Chapter 12

THE INTERNET OF THINGS: A SURVEY FROM THE DATA-CENTRIC PERSPECTIVE

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY
charu@us.ibm.com

Naveen Ashish
University of California at Irvine
Irvine, CA
ashish@ics.uci.edu

Amit Sheth
Wright State University
Dayton, OH
amit.sheth@wright.edu

Abstract Advances in sensor data collection technology, such as pervasive and embedded devices, and RFID Technology have lead to a large number of *smart devices* which are connected to the net and continuously transmit their data over time. It has been estimated that the number of internet connected devices has overtaken the number of humans on the planet, since 2008. The collection and processing of such data leads to unprecedented challenges in mining and processing such data. Such data needs to be processed in real-time and the processing may be highly distributed in nature. Even in cases, where the data is stored offline, the size of the data is often so large and distributed, that it requires the use of *big data analytical tools* for processing. In addition, such data is often sensitive, and brings a number of privacy challenges associated with it. This chapter will discuss a data analytics perspective about mining and managing data associated with this phenomenon, which is now known as the *internet of things*.

Keywords: The Internet of Things, Pervasive Computing, Ubiquitous Computing

1. Introduction

The internet of things [14] refers to uniquely addressable objects and their virtual representations in an Internet-like structure. Such objects may link to information about them, or may transmit real-time sensor data about their state or other useful properties associated with the object. *Radio-Frequency Identification Technology (RFID)* [23, 47, 93, 94] is generally seen as a key enabler of the internet of things, because of its ability to track a large number of uniquely identifiable objects with the use of *Electronic Product Codes (EPC)*. However, other kinds of ubiquitous sensor devices, barcodes, or 2D-codes may also be used to enable the *Internet of Things (IoT)*. The concepts of *pervasive computing* and *ubiquitous computing* are related to the internet of things, in the sense that all of these paradigms are enabled by *large-scale embedded sensor devices*.

The vision of the internet of things is that individual objects of everyday life such as cars, roadways, pacemakers, wirelessly connected pill-shaped cameras in digestive tracks, smart billboards which adjust to the passersby, refrigerators, or even cattle can be equipped with sensors, which can track useful information about these objects. Furthermore, if the objects are *uniquely addressable* and *connected to the internet*, then the information about them can flow through the same protocol that connects our computers to the internet. Since these objects can sense the environment and communicate, they have become tools for understanding complexity, and may often enable autonomic responses to challenging scenarios without human intervention. This broader principle is popularly used in IBM's *Smarter Planet* initiative for autonomic computing.

Since the internet of things is built upon the ability to uniquely identify internet-connected objects, the addressable space must be large enough to accommodate the uniquely assigned IP-addresses to the different devices. The original internet protocol IPv4 uses 32-bit addresses, which allows for only about 4.3 billion unique addresses. This was a reasonable design at the time when IPv4 was proposed, since the total number of internet connected devices was a small fraction of this number. With an increasing number of devices being connected to the internet, and with each requiring its IP-address (for full peer-to-peer communication and functionality), the available IP-addresses are in short supply. As of 2008, the number of internet connected devices exceeded the total number of people on the planet. Fortunately, the new IPv6 protocol which is being adopted has 128-bit addressability, and therefore has an address space of 2^{128}. This is likely to solve the addressability bottleneck being faced by the internet of things phenomenon.

It is clear that from a data centric perspective, *scalability*, *distributed processing*, and *real time analytics* will be critical for effective enablement. The large number of devices simultaneously producing data in an automated way will greatly dwarf the information which individuals can enter manually. Humans are constrained by time and physical limits in terms of how much a single human can enter into the system manually, and this constraint is unlikely to change very much over time. On the other hand, the physical limitations on how much data can be effectively collected from embedded sensor devices have steadily been increasing with advances in hardware technology. Furthermore, with increasing numbers of devices which are connected to the internet, the number of such streams also continue to increase in time. Simply speaking, automated sensor data is likely to greatly overwhelm the data which are available from more traditional human-centered sources such as social media. In fact, it is the trend towards ubiquitous and pervasive computing, which is the greatest driving force towards *big data analytics.*

Aside from scalability issues, privacy continues to be a challenge for data collection [40, 58–62, 69, 71, 78, 81, 82, 111]. Since the individual objects can be tracked, they can also lead to privacy concerns, when these objects are associated with individuals. A common example in the case of RFID technology is one in which a tagged object (such as clothing) is bought by an individual, and then the individual can be tracked because of the presence of the tag on their person. In cases, where such information is available on the internet, the individual can be tracked from almost anywhere, which could lead to unprecedented violations of privacy.

The material in this chapter is closely related to two other chapters [8, 9] in this book corresponding to *social sensing* and *RFID processing* respectively. However, we have devoted a separate chapter to the internet of things, since it is a somewhat separate concept in its own right, though it is related to the afore-mentioned technologies in the following ways:

- *RFID technology* is a key enabler for the internet of things, because it allows the simultaneous identification of large numbers of objects with cost-effective tags [108]. However, in practice many other kinds of embedded sensor technology may be used for enablement. Furthermore, where more sophisticated sensor information is required about the object, *RFID* technology can only provide a partial component of the data required for full enablement.

- *Social sensing* is a paradigm which refers to the interaction between people with embedded sensor devices, which are typically mobile phones. However, the internet of things is a more general concept, where even mundane objects of everyday life such as refrigerators, consumer products, televisions, or cars may be highly connected, and may be utilized for making smarter and automated decisions.

1.1 The Internet of Things: Broader Vision

The *Internet of Things* is a vision, which is currently being built– there is considerable diversity in its interpretation by different communities, who are involved in an inherently cross-disciplinary effort, involving sensor networking, data management and the world wide web. This diversity is also a result of the technical breadth of the consortiums, industries and communities which support the vision. Correspondingly, this is also reflected in the diversity of the technologies, which are being developed by the different communities. Nevertheless, there are numerous common features across the different visions about what the internet of things may constitute, and it is one of the goals of this paper to bring together these visions from a data-centric perspective.

A simple and broad definition of the internet of things [41, 16] is as follows: "*The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals*". The process of machines communicating with one another, is also referred to as the Machine-to-Machine (M2M) paradigm. This requires tremendous *data-centric* capabilities, which is the primary medium of communication between the different entities. Therefore, the ability to securely and privately collect, manage, index, query and process large amounts of data is critical.

In order to enable these goals, a variety of research efforts have been initiated supporting various aspects of these goals. Each of these visions has a slightly different emphasis on different parts of this data-centric pipeline. There are three primary visions [16] around which most of the research in this area is focussed:

- **Things-oriented Vision:** This vision is largely supported by the RFID vision of tracking objects with tags [108]. This vision supports the use of the *Electronic Product Code (EPC)* in conjunction with RFID technology to collect and track sensor data. The EPC-global framework [118] is based on this vision of unique product identification and tracking.

  The *things-oriented vision* is by far the dominant vision today, and RFID technology is often (mistakenly) assumed to be synonymous with the internet of things. It is important to note that while RFID technology will continue to be a very important enabler of this phenomenon (especially because of the unique identifiability provided by the EPC), it is certainly not the only technology which can be used for data collection. The things-vision includes data generated by other kinds of embedded sensor devices, actuators, or mobile phones. In fact, more sophisticated sensor technology (beyond tags) is usually required in conjunction with RFID in order to collect and transmit useful information about the objects being tracked. An example of this is the *Wireless Identification and Sensing Platform (WISP)* [121] being constructed at Intel. WISPs are powered by standard RFID readers, and can be used to measure sensing quantities in the physical environment, such as temperature. The overall vision is that of *RFID-based Sensor Networks* [22], which integrate RFID technology, small sensing and computing devices, RFID readers (which provide a key intermediate layer between the "things" and the "internet"), and internet connectivity.

- **Internet-oriented Vision:** The internet-oriented vision corresponds to construction of the IP protocols for enabling *smart objects*, which are internet connected. This is typically spearheaded by the *IPSO* alliance [122]. Typically, this technology goes beyond RFID.

  A theoretical concept, which has emerged in this direction is that of the *spime*, [99] an object, which is uniquely identifiable, and may of its real-time attributes (such as location) can be continuously tracked. Examples of this concept include *smart objects*, which are tiny computers which have sensors or actuators, and a communication device. These can be embedded in cars, light switches, thermometers, billboards, or machinery. Typically these objects have CPU, memory, a low power communication device, and are battery operated. Since, each of these devices would require its own IP-address, a large part of this vision is also about developing the internet infrastructure to accommodate the ever-expanding number of "things" which require connectivity. A classic example of the efforts in this space include the development of IPv6, which has a much larger addressable IP-space. This vision also supports the development of the *web of things*, in which the focus is to re-use the web-based internet standards and protocols to connect the expanding eco-system of embedded devices built into everyday smart objects [45]. This re-use ensures that widely accepted and understood standards such as URI, HTTP, etc. are used to access the functionality of the smart objects. This approach exposes the synchronous functionality of smart objects through a *REST interface.* The REST interface defines the notion of a resource as any component of an application that is worth being uniquely identified and linked to. On the Web, the identification of resources relies on Uniform Resource Identifiers (URIs), and representations retrieved through resource interactions contain links to other resources [46]. This means that applications can follow links through an interconnected web of resources. Similar to the web, clients of such services can follow these links in order to find resources to interact with. Therefore, a client may explore a service by browsing it, and the services will use different link types to represent different relationships.

