for one failed edge. However, no clear linear relationship is found between the two. The largest gap occurs in the `Oregon` dataset, which indicates that the label pruning process on the affected vertices is quite powerful, leading to much fewer label entries per affected vertex. Meanwhile, the smallest gap happens in the `Gnutella` dataset and this indicates that label pruning is not very effective in this dataset.

Note that, although the proportion of affected vertices for a single-edge failure case could be large, as having been clarified in Figure 3.6, the final SIEF index for all single-edge failure cases is still of moderate sizes compared with the original index.

Table 3.3 Affected Vertices

Dataset	**Avg** $\|AU\|/\|V\|$	**Avg** $\|AU\|$	**Avg SLEN**
Gnutella	6.053%	381.386	78.445
Facebook	16.099%	650.241	47.042
Wiki-Vote	35.841%	2,550.090	396.971
Oregon	25.605%	2,861.070	45.323
Ca-HepTh	2.743%	270.881	51.095
Ca-GrQc	1.486%	77.884	13.064

Query Time

Table 3.4 shows the average BFS query time and the average SIEF query time. The former represents query time without using indexes proposed in this work, while the latter represents the query time when using SIEF indexes. From the table, we can see that the difference for `Oregon` dataset is the least. But using SIEF indexes still achieves at least 40 times faster when compared with the traditional BFS query approach. The largest gap occurs in the `Facebook` dataset, where the average BFS query time is around 500 times more than the SIEF query time. These results show that when using SIEF indexes, the query efficiency can be improved significantly and the query response times are normally no more than 5 μs. As mentioned in Section

3.4, we use supplemental indexes to support edge failures, the query process needs to examine the supplemental indexes first. When examining the supplemental indexes, SIEF checks whether the querying source and querying destination are both affected vertices given the edge failure constraint using binary search strategy. Based on the searching result, SIEF can find out whether we can compute the shortest path distance based only on the supplemental indexes, based on both the supplemental indexes and the original indexes, or based only on the original indexes. Nevertheless, the querying process is still much faster. The main reason is that the number of affected vertices for each single-edge failure case is typically small (more details are presented in Section 3.5.2) and hence the binary search process finishes quickly. This results in fast query responses in SIEF.

Table 3.4 Average Query Time

Dataset	**BFS Query Time**	**SIEF Query Time**
Gnutella	140.329 μs	0.452 μs
Facebook	243.060 μs	0.522 μs
Wiki-Vote	284.867 μs	1.100 μs
Oregon	163.465 μs	4.985 μs
Ca-HepTh	325.196 μs	0.689 μs
Ca-GrQc	159.412 μs	0.479 μs

Identification Time

Table 3.5 shows the total time for identifying affected vertices for all single-edge failure cases. From the figure, we can see that, for the most datasets, the identification process can be done fairly fast and is normally finished within 80 seconds. The exception is `Wiki-Vote`, which requires a bit more than 600 seconds. The fast identification time is mainly because the affected vertices can be identified in a BFS manner and we only need to examine the distances between the affected vertices to one of the end vertices of a failed edge.

Table 3.5 Average Identification Time

Dataset	Identification Time
Gnutella	43.3708 s
Facebook	80.6844 s
Wiki-Vote	612.522 s
Oregon	35.6307 s
Ca-HepTh	36.2022 s
Ca-GrQc	4.32942 s

Labeling Time

Figure 3.7 shows the time for relabeling the affected vertices, which need extra distance label information to maintain correct distances to some other vertices due to a single-edge failure. Here, we used the estimated time for naive method (shown as "Estd Time for Naive Method" in the figure) as the baseline. The naive method refers to the method that we recompute a complete distance labeling index for each single-edge failure case. The process of labeling a new graph with a single-edge failure should be almost the same as the process of labeling the original graph. Therefore, the total labeling time of the naive method can be estimated by multiplying the total edge number in the original graph, i.e., the total number of single-edge failure cases, with the index time of the original graph (see Table 3.2).

Then, we compared the labeling times of the naive method and the the labeling method proposed in our work, BFS ALL. From the figure, we can see that on some datasets, such as `Gnutella`, `Wiki-Vote`, `Ca-HepTh` and `Ca-GrQc`, BFS ALL outperforms the naive method an order of magnitude. On `Facebook`, BFS ALL can also perform close to an order of magnitude faster than the naive one. However, BFS ALL is only slightly better than the naive method on `Oregon` and the cause is that this dataset contains a large number of affected vertices and the label-pruning strategy in BFS ALL does not work well. Overall, BFS ALL method outperforms the naive method in most datasets and can reduce the labeling time by an order of magnitude.

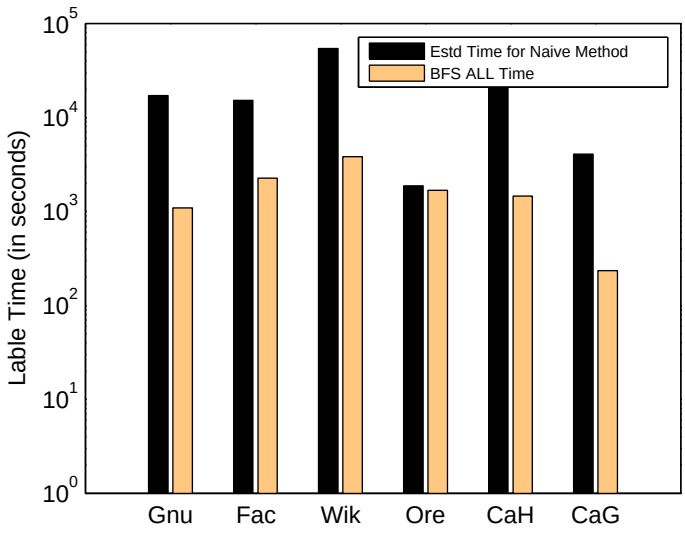


Fig. 3.7 Labeling Time

3.5.3 Performance on Multi-Edge Failure

We have conducted experiments to evaluate SIEF in multi-edge failure graphs, including correctness and disconnectedness, distance error ratio, and average query time. All experimental results reported here were obtained by performing separate multi-edge failures along 100,000 shortest paths between 100,000 random pairs of vertices.

When we performed multi-edge failures along a shortest path, we randomly chose the failed edges. If the shortest path contained insufficient edges, we would looked up a different pair of vertices and tried to perform multi-edge failure again. Such processes finished when we had identified a large enough number of multi-edge failure tests.

Correctness and Disconnectedness

We study correctness and disconnectedness in Figure 3.8 for dual-edge failure. Here, correctness refers to the proportion of correct answers provided by the SIEF frame-

work, including disconnectedness, which is the proportion of disconnections occurred in all 100,000 tests.

From Figure 3.8, we see that the correctness is more than 30% for five datasets and some correctness may exceed 40%. The exception is `Gnutella`, whose correctness is slightly better than 20%. In terms of disconnectedness, we can easily spot that the disconnectedness is very high in `Gnutella`, `Wiki-Vote` and `Oregon` and is very close to the correctness. In contrast, the gaps between correctness and disconnectedness are much larger in `Facebook`, `Ca-HepTh` and `Ca-GrQc`. This indicates that in these datasets, SIEF can correctly answer a large proportion of distance queries for pairs of vertices that are still connected after dual-edge failure.

We then study correctness and disconnectedness in Figure 3.9 for triple-edge failure. It is interesting to observe that the correctness in `Gnutella`, `Wiki-Vote` and `Oregon` improves with an increase about 10% to 15% compared with dual-edge failure, largely due to the increase of disconnectedness. In the meantime, the correctness in `Facebook`, `Ca-HepTh` and `Ca-GrQc` drops dramatically while disconnectedness is similar to that of dual-edge failure. Compared with the correctness for dual-edge failure, this indicates that SIEF fails to answer a large proportion of distance queries exactly for pairs of vertices that are still connected after introducing one more failed edge.

Due to the fact that the more failed edges we have on a shortest path, the more disconnections we would have, we just omit the experimental results of other types of multi-edge failure.

Distance Error Ratio

In Table 3.6, average distance error ratio is studied. Here, distance error ratio is defined as $\frac{|estimated\ distance - actual\ distance|}{actual\ distance}$. From the table, we can see that for dual-edge failure, the average distance error ratio is normally less than 0.5, except `Gnutella`,

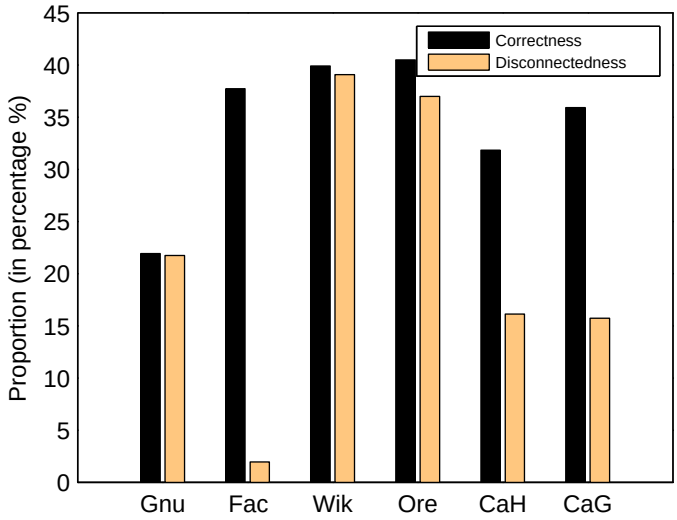


Fig. 3.8 Correctness and Disconnectedness on Dual-Edge Failure

where the average distance error ratio is 0.663. For triple-edge failure, the average distance error ratio is about doubled compared to that of dual-edge failure.

Table 3.6 Average Distance Error Ratio

Dataset	**Dual-Edge Failure**	**Triple-Edge Failure**
Gnutella	0.663	1.311
Facebook	0.194	0.386
Wiki-Vote	0.454	0.944
Oregon	0.379	0.760
Ca-HepTh	0.223	0.407
Ca-GrQc	0.195	0.378

Query Time on Multi-Edge Failure

The query time performance on multi-edge failure is presented in Table 3.7. Compared with the query time performance in Table 3.4, we see that, the query time on dual-edge failure is around 4 times of that on single-edge failure while the query time on triple-edge failure is around 6 to 7 times of that on single-edge failure. This indicates a linear growth of query time for multi-edge failure cases.

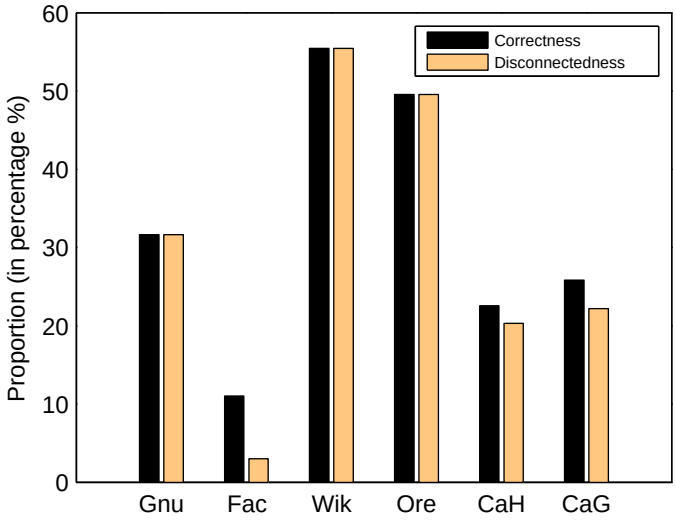


Fig. 3.9 Correctness and Disconnectedness on Triple-Edge Failure

Table 3.7 Average Query Time

Dataset	**Dual-Edge Failure**	**Triple-Edge Failure**
Gnutella	1.785 μs	3.041 μs
Facebook	2.051 μs	3.502 μs
Wiki-Vote	4.332 μs	7.453 μs
Oregon	19.741 μs	33.297 μs
Ca-HepTh	2.734 μs	4.620 μs
Ca-GrQc	1.882 μs	3.109 μs

3.6 Summary

In this chapter, we have studied the computation of shortest path distance in failure-prone graphs, a fundamental and challenging problem in many applications in the smart city computing domain, such as dynamic taxi ridesharing, control of traffic congestion, urban road network planning, etc.

In particular, we have focused on the constructions of compact distance labeling for all possible single-edge failure cases. A generic framework, SIEF, has been designed for this purpose. Based on the most recent technique Pruned Landmark Labeling (PLL) [80] that handles only static graphs, we have implemented an extended version using the SIEF framework developed in this chapter. Extensive experiments

have also been performed using six real-world graphs to confirm its effectiveness and efficiency. SIEF is able to support compact index construction for all single-edge failure cases in graphs efficiently. Specifically, the SIEF index size is comparable to that of the indexes constructed for original static graphs, which is very compact. SIEF can answer distance queries with single-edge failure constraints several orders of magnitude faster than traditional Breadth-First-Search (BFS) algorithms. Moreover, SIEF can answer distance queries on multi-edge failure with high accuracy and fast response time.

Chapter 4

Matching over Linked Data Streams

This chapter leverages semantic technologies, such as Linked Data, which can facilitate machine-to-machine (M2M) communications to build an efficient information dissemination system for semantic IoT. The system integrates Linked Data streams generated from various data collectors and disseminates matched data to relevant data consumers based on triple pattern queries registered in the system by the consumers. We also design two new data structures, *TP-automata* and *CTP-automata*, to meet the high performance needs of Linked Data dissemination. We evaluate our system using a real-world dataset generated from a Smart Building Project. With the new data structures, the proposed system can disseminate Linked Data faster than the existing approach with thousands of registered queries. This chapter is based on our research reported in [96, 97].

4.1 Overview

As of 2012, 2.5 quintillion (2.5×10^{18}) bytes of data were being created daily[1]. In IoT, connecting all of the things that people care about in the world becomes possible [98]. However, all these things will produce much more data than nowadays. The

[1] http://www-01.ibm.com/software/data/bigdata/

volumes of data are vast, the generation speed of data is fast and the data/information space is global. By exploiting such data in IoT, cities will become smarter and more efficient. Some promising IoT applications in future smart cities include resources management issues for modern cities [99], and effective urban street-parking management for reducing traffic congestion and fuel consumption [100]. Indeed, IoT is one of the major driving forces for *big data analytics*.

Given the scale of IoT, topics such as distributed processing, real-time data stream analytics, and event processing are all critical, and need to be revisited in order to improve upon existing technologies for applications of this scale [2, 101]. In this context, semantic technologies such as Linked Data (see http://linkeddata.org/), which aim to facilitate machine-to-machine (M2M) communications, play an increasingly important role [102]. Linked Data is part of a growing trend towards highly distributed systems, with thousands or potentially millions of independent sources providing structured data. Due to the large amount of data produced by various kinds of things, one challenging issue is how to *efficiently disseminate data* to relevant data consumers.

In this chapter, we focus on studying the Internet of Things (IoT) from a *data perspective*. As in IoT, data is processed differently compared with the processing in the traditional Internet environments (i.e., Internet of Computers). In the Internet of Computers, both the main data producers and consumers are human beings. However, in the Internet of Things, the main actors become *things*, which means things are the majority of data producers and consumers. Therefore, in IoT, 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 deal with such challenge, it is imperative to efficiently retrieve the most relevant data from the big data generated in IoT and effectively extract useful information

(e.g., in the process converting "data" into "information" or "knowledge"). We propose in this chapter an efficient data dissemination system for semantic IoT by leveraging semantic technologies, such as Linked Data. Our system will be very helpful and efficient in the retrieval of relevant data from the deluge of IoT data, which can then facilitate the extraction of required information. The system firstly integrates data generated from various data collectors. Then it transforms all the data into Linked Data streams in Resource Description Framework (RDF) format (see www.w3.org/RDF). Meanwhile, data consumers can register their interests in the form of Basic Graph Patterns (BGPs) composed of simple triple patterns 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. It should be noted that we will not discuss the data processing at the user side, instead we focus ourselves on how to efficiently match a large number of BGP queries against Linked Data streams in batch mode. We highlight our main contributions in the following.

- We identify new Linked Data dissemination needs in the context of the Internet of Things, which requires to process continuous data requests in batch mode efficiently.

- We develop two new data structures, Triple Pattern automata (TP-automata) and Conjunctive Triple Pattern automata (CTP-automata), for efficiently matching Linked Streams against a large number of single or conjunctive triple pattern queries based on automata techniques. We also develop novel techniques to evaluate these queries efficiently.

- We conduct extensive experiments using a real-world dataset generated in a Smart Building Project. The results show that: 1) when processing single triple patterns using TP-automata, the system can disseminate Linked Data at one

million triples per second with 100,000 registered user queries, which is several orders of magnitude faster than existing techniques; 2) when processing conjunctive triple patterns using CTP-automata, the system can disseminate Linked Data at a speed of an order of magnitude faster than the existing approaches with thousands registered conjunctive queries.

The rest of this chapter is organized as follows. We present the framework and the technical details of our approach in Section 4.2. In Section 4.3, we report the results of an extensive experimental study. In Section 4.4, we review the related work. Finally, we present some concluding remarks in Section 4.5.

4.2 Linked Data Dissemination System

In order to disseminate high-quality information and provide high-performance matching services to data consumers (or subscribers), we aim to design a system that will not return false-negative match results. Therefore, we investigate pattern matching in this article. Pattern matching performs individual component matching between RDF triples and BGPs. It does not consider semantic relatedness between an RDF triple and a BGP. It may return false positive matching results but not false negative ones. Recent work on pattern matching includes Linked Data stream processing [31] and stream reasoning [30]. However, since these solutions are mainly designed for optimizations of individual query evaluations, they are not 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 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 summarization techniques among multiple datasets [103] to avoid querying all known datasets directly. In contrast, semantic matching will match semantically related triples compared to a specified pattern [104]. 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.

4.2.1 System Overview

Figure 4.1 shows an overview of our system in the smart city scenario. We assume that data generated by all kinds of things will be represented in the form of Linked Data streams using RDF (for the purpose of facilitating (M2M communications). Our system consists of two main 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. The index construction component constructs an index for all user queries. The matching component evaluates the incoming Linked Data streams against the constructed index for efficiently matching triples to the user queries. Finally, the system disseminates the matched data to relevant data consumers for their further processing.

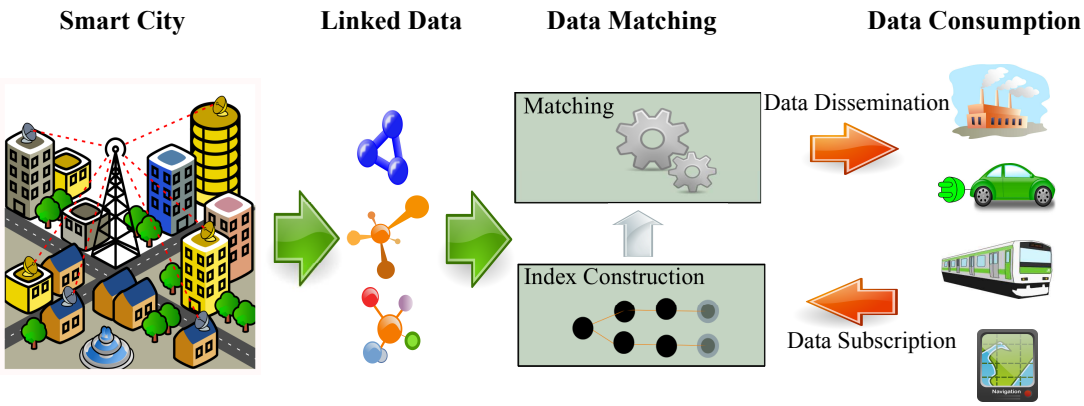


Fig. 4.1 System Overview

User Queries. Similar to [105, 106], triple patterns are adopted as the basic units of user queries in our system. A triple pattern is an expression of the form `(s, p, o)` where `s` and `p` are URIs or variables, and `o` is a URI, a literal or a variable. The eight possible triple patterns 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 in [103], we can also apply hash functions[2] to map these patterns into numerical values.

When a user query contains only one triple pattern, such queries are called single triple pattern queries. Meanwhile, a user query can also be expressed as a *conjunctive* triple pattern query composed of multiple triple patterns [105, 106]. Conjunctive queries can express data needs much more accurately compared with single triple pattern queries. A conjunctive query q has the form of:

$$?x_1, \ldots, ?x_n : (s_1, p_1, o_1) \wedge (s_2, p_2, o_2) \wedge \cdots \wedge (s_n, p_n, o_n)$$

where $?x_1, \ldots, ?x_n$ are variables, each (s_i, p_i, o_i) is a triple pattern, and each variable $?x_i$ appears in at least one triple pattern (s_i, p_i, o_i). Variables will always start with the '?' character. Variables $?x_1, \ldots, ?x_n$ are also called *answer variables* in order to distinguish them from other variables in the query.

Representations of Queries and Triples. In our Linked Data dissemination system, when the user queries (in the form of single or conjunctive triple pattern queries) are registered, all queries are transformed into numerical values. The reason for this is that the comparisons between numbers are faster than strings [103]. Note that, in such case, we will have three numbers for the three components in a query as described above. Then a suitable index can be constructed for efficient evaluation

[2]There are many different hash functions that are suitable for this purpose. For more details, please refer to [103].

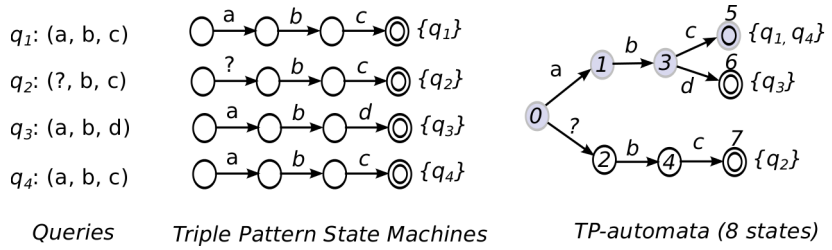


Fig. 4.2 Structure of TP-automata

between Linked Data streams and user queries. Before a matching process starts, RDF triples in the data streams will be mapped into numerical values as well. Then, these numerical represented triples will be matched with conjunctive queries represented as numerical values in the constructed indexes.

4.2.2 TP-automata for Single Triple Pattern Query Matching

Automata techniques have been adopted to process XML data streams [107]. They are mainly based on languages with SQL-like syntax, and relational database execution models adapted to process streaming data. In our system, to support *pattern matching*, we also apply automata to match each individual component of a triple with its counterparts of triple patterns in single triple pattern queries efficiently. We call this approach *Triple Pattern automata* (TP-automata).

As mentioned, operating on numbers is more efficient than operating on strings. Note that when we map triple patterns into numerical values, we treat variables in a triple pattern as a universal match indicator, e.g., 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 4.2 depicts the construction process of TP-automata. Firstly, user queries will be transformed into triple pattern state machines as shown in the middle of Figure 4.2. 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 [107], these state machines can be combined into one machine by exploiting shared common states with same transitions. The combined machine, TP-automata, is shown on the right of Figure 4.2. The shaded circles represent combined states.

To perform pattern matching over TP-automata, triples in the Linked Data stream will be firstly 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 firstly 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 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 match process stops if and only if all current active states are final states or states with no satisfied transition.

4.2.3 CTP-automata for Conjunctive Triple Pattern Query Matching

We also apply automata to match each individual component of a triple with its counterparts of triple patterns in conjunctive triple pattern query efficiently. Similarly, we call this approach *Conjunctive Triple Pattern automata* (CTP-automata).

Construction of CTP-automata. Figure 4.3 depicts the construction process of CTP-automata. There are two conjunctive queries, $q_1 : (?x1, b, c) \wedge (?x1, d, e)$ and $q_2 : (?x2, b, c) \wedge (?x2, d, e) \wedge (a, d, ?x2)$. Accordingly, there are two triple patterns in q_1 and three triple patterns in q_2. Firstly, all the triple patterns in the conjunctive queries will be transformed into triple pattern state machines as shown in the middle of Figure 4.3. As can be seen from the middle part of 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 represent three transitions between different states.

Suppose there are n conjunctive queries for the construction of CTP-automata and each query contains p patterns on average. Then the time complexity of the CTP-automata construction process will be $O(np)$. This is because we can add each pattern into the CTP-automata in an incremental manner and each pattern will require constant time to be inserted (we can adopt hashing based data structures to achieve constant time insertion of each pattern).

It is worth mentioning that we ignore variable names at this stage due to the fact that when processing triples in the Linked Data stream, at the first step, we have to evaluate these triples one by one and that variable naming does not have any relationships between different conjunctive queries. For example, $(?x1, b, c) \wedge (?x1, d, e)$ and $(?x2, b, c) \wedge (?x2, d, e)$ actually refer to the same conjunctive query. However, the variable naming does matter within the same conjunctive query. For example, in $(?x1, b, ?x2) \wedge (?x2, d, e)$, variables $?x1$ and $?x2$ refer to different triple components. We leave the resolution of different variable names within the same conjunctive query in the later stage, called `Conjunctive Constraints Resolution` (CCR) stage. Before that, we need to evaluate each triple against each single triple state machine first, which is the `Triple Pattern Matching` (TPM) stage.

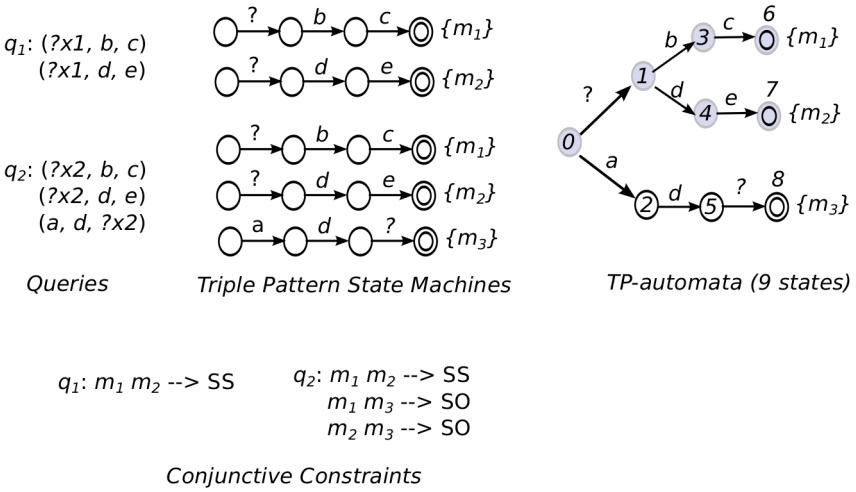


Fig. 4.3 Index Structure and Conjunctive Constraints of CTP-automata

Similar to [107], the multiple single triple state machines shown in Figure 4.3 can be combined into one triple state machine by exploiting shared common states with same transitions. The combined machine, CTP-automata, is shown on the right of Figure 4.3. The shaded circles represent combined states. We can see from the figure that, although we have five single triple state machines, after the combination, the number of single triple state machines drops to three, which have been labeled as m_1,m_2 and m_3, respectively.

Matching Triple Streams against CTP-automata. During the TPM stage, in order to perform *pattern matching* over CTP-automata, when a triple `(a, b, c)` arrives, our system firstly checks the initial state of CTP-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 state 1 and state 2. Following the transitions, only state 3 becomes active state and there is no transition triggered from state 2. Finally, following the transition with condition `c` or `?` from state 3, one final state, state 6, is reached. By checking this final state, the system returns $\{m_1\}$ as the matching result.

The matching process stops if and only if all current active states are final states or states with no satisfied transition.

At this TPM stage, the matching results are only intermediate results and the matched triples are just possible candidates which may satisfy some conjunctive queries. In order to determine which conjunctive queries have been satisfied, we need to further evaluate some `conjunctive constraints`, which will be detailed next.

Conjunctive Constraints Resolution (CCR) of CTP-automata. It should be noted that in order to match q_1 and q_2 in Figure 4.3, all triple patterns they contain must be matched first. Take query $q_1 : (?x1, b, c) \wedge (?x1, d, e)$ as an example. Suppose that triples t_1 and t_2 match triple patterns $(?x1, b, c)$ and $(?x1, d, e)$, respectively. To ensure that query q_1 can be satisfied by t_1 and t_2, we need to check first that whether we have $t_1.s = t_2.s$. We call such conditions as *conjunctive constraints* of a conjunctive query. All conjunctive constraints must be satisfied before we can assure that a conjunctive query is satisfied. As mentioned before, the conjunctive constraints checking occurs at the CCR stage.

In this chapter, we have identified ten conjunctive constraints, including SS, PP, OO, SO, OS, SSPP, SSOO, PPOO, SOPP, OSPP. Constraint SS means that the subjects of two candidate triples must be matched. More details are shown in Table 4.1. These constraints can be used to determine whether a conjunctive query has been satisfied or not so far in the stream.

For example, in Figure 4.3, query q_1's conjunctive constraint is $m_1m_2 \rightarrow$SS and query q_2's conjunctive constraints are $m_1m_2 \rightarrow$SS, $m_1m_3 \rightarrow$SO and $m_2m_3 \rightarrow$SO. Suppose that triples t_1, t_2, t_3 match triple pattern machines m_1, m_2, m_3, respectively. According to Table 4.1, for t_1, t_2 to satisfy q_1, we need to have $t_1.s = t_2.s$. Similarly, for t_1, t_2, t_3 to satisfy q_2, we need to have $t_1.s = t_2.s$, $t_1.s = t_3.o$ and $t_2.s = t_3.o$.

Table 4.1 Conjunctive Constraints

Conjunctive Constraints	**Description**	**Checking Details**
SS	The subjects of two candidate triples must be matched	$t_1.s = t_2.s$
PP	The predicates of two candidate triples must be matched	$t_1.p = t_2.p$
OO	The objects of two candidate triples must be matched	$t_1.o = t_2.o$
SO	The subject of a candidate triple in the first pattern machine matches the object of a candidate triple in the second pattern machine	$t_1.s = t_2.o$
OS	The object of a candidate triple in the first pattern machine matches the subject of a candidate triple in the second pattern machine	$t_1.o = t_2.s$
SSPP	The conjunction of both SS and PP constraints	$t_1.s = t_2.s$ and $t_1.p = t_2.p$
SSOO	The conjunction of both SS and OO constraints	$t_1.s = t_2.s$ and $t_1.o = t_2.o$
PPOO	The conjunction of both PP and OO constraints	$t_1.p = t_2.p$ and $t_1.o = t_2.o$
SOPP	The conjunction of both SO and PP constraints	$t_1.s = t_2.o$ and $t_1.p = t_2.p$
OSPP	The conjunction of both OS and PP constraints	$t_1.o = t_2.s$ and $t_1.p = t_2.p$

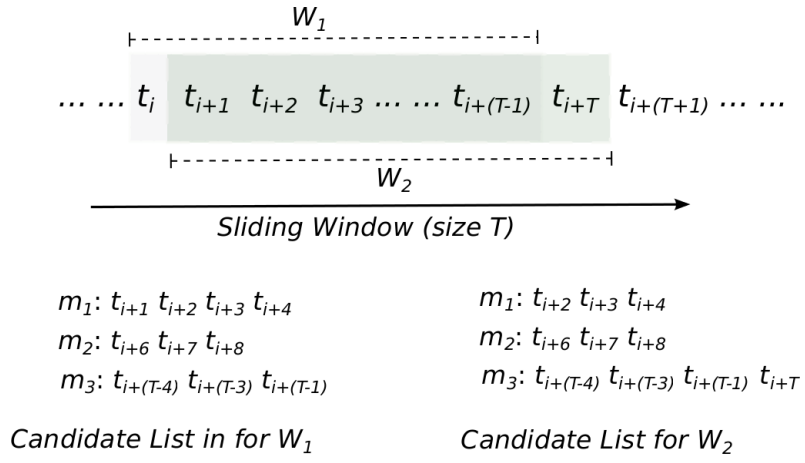


Fig. 4.4 Maintenance of Candidate Triple List

Dynamic Maintenance of the Matching Candidate List. In order to check conjunctive constraints, triples in the stream that match some triple pattern machines will be buffered for this purpose. Since the Linked Stream can be considered infinite, the buffered triple lists for triple pattern machines may grow all the time. To avoid this issue, we need to specify a sliding window to confine our matching scope. That is, we only consider matching within the sliding window.

Figure 4.4 shows two sliding windows with size T: w_1 and w_2, where only the most recent T triples will be considered for our matching. In order to evaluate conjunctive constraints, we need to update the buffered candidate triple list each time a triple arrives in or leaves the window. In Figure 4.4, for w_1, we have got matching results for all three single triple pattern machines, $m1, m2, m3$, in Figure 4.3. After receiving a new tripe, t_{i+T}, the oldest triple t_i will be removed from all candidate lists that contain t_i. In this example, only candidate list for $m1$ contains t_i and hence t_i will be removed from that candidate list. Further, suppose the new arriving triple t_{i+T} will be matched with machine $m3$. Then t_{i+T} will be added to the candidate list for $m3$. It is obvious that, each time when the sliding window moves forward by one triple, we should consider all the buffered lists affected by the leaving triple and the joining triple in the sliding window to verify conjunctive constraints.

4.3 Experimental Evaluation

In this section, we report our experimental evaluation of the proposed approach. We will first describe the experimental settings, and then report the experimental results.

4.3.1 Experimental Setup

The dataset used in our experiments was generated in a Smart Building Energy Project [108]. The energy readings were collected from 4–19 August 2014. In total, there are around 6.2 million triples in the dataset. An event example is depicted in Figure 4.5. 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.

We evaluated the performance of our approach in terms of *average construction time* (in milliseconds) of the indexes and *average throughput* (in number of triples per second). We compared hash-based implementation (i.e., mapping triples and queries into numerical values, denoted as *HashMat* in the figures) with string-based implementation (i.e., using triples and queries as it is, denoted as *StringMat* in the figures). We also compared our methods with an existing approach, CQELS [31], which supports parallel query evaluation on Linked Data streams. Both TP-automata and CTP-automata engines, and CQELS[3], were all 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 main memory. We ran each experiment 10 times in order to ensure consistency of results and reported the average experimental results.

[3]The source code and documentation of CQELS can be obtained via *http://code.google.com/p/cqels/*

```
@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"    .
```

Fig. 4.5 An Event Example

4.3.2 Evaluation of TP-automata

As an initial work, we used simple BGPs that contain only a single triple pattern in a query as queries in the experiment. We randomly generated BGPs using the seven patterns mentioned in Section 2 based on our dataset. We did not consider (?s, ?p, ?o) in our experiment as it requires every triple in the Linked Data stream. In such case, no query index is needed. We generated from 10,000 queries to 100,000 queries.

Firstly, average construction time is compared in Figure 4.6. The construction times for both hash-based TP-automata and string-based TP-automata are similar to each other in most settings. For larger numbers of queries, such as 75k and 100k queries, the construction of string-based indexes takes slightly longer time. Normally, the construction can be completed within a few hundred milliseconds. However, the construction time of CQELS takes much longer, which normally requires around ten thousand milliseconds.

Throughput performance of pattern matching is depicted in Figure 4.7. It shows some large 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 optimizing evaluation of queries with complex operators

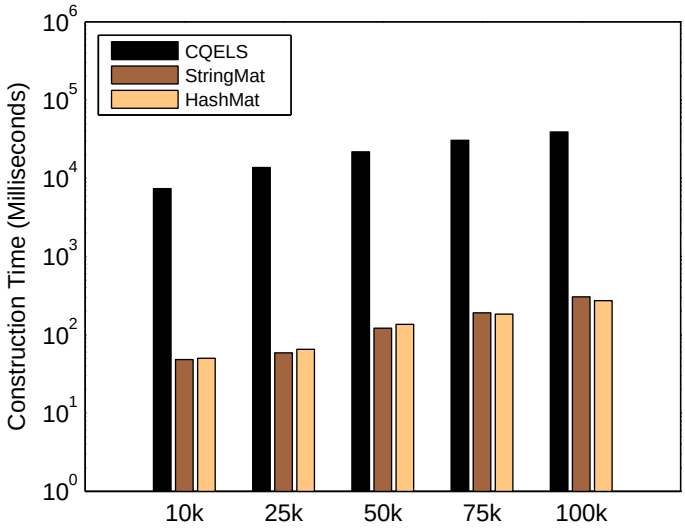


Fig. 4.6 Average Construction Time of TP-automata

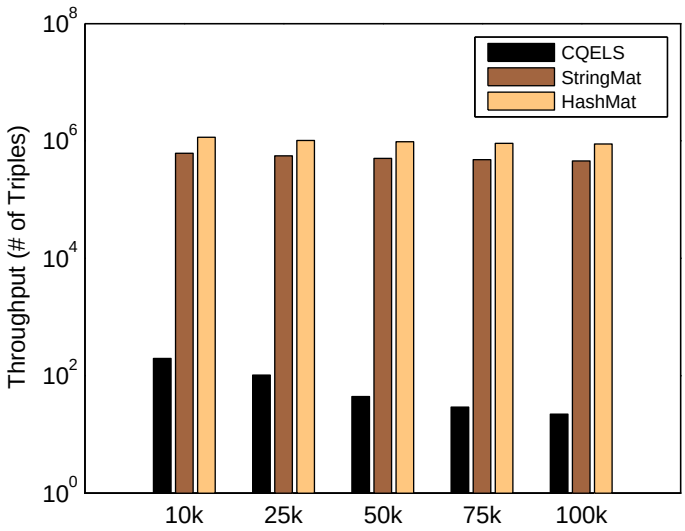


Fig. 4.7 Average Throughput Evaluation of TP-automata

and semantics but not on evaluation of a large set of concurrent and simple queries over Linked Data streams. In this regard, our approach can also be adapted to complement CQELS for dealing with our Linked Data dissemination scenario. Regarding HashMat and StringMat, in most cases, HashMat is about twice throughput speed compared with StringMat.

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%

Table 4.2 Matching Quality of HashMat (When the Recall is 100%) in TP-automata

Finally, we also investigated the matching quality of hash-based TP-automata (HashMat) via Precision, Recall and F_1 Score [109]. 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 look into Precision and F_1 Score when the Recall is 100%. As can be seen in Table 4.2, the Precision and F1 Score are 100% when the number of queries is 10k or 25k. For larger numbers of queries (e.g., 50k, 75k and 100k), both Precision and F_1 Score are still greater than 99.99950%. This demonstrates that HashMat provides very high matching quality.

4.3.3 Evaluation of CTP-automata

Again, we used random walk method to generate conjunctive triple pattern queries in the experiments according to the data graph of the event data. The details of parameters we used for generating these queries are shown in Table 4.3. The parameters include `query number`, `pattern number`, and `window size`.

Construction Time. The average construction times of CTP-automata engines and CQELS engine is presented in Figure 4.8. The construction times for both hash-based CTP-automata matching engine (HashMat) and string-based CTP-automata matching engine (StringMat) are close to each other in most settings and are always under 50 milliseconds. The construction of the string-based indexes takes slightly longer time. By contrast, the construction times of CQELS are much longer than CTP-automata engines. The main reason is that CQELS has to parse the conjunctive triple pattern

Parameter	Range	Default	Description
Query Number	100 to 2000	1000	The number of conjunctive triple pattern queries
Pattern Number	1 to 5	3	The maximum number of triple patterns in a query
Window Size	10 to 500	100	The window size, in terms of number of triples, for the evaluation of conjunctive triple pattern queries

Table 4.3 Workload Parameters for the Experiments of CTP-automata

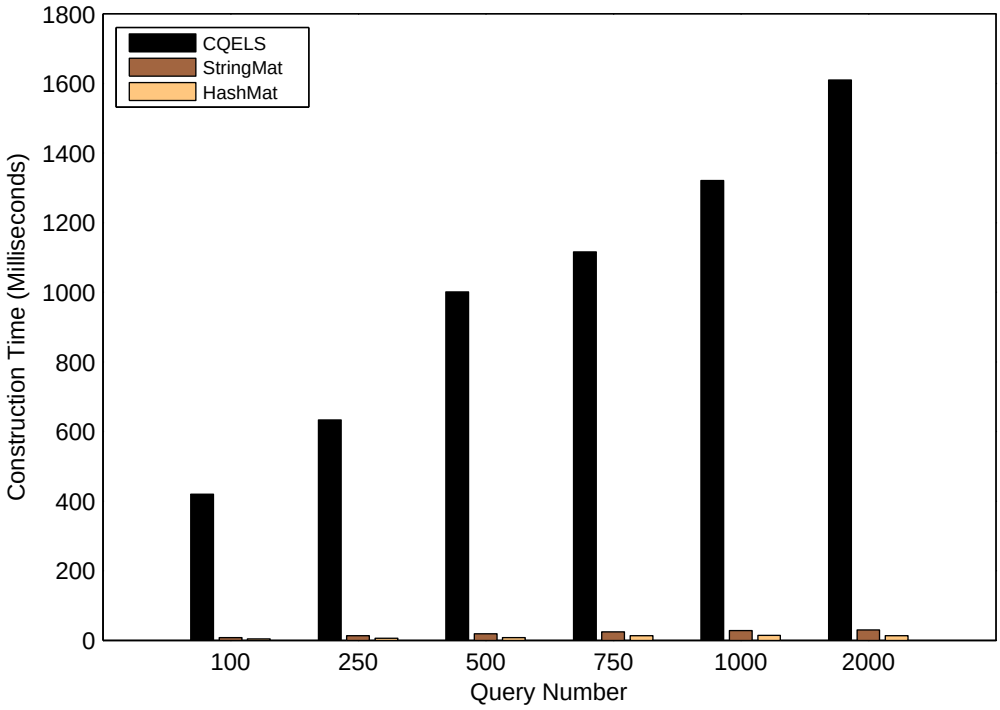


Fig. 4.8 Average Construction Time of CTP-automata

queries using a SPARQL-like parser and then register the parsed queries in the processing engine. As shown from Figure 4.8, the construction times of CQELS grow linearly with the number of conjunctive queries. When the query number is 100, the construction time is around 400 milliseconds. When the number of queries increases to 2000, the construction time reaches above 1610 milliseconds. This indicates that the construction of our CTP-automata engines is very fast.

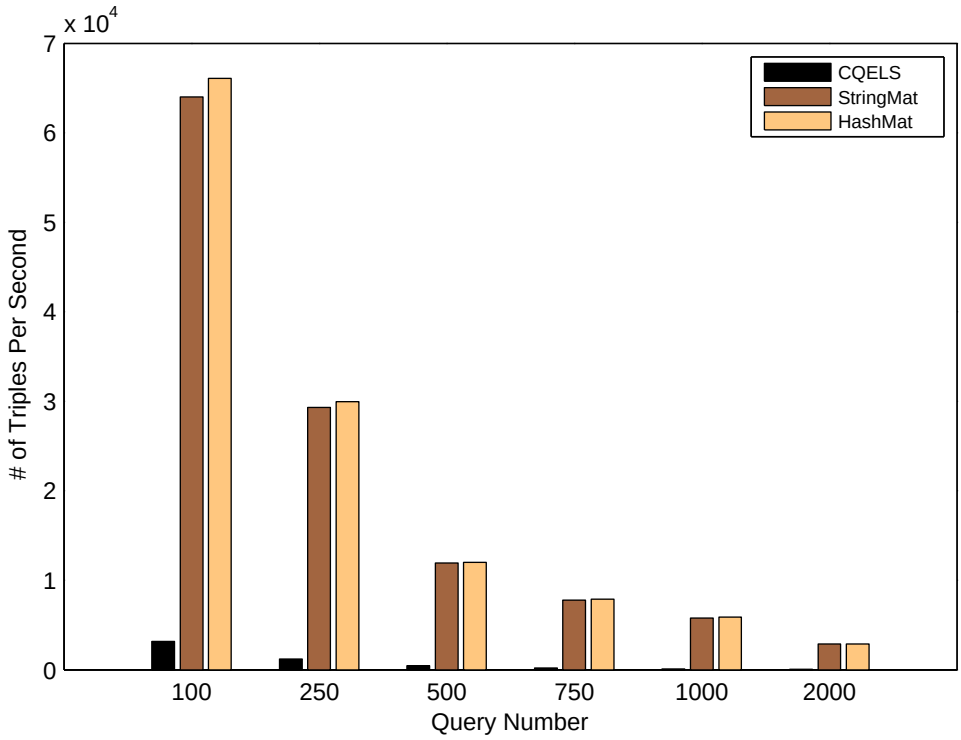


Fig. 4.9 Average Throughput Evaluation of CTP-automata (Varying Query Number)

Throughput. The throughput performance of *pattern matching* under varying query numbers is depicted in Figure 4.9. It shows some similarities between HashMat and StringMat. In most cases, HashMat shows slightly better throughput speed compared with StringMat. This indicates that although comparisons on strings are slower than those on numbers, the differences betwen HashMat and StringMat are negligible. The main reason for this is that the evaluation process of conjunctive queries spends a large proportion of time to evaluate the conjunctive constraints on each query and both HashMat and StringMat use the same strategy to evaluate all these conjunctive constraints.

However, when compared with CQELS, both HashMat and StringMat outperform CQELS significantly. To be specific, when the number of conjunctive queries is 100, the throughput of HashMat and StringMat is more than 64,000 triples per second, and for CQELS, just slightly more than 3,000. When the number of conjunctive queries is

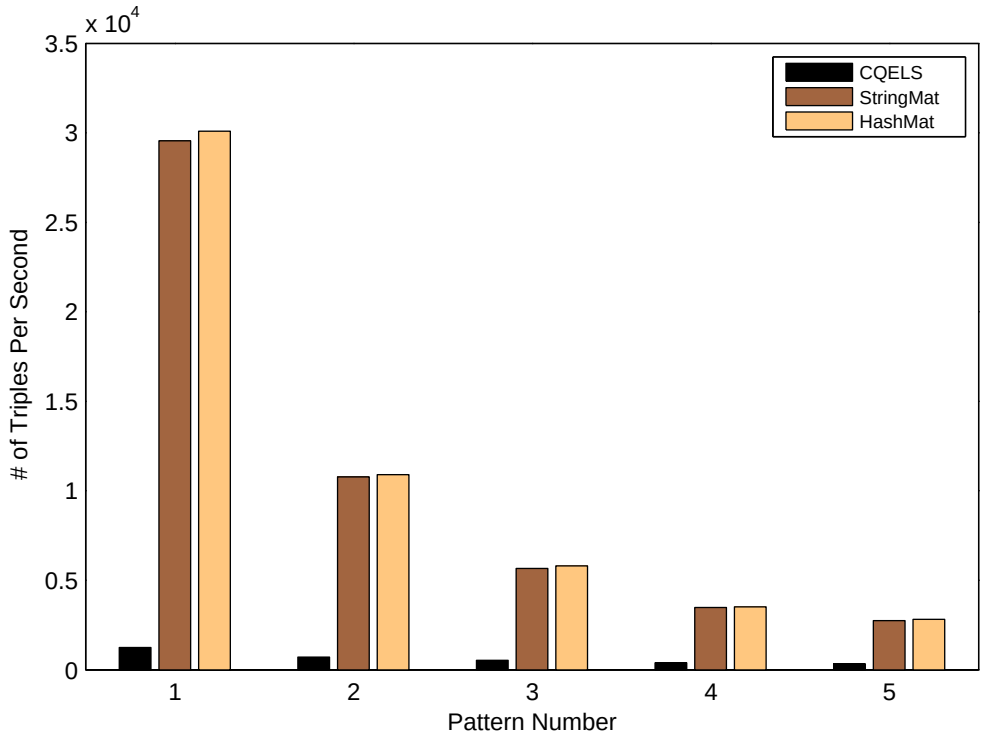


Fig. 4.10 Average Throughput Evaluation of CTP-automata (Varying Pattern Number)

2,000, the throughput of HashMat and StringMat drops to slightly below 3,000 triples per second while CQELS has a throughput about just 50 triples per second. From Figure 4.9, we can observe that (1) HashMat and StringMat are normally 20 to 50 times faster than CQELS; (2) the throughput of HashMat, StringMat and CQELS all drops greatly when increasing the number of conjunctive queries. This also indicates that the evaluation of conjunctive constraints on each query takes a large amount of time and is difficult to share evaluation results between different conjunctive queries.

Figure 4.10 further demonstrates this finding. In the figure, we vary the maximum number of patterns of each conjunctive query. For the same amount of conjunctive queries, when the pattern number is only 1, the throughput of HashMat and StringMat is around 30,000 triples per second and for CQELS, it is around 1,200 triples per second, which is more than 20 times slower. When the pattern number is set to 5, the throughput of HashMat and StringMat drops to slightly lower than 3,000 triples per

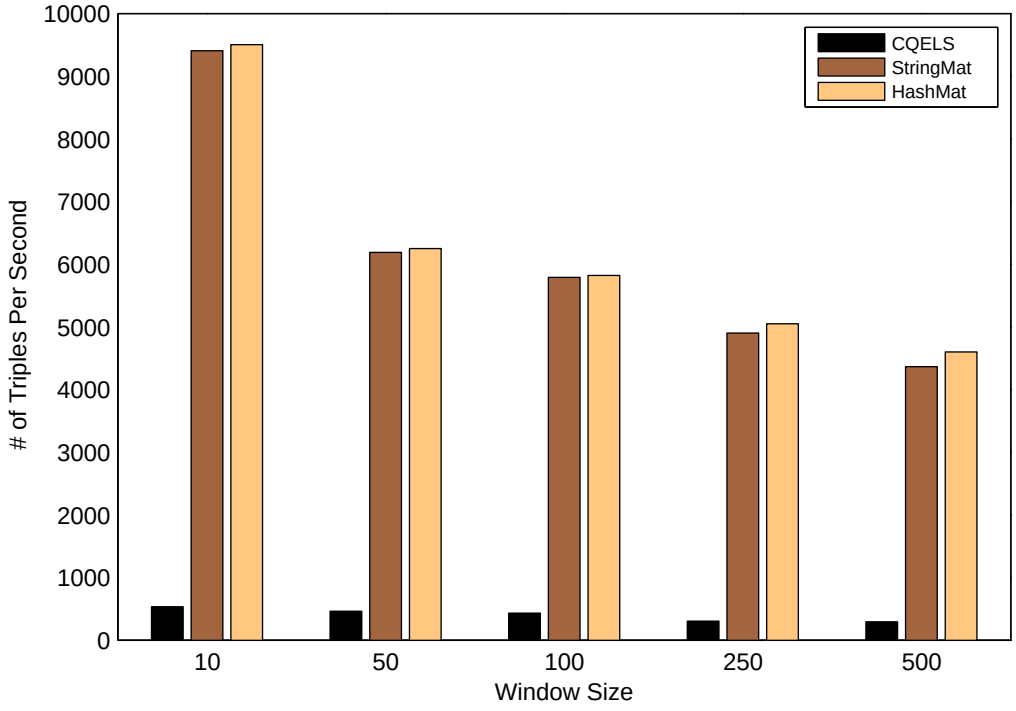


Fig. 4.11 Average Throughput Evaluation of CTP-automata (Varying Window Size)

second and for CQELS, it drops to around 300 triples per second. This confirms that the evaluation of conjunctive constraints is time consuming. Similarly, HashMat and StringMat are both around an order of magnitude faster than CQELS.

Finally, Figure 4.11 depicts the effect of window size, which is varied from 10 to 500. From the figure, we can observe that when the window size increases from 10 to 50, the throughput of HashMat and StringMat drops from 9,500 triples per second to around 6,200 triples per second. But when the window size increases from 50 to 500, the throughput of HashMat and StringMat only drops to around 4,500 triples per second. This indicates that the window size does not affect the throughput heavily like query number and pattern number. Similar effect of window size can be found on CQELS. When the window size increases from 10 to 500, the throughput of CQELS drops from around 500 triples per second to slightly lower than 300 triples per second. Still, HashMat and StringMat are both an order of magnitude faster than CQELS.

From our experimental study, we can conclude that CTP-automata indexes for conjunctive queries can be constructed much faster than the query registration process in CQELS. More importantly, CTP-automata (HashMat and StringMat) significantly outperforms CQELS in terms of throughput. Further, by using hashing techniques, HashMat performs slightly better than StringMat.

4.3.4 Limitations

From the experiments, we can see that our system can match Linked Data streams with single triple pattern queries at high performance, e.g., arriving at close to one million triples processed per second with 100,000 user queries registered in the system. However, our system cannot scale well when processing conjunctive triple pattern queries. As for conjunctive queries, the throughput of our system can only reach a few thousand triples processed per second with only 2,000 user queries registered. Such findings in our experiments suggest that there is an imperative need for developing more advanced techniques for handling a large number of conjunctive queries in Linked Data streams dissemination systems.

4.4 Related Work

Recent work on data summaries on Linked Data [103] transforms RDF triples into a numerical space. Then data summaries are built upon numerical data instead of strings as summarizing numbers is more efficient than summarizing strings. In order to transform triples into numbers, hash functions are applied on 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 [103], which is evolved from standard R-tree [110], is adopted as the

basic index. Data summaries are designed mainly 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 dissemination system. 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 [103], they just provide *match estimation*. Hence, query evaluation on them would return false negative results, which is not allowed in our system.

Pattern matching over streams has been studied in [59]. In order to represent each pattern query, a new query evaluation model is designed for processing pattern matching over RFID streams, by employing a new type of automaton that comprises a non-deterministic finite automaton (NFA) and a match buffer, named NFA^b. However, their techniques are not directly applicable to Linked Data stream processing as they do not specifically consider the characteristics of RDF data and conjunctive triple pattern matching.