- **Semantic-oriented Vision:** The semantic vision addresses the issues of data management which arise in the context of the vast amounts of information which are exchanged by smart objects, and the resources which are available through the web interface. The idea is that standardized resource descriptions are critical to enable interoperability of the heterogeneous resources available through the web of things. The semantic vision is really about the separation of the meanings of data, from the actual data itself. The idea here is that the semantic meanings of objects are stored separately from the data itself, and effective tools for the management of this information. A key capability that this enables in *semantic interoperability and integration* 5semantic i.e., across the sensor data from various sensors.

The diversity of these visions is a result of the diversity in the stakeholders involved in the building of this vision, and also because the vast infrastructure required by this vision naturally requires the technical expertise from different areas of data analytics, and networking.

This chapter is organized as follows. The next section will discuss applications supported by the internet of things. In section 3, we will present networking issues, and their relationship to the data collection process. Section 4 will discuss issues in data management. This includes methods for querying, indexing, and real-time data analytics. Privacy issues are discussed in section 5. Section 6 contains the conclusions and summary.

2. Applications: Current and Future Potential

The ability of machines and sensors to collect, transmit data and communicate with one another can lead to unprecedented flexibility in terms of the variety of applications which can be supported with this paradigm. While the full potential of the IoT vision is yet to be realized, we will review some of the early potential of existing applications, and also discuss future possibilities. The latter set of possibilities are considered ambitious, but reasonable goals in the longer term, as a part of this broader vision.

Product Inventory Tracking and Logistics This is perhaps one of the most popular applications of the internet of things, and was one of the first large scale applications of RFID technology. The movements of large amounts of products can be tracked by inexpensive RFID tags. For large franchises and organizations, the underlying RFID readers may serve as an intermediate layer between the data collection and internet-connectivity. This provides unprecedented opportunities for product tracking in an automated way. In addition, it is possible to design software, which uses the information from the transmitted data in order to trigger alerts in response to specific events.

Smarter Environment More sophisticated embedded sensor technology can be used in order to monitor and transmit critical environmental parameters such as temperature, humidity, pressure etc. In some cases, RFID technology can be coupled with more sophisticated sensors, in order to send back information which is related to specific objects [106, 107]. Such information can also be used to control the environment in an energy-efficient way. For example, smart sensors in a building can be used in order to decide when the lights or air-conditioning in a room in the building should be switched off, if the room is not currently being used.

Social Sensing Social sensing is an integral paradigm of the internet of things, when the objects being tracked are associated with individual people. Examples of such sensing objects include mobile phones, wearable sensors and piedometers. Such paradigms have tremendous value in enabling social networking paradigms in conjunction with sensing. The increasing ability of commodity hardware to track a wide variety of real-life information such as location, speed, acceleration, sound, video and audio leads to unprecedented opportunity in enabling an increasingly connected and mobile world of users that are ubiquitously connected to the internet. This is also a natural mode in which humans and things can interact with one another in a seamless way over the internet. A detailed discussion on social sensing may be found in [8].

Smarter Devices In the future, it is envisioned that a variety of devices in our day-to-day life such as refrigerators, televisions and cars will be smarter in terms of being equipped with a variety of sensors and will also have internet connectivity in order to publish the collected data. For example, refrigerators may have smart sensors which can detect the quantities of various items and the freshness of perishable items. The internet connectivity may provide the means to communicate with and alert the user to a variety of such information. The user may themselves be connected with the use of one a social sensing device such as a mobile phone. Similarly, sensor equipped and internet connected cars can both provide information to and draw from the repository of data on traffic status and road conditions. In addition, as has recently been demonstrated by the *Google Car* project, sensor-equipped cars have the capability to perform assisted driving for a variety of applications [124]. A further advancement of this technology and vision would be the development of internet connected cars, which can perform automated driving in a way which is sensitive to traffic conditions, with the use of aggregate data from other network connected cars.

Identification and Access Control RFID tags can be used for a wide variety of access control applications. For example, RFID sensors can be used for fast access control on highways, instead of manual toll booths. Similarly, a significant number of library systems have implemented *smart check out* systems with tags on items. When the collected data is allowed to have network connectivity for further (aggregate) analysis and processing, over multiple access points, this also enables significant tracking and analysis capabilities for a variety of applications. For example, in a network of connected libraries, automated tracking can provide the insights required to decide which books to acquire for the different locations, based on the aggregate analysis.

Electronic Payment Systems Numerous electronic payment systems are now being developed with the use of a variety of smart technologies. The connectivity of RFID readers to the internet can be used in order to implement payment systems. An example is the Texas Instruments's *Speedpass*, pay-at-pump system, which was introduced in Mobil stations in the mid-nineties. This system uses RFID technology in order to detect the identity of the customer buying gas, and this information is used in order to debit the money from the customer's bank account. Another popular payment system, which is becoming available with many mobile phones is based on *Near Field Communications (NFC).* Many of the latest Android phones have already implemented such systems for mobile payments.

Health Applications RFID and sensor technology have been shown to be very useful in a variety of health applications [100]. For example, RFID chips can be implanted in patients in order to track their medical history. Sensor technology is also very useful in automated monitoring of patients with heart or alzheimer's conditions, assisted living, emergency response, and health monitoring applications [31, 36, 74]. Internet-connected devices can also directly communicate with the required emergency services when required, in order ro respond to emergences, when the sensed data shows the likelihood of significant deterioration in the patient's condition. Smart healthcare technology has the potential to save lives, by significantly improving emergency response times.

3. Networking Issues: Impact on Data Collection

The primary networking issues for the internet of things arise during the data collection phase. At this phase, a variety of technologies are used for data collection, each of which have different tradeoffs in terms of capabilities, energy efficiency, and connectivity, and may also impact both the cleanliness of the data, and how it is transmitted and managed. Therefore, we will first discuss the key networking technologies used for data collection. This will further influence our discussion on data-centric issues of privacy, cleaning and management:

3.1 RFID Technology

At the most basic level, the definition of Radio Frequency Identification (RFID) is as follows: *RFID is a technology which allows a sensor (reader) to read, from a distance, and without line of sight, a unique product identification code (EPC) associated with a tag.* Thus, the unique code from the tag is transmitted to one or more sensor reader(s), which in turn, transmit(s) the readings to one or more server(s). The data at the server is aggregated in order to track all the different product codes which are associated with the tags. We note that such RFID tags do not need to be equipped with a battery, since they are powered by the sensor reader. This is a key advantage from the perspective of providing a high life time to the tracking process. The sensor readers provide a key intermediate layer between the data collection process and network connectivity. The RFID tags typically need to be present at a short distance from the readers in order for the reading process to work effectively. From a data-centric perspective the major limitations of the basic RFID technology are the following:

- The basic RFID technology has limited capabilities in terms of providing more detailed sensing information, especially when passive tags are used.

- The range of the tags is quite small, and is typically of the order of between 5 to 20 meters. As a result significant numbers of readings are dropped.

- The data collected is massively noisy, incomplete and redundant. Sensor readers may repeatedly scan EPC tags which are at the same location (with no addition of knowledge), and multiple readers in the same locality may scan the same EPC tag. This leads to numerous challenges from the perspective of data cleaning. This cleaning typically needs to be performed in the middleware within the sensor reader.

- RFID collection technology leads to considerable privacy challenges, especially when the tags are associated with individual. The tags are susceptible to a wide variety of eavesdropping mechanisms, since covert readers can be used in order to track the locations of individuals.

A detailed discussion of the data-centric issues associated with RFID technology may be found in [9].

3.2 Active and Passive RFID Sensor Networks

The major limitation of the basic RFID sensor technology is that it does not enable detailed sensing information. However, a number of recent methods have been proposed to incorporate sensing into RFID capabilities. One possibility is to use an onboard battery [106, 107] in order to transmit more detailed sensing information about the environment. This is referred to as an *active* RFID tag. Of course, the major limitation of such an approach is that the life-time of the tag is limited by the battery. If a large number of objects are being tracked at given time, then it is not practical to replace the battery or tag on such a basis. Nevertheless, a significant amount of smart object technology is constructed with this approach. The major challenge from the data-centric perspective is to clean or impute the missing data from the underlying collection.

Recently, a number of efforts have focussed on the creating the ability to perform the sensing with *passive* RFID tags. Recently, a number of efforts in this direction [22, 121] are designed to sense more detailed information with the use of passive tags. The major challenge of this approach is that the typical *range* at which the reader must be placed to the tag is even smaller than the basic RFID technology, and may sometimes be less than three meters. This could lead to even more challenges in terms of the dropped readings in a wide variety of application scenarios. On the other hand, since the tag is passive, there are no limitations on the life time because of battery-power consumption.

3.3 Wireless Sensor Networks

A possible solution is to use conventional wireless sensing technology for building the internet of things. One, some, or all nodes in the sensor network may function as gateways to the internet. The major advantage is that peer-to-peer communications among the nodes are possible with this kind of approach. Of course, this kind of approach is significantly more expensive in large-scale applications and is limited by the battery life. The battery-life would be further limited by the fact, that most IP protocols cannot accommodate the sleep modes required by sensor motes in order to conserve battery life. Since the network connectivity of the internet of things is based on the IP protocols, this would require the sensor devices to be on constantly. This would turn out to be a very significant challenge in terms of battery life. The energy requirements can reduced by a variety of methods such as lower sampling or transmission rates, but this can impact the timeliness and quality of the data available for the underlying applications. Wireless sensor networks also have some quality issues because of the conversion process from voltages to measured values, and other kinds of noise. Nevertheless, from a comparative point of view, wireless sensor networks do have a number of advantages in terms of the quality, range, privacy and security of the data collected and transmitted, and are likely to play a significant role in the internet of things.

3.4 Mobile Connectivity

A significant number of objects in the internet of things, such as mobile phones can be connected by 3G and WiFi connectivity. However, the power usage of such systems is quite high. Such solutions are of course sometimes workable, because such objects fall within the *social sensing* paradigm, where each mobile object belongs to a participant who is responsible for maintaining the battery and other connectivity aspects of the sensing object which is transmitting the data. In such cases, however, the privacy of the transmitted data (eg. GPS location) becomes sensitive, and it is important to design privacy preservation paradigms in order to either limit the data transmission, or reduce the fidelity of the transmitted data. This is of course not desirable from the data analytics perspective, because it reduces the quality of the data analytics output. Correspondingly, the user-trust in the data analytics results are also reduced.

Since mobile phones are usually designed for communication-centric applications, they may only have certain sensors such as GPS, accelerometers, microphones, or video-cameras, which are largely user centric. Also they may allow direct human input into the sensor process. Nevertheless, they do have a number of limitations in not being able to collect arbitrarily kinds of sensed data (eg. humidity). Therefore, the applicability of such devices is often in the context of user-centric applications such as social sensing [8], or working with other smart devices in the context of a broader smart infrastructure.

Since such connectivity has high power requirements, it is important to make the data collection as energy efficient as possible. A salient point to be kept in mind is that data collection can sometimes be performed with the use of multiple methods in the same devices (eg. approximate cell phone tower positioning vs. accurate GPS for location information). Furthermore, tradeoffs are also possible during data *transmission* between timeliness and energy consumption (eg. real-time 3G vs. opportunistic WiFi). A variety of methods have been proposed in recent years, for calibrating these different tradeoffs, so that the energy efficiency is maximized with significantly compromising the data-centric needs of the application [30, 84, 91, 117]. Examples of specific methods include energy-timeliness tradeoffs [91], adaptive sampling [84], and application-specific collection modes [117]. We note that the impact of such collection policies on data management and processing applications is likely be significant. Therefore, it is critical to design appropriate data cleaning and processing methods, which take such issues of data quality into consideration.

4. Data Management and Analytics

The key to the power of the internet of things paradigm is the ability to provide real time data from many different distributed sources to other machines, smart entities and people for a variety of services. One major challenge is that the underlying data from different resources are extremely heterogeneous, can be very noisy, and are usually very large scale and distributed. Furthermore, it is hard for other entities to use the data effectively, without a clear description of what is available for processing. In order to enable effective use of this very heterogeneous and distributed data, frameworks are required to describe the data in a sufficiently intuitive way, so that it becomes more easily usable i.e., the problem of semantic interoperability is addressed. This leads to unprecedented challenges both in terms of providing high quality, scalable and real time analytics, and also in terms of intuitively describing to users information about what kind of data and services are available in a variety of scenarios. Therefore, methods are required to clean, manage, query and analyze the data in the distributed way. The cleaning is usually performed at data collection time, and is often embedded in the middleware which interfaces with the sensor devices. Therefore, the research on data cleaning is often studied in the context of the *things-oriented vision.* The issues of providing standardized descriptions and access to the data for smart services are generally studied in the context of standardized web protocols and interfaces, and description/querying frameworks such as offered by *semantic web* technology. The idea is to reuse the existing web infrastructure in an intuitive way, so the heterogeneity and distributed nature of the different data sources can be seamlessly integrated with the different services. These issues are usually studied in the context of the *web of things* and the *semantic web* visions. Thus, the end-to-end data management of IoT technology requires the unification and collaboration between the different aspects of how these technologies are developed, in order to provide a seamless and effective infrastructure.

Unlike the world wide web of documents, in which the objects themselves are described in terms of a natural lexicon, the objects and data in the internet of things, are heterogeneous, and may not be naturally available in a sufficiently descriptive way to be searchable, unless an effort is made to create standardized descriptions of these objects in terms of their properties. Frameworks such as RDF provide such a standardized descriptive framework, which greatly eases various functions such as search and querying in the context of the underlying heterogeneity and lack of naturally available descriptions of the objects and the data. Semantic technologies are viewed as a key to resolving the problems of inter-operability and integration within this heterogeneous world of ubiquitously interconnected objects and systems [65]. Thus, the *Internet of Things* will become a *Semantic Web of Things*. It is generally recognized that this interoperability cannot be achieved by making everyone comply to *too many* rigid standards in ubiquitous environments. Therefore, the interoperability can be achieved by designing middleware [65], which acts as a seamless interface for joining heterogeneous components together in a particular IoT application. Such a middleware offers application programming interfaces, communications and other services to applications. Clearly, *some* data-centric standards are still necessary, in order to represent and describe the properties of the data in a homogenous way across heterogeneous environments.

The internet of things requires a plethora of different middlewares, at different parts of the pipeline for data collection and cleaning, service enablement etc. In this section, we will study the data management issues at different stages of this pipeline. First, we will start with data cleaning and pre-processing issues, which need to be performed at data collection time. We will follow this up with issues of data and ontology representation. Finally, we will describe important data-centric applications such as mining with big data analytics, search and indexing.

4.1 Data Cleaning Issues

The data cleaning in IoT technology may be required for a variety of reasons: (a) When is data is collected from conventional sensors, it may be noisy, incomplete, or may require probabilistic uncertain modeling [34]. (b) RFID data is extremely noisy, incomplete and redundant because a large fraction of the readings are dropped, and there are cross-reads from multiple sensor readers. (c) The process of privacy-preservation may require an intentional reduction of data quality, in which case methods are required for privacy-sensitive data processing [6].

Conventional sensor data is noisy because sensor readings are often created by converting other measured quantities (such as voltage) into measured quantities such as the temperature. This process can be very noisy, since the conversion process is not precise. Furthermore, systematic errors are also introduced, because of changes in external conditions or ageing of the sensor. In order to reduce such errors, it is possible to either re-calibrate the sensor [25], or perform data-driven cleaning and uncertainty modeling [34]. Furthermore, the data may sometimes be incomplete because of periodic failure of some of the sensors. A detailed discussion of methods for cleaning conventional sensor data is provided in Chapter 2 of this book.

RFID data is even noisier than conventional sensor data, because of the inherent errors associated with the reader-tag communication process. Furthermore, since RFID data is repeatedly scanned by the reader, even when the data is stationary, it is *massively redundant*. Techniques for cleaning RFID data are discussed in [9]. Therefore, we will provide a brief discussion of these issues and refer the readers to the other chapters for more details. In the context of many different kinds of sources such as conventional sensor data, RFID data, and privacy-preserving data mining, uncertain probabilistic modeling seems to be a solution, which is preferred in a variety of different contexts [6, 34, 66], because of recent advances in the field of probabilistic databases [7]. The broad idea is that when the data can be represented in probabilistic format (which reflects its errors and uncertainty), it can be used more effectively for mining purposes. Nevertheless, probabilistic databases are still an emerging field, and, as far as we are aware, all commercial solutions work with conventional (deterministic) representations of the sensor data. Therefore, more direct solutions are required in order to clean the data as deterministic entities.

In order to address the issue of lost readings in RFID data, many data cleaning systems [47, 120] is to use a temporal smoothing filter, in which a sliding window over the reader's data stream interpolates for lost readings from each tag within the time window. The idea is to provide each tag more opportunities to be read within the smoothing window. Since the window size is a critical parameter, the work in [55] proposes *SMURF (Statistical sMoothing for Unreliable RFid data)*, which is an adaptive smoothing filter for raw RFID data streams. This technique determines the most effective window size automatically, and continuously changes it over the course of the RFID stream. Many of these cleaning methods use *declarative methods* in the cleaning process are discussed in [54, 56, 55]. The broad idea is to specify cleaning stages with the use of high-level declarative queries over relational data streams.

In addition, RFID data exhibits a considerable amount of redundancy because of multiple scans of the same item, even when it is stationary at a given location. In practice, one needs to track only interesting movements and activities on the item. The work in [42] proposes methods for reducing this redundancy. RFID tag readings also exhibit a considerable amount of *spatial redundancy* because of scans of the same object from the RFID readers placed in multiple zones. This is primarily because of the spatial overlap in the range of different sensor readers. This provides seemingly inconsistent readings because of the inconsistent (virtual) locations reported by the different sensors scanning the same object. While the redundancy causes inconsistent readings, it also provides useful information about the location of an object in cases, where the intended reader fails to perform its intended function. The work in [28] proposes a Bayesian inference framework, which takes full advantage of the duplicate readings, and the additional background information in order to maximize the accuracy of RFID data collection.