In terms of triple pattern matching, a large body of work which focuses on optimizing individual query processing has also been put forward [106, 111–113]. Specifically, the problem of evaluating conjunctive triple pattern queries is studied in [106] in the context of Peer-to-Peer (P2P) networks. In [111], an indexing technique for efficient join processing on RDF graphs is proposed. The index is constructed upon RDF data directly but not join queries. Similarly, the work in [112] focuses on optimizing the processing of conjunctive triple pattern queries, especially star-shaped group based queries individually. Furthermore, optimization on RDF graph pattern matching on MapReduce is also studied in [113]. However, the common problem shared by these research efforts is that, they have not considered the scenarios of optimizing conjunctive triple pattern queries in batch mode, which is the focus of our work in this chapter.

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 false negative. Both approximate event matching [104] and thematic event processing [114] apply semantic matching. Similarly, all these techniques will return false-negative matching results, which is not allowed in our system.

Some existing work on *pattern matching* of Linked Data, such as stream reasoning [30] and Linked Data stream processing [31], 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. Other existing work only studies pattern matching of multiple single triple patterns [115, 96], but not multiple conjunctive triple patterns. Therefore, the issue of supporting *pattern matching* over a large number of conjunctive triple patterns against Linked Data streams still remains open in these approaches.

4.5 Summary

In this chapter, we have leveraged semantic technologies, such as Linked Data, to build an efficient information dissemination system for semantic IoT. Firstly, in order to efficiently match a large number of user queries that contain only single triple patterns 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 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. Secondly, we have also investigated how to handle user queries with conjunctive triple patterns. In order to efficiently match a large number of conjunctive triple pattern queries against

Linked Data streams in batch mode, similarly, we have proposed CTP-automata. In our evaluation, we show that CTP-automata can disseminate Linked Data an order of magnitude faster than the existing approaches. This confirms the efficiency and effectiveness of our proposed approach.

In the next two chapters, we will focus on addressing data sharing needs in the context of IoT. Chapter 5 will address data sharing with Linked Data (in RDF format) and Chapter 6 will address data sharing via exchanging XML format data. Both RDF and XML formats are prevalent data formats for data sharing and exchanging over the Internet. Therefore, we investigate data sharing techniques for both formats in the following two chapters.

Chapter 5

Data Sharing in IoT Environments

Part I

In this chapter, we consider large-scale information sharing scenarios among mobile objects in IoT by leveraging semantic techniques. We propose broadcasting Linked Data on-air using RDF format to allow simultaneous access to the information and to achieve better scalability. We introduce a novel air indexing method to reduce the information access latency and energy consumption. To build air indexes, we firstly map RDF triples in the Linked Data into points in a 3D space and build B^+-trees based on 3D Hilbert curve mappings for all of the 3D points. We then convert these trees into linear sequences so that they can be broadcast over a wireless channel. A novel search algorithm is also designed to efficiently evaluate queries against the air indexes. Experiments show that our indexing method outperforms the air indexing method based on traditional 3D R-trees. The research presented in this chapter was published in [116].

5.1 Overview

In the era of the Internet of Things [98], due to a large amount of information produced by all kinds of things, one of the significant challenges is how to efficiently consume the large amount of information that is generated. In this context, semantic technologies such as Linked Data, which aim at facilitating machine-to-machine communications, play an increasingly important role [102]. Linked Data is taking an active role in a trend towards highly distributed systems, with potentially millions of independent sources providing small amounts of structured data [117]. Event detection and entity discovery requires an effective way of information sharing. Compared with the *point-to-point* communication paradigm, *broadcast* (or *point-to-multipoint*) allows simultaneous access by an arbitrary number of listeners (or clients) without causing contention of server resources [118]. Considering information sharing among a large number of mobile and smart objects in the Internet of Things, broadcast is an attractive mechanism of information dissemination.

To further illustrate the motivation, let us consider the scenario of a future smart city, where intelligent objects will be acting as data collectors in different places of the city. They will be able to sense their vicinities (e.g., for air pollution information) and produce related Linked Data that is understandable by machines. These objects send such data to a nearby base station for further processing. The base station can then integrate and process all the data from various data collectors and broadcast to a much wider audience such as smart objects, which are not direct data collectors but are interested in such data. Since a smart object is normally only interested in a small part of the broadcast data, blindly checking every triple on air would lead to very long access latency and unnecessary energy consumption. As the batteries of smart objects are often limited, an efficient way to reduce energy consumption and lower access latency is imperative. Our work focuses on solving this challenge by designing effective and efficient air indexes for broadcasting Linked Data on air.

Inspired by recent work on data summaries on Linked Data using RDF format [103, 117], we adopt a similar technique to index Linked Data. This is mainly because it can be used to construct very concise indexes, which is critical in saving mobile clients' energy consumption. On the server side, we firstly map the RDF triples contained in Linked Data streams into points in a 3D space and then build indexes on these 3D points. A straightforward indexing solution is to use traditional spatial indexes, such as R-tree [110], which recursively index spatial points/objects in sub-regions. However, it is shown that the performance of R-tree degrades in higher dimension space [119]. Moreover, R-tree is designed for random access while, in a wireless broadcast system, indexes can only be accessed sequentially. Therefore, the R-tree approach is not suitable for indexing Linked Data on air. Motivated by this, we introduce a novel method by adopting 3D Hilbert curve mappings [120] 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. We also build B^+-trees [121] for the one-dimensional points to facilitate point access on air. Finally, we convert these trees into linear sequences so that they can be broadcast on a wireless channel. On the client side, a novel search algorithm is designed to efficiently evaluate queries against the air indexes. The experimental results show that our indexing method outperforms the 3D R-tree based methods. Based on these trees and the search algorithm, clients can determine what data is of interest and when the data will be broadcast on the wireless channel. This allows clients to set the duty cycle to reduce power expenditure. We compare our indexing method with the method based on traditional 3D R-trees. The experimental results show that our indexing method outperforms the 3D R-tree based methods.

The rest of this chapter is organized as follows. Section 5.2 discusses related work. Section 5.3 describes our indexing method in detail. Section 5.4 demonstrates and discusses experimental results. Finally, some concluding remarks are presented in Section 5.5.

5.2 Related Work

The rapid development of Linked Data has accelerated the advent of thousands or potentially millions of independent data sources, which are crucial for answering queries.

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. Light weight data summaries on Linked Data [103, 117] 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 C*k*NN (Continuous *k* Nearest Neighbor) query processing on air in [122] has introduced indexes and search algorithms for continuous *k*NN queries in 2D spaces. It uses a similar technique discussed in this chapter to index moving objects modeled in 2D spaces. Their work differs from this work as their main goal is to support continuous *k*NN queries in 2D spaces, whilst the main goal of this chapter is to support queries on Linked Data broadcast in a wireless channel.

Our work differs from the C*k*NN query processing in the following aspects: 1) C*k*NN query processing is for C*k*NN queries while our work aims at processing Basic Graph Patterns (BGPs) on Linked Data; 2) the search algorithms are different as query evaluation for C*k*NN queries and BGPs is different; 3) the indexing space is different as C*k*NN 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 C*k*NN query processing aims to handle variants of NN queries.

5.3 Air Indexing for Linked Data

5.3.1 Wireless Broadcast Model

In a wireless data broadcast system, generally 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. The wireless channel can be shared by all mobile clients. In this way, the broadcast system could be able to serve an arbitrary number of mobile clients simultaneously.

In order for clients to efficiently locate data of interest, air indexing techniques are used to facilitate the searching of data on air. Air indexes are normally 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. Since the wireless channel is linear and data is broadcast in the form of packets, air indexes can be serialized and separated into packets. In order to improve performance, the number of air index packets required for evaluating a query should be minimized.

Similar to the existing work in data broadcast, we use *access latency* and *tuning time* as the primary performance metrics [118]. *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 of time that a client has to stay active in order to complete a query. During the tuning time, a client downloads the relevant index parts, queries against the downloaded index, or downloads the matched data from the broadcast channel. When the data on air is not of interest, a client can switch to sleep mode to save power and wait for the requested data to be broadcast.

5.3.2 Mapping between Triples and 3D Points

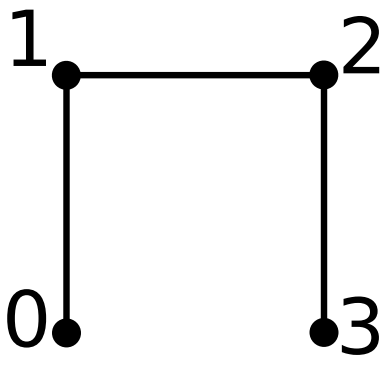
Existing light weight data summaries (e.g., [103, 117]) have been proven to be effective to index Linked Data. However, they are not suitable in a wireless broadcast system. In order to develop a new index structure for broadcasting Linked Data, similar to data summaries, we choose to use hash functions[1] to map RDF triples into numerical values. These numerical values can be regarded as *coordinates* in a 3D space. Specifically, 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* in order to differ 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.

Basic Graph Patterns (BGPs) [117] are adopted as queries in our system. Similar to 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 our work, as it will be mapped into the whole 3D space and require a traversal of all the data points in the whole 3D space, where air indexing is not required.

5.3.3 3D Hilbert Curve Index

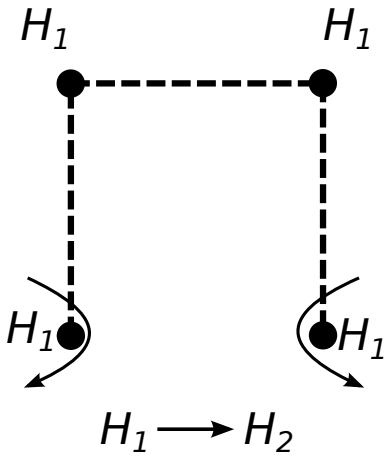
A space-filling curve in d dimensions is a continuous, surjective mapping between one-dimensional space and d-dimensional space. A Hilbert curve is an example of a

[1]There are many options of hash functions. For more details, please refer to [103, 117].

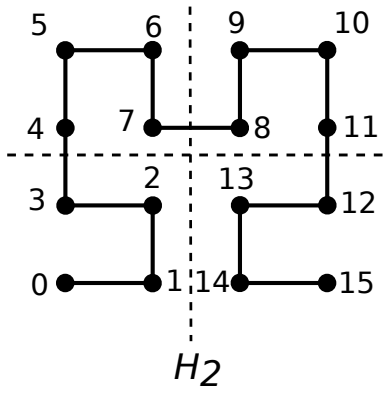


H_1

(a)



(b)



(c)

Fig. 5.1 2D Hilbert Curves of order 1 and 2. (a) H_1, (b) H_1 to H_2, (c) H_2

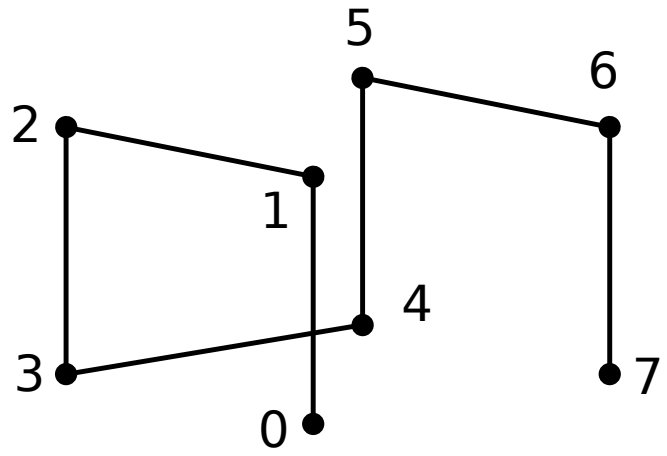


Fig. 5.2 3D Hilbert Curve of order 1

space-filling curve. It generally has good locality properties [120] 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 generalized to 3D Hilbert curves. Figure 5.1 shows 2D Hilbert curves for order 1 and 2, i.e., H_1 and H_2, respectively. Note that, a k order Hilbert curve, denoted as H_k, passes all center points of 2^{kd} subdividing squares (or hypercubes) in a d-dimensional space. In Figure 5.1a, each center 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 center points and Hilbert values are bijective, which means for a given Hilbert curve, we can freely convert between center points and Hilbert values in constant time.

From Figure 5.1b, we can see that high order Hilbert curves can be easily derived using transformation from low order Hilbert curves similar to the one shown in Figure 5.1b. In order to derive H_2 from H_1, in Figure 5.1b, the 2D space is divided into 2^d ($d = 2$ in this case) sub-regions, where each sub-region contains an H_1. After rotating the lower two H_1 curves, an H_2 Hilbert curve is derived (see Figure 5.1c). Since H_1 has 2^2 subdividing squares, H_2 has totally $2^{2\times 2}$ subdividing squares.

Figure 5.2 presents an example of a 3D Hilbert curve of order 1. Higher order 3D Hilbert curves can be derived using a similar process described above. In order to accommodate a larger 3D data space, i.e., the hashing space for RDF triples, we need to utilize higher order 3D Hilbert curves. We can easily check that a k order 3D Hilbert curve can have up to $2^{3\times k}$ data points. In other words, when mapping to a k order 3D Hilbert curve, all RDF triples will be mapped into at most $2^{3\times k}$ data points (also center 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 utilize B^+-trees to index one-dimensional points on a 3D Hilbert curve. An example is depicted in Figure 5.3. 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 serialized and broadcast on the linear wireless channel as air indexes for the Linked Data on 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 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. A naive solution to this would be to map BGPs to points on Hilbert curves as well. However, there will be too many points that fall into a line or a plane and require such mapping. Hence, for those BGPs with variables, it may require significant computing effort to map such BGPs into points on Hilbert curves. Because of this, we propose to map BGPs into a 3D space directly. 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: We give an example of the query evaluation process in the following. Suppose a triple `(a, b, c)` can be hashed as `(112, 31, 92)` in a 3D space and its Hilbert value is 1137. Also 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 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 a query evaluation. Given an air index like the one shown in Figure 5.3, 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 $[12,h_{max}]$ (here h_{max} 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 Figure 5.4, where two dash lines represent two BGPs). If such minimal

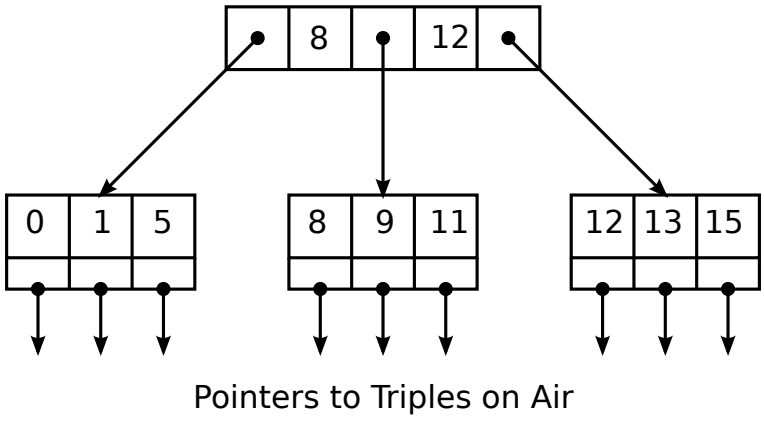


Fig. 5.3 B^+-tree for some Points on Hilbert Curve

sub-region intersects with 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 Figure 5.4 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 Figure 5.3 will match the two BGPs shown as two dash lines in Figure 5.4.

It is reasonable that the mobile clients would not want to download too much data as energy is limited. In such case, the mobile clients can firstly download the index on air only and then evaluate their queries on the index. Such evaluation will provide them with an idea on how much data they need to download. If they find out that their queries will match too much data, they can revise their queries to reduce the amount of data for downloading.

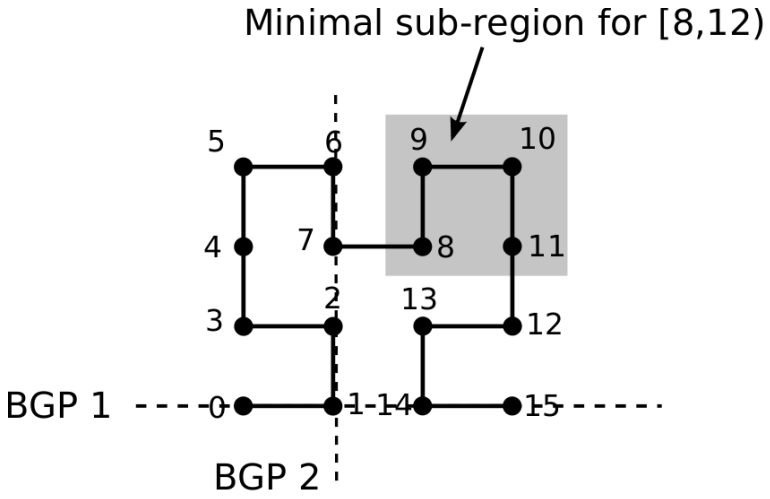


Fig. 5.4 Minimal sub-region

5.4 Experiments

5.4.1 Experimental Setup

The data set used in our experiment is a subset of the current version of the English DBpedia[2]. 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. We used data from DBpedia in our experiment because DBpedia is one of the free and public datasets. DBpedia also provides information that attracts public interest, which well fits the motivation of this work.

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 Section 5.3 based on our dataset. We generated 100,000 queries and reported the average experimental results.

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

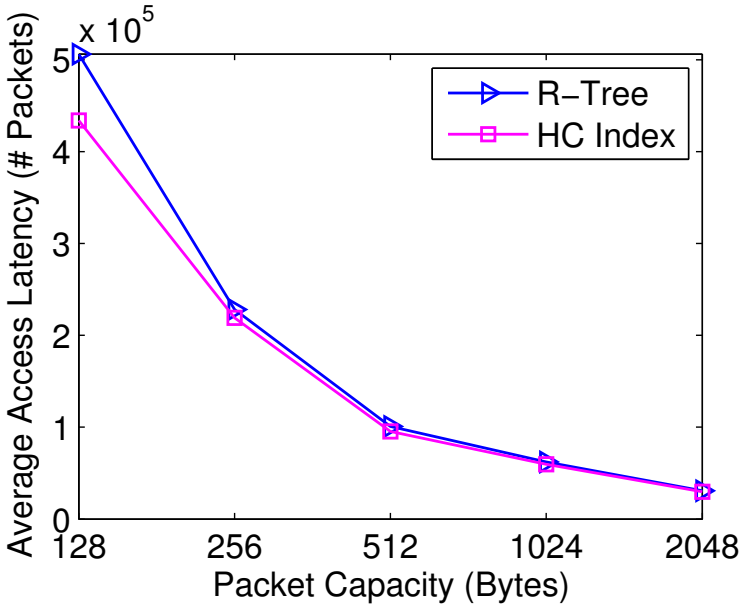
R-tree [110] often requires to maintain a queue to keep track of overlapped regions during the query evaluation which may require considerable memory consumption. Hence we broadcast R-tree indexes in a depth-first order to avoid that issue. We broadcast B^+-tree HC (Hilbert Curve) indexes in a breadth-first order to facilitate the continuous access to linked leaf nodes.

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.

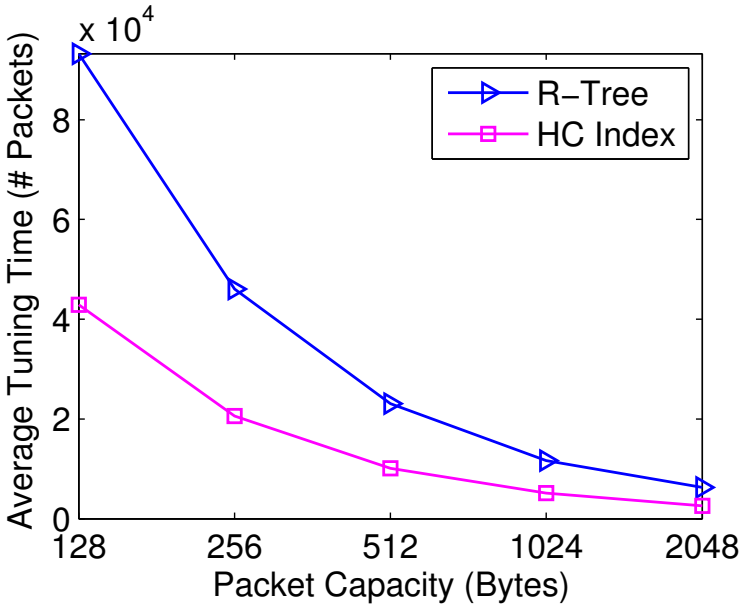
We evaluated the performance of our method using metrics of access latency, tuning time, and index size. Given a fixed bandwidth of the wireless channel, the access latency depends on the number of packets that have been broadcast during the clients' waiting time, and the tuning time depends on the number of packets that have been downloaded by clients. Hence, for simplicity, we measured the access latency and tuning time with the number of packets (denoted as `# Packets`).

5.4.2 Performance Analysis

Figure 5.5 shows the results of average access latency and tuning time under different packet capacities. From Figure 5.5a, we can see that the access latency for our Hilbert Curve (HC) based method is slightly better than the R-tree based method. The reason is that the index size is much smaller than the content (Linked Data stream) on air and hence the dominant factor of access latency is the content but not the index. Nevertheless, the size of HC based index is smaller (see also Figure 5.6b) than R-tree based method, resulting in lower access latency.



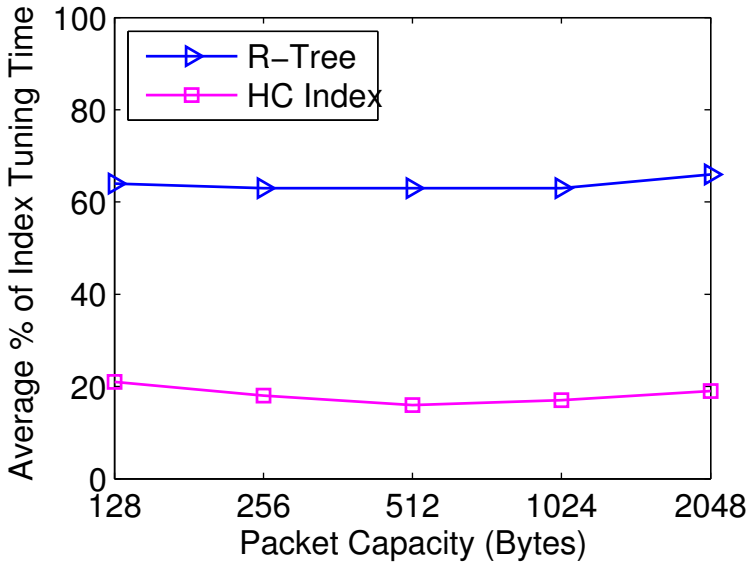
(a) Access Latency



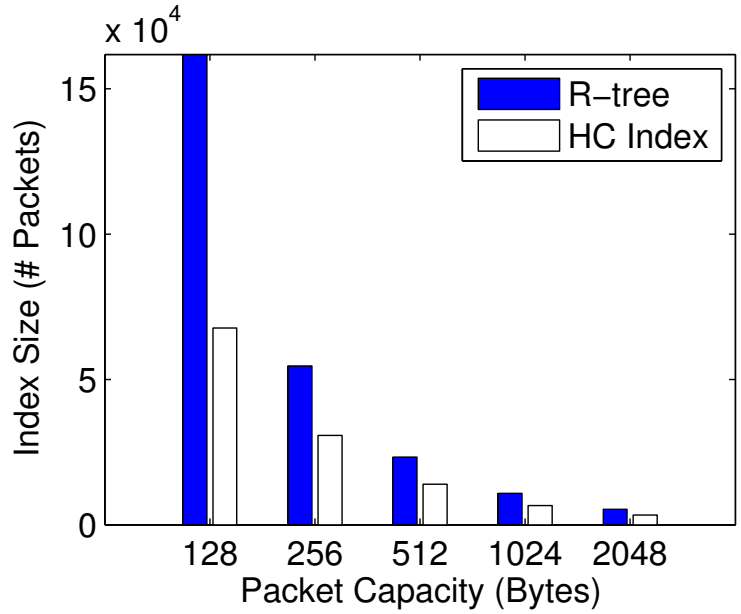
(b) Tuning Time

Fig. 5.5 Access Latency and Tuning Time

Figure 5.5b shows the comparisons of the tuning time. From the figure, we can clearly identify that HC based index outperforms R-tree based index. The reason for this is two-fold: firstly, by using our novel search algorithm, the searching space



(a) Index Tuning Time



(b) Index Size

Fig. 5.6 Index Tuning Time and Index Size

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 Figure 5.6a. 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 for the effectiveness and efficiency of an index. The lower percentage we get, the better effectiveness we can achieve. Figure 5.6a 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 Figure 5.6b. 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 R-tree index has to store 3D points in its nodes while the HC based index only stores one-dimensional points.

5.5 Summary

In this chapter, we have proposed an effective and efficient air indexing method for broadcasting Linked Data on air, which can be used in data sharing among a large number of mobile and smart objects in the era of Internet of Things. 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 serialize these trees in order to accommodate them on the linear wireless channels. An efficient search algorithm has also 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.

Chapter 6

Data Sharing in IoT Environments

Part II

XML has become prevalent in today's ubiquitous and mobile computing devices and applications. To integrate all these mobile computing devices and applications as a significant part of the IoT big family, it is imperative to incorporate XML data techniques in IoT to enable efficient and effective data exchanging and sharing among things that speak and understand the XML language. This chapter studies the data placement problem of periodic XML data broadcast in mobile and wireless environments. Taking advantage of the structured characteristics of XML data, effective broadcast programs can be generated based on the XML data on the server only. An XML data broadcast system is developed and a theoretical analysis on the XML data placement on a wireless channel is also presented, which forms the basis of the novel data placement algorithm in this chapter. The proposed algorithm is validated through a set of experiments. The results show that the proposed algorithm can effectively place XML data on air and significantly improve the overall access efficiency. The research presented in this chapter was initially published in [123].

6.1 Overview

Wireless technologies have become deeply embedded in our daily lives [124]. At the end of 2011, there were 6 billion mobile subscriptions, estimated by the International Telecommunication Union (2011). That is equivalent to 87 percent of the world population, and is a huge increase from 5.4 billion in 2010 and 4.7 billion mobile subscriptions in 2009.

Broadcast is one of the basic ways of information access via wireless technologies. In a wireless data broadcast system, the server broadcasts public information to all mobile devices within its transmission range via a downlink broadcast channel. Mobile clients “listen" to the downlink channel and access information of their interest directly when the desired information arrives. Broadcast is bandwidth efficient because all mobile clients can share the same downlink channel and retrieve data from it simultaneously. Broadcast is also energy efficient at the clients because downloading data costs much less energy than sending data [125].

Wireless data broadcast services have been available as commercial products for many years (e.g., StarBand and Hughes Network). Recently, there has been a push for such systems from industry and various standard bodies. For example, born out of the ITU “IMT-2000” initiative, the Third Generation Partnership Project 2 is developing Broadcast and Multicast Service in CDMA2000 Wireless IP network. Systems for Digital Audio Broadcast (DAB) and Digital Video Broadcast (DVB) are capable of delivering wireless data services. Recent news also reported that SiriusXM Satellite Radio (www.siriusxm.com) and Raytheon have jointly built a communication system, known as the Mobile Enhanced Situational Awareness Network (MESA), that would use SiriusXM satellites to relay information to soldiers and emergency responders during a homeland security crisis.