4.2 Semantic Sensor Web

Sensor networks provide the challenge of too much data, and too little inter-operability and also too little knowledge about the ability to use the different resources which are available in real time. The *Sensor Web Enablement* initiative of the *Open Geospatial Consortium* defines service interfaces which enable an interoperable usage of sensor resources by enabling their discovery, access, tasking, eventing and alerting [21]. Such standardized interfaces are very useful, because such a web hides the heterogeneity of the underlying sensor network from the applications that use it. This initiative defines the term *Sensor Web* as an *"infrastructure enabling access to sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application programming interfaces."* This is critical in order to ensure that the low level sensor details become transparent to application programmers, who may now use higher level abstractions in order to write their applications. Clearly, the goal of the sensor web is to enable real time situation awareness in order to ensure timely responses to a wide variety of events. The main services and language suite specifications include the following:

- *Observations and Measurements (O&M):* These are the standard models and schema, which are used to encode the real-time measurements from a sensor.

- *Sensor Model Language (SML):* These models and schema describe sensor systems and processes. These provide the information needed for discovering sensors, locating sensor observations, processing low level sensor observations, and listing taskable properties.

- *Transducer Model Language (TML):* These are standard models and XML schema for describing transducers and supporting real-time streaming of data to and from sensor systems.
- *Sensor Observation Service (SOS):* This is the standard Web service interface for requesting, filtering, and retrieving observations and sensor system information.
- *Sensor Alert Service (SAS):* This is the standard Web service interface for publishing and subscribing to alerts from sensors.
- *Sensor Planning Service (SPS):* This is the standard Web service interface for requesting user-driven acquisitions and observations.
- *Web Notification Services (WNS):* This is the standard Web service interface for delivery of messages or alerts from *Sensor Alert Service* and *Sensor Planing Services.*

We note that all of the above services are useful for different aspects of sensor data processing, and this may be done in different ways based on the underlying scenario. For example, the *discovery* of the appropriate sensors is a critical task for the user, though it is not always easy to know a-priori about the nature of the discovery that a user may request. For example, a user may be interested in discovering physical sensors based on specific criteria such as location, measurement type, semantic meta-information etc., or they may be interested in specific sensor related functionality such as alerting [57]. Either goal may be achieved with an appropriate implementation of the SML module [21, 57]. Thus, the specific design of each module will dictate the functionality which is available in a given infrastructure.

The World Wide Web Consortium (W3C) has also initiated the *Semantic Sensor Networks Incubator Group (SSN-XL)* to develop Semantic Sensor Network Ontologies, which can model sensor devices, processes, systems and observations. This ontology enables expressive representation of sensors, sensor observations, and knowledge of the environment. This is already being adopted widely by the sensor networking community, and has resulted in improved management of sensor data on the Web, involving annotation, integration, publishing, and search. In the case of sensor data, the amounts of data are so large, that the semantic annotation of the underlying data is extremely important in order to enable effective discovery and search of the underlying resources. This annotation can be either spatial, temporal, or may be semantic in nature. Interesting discussions of research issues which arise in the context of the semantic and database management issues of the sensor web may be found in [96, 17].

The semantic web encodes meta-data about the data collected by sensors, in order to make it effectively searchable and usable by the underlying services. This comprises the following primary components:

- The data is encoded with self-describing XML identifiers. This also enables a standard XML parser to parse the data.

- The identifiers are expressed using the *Resource Description Framework (RDF).* RDF encodes the meaning in sets of triples, with each triple being a subject, verb, and object of an element. Each element defines a Uniform Resource Identifier on the Web.

- Ontologies can express relationships between identifiers. For example, one accelerometer sensor, can express the speed in miles per hour, whereas another will express the speed in terms of Kilometers per hour. The ontologies can represent the relationships among these sensors in order to be able to make the appropriate conversion.

We will describe each of these components in the description below.

While the availability of real-time sensor data on a large scale in domains ranging from traffic monitoring to weather forecasting to homeland security to entertainment to disaster response is a reality today, major benefits of such sensor data can only be realized if and only if we have the infrastructure and mechanisms to synthesize, interpret, and apply this data intelligently via automated means. The Semantic Web vision [73] was to make the World Wide Web more intelligent by layering the networked Web content with semantics. The idea was that a semantic layer would enable the realization of automated agents and applications that "understand" or "comprehend" Web content for specific tasks and applications. Similarly the Semantic Sensor Web puts the layer of intelligence and semantics on top of the deluge of data coming from sensors. In simple terms, it is the Semantic Sensor Web that allows automated applications to understand, interpret and reason with basic but critical semantic notions such as "nearby", "far", "soon", "immediately", "dangerously high", "safe", "blocked", or "smooth", when talking about data coming from sensors, and the associated geo-spatial and spatio-temporal reasoning that must accompany it. In summary, it enables true semantic interoperability and integration over sensor data. In this section, we describe multiple aspects of Semantic Sensor Web technology that enables the advancement of sensor data mining applications in a variety of critical domains.

4.2.1 Ontologies. Ontologies are at the heart of any semantic technology, including the Semantic Sensor Web. An ontology, defined formally as a specification of a conceptualization [43], is a mechanism for knowledge sharing and reuse. In this chapter, we will illustrate two important ontologies that are particularly relevant to the sensor data domain. Our aim is to provide an understanding of ontologies and ontological frameworks per se, as well as highlight the utility of existing ontologies for (further) developing practical sensor data applications. Ontologies are essentially knowledge representation systems. Any knowledge representation system must have mechanisms for (i) Representation and (ii) Inference. In this context, we provide a brief introduction to two important Semantic-Web ontology representation formalisms - namely *RDF* and *OWL*.

RDF stands for the "*Resource Description Framework*" and is a language to describe resources [76]. A resource is literally any thing or concept in the world. For instance, it could be a person, a place, a restaurant entree etc. Each resource is uniquely identified by a *URI*, which corresponds to a Unique Resource Identifier. What RDF enables us to do is to:

- Unambiguously describe a concept or a resource.
- Specify how resources are related.
- Do inferencing.

The building blocks of RDF are *triples*, where a triple is a 3-tuple of the form $< subject, predicate, object >$ where subject, predicate and object are interpreted as in a natural language sentence. For instance the triple representation of the sentence "*Washington DC is the capital of the United States*" is illustrated in Figure 12.1.

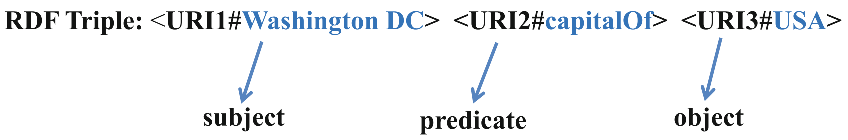

RDF Triple: <URI1#Washington DC> <URI2#capitalOf> <URI3#USA>
subject
predicate
object

Figure 12.1. RDF Triples

The subject and predicate must be resources. This means that they are things or concepts having a URI. The object however can be a resource or a literal (such as the string "USA" or the number "10").

It is most helpful to perceive RDF as a graph, where subject resources are represented in ovals, literals in rectangles, and predicate (relationships) represented as directed edges between ovals or between ovals and rectangles. An example is illustrated in Figure 12.2.

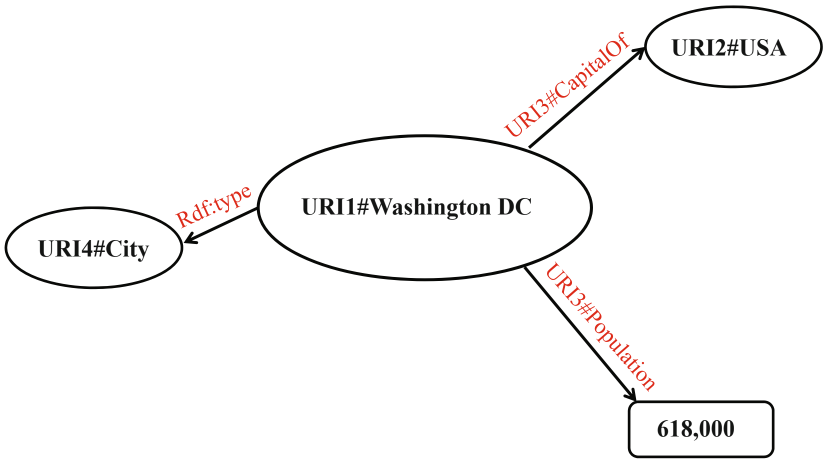


Figure 12.2. RDF as a Graph

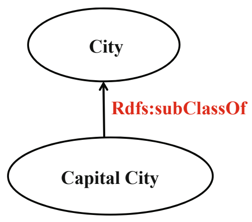
The most popular representation for RDF is RDF/XML. In this case, the RDF is represented in XML format, as illustrated in Figure 12.3, where XML elements are used to capture the fundamental resources and relationships in any RDF triple.

```
<rdf:Descriptionrdf:about="URI1#WashingtonDC">
<rdf:typerdf:resource="URI4#City"/>
<URI2#isCapitalOfrdf:resource="URI3#USA"/>
</rdf:Description>
```

Figure 12.3. RDF XML Representation

RDF(S) stands for RDF *(Schema)* [76]. This can be viewed as a meta-model that is used to define the vocabulary used in an RDF document.

RDF(S) is used for defining classes, properties, hierarchies, collections, reification, documentation and basic entailments for reasoning.



```
<rdfs:Classrdf:ID="someURI#CapitalCity">
            <rdfs:subClassOfrdf:resource="someURI#City"/>
</rdfs:Class>
```

Figure 12.4. RDF Schema

For instance, let us say that we need to define a separate collection of cities that are capital cities of any country. A capital city is of course a sub-class of cities in general. This is represented in RDF(S) as shown in Figure 12.4.

OWL stands for *Web Ontology Language* [76]. This is another ontology formalism that was developed to overcome the challenges with RDF. RDF (and RDF Schema) are limited in that they do not provide ways to represent constraints (such as domain or range constraints). Further, transitive, inverse or closure properties cannot be represented in RDF(S). Extending RDF(s) with the use of standards (XML, RDF etc.,), making it easy to use and understand, and providing a Formal specification is what results in OWL. Both RDF and OWL ontology formats have extensive developer community support in terms of the availability of tools for ontology creation and authoring. An example is *Protege* [101], which supports RDF and OWL formats, data storage and management stores such as *OpenSesame*, for efficient storage and querying of data in RDF or OWL formats. Furthermore, there is significant availability of actual ontologies in a variety of domains in the RDF and OWL formats.

Specific ontologies: We now describe two such ontologies – *SSN* [119] and *SWEET* [92] that are particularly relevant to sensor data semantics. Both these ontologies have been created with the intention of being generic and widely applicable for practical application tasks. SSN is more sensor management centric, whereas SWEET has a particular focus on earth and environmental data (a vast majority of the data collected by sensors). The *Semantic Sensor Network (SSN)* ontology [119] is an OWL ontology developed by the W3C Semantic Sensor Network Incubator group (the SSN-XG) [119] to describe sensors and observations. The SSN ontology can describe sensors in terms of their capabilities, measurement processes, observations and deployments. The SSN ontology development working group (SSN-XG) targeted the SSN ontology development towards four use cases, namely (i) Data discovery and linking, (ii) Device discovery and selection, (iii) Provenance and diagnosis, and (iv) Device operation, tasking and programming. The SSN ontology is aligned with the DOLCE Ultra Lite (DUL) *upper ontology* [39] (an upper ontology is an ontology of more generic, higher level concepts that more specific ontologies can anchor their concepts to) . This has helped to normalize the structure of the ontology to assist its use in conjunction with ontologies or linked data resources developed elsewhere. DUL was chosen as the upper ontology because it is more lightweight than other options, while having an ontological framework and basis. In this case, qualities, regions and object categories are consistent with the group's modeling of SSN. The SSN ontology itself, is organized, conceptually but not physically, into ten modules as shown in Figure 12.5. The SSN ontology is built around a central Ontology Design Pattern (ODP) describing the relationships between sensors, stimulus, and observations, the Stimulus-Sensor- Observation (SSO) pattern. The ontology can be seen from four main perspectives:

- A sensor perspective, with a focus on what senses, how it senses, and what is sensed.
- An observation perspective, with a focus on observation data and related metadata.
- A system perspective, with a focus on systems of sensors and deployments.
- A feature and property perspective, focusing on what senses a particular property or what observations have been made about a property.

The full ontology consists of 41 concepts and 39 object properties, directly inherited from 11 DUL concepts and 14 DUL object properties. The ontology can describe sensors, the accuracy and capabilities of such sensors, observations and methods used for sensing. Concepts for operating and survival ranges are also included, as these are often part of a given specification for a sensor, along with its performance within those ranges. Finally, a structure for field deployments is included to describe deployment lifetimes and sensing purposes of the deployed macro instrument.

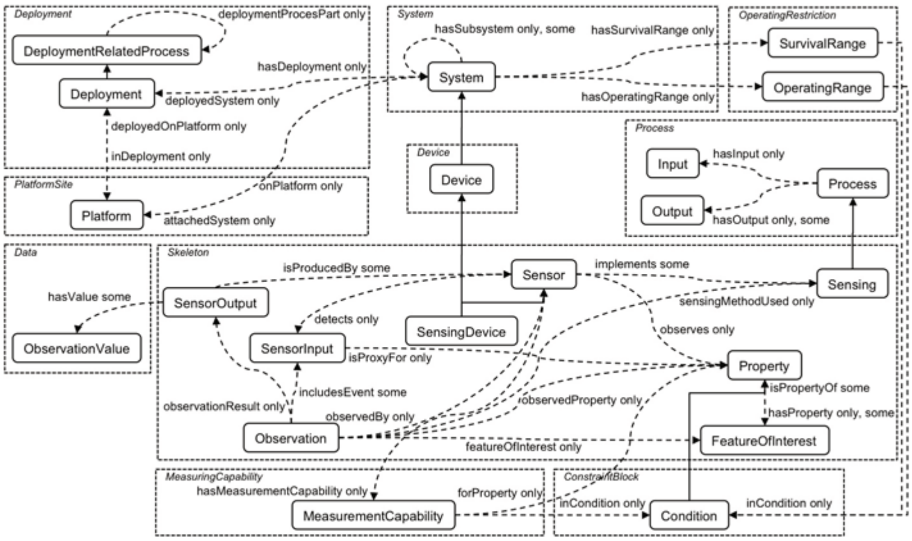
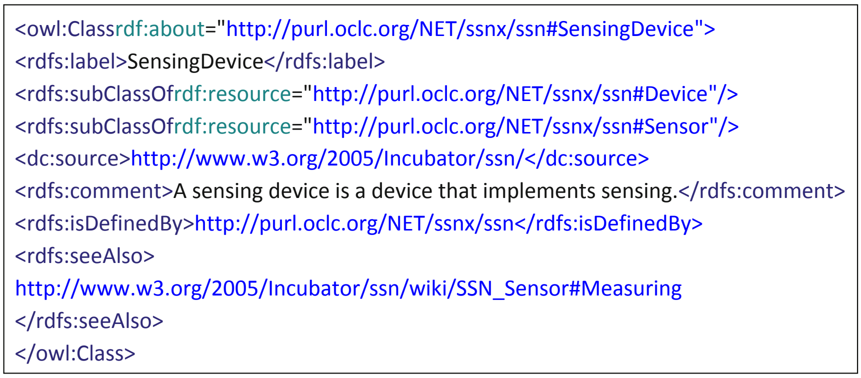


Figure 12.5. The Ten Modules in the SSN Ontology



```
<owl:Classrdf:about="http://purl.oclc.org/NET/ssnx/ssn#SensingDevice">
<rdfs:label>SensingDevice</rdfs:label>
<rdfs:subClassOfrdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Device"/>
<rdfs:subClassOfrdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Sensor"/>
<dc:source>http://www.w3.org/2005/Incubator/ssn/</dc:source>
<rdfs:comment>A sensing device is a device that implements sensing.</rdfs:comment>
<rdfs:isDefinedBy>http://purl.oclc.org/NET/ssnx/ssn</rdfs:isDefinedBy>
<rdfs:seeAlso>
http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Sensor#Measuring
</rdfs:seeAlso>
</owl:Class>
```

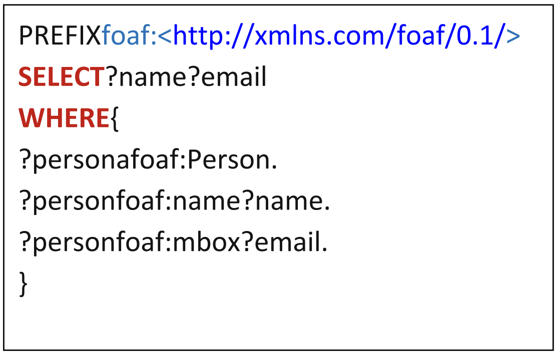
Figure 12.6. Schema for the Sensor Class

SWEET: The motivation for developing *SWEET* (*The Semantic Web of Earth and Environmental Terminology*) stemmed from the realization of making vast amounts of earth science related sensor data collected continuously by NASA more understandable and useful [92]. This effort resulted in a) a collection of ontologies for describing Earth science data and knowledge, and b) an ontology-aided search tool to demonstrate the use of these ontologies. The set of keywords in the NASA Global Change Master Directory (GCMD) (Global Change Master Directory, 2003) form the starting point for the SWEET ontology. This collection includes both controlled and uncontrolled keywords. The controlled keywords include approximately 1000 Earth science terms represented in a subject taxonomy. Several hundred additional controlled keywords are defined for ancillary support, such as: instruments, data centers, missions, etc. The controlled keywords are represented as a taxonomy. The uncontrolled keywords consist of 20,000 terms submitted by data providers. These terms tend to be more general than or synonymous with the controlled terms. Examples of frequently submitted terms include: climatology, remote sensing, EOSDIS, statistics, marine, geology, vegetation, etc.

Some of the SWEET ontologies represent the Earth realm and phenomena and/or physical aspects and phenomena. These include the "Earth Realm" ontology which has elements related to "atmosphere", "ocean" etc., Physical aspects ontologies represent things like substances, living elements and physical properties. However the ontologies most relevant to sensor data are those representing (i) Units, (ii) Numerical entities, (iii) Temporal entities, (iv) Spatial entities, and (v) Phenomena.

4.2.2 Query Languages. While RDF, OWL and other formalisms serve the purpose of data and knowledge representation, one also needs a mechanism for querying any data and knowledge stored. SPARQL (SPARQL Protocol and RDF Query Language) [88] is an RDF query language for querying and manipulating data stored in the RDF format. SPARQL allows writing queries over data as perceived as triples. It allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns. SPARQL closely follows SQL syntax. As a result, its query processing mechanisms are able to inherit from standard database query processing techniques. A simple example of an SPARQL query, which returns the name and email of every person in a data set is provided in Figure 12.7. Significantly, this query can be distributed to multiple SPARQL endpoints for computation, gathering and generation of results. This is referred to as a *Federated Query*.