On the other hand, information expressed in semi-structured formats is widespread over the past years. XML has rapidly gained popularity as a de facto standard to rep-

resent semi-structured information. Popular Web browsers provide support for XML and nearly all major IT companies (e.g., Microsoft, Oracle, and IBM) have integrated XML into their software products. Delivering information in XML format is also popular in Web services and in various kinds of Publish/Subscribe systems.

Combining both trends of the proliferation of mobile computing technologies and XML data, broadcasting information in XML format in a wireless environment would be a preferable way of information delivering and sharing. Consequently, the research of XML data broadcast is of great importance and in fact it has been attracting more and more research interests [126–133]. To further demonstrate the practicability of XML data broadcasts, we will present a potential application by detailing a real life scenario in Section 6.3.

There are two typical data broadcast modes: (i) *Periodic Broadcast Mode* and (ii) *On-Demand Broadcast Mode* [125]. In the periodic broadcast mode, data is periodically broadcasted on a downlink channel via which the server sends data to clients. Clients only need to “listen" to that channel and download the data that they are interested in. In contrast, for the on-demand broadcast mode, clients send their queries to the server via an uplink channel and the server considers all submitted requests and decides the content of the next broadcast cycle. In this work, we focus on the periodic broadcast mode since it has many benefits such as saving uplink bandwidth, reducing power consumption at the clients and effectively delivering information to an unlimited number of clients simultaneously.

Data placement algorithms determine what data items to be broadcasted by the server and their order on wireless channels, aiming to reduce average waiting time for mobile clients. To a large extent, the data placement problem of XML data is similar to that in multi-item contexts [134, 135] where mobile clients may request multiple items each time. However, there are some drawbacks of existing data placement approaches in the traditional data broadcast models.

Firstly, the previous work on multi-item placement problems generally assumes that the clients' queries are known in advance [136, 137, 134, 135]. For example, the clients can provide a profile of their interests to the servers [136, 137]. However, such models may limit the practicability of the proposed placement algorithms in real situations because: (i) new mobile clients may join in the network at anytime; and (ii) mobile users may be reluctant to disclose their queries to the server via an uplink channel due to expensive communication cost and privacy concerns. Secondly, in the traditional data broadcast systems, an appropriate placement can hardly be generated based only on information of data items themselves on the server. Hence, user queries must be known in advance for the design of data placement algorithms. Alternatively, some work applies data mining techniques to discover association rules from the history access patterns of a set of data [138]. This avoids the need to obtain access patterns of mobile clients on-the-fly. However, the availability of such historical access patterns of mobile clients is a necessity, which may not always be available.

In contrast, in XML data broadcast, data items (or XML documents) usually share parts of their structure. Taking structural sharing between XML documents into consideration, we are able to analyze and estimate clients' access patterns. Then we can effectively place XML data on wireless channels based only on the XML data on the server, which is important for practical usage. To the best of our knowledge, little work has addressed similar data placement strategies in the context of wireless data broadcast. Therefore, in this chapter, we study the data placement problem of periodic XML data broadcast. Firstly, we overview an XML broadcast system and present a theoretical analysis on the data placement problem of periodic XML data broadcasts. Secondly, based on the analysis, we design a novel greedy data placement algorithm. The main contributions of this chapter are summarized as follows:

- In the context of periodic XML data broadcasts, by taking advantage of the structural characteristics of XML data, we are able to generate appropriate data placement results based only on the XML data on the server.

- We perform a theoretical analysis on the data placement problem of periodic XML data broadcasts. Based on the analysis, a novel greedy data placement algorithm which organizes XML data on air is presented.

- Extensive experiments are conducted to show the effectiveness of our proposed data placement algorithm.

The remainder of this chapter is organized as follows. Section 6.2 describes some background information of this work, including an application scenario, the wireless broadcast system and the concept of XML structural sharing. Then a theoretical analysis of the data placement problem is presented in Section 6.3. Section 6.4 discusses the structural sharing property of XML data and then proposes a novel greedy data placement algorithm. Section 6.5 presents our experimental study for evaluating the performance of the proposed data placement algorithm. Finally, Section 6.6 discusses the related work and Section 6.7 gives some concluding remarks.

6.2 Background

In this section, we first describe a potential application scenario. Then we show an overview of the broadcast system used in this work and introduce background knowledge of XML structural sharing.

6.2.1 Application Scenario

We use the following scenario to show potential applications of XML data broadcast in real life.

Consider a live basketball game. Information about the game and the players on the court is usually the interest of a large audience. When the game progresses, the volume of such information is expected to increase, which means the information content is dynamic. Normally, a basketball stadium can accommodate 10, 000 to 60, 000 audience at the same time. The audience is likely to be interested in similar game information. They may even want to access the game information at the same time (suppose most of them are with mobile devices). In this context, data broadcast is a preferable way of delivering latest information to the audience. Then thousands of audience in the stadium can access game information simultaneously by just “listening" to the broadcast channel. The audience does not need to contend limited bandwidth (i.e., the use of the uplink channel) and other system resources (i.e. the server processing capacity) with each other.

In this scenario, we assume that (1) the audience is not only interested in the real-time information about the basketball game, but also interested in some historical records/statistics information about the players, the basketball teams or the coach teams, etc; (2) the audience may want more statistics information about the current basketball game than the limited live statistics information shown on the large screens inside the stadium. Therefore, a broadcast service would be very helpful for the audience to obtain more desired information about the game.

Meanwhile, audience could be outside of the stadium, such as basketball fans who are watching live text information about the game via the Internet at their homes. Therefore, the game information could also be delivered via the Internet to online audience and other Web service providers who have subscribed this basketball game. Using XML format to represent game information can satisfy all these needs and realize simplicity, generality, and usability of game information at the same time.

Apart from broadcasting to people, broadcasting to smart objects in the Internet of Things is also important in enabling information sharing among smart objects. Specially in the Smart City environments, smart objects in the same region of a city

might be interested in similar information related to the region they are currently in. In such situation, broadcasting is a preferable way to disseminate information to these smart objects.

6.2.2 A Wireless XML Data Broadcast System

Figure 6.1 shows a wireless XML data broadcast system. The system includes an XML Data Center (the broadcast server), a broadcast program scheduler, broadcast listeners (mobile clients) and a downlink channel (the server broadcasts information to mobile clients via it). If mobile clients are interested in some data on the server, they can listen to the downlink channel and download data that they need. Note that, downloading the data from the downlink channel does not mean each mobile client would need to set up a different downlink connection. All mobile clients only need to tune in the downlink channel and listen to it for the desired information. Thus this downlink channel can be shared by all mobile clients. Mobile clients cannot send their individual queries to the server in this model as no uplink channel is available.

There are several notable advantages of such a system model. Firstly, it can serve an arbitrary large number of mobile clients simultaneously. Secondly, extra energy cost at the client side for using an uplink channel can be avoided. Further, after applying air indexing techniques [139, 129], mobile clients will also be able download an air index and determine when the desired data will be broadcasted on the wireless channel (for more details about air indexing, please refer to Section 6.4.3). Before the desired data is available, they can switch to energy saving mode to reduce power consumption. As a result, the battery life of mobile clients can be extended.

From the figure, we can see that the XML Data Center could be connected to the Internet and deliver information to online users, Web service providers and Publish/Subscribe systems, etc. With the use of XML data, these different applications

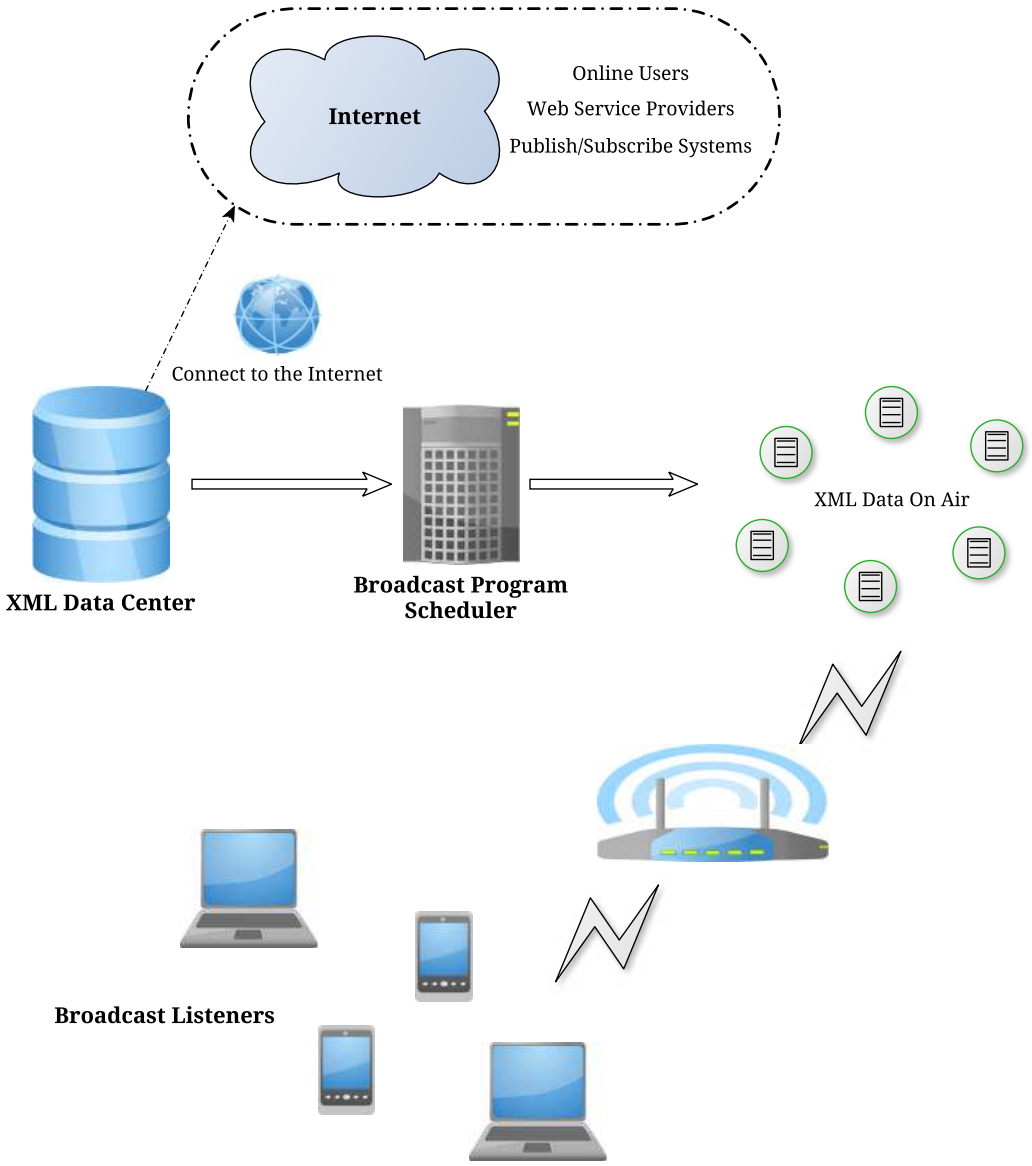


Fig. 6.1 A wireless XML data broadcast system

can be integrated seamlessly with our wireless data broadcast system for the purpose of sharing and delivering the same information to different users.

6.2.3 XML Structural Sharing

Our goal is to place XML documents on the broadcast channel based on the information at the server side. We propose to explore structural sharing between differ-

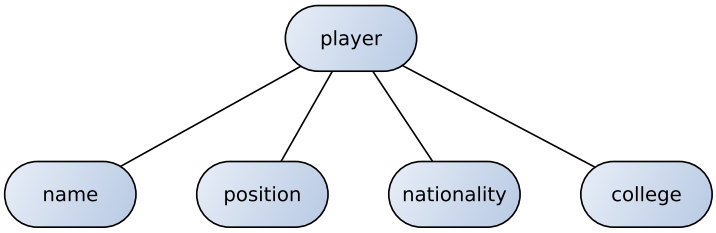


Fig. 6.2 An XML structure tree

ent XML documents and place documents according to the structural sharing results. Here, structural sharing refers to overlaps of path sets (defined in the below) of XML documents.

Some existing work on measuring structural sharing between XML documents can be found in [140, 141]. The main idea of their work is based on the concept of *path sets*. Here, a path set of an XML document contains all full paths (paths that are from the root element to the leaves) and their subpaths. A simple example is depicted in Fig. 6.2. The path set of this example is: {*/player/name, /player/position, /player/nationality, /player/college, /player, /name, /position, /nationality, /college*}. We denote a path set of an XML document d as $PS(d)$.

Different types of metric can be adopted, such as Jaccard metric [142, 143], Dice's coefficient [144] and Lian's metric [145], to measure the structural sharing or similarity between two XML documents d_i and d_j. The exact forms of these metrics based on PS are as follows (Jaccard metric is denoted as $J(d_i,d_j)$, Dice's coefficient is denoted as $D(d_i,d_j)$ and Lian's metric is denoted as $L(d_i,d_j)$):

$$J(d_i,d_j) = \frac{|PS(d_i)\bigcap PS(d_j)|}{|PS(d_i)\bigcup PS(d_j)|} \tag{6.1}$$

$$D(d_i, d_j) = \frac{2 \cdot |PS(d_i) \bigcap PS(d_j)|}{|PS(d_i)| + |PS(d_j)|} \tag{6.2}$$

$$L(d_i, d_j) = \frac{|PS(d_i) \bigcap PS(d_j)|}{max\{|PS(d_i)|, |PS(d_j)|\}} \tag{6.3}$$

From the above definitions, we can see that both Jaccard metric and Dice's coefficient give more weight to the total structural information of two comparing documents while Lian's metric emphasizes on the difference of these documents. All three metrics can vary in the interval $[0,1]$. If $PS(d_i) = PS(d_j)$, we have $J(d_i, d_j) = D(d_i, d_j) = L(d_i, d_j) = 1$. Clearly, the larger the values of these metrics are, the more structural sharing the two comparing XML documents have.

6.3 Analysis of the Data Placement Problem

In this section, we present a theoretical analysis on the data placement problem in periodic XML data broadcasts.

In the literature, two critical metrics, namely *access time* and *tuning time*, are used to measure a system's performance [146]. *Access time*[1] refers to the time elapsed from the moment a query is issued to the moment it is answered, while *tuning time* refers to the time a mobile client stays in active mode to receive the requested XML documents and the index information. Data placement mainly affects access time because tuning time depends on the total content downloaded by mobile clients but not on the order of data. Hence, we use access time as our metric in this analysis. In periodic broadcast, queries are used to describe the interests of mobile clients and

[1]Note that, in [139], *Access latency* is also used for *Access time*, which can be considered the same concept.

Table 6.1 Symbols Overview

Symbol	Description
$\mathscr{D}$	XML document set. $\{d_1, d_2, d_3, \ldots, d_n\}$.
q	query issued by a mobile client.
k	number of documents required by q.
σ	a complete broadcast program (or broadcast sequence). It is the result of a data placement algorithm running on a given $\mathscr{D}$. $< d_1, d_2, d_3, \ldots, d_n >$. (Note: It is different from $\mathscr{D}$ as $\mathscr{D}$ is a set but not a sequence.)
gap	unmatched documents of q that are placed between two adjacent matched documents in σ.
$\mathscr{L}$	the length of documents.
$\mathscr{AT}^q_{exp}$	the expected access time of q.

help mobile clients to skip irrelevant data on air, but they are not actually submitted to the broadcast server.

Table 6.1 lists the symbols used in the rest of the chapter and Fig. 6.3 shows a broadcast program (or a broadcast sequence) σ on the wireless channel which is broadcasted periodically. The broadcast program σ can start from any XML document d_i. However, we assume that σ starts from d_1 (this will then comply with the definition of σ in Table 6.1) to simplify our analysis.

With the basic assumption that queries can be issued at any time with an equal probability (this means the issue time of queries follows a uniform distribution), we can calculate the expected access time of q, denoted as $\mathscr{AT}^q_{exp}$, in the following:

$$\begin{aligned}
\mathscr{AT}^q_{exp} &= \sum_{i=1}^{k}\left(\frac{\mathscr{L}_{d_{n_i}}}{\mathscr{L}_\sigma}\cdot\mathscr{L}_\sigma + \frac{\mathscr{L}_{gap_i}}{\mathscr{L}_\sigma}\cdot\left(\mathscr{L}_\sigma - \frac{1}{2}\cdot\mathscr{L}_{gap_i}\right)\right) \\
&= \sum_{i=1}^{k}\mathscr{L}_{d_{n_i}} + \sum_{i=1}^{k}\mathscr{L}_{gap_i} - \frac{1}{\mathscr{L}_\sigma}\cdot\sum_{i=1}^{k}\frac{1}{2}\cdot\mathscr{L}^2_{gap_i} \\
&= \mathscr{L}_\sigma - \frac{1}{2\cdot\mathscr{L}_\sigma}\cdot\sum_{i=1}^{k}\mathscr{L}^2_{gap_i} \qquad (6.4)
\end{aligned}$$

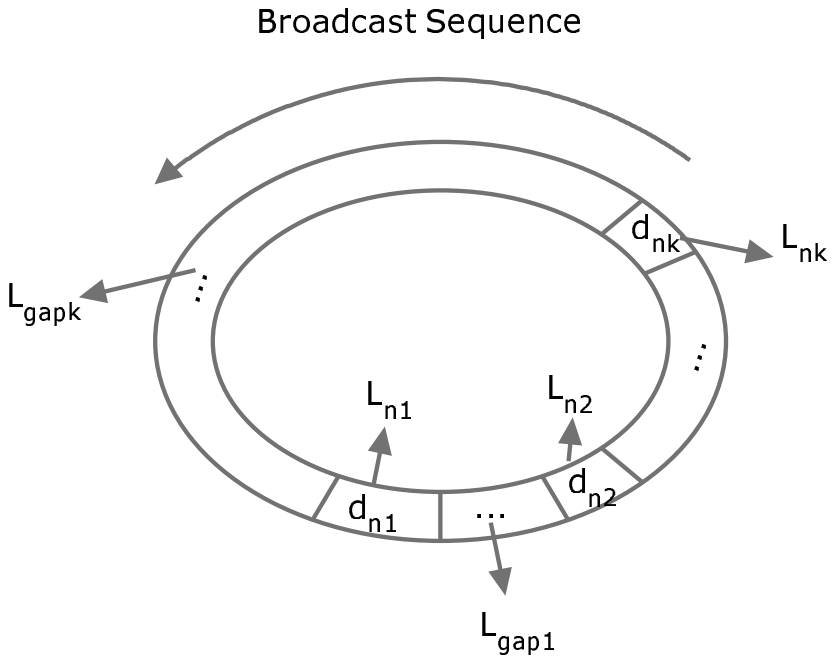


Fig. 6.3 A broadcast program showing positions of documents required by query q

In this equation, $\mathscr{L}_{d_{n_i}}$ refers to the length of document d_{n_i}, $\mathscr{L}_\sigma$ refers to the total length of all documents in σ and $\mathscr{L}_{gap_i}$ refers to the total length of all documents in gap_i.

According to Equation (6.4)[2] and a given broadcast program σ, we can calculate $\mathscr{AT}^q_{exp}$ simply according to the gaps between consecutive documents required by q. Further, from the above equation, we can see that in order to improve the expected access efficiency, $\sum_{i=1}^{k} \mathscr{L}^2_{gap_i}$ should be as large as possible.

Moreover, the sum of all gaps, denoted as $\mathscr{L}_{gaps}$, can be computed as

$$\mathscr{L}_{gaps} = \sum_{i=1}^{k} \mathscr{L}_{gap_i} = \mathscr{L}_\sigma - \sum_{i=1}^{k} \mathscr{L}_{d_{n_i}} = \mathscr{L}_\sigma - \mathscr{L}_{\sigma_q} \tag{6.5}$$

[2]This result is exactly the same as [147] although the deduction process is different. The further analysis on this result in the following is new.

Here, $\mathscr{L}_{\sigma_q}$ refers to the total length of all documents required by q, which is $\sum_{i=1}^{k} \mathscr{L}_{d_{n_i}}$. Note that $\mathscr{L}_{\sigma_q}$ is independent of any data placement results. In other words, $\mathscr{L}_{\sigma_q}$ is fixed for a given q, which in turn indicates that $\mathscr{L}_{gaps}$ is fixed.

In order to derive the lower and upper bounds of $\sum_{i=1}^{k} \mathscr{L}_{gap_i}^2$ and to analyze our data placement strategy, we first present the following propositions for the below function $f(\mathbf{X})$.

Function $f(\mathbf{X}) = {x_1}^2 + {x_2}^2 + \ldots + {x_k}^2$ is with the following constraints:

1. $x_1 + x_2 + \ldots + x_k = M$
2. $x_1, x_2, \ldots, x_k \geq 0$

where $M > 0$. We also denote the lower bound and the upper bound of $f(\mathbf{X})$ as $\underline{f(\mathbf{X})}$ and $\overline{f(\mathbf{X})}$ respectively.

Proposition 1 *Given $f(\mathbf{X})$ defined as above, we must have*

$$f(\mathbf{X}) \leq M^2$$

When $x_1 = x_2 = \ldots = x_{k-1} = 0$ and $x_k = M$ (or any other kind of combinations like this), $f(\mathbf{X})$ reaches its upper bound, i.e., $\overline{f(\mathbf{X})} = M^2$.

Proof: According to the two constraints of function $f(\mathbf{X})$, we have

$$\begin{aligned} f(\mathbf{X}) &= (x_1 + x_2 + \ldots + x_n)^2 - 2 \cdot \sum_{i \neq j} x_i \cdot x_j \\ &\leq M^2 \end{aligned}$$

When $x_1 = x_2 = \ldots = x_{k-1} = 0$ and $x_k = M$ (or any other kind of combinations like this, which means we have only one positive variable[3].), $f(\mathbf{X})$ reaches its upper bound $\overline{f(\mathbf{X})} = M^2$. In all other cases, if two or more variables are positive, i.e. $x_1 > 0$ and $x_2 > 0$, we have $2 \cdot \sum_{i \neq j} x_i \cdot x_j \geq 2 \cdot (x_1 \cdot x_2) > 0$ which indicates $f(\mathbf{X}) < M^2$. □

[3]Note that we cannot have all variables set to be 0 since M is positive.

Proposition 2 *Given $f(\mathbf{X})$ defined as above, we must have*

$$f(\mathbf{X}) \geq k \cdot (\frac{M}{k})^2$$

When $x_1 = x_2 = \ldots + x_k = \frac{M}{k}$, $f(\mathbf{X})$ reaches its lower bound, i.e., $\underline{f(\mathbf{X})} = k \cdot (\frac{M}{k})^2$.

Proof: We utilize mathematical induction to prove it (for all $k \geq 1$).

For the base step, when $k = 1$, $f(\mathbf{X}) = x_1^2$, it is trivial to prove that $f(\mathbf{X}) = M^2 \geq 1 \cdot (\frac{M}{1})^2$ and $\underline{f(\mathbf{X})} = M^2$ since $x_1 = M = \frac{M}{1}$.

For the inductive step, we assume that when $k = n$, $f(\mathbf{X}) \geq n \cdot (\frac{M}{n})^2$ and when $x_1 = x_2 = \ldots = x_n = \frac{M}{n}$, $f(\mathbf{X})$ reaches its lower bound $\underline{f(\mathbf{X})} = n \cdot (\frac{M}{n})^2$. Then for $k = n+1$, we have

$$\begin{aligned} f(\mathbf{X}) &= \sum_{i=1}^{n} x_i^2 + x_{n+1}^2 \\ &\geq n \cdot (\frac{M - x_{n+1}}{n})^2 + x_{n+1}^2 \\ &= \frac{(n+1) \cdot x_{n+1}^2 - 2 \cdot M \cdot x_{n+1} + M^2}{n} \\ &= \frac{(n+1) \cdot (x_{n+1} - \frac{M}{n+1})^2 + \frac{n \cdot M^2}{n+1}}{n} \\ &\geq (n+1) \cdot (\frac{M}{n+1})^2 \end{aligned}$$

From the above induction, we can see that when $x_{n+1} = \frac{M}{n+1}$ and $x_1 = x_2 = \ldots = x_n = \frac{M - x_{n+1}}{n} = \frac{M}{n+1}$, $f(\mathbf{X})$ reaches its lower bound $\underline{f(\mathbf{X})} = (n+1) \cdot (\frac{M}{n+1})^2$.