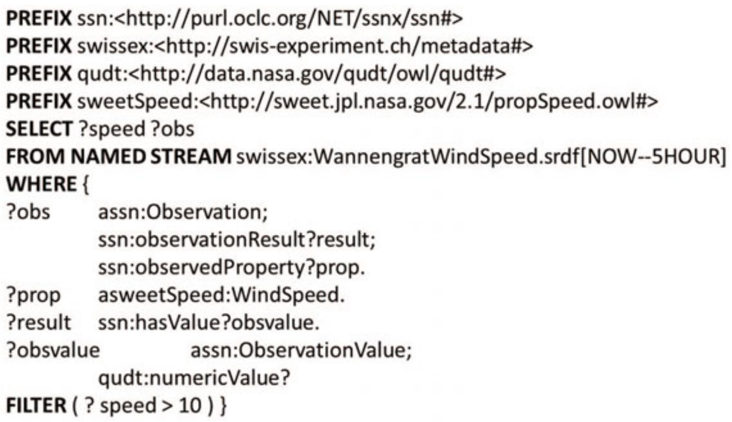
SPARQLstream [89] is an extension of SPARQL that facilitates querying over RDF streams. This is particularly valuable in the context of *sensor data*, which is generally stream-based. An RDF stream is defined as a sequence of pairs (T_i, i) where T_i is an RDF triple $< h_{si}; p_i; o_{ii} >$ and i is a time-stamp which comes from a monotonically non-decreasing sequence.



```
PREFIXfoaf:<http://xmlns.com/foaf/0.1/>
SELECT?name?email
WHERE{
?personafoaf:Person.
?personfoaf:name?name.
?personfoaf:mbox?email.
}
```

Figure 12.7. Simple SPARQL Example

An RDF stream is identified by an IRI, which provides the location of the data source. An example SPARQL stream query is provided in Figure 12.8 whichillustrates a query that obtains all wind-speed observation values greater than some threshold (e.g., 10) in the last 5 hours, from the *sensors* virtual rdf stream **swissex:WannengratWindSensors.srdf**.



```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX swissex:<http://swis-experiment.ch/metadata#>
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#>
PREFIX sweetSpeed:<http://sweet.jpl.nasa.gov/2.1/propSpeed.owl#>
SELECT ?speed ?obs
FROM NAMED STREAM swissex:WannengratWindSpeed.srdf[NOW--5HOUR]
WHERE {
?obs        assn:Observation;
            ssn:observationResult?result;
            ssn:observedProperty?prop.
?prop       asweetSpeed:WindSpeed.
?result     ssn:hasValue?obsvalue.
?obsvalue               assn:ObservationValue;
            qudt:numericValue?
FILTER ( ? speed > 10 ) }
```

Figure 12.8. SPARQL Stream Example

4.2.3 Linked Data. Realization of the Semantic-Web vision has indeed faced challenges on multiple fronts, some impediments including having to define and develop ontologies that domain experts and representatives can agree upon, ensuring that data on the Web is indeed marked up in semantic formats, etc. The *Linked Data* vision

[109] is a more recent initiative that can perhaps be described as a "light-weight" Semantic Web. In a nutshell, Linked Data describes a paradigm shift from a Web of linked documents towards a Web of linked data. Flexible, minimalistic, and local vocabularies are required to interlink single, context-specific data fragments on the Web. In conjunction with ontologies, such raw data can be combined and reused on-the-fly. In comparison to SDIs, the Linked Data paradigm is relatively simple and, therefore, can help to open up SDIs to casual users. Within the last years, Linked Data has become the most promising vision for the Future Internet and has been widely adopted by academia and industry. The Linking Open Data cloud diagram provides a good and up-to-date overview. Some of the foundational work for taking sensor data to the Linked Data paradigm has been in the context of Digital Earth [109], which calls for more dynamic information systems, new sources of information, and stronger capabilities for their integration. Sensor networks have been identified as a major information source for the Digital Earth, while Semantic Web technologies have been proposed to facilitate integration. So far, sensor data is stored and published using the Observations and Measurements standard of the Open Geospatial Consortium (OGC) as data model. With the advent of Volunteered Geographic Information and the Semantic Sensor Web, work on an ontological model gained importance within Sensor Web Enablement. In contrast to data models, an ontological approach abstracts from implementation details by focusing on modeling the physical world from the perspective of a particular domain. Ontologies restrict the interpretation of vocabularies towards their intended meaning. The ongoing paradigm shift towards Linked Sensor Data complements this attempt. Two questions need to be addressed:

- How to refer to changing and frequently updated data sets using Uniform Resource Identifiers.

- How to establish meaningful links between those data sets, i.e., observations, sensors, features of interest, and observed properties?

The work in [109] presents a Linked Data model and a RESTful proxy for OGC's Sensor Observation Service to improve integration and interlinkage of observation data for the Digital Earth.

In summary, today with the existence of practical and real-world sensor domain ontologies (such as SSN and SWEET), RDF storage and streaming query language mechanisms, and the availability of linked sensor data - we are in a position to use such infrastructure for building practical sensor data mining applications.

4.3 Semantic Web Data Management

One of the most challenging aspects of RDF data management is that they are represented in the form of *triples* which conceptually represents a graph structure of a particular type. The conventional method to represent RDF data is in the form of triple stores. In these cases, giant triples tables are used in order to represent the underlying RDF data [11, 12, 18, 24, 49, 50, 85, 113, 114]. In these systems, the RDF data is decomposed into a large number of statements or triples that are stored in conventional relational tables, or hash tables. Such systems can effectively support statement-based queries, in which the query is missing some parts of the triple, and these parts are then provided by the response. On the other hand, many queries cannot be answered from a single property table, but from multiple property tables. One major problem with such solutions is that because a relational structure is imposed on inherently structured data, it results in sparse tables with many null values. This causes numerous scalability challenges, because of the computational overhead in processing such sparse tables.

A natural solution is to index the RDF data directly as a graph. This has the virtue of recognizing the inherently structured nature of the data for storage and processing [13, 20, 53, 103]. A number of graph-based methods also use the measurement of similarity within the Semantic Web [67], and selectivity estimation techniques for query optimization of RDF data [97]. Many of these techniques require combinatorial graph exploration techniques with main memory operations necessitated by the random storage access inherent in graph analytics. Such approaches can doom the scalability of RDF management. Other methods use path-based techniques [68, 75] for storing and retrieving RDF data. These methods essentially store subgraphs into relational tables. As discussed earlier, approaches which are based on relational data have fundamental limitations which cannot be addressed by these methods.

A different approach is to use multiple indexing approaches [51, 116] in which information about the context is added to the triple. Thus, we now have a *quad* instead of a *triple* which has $2^4 = 16$ possible access patterns. The work in [51] creates six indexes which cover all these 16 access patterns. Thus, a query, which contains any subset of these variables can be easily satisfied with this approach. These methods are also designed for statement-based queries, and do not provide efficient support for more complex queries.

4.3.1 Vertical Partitioning Approach. A fundamental paradigm shift in the management of RDF data is with the use of a vertical partitioning approach [1]. This is closely related to the development of column-oriented databases for sensor data management [98, 2, 3]. Consider a situation in which we have m different properties in the data. In such a case, a total of m two-column tables are created. Each table contains a subject and object column, and if a subject is related to multiple objects, this corresponds to the different rows in the table. The tables may be stored by subject, and this can enable quick location of a specific subject. Furthermore, each table is sorted by subject, so that particular subjects can be located quickly, and fast merge-joins can be used to reconstruct information about multiple properties for subsets of subjects. This approach is combined with a column-oriented database system [98] in order to achieve better compression and performance. In addition, the object columns of the scheme can be indexed with the use of a B^+-Tree or any other index. It was argued in [110] that the scheme in [1] is also not particularly effective, unless the properties appear as bound variables.

It was observed in [110] that while the work in [1] argued against conventional property-table solutions, their solution turned out to be a special variation of property tables, and therefore share all its disadvantages. The two-column tables of [1] are similar to the multi-valued property tables introduced in [113], and the real novelty of the work in [1] was to integrate the column-oriented database systems into two-column property tables. Therefore, the work in [110] combines a multiple-indexing scheme with the vertical partitioning approach proposed in [1] in order to obtain more effective results. The use of multiple indexes has tremendous potential to be extremely effective for semantic web management, because of its simultaneous exploitations of different access patterns, while incorporating the virtues of a vertical approach. Multiple index-based techniques have also been used successfully for a variety of other database applications such as join processing [15, 79, 80].

4.4 Real-time and Big Data Analytics for The Internet of Things

Since RFID and conventional sensors form the backbone of the data collection mechanisms in the internet of things, the volume of the data collected is likely to be extremely large. We note that this large size is not just because of the streaming nature of the collected data, but also because smart infrastructures typically have a large number of objects simultaneously collecting data and communicating with one another. In many cases, the communications and data transfers between the objects may be required to enable smart analytics. Such communications and transfers may require both bandwidth and energy consumption, which are usually a limited resource in real scenarios. Furthermore, the analytics required for such applications is often real-time, and therefore it requires the design of methods which can provide real-time insights in a distributed way, with communication requirements. Discussions of such techniques for a wide variety of data mining problems can be found in the earlier chapters of thus book, and also in [5].

In addition to the real-time insights, it is desirable to glean *historical* insights from the underlying data. In such cases, the insights may need to be gleaned from massive amounts of archived sensor data. In this context, Google's *MapReduce* framework [33] provides an effective method for analysis of the sensor data, especially when the nature of the computations involve linearly computable statistical functions over the elements of the data streams (such as MIN, MAX, SUM, MEAN etc.). A primer on the *MapReduce* framework implementation on *Apache Hadoop* may be found in [115]. Google's original *MapReduce* framework was designed for analyzing large amounts of web logs, and more specifically deriving such linearly computable statistics from the logs. Sensor data has a number of conceptual similarities to logs, in that they are similarly repetitive, and the typical statistical computations which are often performed on sensor data for many applications are linear in nature. Therefore, it is quite natural to use this framework for sensor data analytics.

In order to understand this framework, let us consider the case, when we are trying to determine the maximum temperature in each year, from sensor data recorded over a long period of time. The *Map* and *Reduce* functions of *MapReduce* are defined with respect to data structured in $(key, value)$ pairs. The *Map* function, takes a list of pairs (k_1, v_1) from one domain, returns a list of pairs (k_2, v_2). This computation is typically performed in parallel by dividing the key value pairs across different distributed computers. For example, in our example above consider the case, where the data is in the form of $(year, value)$, where the year is the key. Then, the *Map* function, also returns a list of $(year, local\_max\_value)$ pairs, where $local\_max\_value$ represents the local maximum in the subset of the data processed by that node.

At this point, the *MapReduce* framework collects all pairs with the same key from all lists and groups them together, thus creating one group for each one of the different generated keys. We note that this step requires communication between the different nodes, but the cost of this communication is much lower than moving the *original* data around, because the Map step has already created a compact summary from the data processed within its node. We note that the exact implementation of this step depends upon the particular implementation of *MapReduce* which is used, and exact nature of the distributed data. For example, the data may be distributed over a local cluster of computers (with the use of an implementation such as *Hadoop*), or it may be geographically distributed because the data was originally created at that location, and it is too expensive to move the data around. The latter scenario is much more likely in the *IoT* framework. Nevertheless, the steps for collecting the intermediate results from the different *Map* steps may depend upon the specific implementation and scenario in which the *MapReduce* framework is used.

The *Reduce* function is then applied in parallel to each group, which in turn produces a collection of values in the same domain. Next, we apply $Reduce(k2, list(v2))$ in order to create $list(v3)$. Typically the Reduce calls over the different keys are distributed over the different nodes, and each such call will return one value, though it is possible for the call to return more than one value. In the previous example, the input to *Reduce* will be a list of the form $(Year, [local\_max1, local\_max2, \ldots local\_maxr])$, where the local maximum values are determined by the execution of the different *Map* functions. The *Reduce* function will then determine the maximum value over the corresponding list in each call of the *Reduce* function.

The *MapReduce* framework is very powerful in terms of enabling distributed search and indexing capabilities across the semantic web. An overview paper in this direction [77] explores the various data processing capabilities of *MapReduce* used by $Yahoo!$ for enabling efficient search and indexing. The *MapReduce* framework has also been used for distributed reasoning across the semantic web [104, 105]. The work in [105] addresses the issue of semantic web compression with the use of the *MapReduce* framework. The work is based on the fact that since the number of RDF statements are rapidly increasing over time (because of a corresponding increase in the number of "things"), the compression of these strings would be useful for storage and retrieval. One of the most often used techniques for compressing data is called *dictionary encoding.* It has been experimentally estimated that the statements on the semantic web require about 150–210 bytes. If this text is replaced with 8 byte numbers, the same statement requires only 24 bytes, which is a significant saving. The work in [105] presents methods for performing this compression with the use of the *MapReduce* framework. Methods for computing the closure of the RDF graph with the use of the *MapReduce* framework are proposed in [104].

The *Hadoop* implementation of the *MapReduce* framework is an open source implementation provided by *Apache.* This framework implements a *Hadoop Distributed File System (HDFS)*, which is similar to Google's file system. HDFS provides a distributed file system, in which data is distributed across multiple machines, with some replication, in order to provide resilience to disk failures. The Hadoop framework handles the process of task sub-division, and mapping the *Map* and *Reduce* sub-tasks to the different machines. This process is completely transparent to the programmer, who can focus their attention on building the *Map* and *Reduce* functions. There are two other related big-data technologies which are very useful for data management in the semantic web.

HBase The *HBase* is a database abstraction within the *Hadoop* framework, which is similar to the original *BigTable* system [27, 126]. The *HBase* has column which serves as the key, and is the only index which may be used to retrieve the rows. The data in *HBase* is also stored as $(key, value)$ pairs, where the content in the non-key columns may be considered the values.

Pig The *Pig* implementation builds upon the *Hadoop* framework in order to provide further database-like functionality. A table in *Pig* is a set of tuples, and each field is either a value or a set of tuples. Thus, this framework allows for nested tables, which is a rather powerful abstraction. *Pig* also provides a scripting language [83] called *PigLatin*, which provides all the familiar constructs of SQL such as projections, joins, sorting, grouping etc. Different from SQL, *PigLatin* scripts are *procedural*, and are rather easy for programmers to pick up. The *PigLatin* language provides a higher abstraction level to the *MapReduce* framework, because a query in *PigLatin* can be transformed into a sequence of *MapReduce* jobs.

One interesting aspect of *Pig* is that its data model and transformation language are similar to RDF and the SPARQL query language respectively. Therefore, *Pig* was recently extended [77] to perform RDF querying and transformations. Specifically, *Load* and *Save* functions were defined to convert RDF into Pig's data model, and a complete mapping was created between SPARQL and *PigLatin*.

All of these technologies play a very useful role in crawling storing and analyzing the massive RDF data sets, which are possible and likely in the massive scale involved in the internet of things. In the next subsection, we will discuss some of the ways in which these technologies can be used for search and indexing.

4.5 Crawling and Searching the Internet of Things

The Internet of Things is the beginning of the *data-centric web era*, where the data could be about events, locations or people, as is collected by the sensor infrastructure, and richly described in the form of RDF meta-data. Therefore, it is natural to move to the next stage of *smart semantic web search*, where data and services about arbitrary "things" such as people, events and locations can be easily accessed. Providing such search functionality will be *extremely* challenging, because the size of the semantic web continues to grow rapidly, and is expected to be several *orders of magnitude* larger than the conventional web. This leads to numerous challenging in crawling, indexing and retrieving search results on the semantic web. While the RDF framework solves the *representation issues* for effective search and indexing, the data scalability issue continues to be an enormous challenge. Nevertheless, such a functionality is critical, because search engines can locate the data and services that other applications may need in a M2M world.

Some early frameworks for semantic web search may be found in [44, 72]. Some real implementations of meta-data search engines are *Swoogle* [35, 129] and *Sindice* [102, 127]. Among these different frameworks and implementations, only the last one is recent enough to incorporate the full advantages of the *MapReduce* framework. Generally speaking, since the semantic web is similar to the conventional web in terms of being a linked entity, algorithms which are similar to *PageRank* can be implemented with a *MapReduce* framework for efficient retrieval. The semantic web may require slightly more sophisticated algorithms for indexing, as compared to the conventional web, because of the greater richness in the semantic web in terms of accommodating different types of links. Other tasks such as crawling, are also very similar to the conventional web, in terms of using the linkage structure during the crawling process. Again, some additional intelligence may be incorporated into the crawling process, depending upon the importance of different links and crawling strategies for resource discovery.

A very recent large-scale framework for search and indexing of the web is *Sindice* [102, 127]. We will discuss this engine in more detail, because the high level of scalability, which is incorporated in all aspects of its design choices. In particular, this is achieved with the use of the *MapReduce* framework. The first step is to harvest the web with a crawler called *SindiceBot*, that collects web and RDF documents. This crawler utilizes *Hadoop* in order to distribute the crawling job across multiple machines. An extension to the *Sitemap* protocol [128] allows the data sets to be a described in such a way, that they can be downloaded as a dump, rather than having to download each references URI individually. Nevertheless, the processing of such dumps in order to create indexed RDF representations is computationally intensive. This is achieved with the use of the *MapReduce* framework [127].

In order to create the index, the first step is to process the raw data from *HBase*. The semantics of the raw data are extracted and represented in RDF. At this point, reasoning is applied to fact sets in order to increase the richness of the indexing for query processing purposes. Finally entities are consolidated with appropriate cross-references between the data and its index.

Once the index is created, traditional information retrieval techniques are used in order answer textual and semantic queries over large collections of documents. We note that this phase is relatively efficient, once the index has been materialized, and does not necessarily require the use of the *MapReduce* framework. However, the initial stage of crawling, processing and indexing the data is extremely computationally intensive, and cannot be easily achieved without efficient distributed techniques.

5. Privacy and Security

Privacy and security are an important concern in systems, which are as open as the internet of things. The issues of data privacy may arise both during data collection, and during data transmission and sharing. Privacy in data *collection* issues typically arise because of the widespread use of RFID technology, in which the tags carried by a person may become a unique identifier for that person. Privacy in data *sharing and management* may arise because much of the information being transmitted (eg. GPS location) can be sensitive, but it may also be required (on an *aggregate* basis) to enable useful real-time applications such as traffic analysis. In this section, we will discuss both issues. In addition, a number of security issues also arise involving the access control of the managed data. We will discuss these issues below.