Because we have shown both the base step and the inductive step, by the principle of mathematical induction the proposition is true. □

Moreover, given $f(\mathbf{X})$ defined as above and suppose that m variables, i.e. $x_1, x_2, \ldots, x_m$, have been determined ($m < k$) while the rest $k - m$ variables are not. We denote $M' = M - \sum_{i=1}^{m} x_i$. Now we are going to determine the next variable. Without loss of generality, we use x_{m+1} as our next variable to be determined and aim to maximize or minimize $f(\mathbf{X})$. We denote $f(\mathbf{X})_{x_{m+1}}$ as the function with $m+1$ determined variables

(x_i, $1 \leq i \leq m+1$) and $k-m-1$ undetermined variables. Then given two values of this variable, i.e. x_{m+1} and x'_{m+1} and suppose $x_{m+1} < x'_{m+1}$, we have the following propositions:

Proposition 3 *For* $\overline{f(\mathbf{X})}_{x_{m+1}}$ *and* $\overline{f(\mathbf{X})}_{x'_{m+1}}$, *we have*

$$\begin{cases} \overline{f(\mathbf{X})}_{x_{m+1}} > \overline{f(\mathbf{X})}_{x'_{m+1}} & x_{m+1} < x'_{m+1} \leq \frac{M'}{2} \\ \overline{f(\mathbf{X})}_{x_{m+1}} < \overline{f(\mathbf{X})}_{x'_{m+1}} & \frac{M'}{2} \leq x_{m+1} < x'_{m+1} \\ Indefinite & Otherwise \end{cases}$$

Proof: According to our definitions of $f(\mathbf{X})_{x_{m+1}}$, we have

$$f(\mathbf{X})_{x_{m+1}} = \sum_{i=1}^{m} x_i^2 + x_{m+1}^2 + \sum_{i=m+2}^{k} x_i^2$$

Since $M' = M - \sum_{i=1}^{m} x_i = \sum_{i=m+1}^{k} x_i$, according to Proposition 1, we have

$$\begin{aligned} \overline{f(\mathbf{X})}_{x_{m+1}} &= \sum_{i=1}^{m} x_i^2 + x_{m+1}^2 + (M' - x_{m+1})^2 \\ &= \sum_{i=1}^{m} x_i^2 + 2 \cdot x_{m+1}^2 - 2 \cdot M' \cdot x_{m+1}^2 + M'^2 \\ &= \sum_{i=1}^{m} x_i^2 + 2 \cdot (x_{m+1} - \frac{M'}{2})^2 + \frac{M'^2}{2} \end{aligned}$$

According to the above result, for any $x_{m+1} < x'_{m+1}$ we can infer that

1. $\overline{f(\mathbf{X})}_{x_{m+1}} > \overline{f(\mathbf{X})}_{x'_{m+1}}$, if $x_{m+1} < x'_{m+1} \leq \frac{M'}{2}$
2. $\overline{f(\mathbf{X})}_{x_{m+1}} < \overline{f(\mathbf{X})}_{x'_{m+1}}$, if $\frac{M'}{2} \leq x_{m+1} < x'_{m+1}$
3. Indefinite for all other cases

□

Proposition 4 *For* $\underline{f(\mathbf{X})}_{x_{m+1}}$ *and* $\underline{f(\mathbf{X})}_{x'_{m+1}}$, *we have*

$$\begin{cases} \underline{f(\mathbf{X})}_{x_{m+1}} > \underline{f(\mathbf{X})}_{x'_{m+1}} & x_{m+1} < x'_{m+1} \leq \frac{M'}{k-m} \\ \underline{f(\mathbf{X})}_{x_{m+1}} < \underline{f(\mathbf{X})}_{x'_{m+1}} & \frac{M'}{k-m} \leq x_{m+1} < x'_{m+1} \\ Indefinite & Otherwise \end{cases}$$

Proof: According to our definitions of $f(\mathbf{X})_{x_{m+1}}$, we have

$$f(\mathbf{X})_{x_{m+1}} = \sum_{i=1}^{m} x_i^2 + x_{m+1}^2 + \sum_{i=m+2}^{k} x_i^2$$

Since $M' = M - \sum_{i=1}^{m} x_i = \sum_{i=m+1}^{k} x_i$, according to Proposition 2, we have

$$\begin{aligned} \underline{f(\mathbf{X})}_{x_{m+1}} &= \sum_{i=1}^{m} x_i^2 + x_{m+1}^2 + \\ &\quad (k-m-1) \cdot (\frac{M' - x_{m+1}}{k-m-1})^2 \\ &= \sum_{i=1}^{m} x_i^2 + \\ &\quad \frac{(k-m)x_{m+1}^2 - 2 \cdot M' \cdot x_{m+1} + M'^2}{k-m-1} \\ &= \sum_{i=1}^{m} x_i^2 + \\ &\quad \frac{(k-m)(x_{m+1} - \frac{M'}{k-m})^2 + \frac{(k-m-1)M'^2}{k-m}}{k-m-1} \end{aligned}$$

According to the above result, for any $x_{m+1} < x'_{m+1}$ we can infer that

1. $\underline{f(\mathbf{X})}_{x_{m+1}} > \underline{f(\mathbf{X})}_{x'_{m+1}}$, if $x_{m+1} < x'_{m+1} \leq \frac{M'}{k-m}$
2. $\underline{f(\mathbf{X})}_{x_{m+1}} < \underline{f(\mathbf{X})}_{x'_{m+1}}$, if $\frac{M'}{k-m} \leq x_{m+1} < x'_{m+1}$
3. Indefinite for all other cases

□

Now according to Proposition 1 and Proposition 2, we have

$$k \cdot (\frac{\mathscr{L}_{gaps}}{k})^2 \leq \sum_{i=1}^{k} \mathscr{L}_{gap_i}^2 \leq \mathscr{L}_{gaps}^2 \tag{6.6}$$

Then according to Equation (6.4), we have

$$\mathscr{L}_\sigma - \frac{\mathscr{L}_{gaps}^2}{2 \cdot \mathscr{L}_\sigma} \leq \mathscr{AT}_{exp}^q \leq \mathscr{L}_\sigma - \frac{\mathscr{L}_{gaps}^2}{2 \cdot k \cdot \mathscr{L}_\sigma} \tag{6.7}$$

From the above two inequalities, we can see that in order to improve the expected access efficiency, $\sum_{i=1}^{k} \mathscr{L}_{gap_i}^2$ should be as large as possible. According to Proposition 1, when we have one of the gaps equals to $\mathscr{L}_{gaps}$ and all other gaps equal to 0, we can achieve the best expected access efficiency. Thus, when all XML documents required by q are placed together and broadcasted in sequence, $\mathscr{AT}_{exp}^q$ can be minimized. Also, according to Equation (6.5), we can rewrite the above inequality to

$$\mathscr{L}_\sigma - \frac{(\mathscr{L}_\sigma - \mathscr{L}_{\sigma_q})^2}{2 \cdot \mathscr{L}_\sigma} \leq \mathscr{AT}_{exp}^q \leq \mathscr{L}_\sigma - \frac{(\mathscr{L}_\sigma - \mathscr{L}_{\sigma_q})^2}{2 \cdot k \cdot \mathscr{L}_\sigma} \tag{6.8}$$

Here Inequality (6.8) shows both the lower and upper bounds of $\mathscr{AT}_{exp}^q$ for q in another form. It is worth mentioning that both bounds are independent of any data placement results. Moreover, we can infer that when k increases, σ_q will include more documents. Then $\mathscr{L}_{\sigma_q}$ increases as well. However, the decrease of difference $\mathscr{L}_\sigma - \mathscr{L}_{\sigma_q}$ leads to larger lower and upper bounds of $\mathscr{AT}_{exp}^q$, which means the system's overall performance will degrade.

The above analysis focuses on a single query. However, generalizing it to multiple queries would be much more complicated, as determining an optimal broadcast sequence for multiple multi-item queries is an NP-Complete problem [147].

When there are multiple queries to consider for a broadcast program, these queries are not likely to require the same XML documents. In such cases, Proposition 1 and Inequality (6.6), which minimizes expected access time for a single query, cannot assist in finding an optimal solution for all queries.

However, from Inequalities (6.6) and (6.8) we learn that the upper and lower bounds of $f(\mathbf{X})$ should be as large as possible which can lead to larger probability of having large results of $f(\mathbf{X})$. This in turn enlarges the probability that we have smaller $\mathscr{AT}^{q}_{exp}$ for q according to Inequality (6.6) and (6.8). Then, for multiple queries, according to Proposition 3 and Proposition 4, we know that when $x_{m+1} \leq \frac{M'}{2}$ and $x_{m+1} \leq \frac{M'}{k-m}$, we should decrease x_{m+1} to have larger $\overline{f(\mathbf{X})}_{x_{m+1}}$ and $\underline{f(\mathbf{X})}_{x_{m+1}}$. In other words, if we progressively reduce each gap as much as possible, we would have larger lower and upper bounds of $\sum_{i=1}^{k} \mathscr{L}^{2}_{gap_i}$. In this way, we can reduce both the lower and upper bounds of the overall expected access time with a higher probability.

For example, if we try to minimize $\sum_{i=1}^{k} \mathscr{L}^{2}_{gap_i}$ for each query in $\{q_1, q_2, q_3\}$, for the first step, we should place XML documents that are required by all three queries together to form an initial broadcast program. In the second step, we should place XML documents required by only two of the three queries together and append them to the initial broadcast program. After that, we append XML documents required by only one query to the broadcast program to form a final broadcast program. Hence, we can construct a final broadcast program in a greedy style.

Now the problem becomes how we can determine which documents should be placed together first as we cannot obtain queries in advance. Our solution will be discussed in the next section.

6.4 The Data Placement Algorithm

In this section, we introduce the data placement algorithm for periodic XML data broadcasts, based on our theoretical analysis in the previous section. We first discuss the structural sharing property of XML data, which is used to estimate the potential access patterns of mobile clients, i.e., the probability of accessing a small set of similar XML documents simultaneously. Then we put forward a novel greedy data placement algorithm based on it.

6.4.1 Structural Sharing in XML Data

Intuitively, for any two given XML documents, we can utilize one of the three structural similarity metrics described in Section 6.2.3 to calculate the similarity between them and the similarity results can be used to approximate the probability that a specific query is matched with both documents at the same time. For example, if two XML elements are under structurally similar paths, then it is more likely that that either both elements satisfy, or none satisfies, a given query [140]. Therefore, if two XML documents are with larger structural similarity, i.e. d_1 and d_2, then they would have a higher probability to be required simultaneously. However, there are still three other cases to be considered, such as requiring d_1 but not d_2, requiring d_2 but not d_1 and requiring neither of d_1 and d_2. Therefore, the above similarity metrics consider only successful match probabilities of both XML documents and do not consider unsuccessful match probabilities.

Nonetheless, unsuccessful match cases have effects on the expected access time as well. According to Proposition 1 and Proposition 2, in order to have better access efficiency, the gaps between any two required documents by a single query should be as less uniform as possible. Based on this, we can infer that in the above example, cases of required d_1 but not d_2 and required d_2 but not d_1 are likely to generate more uniform gaps while other two cases (required both documents or neither) are likely

to have less uniform gaps. Observing this, we define a new similarity metric called *Cohesion* to give a more accurate estimation of access patterns of mobile clients in the following.

Note that, for any query q requiring at least one of the documents in $\mathscr{D}$, q must match some paths in $PS(\mathscr{D})$ and it has a probability of $\frac{|PS(d)|}{|PS(\mathscr{D})|}$ of matching d. If a query q fails to match any document in $\mathscr{D}$, the issuer of q only needs to locate and download an air index to confirm that his/her query does not match any document. Then he/she can stop waiting for the result to be broadcasted. All these queries only need to access the index information on air and therefore, their expected access time depends heavily on the index distribution, which is not the focus of our work. To estimate their expected access time, interested readers are referred to [139] for more details. Hence, we only consider successful queries in this work.

Now suppose we have a set of n XML documents $\mathscr{D} = \{d_1, d_2, \ldots, d_n\}$ on the server, we can approximate the access probability of any document d for queries which successfully match at least one document in the set $\mathscr{D}$ as follows:

$$Pr(d) = \frac{|PS(d)|}{|PS(\mathscr{D})|} \tag{6.9}$$

and for any i, j $(1 \leq i, j \leq n)$

$$Pr(d_i - d_j) = \frac{|PS(d_i) - PS(d_j)|}{|PS(\mathscr{D})|} \tag{6.10}$$

Here, $PS(\mathscr{D}) = \bigcup_{i=1}^{n} PS(d_i)$.

There would be many different matching cases for a given set $\mathscr{D}$. Take two XML documents d_1 and d_2 in $\mathscr{D}$ as an example. As mentioned previously, there would be four cases of matching of them and the probability of each case is shown in Table 6.2.

Table 6.2 Matching Cases for Document d_1 and d_2 in a Document Set $\mathscr{D}$

Case	Probability	Effect on AT_{exp}
Matched both d_1, d_2	$Pr(d_1 \bigcap d_2)$	Positive
Matched none of d_1, d_2	$1 - Pr(d_1 \bigcup d_2)$	Positive
Matched d_1, but not d_2	$Pr(d_1 - d_2)$	Negative
Matched d_2, but not d_1	$Pr(d_2 - d_1)$	Negative

In this table, we also include positive and negative effects on the expected access time (AT_{exp}) for each case.

Based on Table 6.2 and the previous analysis, we can see that, given two documents, if both documents are matched against the same query or neither of the documents is matched, the two documents will tend to be more similar to each other, i.e., $Pr(d_1 \bigcap d_2)$ or $1 - Pr(d_1 \bigcup d_2)$ is higher. On the other hand, if only one of the two documents is matched against a given query, the similarity between them tends to be smaller, i.e., $1/Pr(d_1 - d_2)$ or $1/Pr(d_2 - d_1)$ is smaller. Based on this observation, we define Cohesion $C(d_i, d_j)$ of XML documents d_i and d_j as follows:

$$C(d_i, d_j) = \frac{Pr(d_i \bigcap d_j) \cdot (1 - Pr(d_i \bigcup d_j))}{max\{Pr(d_i - d_j), Pr(d_j - d_i)\}} \tag{6.11}$$

Here d_i and d_j are both in set $\mathscr{D}$. It is easy to see that $C(d_i, d_j) = C(d_j, d_i)$. According to Equation (6.9), Equation (6.10) and Equation (6.11), we can calculate $C(d_i, d_j)$ after finding path sets of d_i, d_j in $\mathscr{D}$. Cohesion values can vary in a wide range which exceeds interval $[0, 1]$. Strictly speaking, Cohesion values only vary in interval $[0, \frac{|PS(\mathscr{D})|}{4}]$ given that $C(d_i, d_j) = \frac{|PS(\mathscr{D})|}{4}$ when $PS(d_i) = PS(d_j)$. The lower bound 0 is trivial. In order to obtain the upper bound, we only consider cases that have $PS(d_i) \neq PS(d_j)$, from which we can infer that $max\{|PS(d_i - d_j)|, |PS(d_j - d_i)|\} \geq 1$. Without loss of generality, let $|PS(d_i)| \geq |PS(d_j)|$, according to Equation (6.9) and Equation (6.10), we can rewrite Equation (6.11) as follows:

Algorithm 3 Initialize structural sharing matrix $S[n][n]$

Input: A set of XML documents $\mathscr{D} : \{d_1, d_2, ..., d_n\}$
Output: Structural sharing matrix $S[n][n]$

1. create matrix $S[n][n]$
2. **for** each document d in $\mathscr{D}$ **do**
3. compute $PS(d)$
4. **end for**
5. **for** each pair of documents $< d_i, d_j >$ in $\mathscr{D}$ $(i < j)$ **do**
6. $S[i][j] \Leftarrow$ structural sharing between d_i and d_j
7. $S[j][i] \Leftarrow S[i][j]$
8. **end for**

$$
\begin{aligned}
C(d_i, d_j) &\leq \frac{\frac{|PS(d_i \bigcap d_j)|}{|PS(\mathscr{D})|} \cdot (1 - \frac{|PS(d_i \bigcup d_j)|}{|PS(\mathscr{D})|})}{\frac{1}{|PS(\mathscr{D})|}} \\
&< \frac{|PS(d_i)| \cdot (|PS(\mathscr{D})| - |PS(d_i)|)}{|PS(\mathscr{D})|} \\
&= \frac{-(|PS(d_i)| - \frac{|PS(\mathscr{D})|}{2})^2 + \frac{|PS(D)|^2}{4}}{|PS(\mathscr{D})|} \\
&\leq \frac{|PS(\mathscr{D})|}{4}
\end{aligned}
$$

Then the above result gives the upper bound of Cohesion $C(d_i, d_j)$. Now we can normalize Cohesion values to interval $[0, 1]$ in the following

$$
C'(d_i, d_j) = \begin{cases} \frac{4 \cdot C(d_i, d_j)}{|PS(\mathscr{D})|} & PS(d_i) \neq PS(d_j) \\ 1 & PS(d_i) = PS(d_j) \end{cases} \tag{6.12}
$$

We can also infer that $C'(d_i, d_j) = 1$ if and only if $PS(d_i) = PS(d_j)$. Similar to the other three similarity metrics, the larger the value of Cohesion is, the more structural sharing the two comparing XML documents have.

Algorithm 4 GDPA

Input: Structural sharing matrix $S[n][n]$
Output: A broadcast program σ for $\mathscr{D}$

1. $\sigma \Leftarrow$ empty sequence
2. select a pair of documents $< d_i, d_j >$ with maximum value $S[i][j]$ in matrix $S[n][n]$
3. **if** $L_{d_i} <= L_{d_j}$ **then**
4. add $< d_i, d_j >$ into σ
5. **else**
6. add $< d_j, d_i >$ into σ
7. **end if**
8. $\mathscr{D}' \Leftarrow \mathscr{D} - d_i - d_j$
9. **while** $\mathscr{D}'$ is not empty **do**
10. $d_{head} \Leftarrow$ the first document in σ
11. select a pair of documents $< d_{i_{max}}, d_{head} >$ with maximum value $S[i_{max}][head]$ $(d_{i_{max}} \in \mathscr{D}')$
12. $d_{rear} \Leftarrow$ the last document in σ
13. select a pair of documents $< d_{j_{max}}, d_{rear} >$ with maximum value $S[j_{max}][rear]$ $(d_{j_{max}} \in \mathscr{D}')$
14. **if** $S[i_{max}][head] >= S[j_{max}][rear]$ **then**
15. append $d_{i_{max}}$ into σ from head
16. $\mathscr{D}' \Leftarrow \mathscr{D}' - d_{i_{max}}$
17. **else**
18. append $d_{j_{max}}$ into σ from rear
19. $\mathscr{D}' \Leftarrow \mathscr{D}' - d_{j_{max}}$
20. **end if**
21. **end while**

6.4.2 The Greedy Data Placement Algorithm

Based on the discussions of structural sharing between XML documents, we can generate a broadcast program for periodic data broadcasts in a greedy way. From previous discussions, we can see that if more structural sharing of two XML documents is observed, we will have a larger probability of matching both XML documents simultaneously. As a result, our Greedy Data Placement Algorithm (GDPA) places XML documents with most structural sharing together first as an initial broadcast program. Then it progressively appends other XML documents to the broadcast program in a descending order of structural sharing. Detailed steps of GDPA are shown in Algorithm 3 and Algorithm 4.

Algorithm 3 initializes a structural sharing matrix $S[n][n]$ for n XML documents on the broadcast server. Note that, all four similarity metrics defined in Section 6.2.3 and 6.4.1 can be used in Algorithm 3 to compute structural sharing between two documents (Line 6). All of them are symmetric which means for any one of these metrics, we must have $S[j][i] = S[i][j]$. Also we have $J(d_i,d_j) = D(d_i,d_j) = L(d_i,d_j) = C'(d_i,d_j) = 1$ if $i = j$. Therefore, we only need to calculate matrix S for entries $S[i][j]$ where $i < j$.

Based on matrix S, Algorithm 4 finds the pair of XML documents with maximum structural sharing value and adds them into the initial empty broadcast program σ (Line 2). As discussed in Section 6.3, the expected access time is determined by the gaps between the required documents but not by the sequence of them. Therefore, the sequence of the first pair of XML documents can be simply placed on the broadcast channel according to the ascending order of document lengths (Line 3 to 7). Next, Algorithm 4 identifies the XML document (called d) with the maximum structural sharing to the head document d_{head} or the rear document d_{rear} of σ. If the maximum structural sharing is obtained between document d and document d_{head}, d will be appended into σ from head; otherwise, d will be appended into σ from rear. This process will be repeated until all XML documents are placed into σ in order (Line 9 to 21).

Time Complexity. Suppose there are n documents in total on the server and each document contains p elements (or in other words, p paths) on average. Then in Algorithm 3, the **for** loop from Line 2 to Line 4 takes $O(np\log p)$ time (we propose the use of sorted sets here in order to speed up the set operations in the following lines). Note that intersections or differences between two sorted sets take linear time, which means Line 6 in Algorithm 3 takes $O(p)$ time. Therefore, the **for** loop from Line 5 to Line 8 takes $O(n^2p)$ time since there are $O(n^2)$ pairs of documents. Finally, in Algorithm 4, Line 2 takes $O(n^2)$ times and the **while** loop from Line 9 to Line 21

takes $O(n^2)$ time as well. Putting all the times together, we will get the total time complexity of both Algorithm 3 and Algorithm 4, which is $O(np\log p + n^2 p + n^2)$.

Now consider the scenarios of adding a new document or removing an existing document from the document set on the server. For adding a new document, the update time complexity would be $O(p\log p + np + n^2)$. This is because it takes $O(p\log p)$ time to compute the sorted path set for the new document, $O(np)$ time to intersect with the n existing documents on the server and finally, it takes $O(n^2)$ to recompute the whole data placement. For removing an existing document from the server, the update time complexity would be just $O(n^2)$ as we only need to recompute the data placement for the rest $n-1$ documents.

6.4.3 Index Distribution Strategy

In order to improve energy conservation, smart mobile devices can switch between two operation modes: *active mode* and *doze mode*. In the active mode, a device can listen, compare, and download the required data; while in the doze mode, it turns off antennas and some processes to save energy. The energy consumed in active mode can be up to 100 times of that in doze mode [148]. However, after we have generated a broadcast program σ, mobile clients cannot locate information relevant to their interests as there is no auxiliary information to assist them, which means mobile clients would need to stay in active mode all the time. Air indexing can help to solve this problem.

Air index is a small amount of auxiliary information of the broadcast program σ and is used to assist mobile clients to calculate the arrival time of information that they are interested in [139]. The basic idea of air index is that the broadcast server pre-computes index information (including searchable attributes and delivery time of data items) and interleaves it with data items (e.g., XML documents) on the broadcast channel. As mentioned, without air index, mobile clients would have to download all

XML documents on air to examine which ones satisfy their requests. Therefore, we need to apply air indexing technique to our generated broadcast program σ. With the technique, mobile clients can avoid to examine documents on air one by one. Instead, they can switch to doze mode when uninterested documents are broadcasted and switch to active mode only when interested documents arrive. After the interested documents have been retrieved, they can switch back to doze mode again. In this way, we can significantly save energy for mobile clients.

The basic idea of air indexing technique is to provide auxiliary index information that annotates the broadcast data on air. Based on index information broadcasted along with XML data, mobile clients are able to selectively skip unwanted data by switching into doze mode and switching back to active mode only when the data of desire arrives. This technique helps to reduce energy consumption of mobile clients. It is true that the clients still need to stay in active mode to receive matched data after introducing broadcast indexes. But the clients can simply skip unmatched data in the meantime. In contrast, if broadcasting data without indexes, the clients will have to examine all the data on the broadcast channel (no matter the data is matched or not). In other words, the broadcast indexes provide a way for the clients to avoid scanning or downloading irrelevant data so as to reduce energy consumption.

In this work, we adopt Compact Index (CI) [129] as our index structure and $(1,m)$ index scheme [139] as our index distribution strategy.

CI provides a two-tier air index scheme [129]. The basic structure of CI is the combination of all the DataGuides of XML documents in σ, which utilizes RoXSum [149] technique to integrate these DataGuides. In this basic index structure, every unique-label path of a document appears exactly once for supporting simple XPath queries (like DataGuides). Thus it contains the entire unique-label paths in all of the XML documents. CI also includes a two-tier structure which enables efficient access protocol at the client which facilitates the index access. Normally, CI is only 1.5% of the original XML data in terms of size. More details can be found in [129].

also

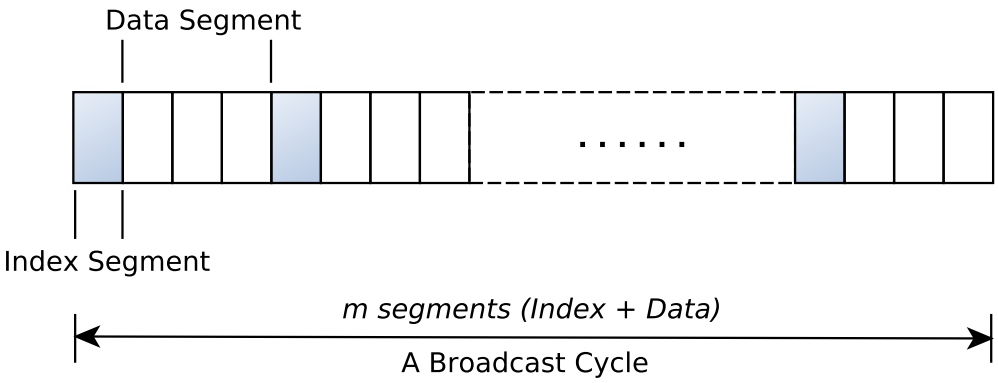


Fig. 6.4 $(1,m)$ index distribution

The idea of $(1,m)$ index scheme [139] is shown in Figure 6.4. From the figure, we can see that the generated broadcast program σ is divided into m data segments and before broadcasting each data segment, an index segment will be broadcasted first. The optimal value m is given by $\sqrt{\frac{\mathscr{L}_{data}}{\mathscr{L}_{index}}}$. Here, $\mathscr{L}_{data}$ refers to the total length of the raw broadcast program σ without any index information and $\mathscr{L}_{index}$ refers to the length of index, which is CI in our work. $(1,m)$ index scheme can best balance access time and tuning time of mobile clients. More details can be found in [139].

6.5 Experiments

In this section, we study the performance of our data placement algorithm. We show its efficiency in terms of access time, which is a common measure of performance in data broadcasts. Since this is the first work that determines broadcast schedules based only on XML data on the server, we compare our algorithm with a common random data placement algorithm (RDPA).

Table 6.3 Data Sets in Our Experiments

Set Name	Length			Remark
	Minimum	Maximum	Average	
*DS*1	2.4KB	8.1KB	5.0KB	6 clusters
*DS*2	0.5KB	45.9KB	12.4KB	miscellaneous
*DS*3	2.4KB	24.8KB	9.9KB	hybrid
*DS*4	0.5KB	55.8KB	12.3KB	miscellaneous
*DS*5	0.3KB	65.6KB	12.7KB	miscellaneous

6.5.1 Experimental Setup

The experiments are run on three data sets (*DS*1, *DS*2, *DS*3) each with 250 XML documents and two more data sets (*DS*4, *DS*5) with 500 and 1000 XML documents, respectively. All generated documents are defined by News Industry Text Format (NITF) DTD [150]. This DTD is published for news copy production, press releases, and Web-based news organizations. The average depth of all five document sets is between 6 and 8 while the maximum depth is 22.

The details of these data sets are shown in Table 6.3. Data in *DS*1 can be well clustered into 6 clusters. Moreover, for any two documents d_i, d_j in two different clusters of *DS*1, the minimum similarity values, the maximum similarity values and the average similarity values of all four metrics (normalized Cohesion is adopted here) are shown in Table 6.4. We can see that all clusters are quite different from each other and share very little structural information. Data in *DS*2 are miscellaneous. Documents in *DS*2 cannot be classified into fine clusters. Data in *DS*3 are a mix of well-clustered data and miscellaneous data, which include 125 XML documents from *DS*1 and 125 XML documents from *DS*2. Similar to *DS*2, data in *DS*4, *DS*5 are miscellaneous as well.

In the experiments, XPath queries are generated using the generator developed by [151]. Queries are allowed to repeat. The generator provides several parameters to generate different types of XPath queries, such as query depth, probability of '*' and

Table 6.4 Similarity between Clusters in *DS*1

Metric	Similarity		
	Minimum	Maximum	Average
Jaccard	0.0097	0.1102	0.0435
Dice	0.0049	0.0583	0.0225
Lian	0.0057	0.1039	0.0345
Cohesion	0.0229	0.4620	0.1457

Table 6.5 Workload Parameters for the Experiments

Parameter	Range	Default	Description
PROB	5% to 30%	10%	probability(* and //)
QIR	0.1 to 5	1	query incoming rate
MQD	5 to 8	7	maximum query depth

'//' and so on. The probability of '*' and '//' appearing in each query's step is between 5% and 30% (denoted *PROB*, and the default value is 10%). Query Incoming Rate (denoted *QIR*) means the number of newly issued queries from mobile clients in a unit of time, *i.e.*, the time that mobile wireless system takes to broadcast a block of 1024-byte XML data. The maximum depth of generated XPath queries (denoted *MQD*) is between 5 and 8. Table 6.5 shows the value range of parameters in the experiments. It should be noted that we assume the user queries follow a uniform distribution.

The random data placement algorithm (denoted RDPA) is compared with GDPA (implemented using all four similarity metrics defined in Equations (6.1), (6.2), (6.3) and (6.12)). In RDPA, the server broadcasts XML documents in a random order. This random order is implemented by a Java class `Random` (JDK version 7). When applying a series of pseudorandom numbers on the order of XML documents in a broadcast program, there would be conflicts. For example, suppose there are N documents in total to be broadcasted. We need to generate pseudorandom numbers between 1 and N. After we have randomly placed k out of N documents, the next chosen document may be one of the first k documents. If such case happens, we simply ignore it and

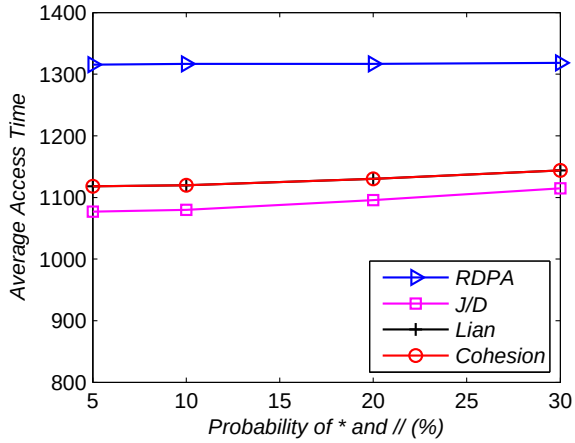
use Random class to generate pseudorandom numbers between 1 and $N-k$ for the rest $N-k$ documents. In this way, we can simulate a random order of N documents.

We implement both RDPA and GDPA on Java Platform Standard Edition 6 running on Windows 7 Enterprise, 64-bit Operating System. All our experiments are obtained by running 30 consecutive broadcast cycles. When we vary *PROB*, we set *QIR* and *MQD* to their default values. When we vary *QIR*, we set *PROB* and *MQD* to their default values. Similarly, when we vary *MQD*, we set *PROB* and *QIR* to their default values.

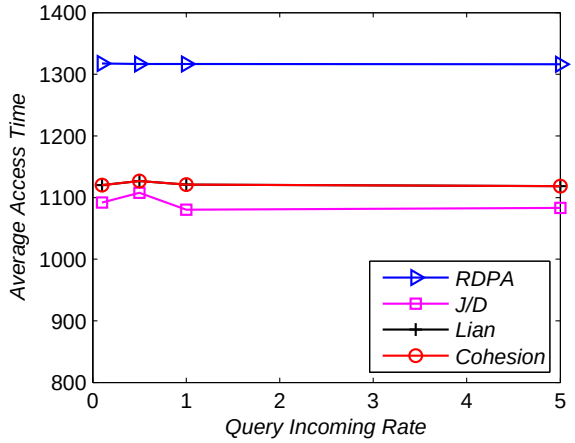
6.5.2 Performance of GDPA

Our experimental results are shown in Fig. 6.5 to 6.9. Average access time (*AAT*) is our performance metric and we seek to minimize it. Also we only consider *AAT* for all successful matched queries and abandon unsuccessful matched queries. The main reason for this is that, *AAT* of unsuccessful queries is determined by index distribution but not by data placement results (more details about this can be found in [139]). Note that, GDPA can be implemented with four different similarity metrics defined in Section 6.4, which are Jaccard metric, Dice's coefficient, Lian's metric and our proposed Cohesion. Through our experiments, Jaccard metric and Dice's coefficient always yield the same results. Therefore, we denote GDPA implemented with them as *J/D* method in all figures. Meanwhile, we denote GDPA implemented with Lian's metric as *Lian* method and denote GDPA implemented with Cohesion as *Cohesion* method.

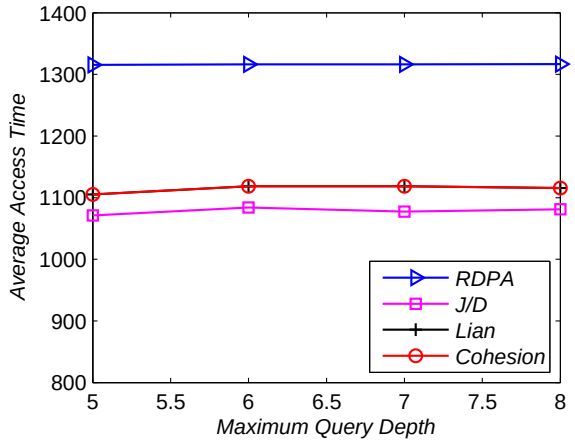
Fig. 6.5 shows the results on *DS*1. From the figure we can see that all GDPA methods outperform RDPA significantly. Specifically, the J/D method achieves the best results while the Lian method and Cohesion method provides similar results. This indicates that the J/D method better fits well-clustered data. Also, the reason for the Lian method and Cohesion method showing similar results is that both methods



(a) Varying *PROB*

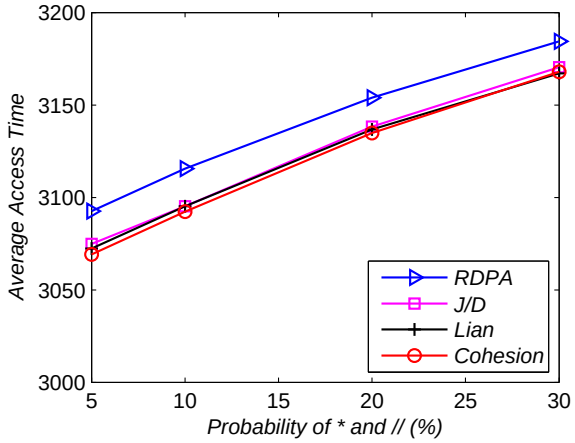


(b) Varying *QIR*

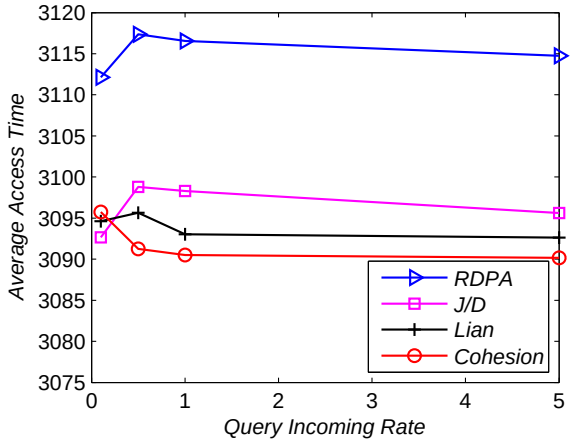


(c) Varying *MQD*

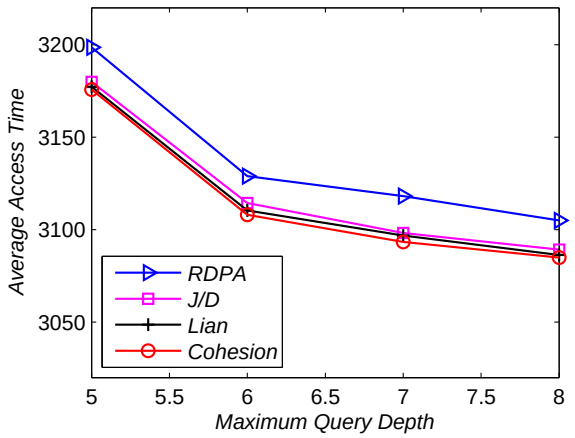
Fig. 6.5 Evaluating *AAT* Performance on *DS*1: well-clustered data set with 250 documents



(a) Varying *PROB*

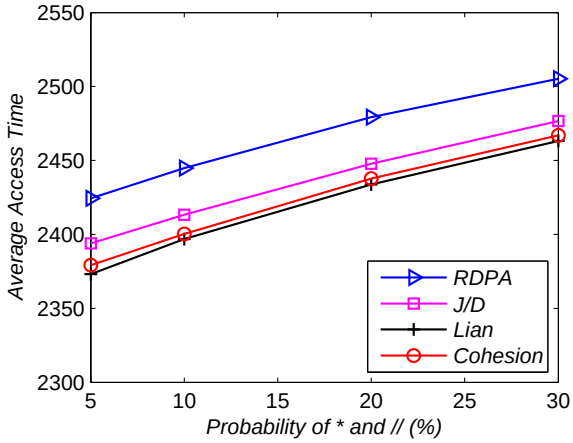


(b) Varying *QIR*

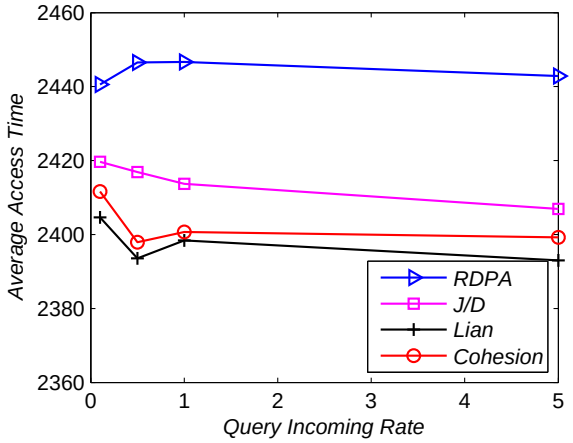


(c) Varying *MQD*

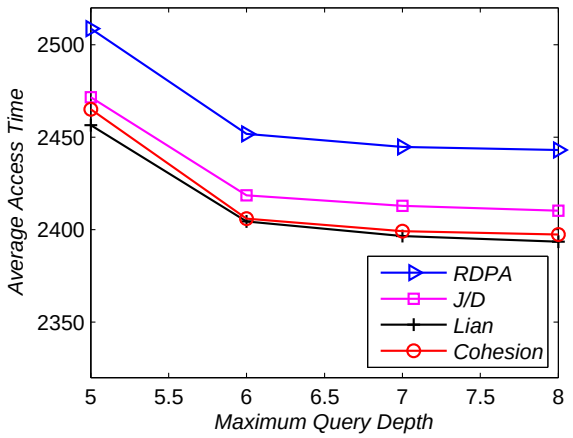
Fig. 6.6 Evaluating *AAT* Performance on *DS*2: miscellaneous data set with 250 documents



(a) Varying *PROB*

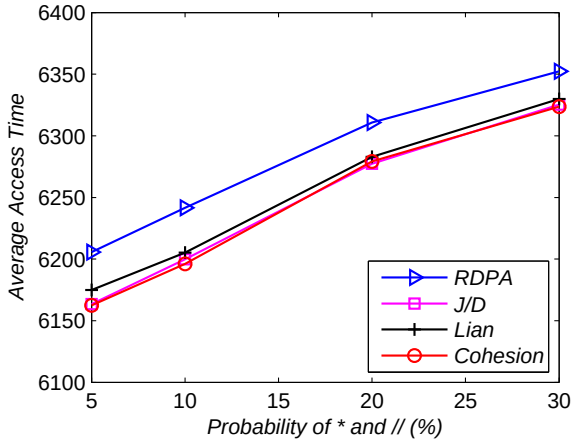


(b) Varying *QIR*

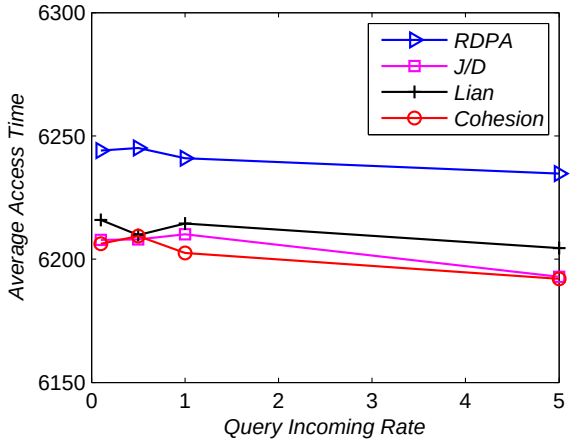


(c) Varying *MQD*

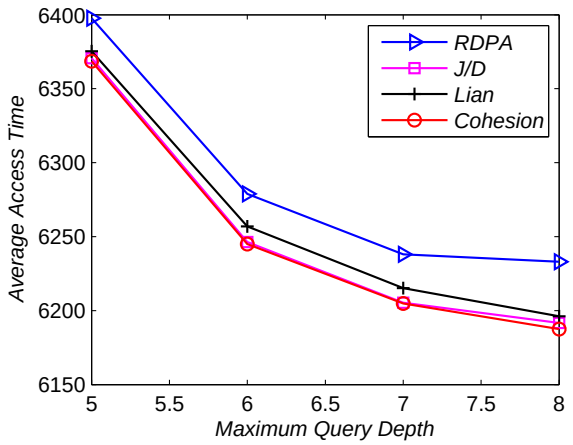
Fig. 6.7 Evaluating *AAT* Performance on *DS*3: a mixed set of well-clustered data and miscellaneous data with 250 documents



(a) Varying *PROB*

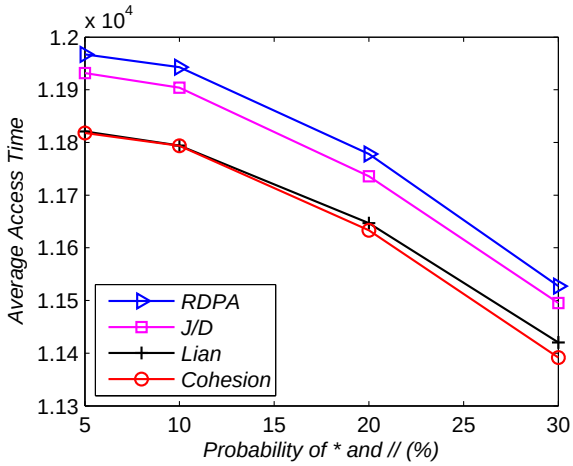


(b) Varying *QIR*

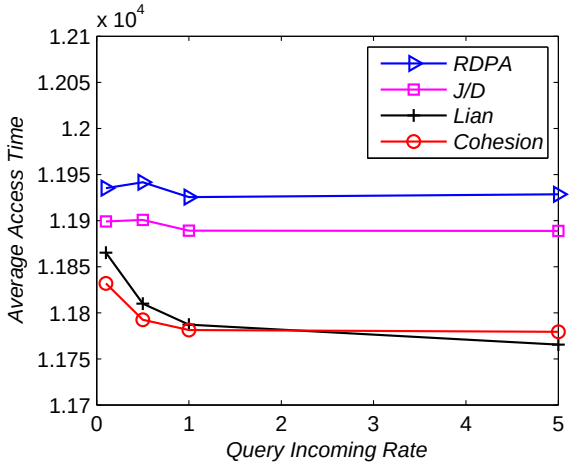


(c) Varying *MQD*

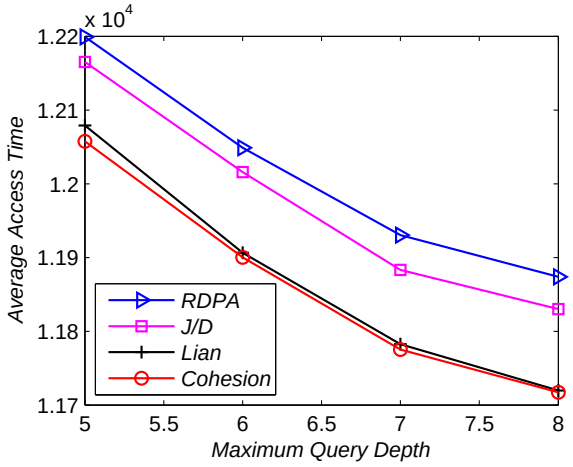
Fig. 6.8 Evaluating *AAT* Performance on *DS*4: miscellaneous data set with 500 documents



(a) Varying *PROB*



(b) Varying *QIR*



(c) Varying *MQD*

Fig. 6.9 Evaluating *AAT* Performance on *DS*5: miscellaneous data set with 1000 documents

place emphasis on the common structure of XML documents against the larger document (see also Equations (6.3) and (6.11)). Further, since *DS*1 is well-clustered, from the definitions of the Lian and Cohesion metrics, we can also infer that the partial order of similarity results using these two methods are similar to each other. Note that, according to the data placement algorithm described in Section 6.4, the data placement results are determined by the partial order of the similarity results. Hence, for *DS*1, Lian method and Cohesion method yield similar results.

In Fig. 6.5a, GDPA methods become slightly worse when *PROB* increases. Since *DS*1 is well-clustered, most queries only require documents in the same clusters. Thus *PROB* has less effect on AAT. In Fig. 6.5b, when *QIR* increases, J/D method becomes slightly better. This indicates that J/D method can achieve better scalability than other methods when accessing well-clustered data. Fig. 6.5c shows that all GDPA methods remain stable as MQD increases. It is interesting to note that for RDPA, AAT always remains stable.

Fig. 6.6 shows the results on *DS*2. From the figure we can see that all GDPA methods achieve better performance when compared with RDPA. Specifically, Cohesion method achieves the best results while J/D method achieves the worst results among GDPA methods. This indicates that Cohesion method better fits miscellaneous data. In Fig. 6.6a, both GDPA methods and RDPA become worse when *PROB* increases. It is clear that *PROB* has more effect on AAT for miscellaneous data. In Fig. 6.6b, when *QIR* increases from 0.1 to 0.5, GDPA methods J/D and Lian together with RDPA perform worse while Cohesion method still performs better. The reason for this is that the number of queries that can take advantage of the clustering results fluctuates when *QIR* is small. After that, when *QIR* increases, all methods performs slightly better. This shows that Cohesion method can achieve best scalability when accessing miscellaneous data. Fig. 6.6c shows that all methods achieve better AAT as MQD increases since selectivity of queries drops with the increase of MQD.

Fig. 6.7 shows the results on *DS*3. Similarly, all GDPA methods achieve better performance when compared with RDPA. Specifically, Lian method achieves the best results while J/D method provides the worst results among GDPA methods. This shows that Lian method better fits hybrid data. However, Cohesion method achieves very similar performance to Lian method. In Fig. 6.7a, both GDPA methods and RDPA become worse when *PROB* increases. *PROB* has more effect on *AAT* for hybrid data. In Fig. 6.7b, when *QIR* increases, all GDPA methods become slightly better and still Lian method provides the best results. Figure 6.7c shows that all methods achieve better *AAT* as *MQD* increases since selectivity of queries drops with the increase of *MQD*.

We also conducted similar experiments using larger data sets (including 500 and 1,000 XML documents respectively). The results of experiments are shown in Fig. 6.8 and Fig. 6.9. From these figures, we can see the similar trends which are observed in Fig. 6.6.

Therefore, from the above experiments, we can see that GDPA methods always achieve better *AAT* when compared with RDPA. When accessing well-clustered data, J/D method achieves the best performance. When accessing miscellaneous data, Cohesion method provides better performance in most cases. This is because the J/D metric emphasizes on the common paths between two XML documents according to their definitions (the Jaccard and Dice similarity definitions). This also means, the J/D metric is more suitable for well-clustered data sets. In contrast, the Cohesion metric emphasizes both the common paths (representing the probability that two documents are matched against a given query at the same time) and the difference in the path sets of two documents (representing the probability that two documents are not matched against the same query at the same time). Therefore, the Cohesion metric is more suitable for miscellaneous data sets. Finally when accessing hybrid data, Lian method shows better performance in most cases since Lian method emphasizes on the difference between XML documents but not on common paths.

Table 6.6 Query Selectivity and Document Coverage Rate

Data Set	Query Selectivity	Coverage Rate
*DS*1	44.8%	100%
*DS*2	33.6%	100%
*DS*3	35.6%	100%
*DS*4	32.6%	100%
*DS*5	33.2%	100%

We also investigate query selectivity and document coverage rate in our experiments. Here, query selectivity refers to the average proportion of documents matched with a user query and document coverage rate refers to the proportion of documents in the entire document set on the server required by at least one user query. The results are obtained using all workload parameters at default values. As can be seen from Table 6.6, the query selectivity is ranging from around 32% to 45%. The main reason for this is that the probability of * and // is 10% by default. Further, for all data sets. the document coverage rate is 100%, which is mainly due to the same reason of query selectivity. This shows that all documents on the server are covered in all our experiments.

Finally, we study the maintenance cost of our data placement algorithms. The maintenance cost is measured when adding a new document to the server or removing an existing document from the server. The maintenance results are shown in Table 6.7. From the table we can see that, when adding a new document in a document set with 250 documents on the server, the time cost to calculate the similarity between the new document and all the existing documents (shown as “Similarity Time" in the table) ranges from 209 milliseconds to 238 milliseconds on average for the Dice, Jaccard and Lian metrics. For the Cohesion metric, it takes 378 milliseconds on average, which is longer due to fact that the Cohesion metric requires more set operations that the other three metrics. Meanwhile, the time cost to readjust the data placement (shown as “Placement Time" in the table) is quite similar among all four metrics,

Table 6.7 Data Placement Update Time (in milliseconds)

No. of Doc.	Metric	Add		Remove
		Similarity Time	Placement Time	Placement Time
250	Dice	228	168	165
	Jaccard	209	169	174
	Lian	238	174	171
	Cohesion	378	181	185
500	Dice	343	745	751
	Jaccard	359	814	811
	Lian	419	844	837
	Cohesion	763	811	801
1000	Dice	781	1212	1233
	Jaccard	640	1152	1187
	Lian	691	1192	1202
	Cohesion	1130	1298	1275

which is ranging from 168 milliseconds to 181 milliseconds. A similar pattern is observed when adding a new document to a larger document set.

On the other hand, when removing an existing document from the server, only Placement Time will be incurred. From the table we can see that, similarly, the Placement Time costs for four metrics are quite similar to each other. For example, when removing an existing document from a document set with 250 documents, the Placement Time is ranging from 165 milliseconds to 185 milliseconds. Again, a similar pattern is observed when removing an existing document from a larger document set.

6.6 Related Work

Many studies have been claimed out to investigate data placement techniques to reduce access time [152–154]. These studies generally assume that each user query requires one data item only. Other studies handle data placement problems for queries that may require multiple data items.

Multi-item data placement problem is related to the data placement problem of XML data which is the focus of our work. It has been proven to be an NP-Complete problem [147]. A data placement method for multi-item queries called QEM is introduced in [155], which opened up a new perspective in this field. In addition, several improved methods are proposed [156, 138]. The above work is all within the scope of periodic broadcast and generally assumes that the clients' queries are already known in advance. However, in some applications, the user demands may be either unknown or costly to collect the related information due to the mobility of mobile users and privacy concerns.

Multi-item data placement problem in the on-demand broadcast mode has also attracted lots of interest [134, 157]. These approaches are in pure on-demand broadcast mode and strictly require that mobile clients submit their queries to the server for desired data. Otherwise, the server will not broadcast related data on air. This is because the server filters and schedules data based solely on submitted queries. However, frequent use of uplink channel leads to high communication cost via uplink channel, which can shorten battery life of mobile clients dramatically.

The above mentioned studies focus on flat data broadcasts, in which indexes of data items are generally key-based and data do not contain structural information. Recently, besides the traditional flat data broadcast, a wealth of work dealing with XML data broadcast has appeared. Some work addresses the performance optimization of query processing of XML streams in wireless broadcast [127, 158], while other work designs indexing techniques for XML data broadcast based on existing XML indexing techniques [128, 129]. However, their work mainly focuses on air indexing techniques similar to content based indexing techniques in XML stream processing or XPath query based indexing techniques. Moreover, this kind of work does not study the data placement problem for XML data broadcast.

Data placement problem for XML data broadcast is investigated in [159]. In that work, the broadcast schedules are generated based on clustering results of XML

data on the server. However, the clustering process requires manually specifying the number of clusters and has to compare different clustering results based on clients' query distribution in order to find the optimal clustering result, which differs from our work.

6.7 Summary

In this chapter, we have studied the data placement problem of periodic XML data broadcast. Taking advantage of the structured characteristics of XML data, we are able to generate effective broadcast programs based only on XML data on the server. This not only distinguishes our work from previous studies, but also enables it to have broader applicability. We have performed a theoretical analysis of the problem and discussed structural sharing in XML data which forms the basis of our novel greedy data placement algorithm (GDPA). Our experiments demonstrated that the proposed algorithm could improve access efficiency and achieve better scalability.

Chapter 7

Emerging IoT Applications and Open Issues

7.1 Emerging IoT Applications

As pointed out by [1] that IoT "has the potential to change the world, just as the Internet did". The ongoing and/or emerging IoT applications show that IoT can bring significant changes in many domains, i.e., cities and homes, environment monitoring, health, energy, and business, etc. IoT can bring the ability to react to events in the physical world in an automatic, rapid and informed manner. This also opens up new opportunities for dealing with complex or critical situations and enables a wide variety of business processes to be optimized. In this section, we overview several representative domains where IoT can make some profound changes.

7.1.1 Smart Cities and Homes

IoT can connect billions of smart things and can help capture information in cities. Based on IoT, cities could become smarter and more efficient. Below are some examples of promising IoT applications in future smart cities. In a modern city, lots

of digital data traces are generated there every second via cameras and sensors of all kinds [160]. All this data can be exploited to provide better services to city inhabitants, if these inhabitants would be able to take advantage of it in an efficient and effective way. For example, IoT can facilitate resources management issues for modern cities. Specifically, static resources (e.g., fire stations, parking spots) and mobile resources (e.g., police cars, fire trucks) in a city can be managed effectively using IoT technologies. Whenever events (fires, crime reports, cars looking for parking) arise, IoT technologies would be able to quickly match resources with events in an optimal way based on the information captured by smart things, thereby reducing cost and saving time. Taxi drivers in the city would also be able to better serve prospective passengers by learning passenger's mobility patterns and other taxi drivers' serving behaviors through the help of IoT technologies [161]. One study estimated a loss of $78 billion in 2007 in the form of 4.2 billion lost hours and 2.9 billion gallons of wasted gasoline in the United States alone [100]. IoT has the potential to bring fundamental changes in urban street-parking management, which would greatly benefit the whole society by reducing traffic congestion and fuel consumption.

With IoT technologies, people can browse and manage their homes via the Web. For example, they would be able to check whether the light in their bedrooms is on and could turn it off by simply clicking a button on a Web page. Similar operations and management could be done in office environments. Plumbing is ranked as one of the ten most frequently found problems in homes [162]. It is important to determine the spatial topology of hidden water pipelines behind walls and underground. In IoT, smart things in homes would be able to report plumbing problems automatically and report to owners and/or plumbers for efficient maintenance and repair.

7.1.2 Environment Monitoring

IoT technologies can also help to monitor and protect environments thereby improving human's knowledge about environments. Take water as an example. Understanding the dynamics of bodies of water and their impact on the global environment requires sensing information over the full volume of water. In such context, IoT technologies would be able to provide effective approaches to study water. Also IoT could improve water management in a city. Drinking water is becoming a scarce resource around the world. In big cities, efficiently distributing water is one of the major issues [160]. Various reports show that on average 30% of drinkable water is lost during transmission due to the aging infrastructure and pipe failures. Further, water can be contaminated biologically or chemically due to inefficient operation and management. In order to effectively manage and efficiently transport water, IoT technologies would be of great importance.

Soil contains vast ecosystems that play a key role in the Earth's water and nutrient cycles, but scientists cannot currently collect the high-resolution data required to fully understand them. Many soil sensors are inherently fragile and often produce invalid or uncalibrated data [163]. IoT technologies would help to validate, calibrate, repair, or replace sensors, allowing to use available sensors without sacrificing data integrity and meanwhile minimizing the human resources required.

Sound is another example where IoT technologies can help. Sound is multidimensional, varying in intensity and spectra. So it is difficult to quantify, e.g., it is difficult to determine what kind of sound is noise. Further, the definitions and feelings of noise are quite subjective. For example, some noises could be pleasant, like flowing water, while others can be annoying, such as car alarms, screeching breaks and people arguing. A device has been designed and built to monitor residential noise pollution to address the above problems [164]. Firstly, noise samples from three representative houses are used, which span the spectrum of quiet to noisy neighborhoods. Secondly,

a noise model is developed to characterize residential noise. Thirdly, noise events of an entire day (24 hours) are compressed into a one minute auditory summary. Data collection, transmission and storage requirements can be minimized in order to utilize low-cost and low-power components, while sufficient measurement accuracy is still maintained.

Intel has developed smart sensors that can warn people about running outside when the air is polluted[1]. For example, if someone is preparing to take a jog along his/her regular route, an application on his/her smartphone pushes out a message: air pollution levels are high in the park where he/she usually runs. Then he/she could try a recommended route that is cleaner. Currently, many cities already have pollution and weather sensors. They are usually located on top of buildings, far from daily human activities.

7.1.3 Health

In future IoT environments, an RFID-enabled information infrastructure would be likely to revolutionize areas such as healthcare, and pharmaceutical. For example, a healthcare environment such as a large hospital or aged care could tag all pieces of medical equipment (e.g., scalpels, thermometers) and drug products for inventory management. Each storage area or patient room would be equipped with RFID readers that could scan medical devices, drug products, and their associated cases. Such an RFID-based infrastructure could offer a hospital unprecedented near real-time ability to track and monitor objects and detect anomalies (e.g., misplaced objects) as they occur. For example, it is suggested that a simple surgery checklist can save lives[2].

As personal health sensors become ubiquitous, they are expected to become interoperable. This means standardized sensors can wirelessly communicate their data to

[1]http://www.fastcoexist.com/1680111/intels-sensors-will-warn-you-about-running-outside-when-the-air-is-polluted

[2]http://content.time.com/time/health/article/0,8599,1871759,00.html

a device many people already carry today (e.g., mobile phones). It is argued by [165] that one challenge in weight control is the difficulty of tracking food calories consumed and calories expended by activity. Then they present a system for automatic monitoring of calories consumed using a single body-worn accelerometer. To be fully benefited from such data for a large body of people, applying IoT technologies in such area would be a promising direction.

Mobile technology and sensors are creating ways to inexpensively and continuously monitor people's health. Doctors may call their clients to schedule an appointment,rather than vice-versa, because the doctors could know their clients' health conditions in real-time. Some projects for such purpose have been initiated. For example, EveryHeartBeat[3] is a project for Body Computing to "connect the more than 5 billion mobile phones in the world to the health ecosystem". In the initial stage, heart rate monitoring is investigated. Consumers would be able to self track their pulse and studies show heart rate monitoring could be useful in detecting heart conditions and enabling early diagnosis. The future goal is to include data on blood sugar levels, and other biometrics collected via mobile devices.

7.1.4 Energy

Home heating is a major factor in worldwide energy use. In IoT, home energy management applications could be built upon embedded Web servers [166]. Through such online web services, people can track and manage their home energy consumption. A system is designed by [167] for augmenting these thermostats using just-in-time heating and cooling based on travel-to-home distance obtained from location-aware mobile phones. The system makes use of a GPS-enabled thermostat which could lead to savings of as much as 7%. In IoT, as things in homes would become smart and connected to the Internet, similar energy savings could be more effective. For

[3] http://join.everyheartbeat.org/

example, by automatically sensing occupancy and sleep patterns in a home, it would be possible to save energy by automatically turning off the home's HVAC (heating, ventilation, and air conditioning) system.

Beside home heating, fuel consumption is also an important issue. GreenGPS, a navigation service that uses participatory sensing data to map fuel consumption on city streets, has been designed by [168]. GreenGPS would allow drivers to find the most fuel-efficient routes for their vehicles between arbitrary end-points. In IoT, fuel consumption would be further reduced by enabling cars and passengers to communicate with each other for ride sharing [161].

7.1.5 Business

IoT technologies would be able to help to improve efficiency in business and bring other impacts on business [169]:

- From a commercial point of view, IoT can help increase the efficiency of business processes and reduce costs in warehouse logistics and in service industries. This is because more complete and necessary information can be collected by interconnected things. owing to its huge and profound impact on the society, IoT research and applications can also trigger new business models involving smart things and associated services.

- From a social and political point of view, IoT technologies can provide a general increase in the quality of life for the following reasons. Firstly, consumers and citizens will be able to obtain more comprehensive information. Secondly, care for aged and/or disabled people can be improved with smarter assistance systems. Thirdly, safety can be increased. For example, road safety can be improved by receiving more complete and real-time traffic and road condition information.

- From a personal point of view, new services enabled by IoT technologies can make life more pleasant, entertaining, independent and also safer. For example, business taking advantages of technologies of search of things in IoT can help locate lost things quickly, such as personal belongs, pets or even other people.

Besides, take improving information handover efficiency in a global supply chain as an example. The concept of *digital object memories* (DOM) is proposed in [170], which can store order-related data via smart labels on the item. Based on DOM, relevant life cycle information could be attached to the product itself. Considering the potential different stakeholders including manufacturer, distributor, retailer, and end customer along the supply/value chain, this approach facilitates information handover.

Further, there are many important bits of information in an IoT-based supply chain, such as the 5W (what, when, where, who, which). It is also necessary to integrate them efficiently and in real-time in other operations. The EPCIS (Electronic Product Code Information System) network is a set of tools and standards for tracking and sharing RFID-tagged products in IoT. However, much of this data remains in closed networks and is hard to integrate [171]. IoT technologies could be used to make it easier to use all this data, to integrate it into various applications, and to build more flexible, scalable, global application for better (even real-time) logistics.

7.2 Open Issues

The development of IoT technologies and applications is merely beginning. Many new challenges and issues have not been addressed, which require substantial efforts from both academia and industry. In this section, we identify some key directions for future research and development from a data-centric perspective.

- *Data Quality and Uncertainty:* In IoT, as data volume increases, inconsistency and redundancy within data would become paramount issues. One of the central problems for data quality is *inconsistency detection* and when data is distributed, the detection would be far more challenging [172]. This is because inconsistency detection often requires shipping data from one site to another. Meanwhile, inherited from RFID data and sensor data, IoT data would be of great uncertainty, which also presents significant challenges.

- *Co-Space Data:* In an IoT environment, the physical space and the virtual (data) space co-exist, and interact simultaneously. Novel technologies must be developed to allow data to be processed and manipulated seamlessly between the real and digital spaces [173]. To synchronize data in both real and virtual worlds, large amount of data and information will flow between co-spaces, which pose new challenges. For example, it would be challenging to process heterogeneous data streams in order to model and simulate real world events in the virtual world. Besides, more intelligent processing is needed to identify and send interesting events in the co-space to objects in the physical world.

- *Transaction Handling:* When the data being updated is spread across hundreds or thousands of networked computers/smart things with differing update policies, it would be difficult to define what the transaction is. In addition, most of things are resource-constrained, which are typically connected to the Internet using light-weight, *stateless* protocols such as CoAP (Constrained Application Protocol)[4] and 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks)[5] and accessed using RESTful Web services. This makes transaction handling in IoT a great challenge. As pointed out by [2] that the problem is that the world is changing fast, the data representing the world is on multiple

[4]http://tools.ietf.org/html/draft-ietf-core-coap-18
[5]http://tools.ietf.org/wg/6lowpan

networked computers/smart things and existing database technologies cannot manage. Techniques developed for streamed and real-time data may provide some hints.

- *Frequently Updated Timestamped Structured (FUTS) Data:* The Internet, and hence IoT, contains potentially billions of Frequently Updated Timestamped Structured (FUTS) data sources, such as real-time traffic reports, air pollution detection, temperature monitoring, crops monitoring, etc. FUTS data sources contain states and updates of physical world things. Current technologies are not capable in dealing with FUTS data sources [2] because: (i) no data management system can easily display FUTS past data; (ii) no efficient crawler or storage engine is able to collect and store FUTS data; and (iii) querying and delivering FUTS data is hardly supported. All these pose great challenges for the design of novel data management systems for FUTS data.

- *Distributed and Mobile Data:* In IoT, data will be increasingly distributed and mobile. Different from traditional mobile data, distributed and mobile data in IoT would be much more highly distributed and data intensive. In the context of interconnecting huge numbers of mobile and smart objects, centralized data stores would not be a suitable tool to manage all the dynamics of mobile data produced in IoT. Thus there is a need for novel ways to manage distributed and mobile data efficiently and effectively in IoT.

- *Semantic Enrichment and Semantic Event Processing:* The full potentials of IoT would heavily rely on the progress of the semantic Web. This is because things and machines should play a much more important role than humans in IoT to process and understand data. This calls for new research in Semantic technologies. For example, there are increasing efforts in building public knowledge bases (such as DBpedia, FreeBase, Linked Open Data Cloud, etc.).

But how these knowledge bases can be effectively used to add to the understanding of raw data coming from sensor data streams and other types of data streams? To resolve this challenge, semantic enrichment of sensing data is a promising research direction. Further, consider the potential excessively large amount of subscriptions to IoT data. To produce proper semantic enrichment to meet different enrichment needs from different subscribers poses great challenges. Finally, how to effectively incorporate semantic enrichment techniques with semantic event processing to provide much better expressiveness in event processing is still at its initial stage. This will also demand a large amount of research effort.

- *Mining:* Data mining aims to facilitate the exploration and analysis of large amounts of data, which can help to extract useful information for huge volume of IoT data. Data mining challenges may include extraction of temporal characteristics from sensor data streams, event detection from multiple data streams, data stream classification, activity discovery and recognition from sensor data streams. Besides, clustering and table summarization in large data sets, mining large (data, information or social) networks, sampling, and information extraction from the Web are also great challenges in IoT.

- *Knowledge Discovery:* Knowledge discovery is the process of extracting useful knowledge from data. This is essential especially when connected things populate their data to the Web. The following issues related to knowledge discovery in IoT have been identified by [174]: (i) automatic extraction of relational facts from natural-language text and multi-modal contexts; (ii) large-scale gathering of factual-knowledge candidates and their reconciliation into comprehensive knowledge bases; (iii) reasoning on uncertain hypotheses, for knowledge dis-

covery and semantic search; and (iv) deep and real-time question answering, e.g., to enable computers to win quiz game shows[6].

- *Security:* Due to the proliferation of embedded devices in IoT, effective device security mechanisms are essential to the development of IoT technologies and applications. The National Intelligence Council [6] argues that, to the extent that everyday objects become information security risks, the IoT could distribute those risks far more widely than the Internet has to date. For example, RFID security presents many challenges. Potential solutions should consider aspects from hardware and wireless protocol security to the management, regulation and sharing of collected RFID data [175]. Besides, it is argued by [176] that there is still no generic framework for deploying and extending traditional security mechanisms over a variety of pervasive systems. Regarding security concerns of the network layer, it is suggested by [177] that the Internet can be gradually encrypted and authenticated based on the observations that the recent advances in implementation of cryptographic algorithms have made general purpose processors capable of encrypting packets at high rates. But how to generalize such algorithms to IoT would be challenging as things in IoT normally only maintain low transmission rates and connections are usually intermittent.

- *Privacy:* Privacy protection is a serious challenge in IoT. One of the fundamental problems is the lack of a mechanism to help people expose appropriate amounts of their identity information. Embedded sensing is becoming more and more prevalent on personal devices such as mobile phones and multi-media players. Since people are typically wearing and carrying devices capable of sensing, details such as activity, location, and environment could become available to other people. Hence, personal sensing can be used to detect their physical activities and bring about privacy concerns [178].

[6] http://www.ibm.com/smarterplanet/us/en/ibmwatson/

- *Social Concerns:* Since IoT connects everyday objects to the Internet, social concerns would become a hot topic in the development of IoT. Further, online social networks with personal things information may incur social concerns as well, such as disclosures of personal activities and hobbies, etc. Appropriate economic and legal conditions and a social consensus on how the new technical opportunities in IoT should be used also represents a substantial task for the future [169].

Chapter 8

Conclusions and Future Work

8.1 Conclusions

It is widely predicted that the next generation of the Internet will be comprised of trillions of connected computing nodes at a global scale. Through these nodes, everyday objects in the world can be identified, connected to the Internet and take decisions independently. In this context, Internet of Things (IoT) is considered a new revolution of the Internet. In IoT, the possibility of seamlessly merging the real and the virtual worlds, through the massive deployment of embedded devices, opens up many new and exciting directions for both research and development. In this article, we have provided an overview of some key research areas of IoT, specifically from a data-centric perspective. It also presents a number of fundamental issues to be resolved before we can fully realize the promise of IoT applications.

This thesis has reviewed the state-of-the-art research efforts in IoT in Chapter 2 from data-centric perspectives, including data stream processing, data storage models, complex event processing, and searching in IoT by identifying an IoT data taxonomy, which includes ten key data elements of IoT data under three categorizations. This thesis has focused on three aspects of data management in IoT, including data dynamics, data velocity, and data incompleteness. More specifically, we study data dynamics in

dynamic graphs, handle data velocity in streams, and tackle data incompleteness via sharing.

Chapter 3 has proposed SIEF for computing the shortest path distance in graphs subject to edge failures. Extensive experiments have also been performed using six real-world graphs to confirm its effectiveness and efficiency. SIEF is able to support compact index construction for all single-edge failure cases in graphs efficiently. Specifically, the SIEF index size is comparable to that of the indexes constructed for original static graphs, which is very compact. SIEF can answer distance queries with single-edge failure constraints several orders of magnitude faster than traditional BFS algorithms. Moreover, SIEF can answer distance queries on multi-edge failure with high accuracy and fast response time.

Chapter 4 has proposed two new data structures, namely TP-automata and CTP-automata, to support efficient pattern matching over Linked Data streams. Firstly, in order to efficiently match a large number of user queries that contain only single triple patterns against Linked Data streams, TP-automata has been designed. The experiments have shown that TP-automata can disseminate Linked Data 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. Secondly, in order to efficiently match a large number of conjunctive triple pattern queries against Linked Data streams in batch mode, similarly, CTP-automata has also been designed. CTP-automata has been experimentally demonstrated that it can disseminate Linked Data an order of magnitude faster than the existing approaches.

Chapter 5 has proposed an effective and efficient air indexing method for broadcasting Linked Data on air, which can be used in data sharing among a large number of mobile and smart objects in the era of IoT. The proposed method is based on 3D

Hilbert curve mappings. RDF triples are mapped into points in a 3D space and then adopt 3D Hilbert curve mappings to convert all the 3D points into one-dimensional points. An efficient search algorithm has also been devised to facilitate query processing over the Linked Data on air. Experiments have been conducted and the proposed method has shown better performance over the traditional R-tree based method in various aspects, including access latency, tuning time, and index size.

Chapter 6 has studied the data placement problem of periodic XML data broadcast. Taking advantage of the structured characteristics of XML data, it is feasible to generate effective broadcast programs based only on XML data on the server. A detailed theoretical analysis of the problem has been provided and structural sharing in XML data has also been discussed, which forms the basis of the proposed GDPA algorithm. The experiments have demonstrated that the proposed algorithm could improve access efficiency and achieve better scalability.

This thesis has also discussed on-going and emerging IoT applications, and open research issues for processing and managing IoT data in Chapter 7. Several representative domains where IoT can make profound changes are explored, and some key directions for future research and development from a data-centric perspective have been identified.

8.2 Directions for Future Work

In Chapter 7, we have already identified a number of open issues in IoT research and development (see Section 7.2). In the following, we further identify some possible research directions for future research in data management in IoT. These directions are basically extensions or further work on top of the research presented in this thesis.

Query Processing in a Graph with Edge Failures

There are several aspects in handling distance queries in graphs with edge failures. The first aspect centers on how to support exact distance queries with more complex edge failure constraints, i.e., dual-failure or triple-failure on edges. The second aspect is to further speed up the index construction process in order to process larger graphs. In addition, it is also interesting to investigate the problem of answering distance queries in graphs with node failures, which is even more challenging than edge failures.

Apart from answering simple point-to-point distance queries in graphs with edge failures, more complex queries based on point-to-point distance can also be investigated. For example, betweenness centrality query processing in graphs with edge failures is also an interesting research direction. As a node with high betweenness centrality has a large influence on the transfer of items through the network, fast betweenness computation in dynamic networks can help to efficiently identify the most influenced nodes in a network.

Linked Data Streams Processing in the Context of IoT

In future, it is imperative to support efficient matching for larger scales of user queries over Linked Data streams, specially user queries containing conjunctive triple patterns, which would be a critical issue in the emerging Internet of Things. In addition, it is also important to investigate how to support *semantic matching* over Linked Data streams to further cater heterogeneous data consumption needs from a rich variety of data consumers in IoT. *Semantic matching* in a hashing space with the use of Locality Sensitive Hashing (LSH) techniques [179] could be an effective way to help to map semantically related data together, thereby enabling matching user queries and Linked Data streams approximately. Both directions will enable the Linked Data dissemination system to provide better semantics richness and to support data con-

sumption needs more efficiently and more accurately, which we believe would be a critical issue in the emerging IoT.

Broadcasting Linked Data in IoT Accurately and Efficiently

The work on broadcasting Linked Data on-air using 3D-Hilbert curves presented in this thesis can be extended to support join queries, where data consumption needs can be expressed more accurately. It is also important to investigate the scalability of the broadcast system in terms of real-time construction of air indexes for broadcasting Linked Data streams with high stream rates and highly dynamic mobile users, which is a challenging issue in the context of IoT.

Broadcasting XML Data in IoT

In future, it is interesting to further improve the performance of our periodic XML data broadcast system by investigating the insights of structural sharing among XML documents. For example, it might be useful to consider details on how to measure structural sharing distribution in an XML document set, how the distribution affects the expected access time of queries and how to choose a similarity metric based on structural sharing distribution in a set of XML documents, etc. This basically targets at improving the system by using different clustering strategies.

More directions of future work may include: (1) It might also be possible to show that GDPA is better than random theoretically by estimating probability of the next item in the program being relevant. (2) We can also generalize our broadcasting scenario to the case that each mobile client has a set of very different queries, i.e., in the sense that answers are very different. It is interesting to see how the system can efficiently handle queries from the same user efficiently. (3) It is also interesting to investigate how to generate a broadcast program by considering both similarity and popularity of broadcasting items.

References

[1] Kevin Ashton. That 'Internet of Things' Thing. http://www.rfidjournal.com/article/view/4986, 2009.

[2] Anne E. James, Joshua Cooper, Keith G. Jeffery, and Gunter Saake. 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). In *Proceedings of the 26th British National Conference on Databases (BNCOD)*, pages 225–233, Birmingham, UK, 2009. Springer.

[3] Bin Guo, Daqing Zhang, Zhu Wang, Zhiwen Yu, and Xingshe Zhou. Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things. *J. Network and Computer Applications*, 36(6):1531–1539, 2013.

[4] Elizabeth Biddlecombe. UN predicts 'Internet of Things'. http://news.bbc.co.uk/2/hi/technology/4440334.stm, 2005.

[5] Julie Bort. 10 technologies that will change the world in the next 10 years. http://www.networkworld.com/news/2011/071511-cisco-futurist.html, 2011.

[6] Anonymous. National Intelligence Council (NIC), Disruptive Civil Technologies: Six Technologies with Potential Impacts on US Interests Out to 2025, Conference Report CR 2008-07, April 2008, http://www.dni.gov/nic/NIC_home.html, 2008.

[7] CASAGRAS. CASAGRAS (Coordination And Support Action for Global RFID-related Activities and Standardisation)., 2000.

[8] ITU. International Telecommunication Union (ITU) Internet Reports, The Internet of Things, November 2005., 2005.

[9] INFSO. INFSO D.4 Networked Enterprise & RFID INFSO G.2 Micro & Nanosystems, in: Co-operation with the Working Group RFID of the ETP EPOSS, Internet of Things in 2020, Roadmap for the Future, Version 1.1, 27 May 2008., 2008.

[10] European Commission. European Commission: Internet of Things, An action plan for Europe. http://europa.eu/legislation_summaries/information_society/internet/si0009_en.htm, 2009.

[11] Jonathan de Andrade Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André Carlos Ponce Leon Ferreira de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13, 2013.

[12] Sharmila Subramaniam and Dimitrios Gunopulos. A Survey of Stream Processing Problems and Techniques in Sensor Networks. In *Data Streams - Models and Algorithms*, pages 333–352. Springer, 2007.

[13] Brian Babcock and Chris Olston. Distributed Top-K Monitoring. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD Conference)*, pages 28–39, 2003.

[14] João Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press, 2010.

[15] Feng Wang and Jiangchuan Liu. Networked Wireless Sensor Data Collection: Issues, Challenges, and Approaches. *IEEE Communications Surveys and Tutorials*, 13(4):673–687, 2011.

[16] Edwin M. Knorr and Raymond T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proceedings of 24rd International Conference on Very Large Data Bases (VLDB)*, pages 392–403, 1998.

[17] Noboru Koshizuka and Ken Sakamura. Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things. *IEEE Pervasive Computing*, 9(4):98–101, 2010.

[18] Q. Z. Sheng, X. Li, and S. Zeadally. Enabling Next-Generation RFID Applications: Solutions and Challenges. *IEEE Computer*, 41(9):21–28, September 2008.

[19] George Roussos. *Networked RFID - Systems, Software and Services*, pages 1–168. Springer, 2008.

[20] Shawn R. Jeffery, Minos N. Garofalakis, and Michael J. Franklin. Adaptive Cleaning for RFID Data Streams. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, pages 163–174, 2006.

[21] Guoqiong Liao, Jing Li, Lei Chen, and Changxuan Wan. KLEAP: an efficient cleaning method to remove cross-reads in RFID streams. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM)*, pages 2209–2212, 2011.

[22] Bettina Fazzinga, Sergio Flesca, Filippo Furfaro, and Francesco Parisi. Cleaning trajectory data of RFID-monitored objects through conditioning under integrity constraints. In *Proc. 17th International Conference on Extending Database Technology (EDBT)*, pages 379–390, 2014.

[23] Thanh T. L. Tran, Charles A. Sutton, Richard Cocci, Yanming Nie, Yanlei Diao, and Prashant J. Shenoy. Probabilistic Inference over RFID Streams in Mobile Environments. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 1096–1107, 2009.

[24] Zhao Cao, Charles A. Sutton, Yanlei Diao, and Prashant J. Shenoy. Distributed inference and query processing for RFID tracking and monitoring. *Proceedings of the VLDB Endowment*, 4(5):326–337, 2011.

[25] Yanming Nie, Richard Cocci, Zhao Cao, Yanlei Diao, and Prashant J. Shenoy. SPIRE: Efficient Data Inference and Compression over RFID Streams. *IEEE Trans. Knowl. Data Eng.*, 24(1):141–155, 2012.

[26] Ying Zhang, Minh-Duc Pham, Óscar Corcho, and Jean-Paul Calbimonte. SR-Bench: A Streaming RDF/SPARQL Benchmark. In *Proceedings of the 11th International Semantic Web Conference (ISWC)*, pages 641–657, 2012.

[27] Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming SPARQL - Extending SPARQL to Process Data Streams. In *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications (ESWC)*, pages 448–462, 2008.

[28] Jean-Paul Calbimonte, Óscar Corcho, and Alasdair J. G. Gray. Enabling Ontology-Based Access to Streaming Data Sources. In *Proceedings of the 9th International Semantic Web Conference (ISWC)*, pages 96–111, 2010.

[29] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, and Michael Grossniklaus. An execution environment for C-SPARQL queries. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 441–452, 2010.

[30] Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. EP-SPARQL: a unified language for event processing and stream reasoning. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 635–644, 2011.

[31] Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In *Proceedings of the 10th International Semantic Web Conference (ISWC)*, pages 370–388, 2011.

[32] Daniel Gerber, Sebastian Hellmann, Lorenz Bühmann, Tommaso Soru, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. Real-Time RDF Extraction from Unstructured Data Streams. In *Proceedings of the 12th International Semantic Web Conference (ISWC)*, pages 135–150, 2013.

[33] Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, and Stanley B. Zdonik. C-Store: A Column-oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 553–564, Trondheim, Norway, 2005. ACM.

[34] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland. The End of an Architectural Era (It's Time for a Complete Rewrite). In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 1150–1160, University of Vienna, Austria, 2007. ACM.

[35] Sudarshan Kadambi, Jianjun Chen, Brian F. Cooper, David Lomax, Raghu Ramakrishnan, Adam Silberstein, Erwin Tam, and Hector Garcia-Molina. Where in the World is My Data? *Proceedings of the VLDB Endowment*, 4(11):1040–1050, 2011.

[36] Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *MONET*, 19(2):171–209, 2014.

[37] Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[38] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2), 2008.

[39] Apache. Apache HBase Project, https://hbase.apache.org/, 2014.

[40] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP)*, pages 205–220, 2007.

[41] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *Operating Systems Review*, 44(2):35–40, 2010.

[42] Luca Mottola. Programming Storage-Centric Sensor Networks with Squirrel. In *Proceedings of the 9th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12, Stockholm, Sweden, 2010. IEEE.

[43] Yong Yang, Lili Wang, Dong Kun Noh, Hieu Khac Le, and Tarek F. Abdelzaher. SolarStore: Enhancing Data Reliability in Solar-Powered Storage-Centric Sensor Networks. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 333–346, Kraków, Poland, 2009. ACM.

[44] Nicolas Tsiftes and Adam Dunkels. A Database in Every Sensor. In *Proceedings of the 9th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 316–332, Seattle, WA, USA, 2011. ACM.

[45] Suman Nath. Energy Efficient Sensor Data Logging with Amnesic Flash Storage. In *Proceedings of the 8th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 157–168, San Francisco, California, USA, 2009. IEEE.

[46] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 851–860, Raleigh, North Carolina, USA, 2010. ACM.

[47] Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. Time is of the Essence: Improving Recency Ranking Using Twitter Data. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 331–340, Raleigh, North Carolina, USA, 2010. ACM.

[48] George Tsatsanifos, Dimitris Sacharidis, and Timos K. Sellis. On Enhancing Scalability for Distributed RDF/S Stores. In *Proceedings of the 14th International Conference on Extending Database Technology (EDBT)*, pages 141–152, Uppsala, Sweden, 2011. ACM.

[49] Sanjay Agrawal, Kaushik Chakrabarti, Surajit Chaudhuri, Venkatesh Ganti, Arnd Christian König, and Dong Xin. Exploiting Web Search Engines to Search Structured Databases. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 501–510, Madrid, Spain, 2009. ACM.

[50] Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. Exploiting Web Search to Generate Synonyms for Entities. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 151–160, Madrid, Spain, 2009. ACM.

[51] Haodong Wang, Chiu Chiang Tan, and Qun Li. Snoogle: A Search Engine for Pervasive Environments. *IEEE Trans. Parallel Distrib. Syst.*, 21(8):1188–1202, 2010.

[52] Kok-Kiong Yap, Vikram Srinivasan, and Mehul Motani. MAX: Wide Area Human-Centric Search of the Physical World. *ACM Trans. on Sensor Networks*, 4(4):34, 2008.

[53] Chiu Chiang Tan, Bo Sheng, Haodong Wang, and Qun Li. Microsearch: A Search Engine for Embedded Devices Used in Pervasive Computing. *ACM Trans. Embedded Comput. Syst.*, 9(4):29, 2010.

[54] B. Maryam Elahi, Kay Römer, Benedikt Ostermaier, Michael Fahrmair, and Wolfgang Kellerer. Sensor Ranking: A Primitive for Efficient Content-based Sensor Search. In *Proceedings of the 8th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 217–228, San Francisco, California, USA, 2009. IEEE.

[55] Christian Frank, Philipp Bolliger, Friedemann Mattern, and Wolfgang Kellerer. The Sensor Internet at Work: Locating Everyday Items Using Mobile Phones. *Pervasive and Mobile Computing*, 4(3):421–447, 2008.

[56] Chun Chen, Feng Li, Beng Chin Ooi, and Sai Wu. TI: An Efficient Indexing Mechanism for Real-Time Search on Tweets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 649–660, Athens, Greece, 2011. ACM.

[57] Alexander Artikis and Georgios Paliouras. Tutorial: Formal Methods for Event Processing. In *Proc. 17th International Conference on Extending Database Technology (EDBT)*, page 675, 2014.

[58] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'06)*, pages 407–418, 2006.

[59] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. Efficient pattern matching over event streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*, pages 147–160, 2008.

[60] Mo Liu, Elke A. Rundensteiner, Daniel J. Dougherty, Chetan Gupta, Song Wang, Ismail Ari, and Abhay Mehta. High-performance nested CEP query processing over event streams. In *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, pages 123–134, 2011.

[61] Thomas Heinze, Yuanzhen Ji, Yinying Pan, Franz Josef Grüneberger, Zbigniew Jerzak, and Christof Fetzer. Elastic complex event processing under varying query load. In *Proceedings of the First International Workshop on Big Dynamic Distributed Data*, pages 25–30, 2013.

[62] Yeye He, Siddharth Barman, and Jeffrey F. Naughton. On Load Shedding in Complex Event Processing. In *Proc. 17th International Conference on Database Theory (ICDT)*, pages 213–224, 2014.

[63] Qunzhi Zhou, Yogesh Simmhan, and Viktor K. Prasanna. Towards an inexact semantic complex event processing framework. In *Proceedings of the Fifth ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 401–402, 2011.

[64] Kia Teymourian, Malte Rohde, and Adrian Paschke. Fusion of background knowledge and streams of events. In *DEBS*, pages 302–313, 2012.

[65] Souleiman Hasan, Seán O'Riain, and Edward Curry. Towards unified and native enrichment in event processing systems. In *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 171–182, 2013.

[66] Souleiman Hasan, Seán O'Riain, and Edward Curry. Approximate semantic matching of heterogeneous events. In *Proceedings of the Sixth ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 252–263, 2012.

[67] Yongrui Qin, Quan Z. Sheng, and Wei Emma Zhang. SIEF: Efficiently Answering Distance Queries for Failure Prone Graphs. In *Proc. of the 18th International Conference on Extending Database Technology (EDBT)*, pages 145–156, 2015.

[68] Sergei Vassilvitskii and Eric Brill. Using Web-Graph Distance for Relevance Feedback in Web Search. In *Proc. of the 29th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 147–153, Seattle, Washington, USA, 2006.

[69] Parthasarathy K., Sreenivasa P. Kumar, and Dominic Damien. Ranked answer graph construction for keyword queries on RDF graphs without distance neighbourhood restriction. In *Proc. of the 20th International Conference on World Wide Web (WWW, Companion Volume)*, pages 361–366, Hyderabad, India, 2011.

[70] Klaus Wehmuth and Artur Ziviani. DACCER: Distributed Assessment of the Closeness CEntrality Ranking in complex networks. *Computer Networks*, 57(13):2536–2548, 2013.

[71] Lina Yao and Quan Z Sheng. Exploiting Latent Relevance for Relational Learning of Ubiquitous Things. In *Proc. of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1547–1551, Maui, Hawaii, USA, 2012.

[72] Hadi Khosravi Farsani, Mohammad Ali Nematbakhsh, and Georg Lausen. SRank: Shortest paths as distance between nodes of a graph with application to RDF clustering. *J. Information Science*, 39(2):198–210, 2013.

[73] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban Computing: Concepts, Methodologies, and Applications. *ACM Trans. Intell. Syst. Technol.*, 5(3), 2014.

[74] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Proc. of the 29th IEEE International Conference on Data Engineering (ICDE)*, pages 410–421, Brisbane, Australia, 2013.

[75] Kexin Xie, Ke Deng, Shuo Shang, Xiaofang Zhou, and Kai Zheng. Finding Alternative Shortest Paths in Spatial Networks. *ACM Trans. Database Syst.*, 37(4):29, 2012.

[76] Jiefeng Cheng and Jeffrey Xu Yu. On-line exact shortest distance query processing. In *EDBT*, pages 481–492, 2009.

[77] Fang Wei. TEDI: efficient shortest path query answering on graphs. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 99–110, Indianapolis, Indiana, USA, 2010.

[78] James Cheng, Yiping Ke, Shumo Chu, and Carter Cheng. Efficient Processing of Distance Queries in Large Graphs: A Vertex Cover Approach. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 457–468, Scottsdale, AZ, USA, 2012.

[79] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Hong Cheng, and Miao Qiao. The exact distance to destination in undirected world. *VLDB J.*, 21(6):869–888, 2012.

[80] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. Fast Exact Shortest-Path Distance Queries on Large Networks by Pruned Landmark Labeling. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 349–360, New York, NY, USA, 2013.

[81] Andy Diwen Zhu, Xiaokui Xiao, Sibo Wang, and Wenqing Lin. Efficient Single-Source Shortest Path and Distance Queries on Large Graphs. In *Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 998–1006, Chicago, IL, USA, 2013.

[82] Ada Wai-Chee Fu, Huanhuan Wu, James Cheng, and Raymond Chi-Wing Wong. IS-LABEL: an Independent-Set based Labeling Scheme for Point-to-Point Distance Querying. *Proc.of the VLDB Endowment*, 6(6):457–468, 2013.

[83] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. In *Proc. of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 937–946, San Francisco, CA, USA, 2002.

[84] Andrei Ciortea, Olivier Boissier, Antoine Zimmermann, and Adina Magda Florea. Reconsidering the social web of things: position paper. In *Proc. the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp) (Adjunct Publication)*, pages 1535–1544, Zurich, Switzerland, 2013.

[85] John Hershberger and Subhash Suri. Vickrey Prices and Shortest Paths: What is an Edge Worth? In *Proc. of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 252–259, 2001.

[86] Kazuo Iwano and Naoki Katoh. Efficient Algorithms for Finding the Most Vital Edge of a Minimum Spanning Tree. *Inf. Process. Lett.*, 48(5):211–213, 1993.

[87] Cristina Bazgan, Sonia Toubaline, and Daniel Vanderpooten. Efficient Algorithms for Finding the k Most Vital Edges for the Minimum Spanning Tree Problem. In *Proc. of the 5th International on Conference Combinatorial Optimization and Applications (COCOA)*, pages 126–140, 2011.

[88] Surender Baswana, Utkarsh Lath, and Anuradha S. Mehta. Single source distance oracle for planar digraphs avoiding a failed node or link. In *Proc. of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232, 2012.

[89] Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proc. of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 506–515, 2009.

[90] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato Fonseca F. Werneck. Hierarchical Hub Labelings for Shortest Paths. In *Proc. of the 20th Annual European Symposium on Algorithms (ESA)*, pages 24–35, Ljubljana, Slovenia, 2012.

[91] Ruoming Jin, Ning Ruan, Yang Xiang, and Victor E. Lee. A highway-centric labeling approach for answering distance queries on large sparse graphs. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 445–456, Scottsdale, AZ, USA, 2012.

[92] Takuya Akiba, Christian Sommer, and Ken Kawarabayashi. Shortest-Path Queries for Complex Networks: Exploiting Low Tree-Width Outside the Core. In *Proc. of the 15th International Conference on Extending Database Technology, (EDBT)*, pages 144–155, Berlin, Germany, 2012.

[93] Ralf Schenkel, Anja Theobald, and Gerhard Weikum. Efficient Creation and Incremental Maintenance of the HOPI Index for Complex XML Document Collections. In *Proc. of the 21st International Conference on Data Engineering (ICDE)*, pages 360–371, Tokyo, Japan, 2005.

[94] Ramadhana Bramandia, Byron Choi, and Wee Keong Ng. Incremental Maintenance of 2-Hop Labeling of Large Graphs. *IEEE Trans. Knowl. Data Eng.*, 22(5):682–698, 2010.

[95] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling. In *Proc. of the 23rd International World Wide Web Conference (WWW)*, pages 237–248, Seoul, Republic of Korea, 2014.

[96] Yongrui Qin, Quan Z. Sheng, and Edward Curry. Matching Over Linked Data Streams in the Internet of Things. *IEEE Internet Computing*, 19(3):21–27, 2015.

[97] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Ali Shemshadi, and Edward Curry. Batch matching of conjunctive triple patterns over linked data streams in the internet of things. In *Proc. of the 27th International Conference on Scientific and Statistical Database Management (SSDBM)*, page 41, 2015.

[98] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Schahram Dustdar, Hua Wang, and Athanasios V. Vasilakos. When Things Matter: A Data-Centric View of the Internet of Things. *CoRR*, abs/1407.2704, 2014.

[99] Jie Gao, Leonidas J. Guibas, Nikola Milosavljevic, and Dengpan Zhou. Distributed Resource Management and Matching in Sensor Networks. In *Proceedings of the 8th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 97–108, San Francisco, California, USA, 2009. IEEE.

[100] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. ParkNet: Drive-by Sensing of Road-Side Parking Statistics. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 123–136, San Francisco, California, USA, 2010. ACM.

[101] Payam M. Barnaghi, Amit P. Sheth, and Cory A. Henson. From Data to Actionable Knowledge: Big Data Challenges in the Web of Things. *IEEE Intelligent Systems*, 28(6):6–11, 2013.

[102] Payam M. Barnaghi, Wei Wang, Cory A. Henson, and Kerry Taylor. Semantics for the Internet of Things: Early Progress and Back to the Future. *International Journal on Semantic Web Information Systems*, 8(1):1–21, 2012.

[103] Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, Kai-Uwe Sattler, and Jürgen Umbrich. Data Summaries for On-Demand Queries over Linked Data. In *WWW*, pages 411–420, 2010.

[104] Souleiman Hasan and Edward Curry. Approximate Semantic Matching of Events for the Internet of Things. *ACM Trans. Internet Techn.*, 14(1), 2014.

[105] A. Seaborne. Rdql - a query language for RDF. In *W3C Member Submission*, 2001.

[106] Erietta Liarou, Stratos Idreos, and Manolis Koubarakis. Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks. In *ISWC*, pages 399–413, 2006.

[107] Yanlei Diao, Mehmet Altinel, Michael J. Franklin, Hao Zhang, and Peter M. Fischer. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. *ACM Trans. Database Syst.*, 28(4):467–516, 2003.

[108] Edward Curry, Souleiman Hasan, and Seán O'Riain. Enterprise energy management using a linked dataspace for Energy Intelligence. In *SustainIT*, pages 1–6, 2012.

[109] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[110] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD*, pages 47–57, 1984.

[111] George H. L. Fletcher and Peter W. Beck. Scalable indexing of RDF graphs for efficient join processing. In *CIKM*, pages 1513–1516, 2009.

[112] Maria-Esther Vidal, Edna Ruckhaus, Tomas Lampo, Amadís Martínez, Javier Sierra, and Axel Polleres. Efficiently Joining Group Patterns in SPARQL Queries. In *ESWC, Part I*, pages 228–242, 2010.

[113] Padmashree Ravindra, HyeongSik Kim, and Kemafor Anyanwu. An Intermediate Algebra for Optimizing RDF Graph Pattern Matching on MapReduce. In *ESWC, Part II*, pages 46–61, 2011.

[114] Souleiman Hasan and Edward Curry. Thematic Event Processing. In *Proceedings of the 15th International Middleware Conference, Bordeaux, France, December 8-12, 2014*, pages 109–120, 2014.

[115] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Ali Shemshadi, and Edward Curry. Towards Efficient Dissemination of Linked Data in the Internet of Things. In *Proc. of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 1779–1782, 2014.

[116] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Wei Emma Zhang, and Hua Wang. Indexing Linked Data in a Wireless Broadcast System with 3D Hilbert Space-Filling Curves. In *Proc. of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 1775–1778, 2014.

[117] Jürgen Umbrich, Katja Hose, Marcel Karnstedt, Andreas Harth, and Axel Polleres. Comparing data summaries for processing live queries over Linked Data. *World Wide Web*, 14(5-6):495–544, 2011.

[118] Tomasz Imielinski, S. Viswanathan, and B. R. Badrinath. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):353–372, 1997.

[119] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An Index Structure for High-Dimensional Data. In *VLDB*, pages 28–39, Mumbai (Bombay), India, September 1996.

[120] Herman J. Haverkort. An Inventory of Three-dimensional Hilbert Space-filling Curves. *Computing Research Repository*, abs/1109.2323, 2011.

[121] Douglas Comer. The ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137, 1979.

[122] Baihua Zheng, Wang-Chien Lee, and Dik Lun Lee. On Searching Continuous k Nearest Neighbors in Wireless Data Broadcast Systems. *IEEE Transactions on Mobile Computing*, 6(7):748–761, 2007.

[123] Yongrui Qin, Quan Z. Sheng, Muntazir Mehdi, Hua Wang, and Dong Xie. Effectively Delivering XML Information in Periodic Broadcast Environments. In *Proc. of the 24th International Conference on Database and Expert Systems Applications (DEXA)*, pages 165–179, 2013.

[124] R. Q. Shaddad, A. B. Mohammad, S. A. Al-Gailani, A. M. Al-hetar, and M. A. Elmagzoub. A survey on access technologies for broadband optical and wireless networks. *J. Network and Computer Applications*, 41:459–472, 2014.

[125] Jianliang Xu, Dik-Lun Lee, Qinglong Hu, and Wang-Chien Lee. *Handbook of Wireless Networks and Mobile Computing*, pages 243–265. John Wiley & Sons, Inc., 2002.

[126] Chang-Sup Park, Chung Soo Kim, and Yon Dohn Chung. Efficient Stream Organization for Wireless Broadcasting of XML Data. In *ASIAN*, pages 223–235, 2005.

[127] Yon Dohn Chung and Ji Yeon Lee. An Indexing Method for Wireless Broadcast XML Data. *Inf. Sci.*, 177(9):1931–1953, 2007.

[128] Yongrui Qin, Weiwei Sun, Zhuoyao Zhang, Ping Yu, Zhenying He, and Weiyu Chen. A Novel Air Index Scheme for Twig Queries in On-Demand XML Data Broadcast. In *DEXA*, pages 412–426, 2009.

[129] Weiwei Sun, Ping Yu, Yongrui Qin, Zhuoyao Zhang, and Baihua Zheng. Two-Tier Air Indexing for On-Demand XML Data Broadcast. In *ICDCS*, pages 199–206, 2009.

[130] Jun Pyo Park, Chang-Sup Park, and Yon Dohn Chung. Energy and Latency Efficient Access of Wireless XML Stream. *J. Database Manag.*, 21(1):58–79, 2010.

[131] Jingjing Wu, Peng Liu, Lu Gan, Yongrui Qin, and Weiwei Sun. Energy-Conserving Fragment Methods for Skewed XML Data Access in Push-Based Broadcast. In *WAIM*, pages 590–601, 2011.

[132] Weiwei Sun, Peng Liu, Jingjing Wu, Yongrui Qin, and Baihua Zheng. An Automaton-Based Index Scheme for On-Demand XML Data Broadcast. In *DASFAA (2)*, pages 96–110, 2012.

[133] Chang-Sup Park, Jun Pyo Park, and Yon Dohn Chung. PrefixSummary: A Directory Structure for Selective Probing on Wireless Stream of Heterogeneous XML Data. *IEICE Transactions*, 95-D(5):1427–1435, 2012.

[134] Weiwei Sun, Zhuoyao Zhang, Ping Yu, and Yongrui Qin. Efficient Data Scheduling for Multi-item Queries in On-Demand Broadcast. In *EUC (1)*, pages 499–505, 2008.

[135] Jun Chen, Victor C. S. Lee, and Kai Liu. On the Performance of Real-time Multi-item Request Scheduling in Data Broadcast Environments. *Journal of Systems and Software*, 83(8):1337–1345, 2010.

[136] Swarup Acharya, Rafael Alonso, Michael J. Franklin, and Stanley B. Zdonik. Broadcast Disks: Data Management for Asymmetric Communications Environments. In *SIGMOD*, pages 199–210, 1995.

[137] Swarup Acharya, Michael J. Franklin, and Stanley B. Zdonik. Balancing Push and Pull for Data Broadcast. In *SIGMOD Conference*, pages 183–194, 1997.

[138] Ye-In Chang and Wu-Han Hsieh. An Efficient Scheduling Method for Query-Set-Based Broadcasting in Mobile Environments. In *ICDCS Workshops*, pages 478–483, 2004.

[139] Tomasz Imielinski, S. Viswanathan, and B. R. Badrinath. Data on Air: Organization and Access. *IEEE Trans. Knowl. Data Eng.*, 9(3):353–372, 1997.

[140] Davood Rafiei, Daniel L. Moise, and Dabo Sun. Finding Syntactic Similarities Between XML Documents. In *DEXA Workshops*, pages 512–516, 2006.

[141] Sven Helmer. Measuring the Structural Similarity of Semistructured Documents Using Entropy. In *VLDB*, pages 1022–1032, 2007.

[142] Dekang Lin. An Information-Theoretic Definition of Similarity. In *ICML*, pages 296–304, 1998.

[143] Prasanna Ganesan, Hector Garcia-Molina, and Jennifer Widom. Exploiting Hierarchical Domain Structure to Compute Similarity. *ACM Trans. Inf. Syst.*, 21(1):64–93, 2003.

[144] Lee R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302, 1945.

[145] Wang Lian, David Wai-Lok Cheung, Nikos Mamoulis, and Siu-Ming Yiu. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. *IEEE Trans. Knowl. Data Eng.*, 16(1):82–96, 2004.

[146] Tomasz Imielinski, S. Viswanathan, and B. R. Badrinath. Power Efficient Filtering of Data an Air. In *EDBT*, pages 245–258, 1994.

[147] Yon Dohn Chung and Myoung-Ho Kim. Effective Data Placement for Wireless Broadcast. *Distributed and Parallel Databases*, 9(2):133–150, 2001.

[148] Marc A. Viredaz, Lawrence S. Brakmo, and William R. Hamburgen. Energy Management on Handheld Devices. *ACM Queue*, 1(7):44–52, 2003.

[149] Zografoula Vagena, Mirella Moura Moro, and Vassilis J. Tsotras. RoXSum: Leveraging Data Aggregation and Batch Processing for XML Routing. In *ICDE*, pages 1466–1470, 2007.

[150] IPTC. International Press Telecommunications Council, News Industry Text Format (NITF), version 2.5. http://www.nitf.org, visited on 20/09/2012, 2000.

[151] Yanlei Diao, Mehmet Altinel, Michael J. Franklin, Hao Zhang, and Peter M. Fischer. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. *ACM Trans. Database Syst.*, 28(4):467–516, 2003.

[152] Weiwei Sun, Weibin Shi, Bole Shi, and Yijun Yu. A Cost-Efficient Scheduling Algorithm of On-Demand Broadcasts. *Wireless Networks*, 9(3):239–247, 2003.

[153] Bingjun Sun, Ali R. Hurson, and John Hannan. Energy-Efficient Scheduling Algorithms of Object Retrieval on Indexed Parallel Broadcast Channels. In *ICPP*, pages 440–447, 2004.

[154] Sang Hyuk Kang. Wireless Data Broadcast Scheduling with Utility Metric Based on Soft Deadline. *IEICE Transactions*, 94-B(5):1424–1431, 2011.

[155] Yon Dohn Chung and Myoung-Ho Kim. QEM: A Scheduling Method for Wireless Broadcast Data. In *DASFAA*, pages 135–142, 1999.

[156] Guanling Lee, Meng-Shin Yeh, Shou-Chih Lo, and Arbee L. P. Chen. A Strategy for Efficient Access of Multiple Data Items in Mobile Environments. In *MDM*, pages 71–78, 2002.

[157] Yongrui Qin, Hua Wang, and Jitian Xiao. Effective Scheduling Algorithm for On-Demand XML Data Broadcasts in Wireless Environments. In *ADC*, pages 95–102, 2011.

[158] Sang-Hyun Park, Jae-Ho Choi, and SangKeun Lee. An Effective, Efficient XML Data Broadcasting Method in a Mobile Wireless Network. In *DEXA*, pages 358–367, 2006.

[159] Yongrui Qin, Hua Wang, and Lili Sun. Cluster-Based Scheduling Algorithm for Periodic XML Data Broadcast in Wireless Environments. In *AINA Workshops*, pages 855–860, 2011.

[160] Dominique Guinard. A Web of Things for Smarter Cities. In *Technical Talk*, pages 1–8, 2010.

[161] Jing Yuan, Yu Zheng, Liuhang Zhang, Xing Xie, and Guangzhong Sun. Where to Find My Next Passenger? In *Proceedings of the 13th International Conference on Ubiquitous Computing (Ubicomp)*, pages 109–118, Beijing, China, 2011. ACM.

[162] Tsung-Te Lai, Yu-Han Chen, Polly Huang, and Hao-Hua Chu. PipeProbe: A Mobile Sensor Droplet for Mapping Hidden Pipeline. In *Proceedings of the 8th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 113–126, Zurich, Switzerland, 2010. ACM.

[163] Nithya Ramanathan, Thomas Schoellhammer, Eddie Kohler, Kamin Whitehouse, Thomas Harmon, and Deborah Estrin. Suelo: Human-Assisted Sensing for Exploratory Soil Monitoring Studies. In *Proceedings of the 7th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 197–210, Berkeley, California, USA, 2009. ACM.

[164] Thomas Zimmerman and Christine Robson. Monitoring Residential Noise for Prospective Home Owners and Renters. In *Proceedings of the 9th International Conference on Pervasive Computing (Pervasive)*, pages 34–49, San Francisco, CA, USA, 2011. Springer.

[165] Jonathan Lester, Carl Hartung, Laura Pina, Ryan Libby, Gaetano Borriello, and Glen Duncan. Validated Caloric Expenditure Estimation Using a Single Body-Worn Sensor. In *Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp)*, pages 225–234, Orlando, Florida, USA, 2009. ACM.

[166] Nissanka B. Priyantha, Aman Kansal, Michel Goraczko, and Feng Zhao. Tiny Web Services: Design and Implementation of Interoperable and Evolvable Sensor Networks. In *Proceedings of the 6th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 253–266, Raleigh, NC, USA, 2008. ACM.

[167] Manu Gupta, Stephen S. Intille, and Kent Larson. Adding GPS-Control to Traditional Thermostats: An Exploration of Potential Energy Savings and Design Challenges. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive)*, pages 95–114, Nara, Japan, 2009. Springer.

[168] Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F. Abdelzaher. GreenGPS: A Participatory Sensing Fuel-Efficient Maps Application. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 151–164, San Francisco, California, USA, 2010. ACM.

[169] Friedemann Mattern and Christian Floerkemeier. From the Internet of Computers to the Internet of Things. In *From Active Data Management to Event-Based Systems and More*, pages 242–259. Springer, 2010.

[170] Peter Stephan, Gerrit Meixner, Holger Koessling, Florian Floerchinger, and Lisa Ollinger. Product-Mediated Communication through Digital Object Memories in Heterogeneous Value Chains. In *Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 199–207, Mannheim, Germany, 2010. IEEE.

[171] Yanbo Wu, Quan Z. Sheng, Hong Shen, and Sherali Zeadally. Modeling Object Flows from Distributed and Federated RFID Data Streams for Efficient Tracking and Tracing. *IEEE Transactions on Parallel and Distributed Systems*, 24(10):2036–2045, 2013.

[172] Wenfei Fan, Floris Geerts, Shuai Ma, and Heiko Müller. Detecting Inconsistencies in Distributed Data. In *Proceedings of the 26th International Conference on Data Engineering (ICDE)*, pages 64–75, Long Beach, California, USA, 2010. IEEE.

[173] Beng Chin Ooi, Kian-Lee Tan, and Anthony K. H. Tung. Sense The Physical, Walkthrough The Virtual, Manage The Co (existing) Spaces: A Database Perspective. *SIGMOD Record*, 38(3):5–10, 2009.

[174] Gerhard Weikum. Database Researchers: Plumbers or Thinkers? In *Proceedings of the 14th International Conference on Extending Database Technology (EDBT)*, pages 9–10, Uppsala, Sweden, 2011. ACM.

[175] Evan Welbourne, Leilani Battle, Garrett Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. Building the Internet of Things Using RFID: The RFID Ecosystem Experience. *IEEE Internet Computing*, 13(3):48–55, 2009.

[176] Brent Lagesse, Mohan Kumar, Justin Mazzola Paluska, and Matthew Wright. DTT: A Distributed Trust Toolkit for Pervasive Systems. In *Proceedings of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–8, Galveston, TX, USA, 2009. IEEE.

[177] Michael E. Kounavis, Xiaozhu Kang, Ken Grewal, Mathew Eszenyi, Shay Gueron, and David Durham. Encrypting the Tnternet. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 135–146, New Delhi, India, 2010. ACM.

[178] Predrag V. Klasnja, Sunny Consolvo, Tanzeem Choudhury, Richard Beckwith, and Jeffrey Hightower. Exploring Privacy Concerns about Personal Sensing. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive)*, pages 176–183, Nara, Japan, 2009. Springer.

[179] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. Streaming First Story Detection with application to Twitter. In *HLT-NAACL*, pages 181–189, 2010.