5.1 Privacy in Data Collection

As discussed above, the ability to track the RFID data with covert readers is a significant challenge in the data collection process. We have discussed details of methods for reducing the privacy risks in the data collection process in the chapter on RFID processing in this book [9]. In this section, we will provide an abbreviated discussion about these issues. Once an RFID-based smart object is carried by a user on their person (as would be natural in many applications), the EPC then becomes a unique identifier for that person. The information about object movement can be used either to track the whereabouts of the person, or even for corporate espionage in a product supply chain.

The simplest solution to privacy with RFID data is the use of the *kill* command. The Auto-Id Center designed the "kill" command, which are intended to be executed at the point of sale. The kill command can be triggered by a signal, which explicitly disables the tag [63, 64]. If desired, a short 8-bit password can be included with the "kill" command. The tag is subsequently "dead" and no longer emits the EPC, which is needed to identify it. However, the killing of a tag, was mostly designed for cases where tags were associated with products, which have a limited lifespan (before point of sale) for tracking purposes. This may not work with smart products, where the tags are essential to its functioning over the entire lifetime [40]. Another mechanism is to use a locking and unlocking mechanism for the tags [111], if the data collection from the tag is known to be needed only in specific periods, where the data collection is relatively secure from eavesdropping. This can work in some smart applications, where such periods are known in advance.

More robust solutions are possible with cryptographic methods. For example, it is possible to encrypt the code in a tag before transmission. However, such a solution may not be very effective, because this only protects the *content* of the tag, but not the ability to *uniquely identify* the tag. For example, the encoded tag is itself a kind of meta-tag, which can be used for the purposes of tracking. Another solution is to embed dynamic encryption ability within the tag. Such a solution, however, comes at a cost, because it requires the chip to have the ability to perform such an encryption computation. Therefore, a recent solution [58] avoids this by performing the cryptographic computations at the reader end, and store the resulting information in the tags. This solution of course requires careful modification of the reader-tag protocols. A number of cryptographic protocols for privacy protection of library RFID activity are discussed in [78]. Some of the cryptographic schemes [62, 69, 82] work with *re-writable* memory in the tags in order to increase security. The tags are encrypted, and the reader is able to decrypt them when they send them to the server, in order to determine the unique meta-information in the tag. The reader also has the capability to re-encrypt the tag with a different key and write it to its memory, so that the (encrypted) tag signal for an eavesdropper is different at different times. Such a scheme provides additional protection because of repeated change in the encrypted representation of the tag, and prevents the eavesdropper from uniquely identifying the tag at different times.

An interesting solution for making it difficult to read tags in an unauthorized way is the use of *blocker tags* [59, 60]. Blocker tags exploit the collision properties of RFID transmission, which are inherent in this technology. The key idea is that when two RFID tags transmit distinct signals to a reader at the same time, a broadcast collision occurs, which prevents the reader from deciphering either response. Such collisions are in fact very likely to occur during the normal operation of the RFID infrastructure. In order to handle this issue, RFID readers typically use anti-collision protocols. The purpose of blocker tags is to emit signals (or spam) which can defeat these anti-collision protocols, thereby causing the reader to stall. The idea is that blocker tags should be implemented in a way, that it will only spam unauthorized readers, thereby allowing the authorized readers to behave normally. Details of the blocking approach are discussed in [9].

It was inferred in [111] that the greater threat to privacy arises from the eavesdropping of signals sent from the reader (which can be detected much further away), rather than reading the tag itself (which can be done only at a much closer distance). In fact, the IDs being read by the tree-walking protocol can be inferred merely by listening to the signals being broadcast by the reader. Therefore, it has been proposed in [111] to encrypt the signals being sent by the reader in order to prevent privacy attacks by eavesdropping of *reader* signals.

It is also possible to modify RFID tags to cycle through a set of *pseudonyms* rather than emit a unique serial number [58]. Thus, the tag cycles through a set of k pseudonyms and emits them sequentially. This makes it more difficult for an attacker to identify the tags, because they may only be able to scan different pseudonyms of the tags at different times. Of course, if the attacker is aware of the method being used in order to mask the tag, they may try to scan the tag over a longer period of time, in order to learn all the pseudonyms associated with the tag. This process can be made more difficult for an attacker by increasing the time it takes for the tag to switch from one pseudonym to another.

5.2 Privacy in Data Sharing and Management

Since the functionality of the internet of things is based on the data communication between different entities, and the underlying data may often be person-centric, the ability to provide privacy during the data transmission and sharing process is critical. For example, in a mobile application, the GPS data for a user may be collected exactly, but may not necessarily be shared exactly. A variety of techniques may be used in order to reduce the privacy challenges during data sharing:

- Many applications may require only *aggregate* information collected by the sensors, rather than exact information about individuals. For example, traffic conditions in a vehicular sensing applications can be inferred with the use of aggregate data. Examples of systems which use aggregate data for privacy-preserving queries in smart vehicular sensing environments are discussed in [87].

- A variety of privacy-preservation mechanisms such as k-anonymity, ℓ-diversity, and t-closeness reduce the accuracy of the data before sharing it with other entities [10]. For example, for video data, the faces in the videos can be blurred in order to reduce the likelihood of identification [112]. In the context of mobile and location data, a variety of methods such as spatial cloaking, spatial delays, adding noise to locations etc. [29, 8] are incorporated in order to increase data privacy. A detailed discussion of methods for increasing location privacy are provided in [8].

In practice, it is desirable to set up a set of policies which can allow users to specify which kinds of data they would like to share about themselves. The W3C group has defined the *Platform for Privacy Preferences (P3P)* [125], which provides a language for description of privacy preferences. This allows the user to set specific privacy requirements, and also allows for automatic negotiation between the personal information needs of a user and their privacy preferences.

The issue of privacy has also been addressed in the context of the semantic web [38, 64]. The broad idea in [38] is that users are able to retain control over who has access to their personal information under different conditions. For instance, one may allow their colleagues to access their calender over the weekend, but not over weekdays. In addition, it is desirable to fine tune the granularity of the query responses, depending upon the identity of the person who is performing the queries. A semantic web architecture is proposed in [38], which supports the automated discovery and access of personal resources for a variety of context-aware applications. Each source of contextual information (e.g. a calendar, location tracking functionality, collections of relevant user preferences, organizational databases) is represented as a semantic web service. A semantic e-Wallet acts as a directory of contextual resources for a given user, while enforcing her privacy preferences. Privacy preferences enable users to specify what information can be provided to whom in different contexts. They also allow users to specify *obfuscation rules*, which control the accuracy or inaccuracy of the information provided in response to different queries under different conditions.

5.3 Data Security Issues

Since the data collection nodes in the internet of things spend a lot of time unattended, it opens up the system to a number of security threats. For example, *data integrity* is often a concern, because a malicious adversary can change the data at various stages in the pipeline. In order to address these issues, a number of methods have been designed to password-protect the writing of the memory in the RFID tags or the sensor nodes. A number of solutions for password protection in the context of sensor data are proposed in [4, 70]. For RFID data, this is a greater challenge because the password-protection process requires the use of energy-intensive cryptographic algorithms. This would require an onboard battery (active tag) for enablement, and larger energy consumption requirements are usually undesirable. In this context, a number of methods, which have low energy requirements for these cryptographic solutions in RFID have been proposed recently [26, 37].

The use of RFID technology also has a number of other security concerns. For example, RFID technology is highly dependent on the use of radio signals which are easily jammed. This can open the system to a variety of infrastructure threats, that can disrupt the data collection process. It has recently been demonstrated [19], that RFID tags can be cloned to emit the same identification code as another tag. This opens the system to fraud, when the RFID tag is used for the purpose of sensitive tasks such as payment, authentication or access control. As in the previous case, a number of cryptographic solutions are being proposed to increase the security of RFID technology [19].

A number of security issues also arise in the context of data representations on the semantic web. The data on the semantic web is dynamic and open, which makes it a challenge from a security perspective. Therefore, methods have been proposed for marking up web entities with a semantic policy language, and the use of distributed policy management as a tool for security [63]. The major challenge which is identified with implementing such security policies for the semantic web is the decentralized nature of the semantic web, with a large number of entities, each with its resources, services, agents, users, and their heterogeneity. The work in [63] proposes a distributed policy framework, in which every entity can specify their own policy, since there is no centralized policy. A policy language is proposed, based on RDF-S, in order to markup security information. The policies are specified in terms of *properties* of users, agents, services or resources, rather than *identities*, since full authentication is not possible on the web. A related privacy-preserving ontology framework, based on OWL-S, is proposed in [64].

6. Conclusions

The internet of things is a vision, which is currently being built. It is based on the unique addressability of a large number of objects which may be RFID-based tags, sensors, actuators, or other embedded devices, which can collect and transmit data in an automated way. The massive scale of the internet of things brings a number of corresponding challenges of scale in terms of IP-addressability, privacy, security, and data management and analytics. The internet-of-things has a long data-processing pipeline in terms of collection, storage, and processing, and the decisions made at the earlier stages of the pipeline can significantly impact the processing at later stages. Numerous research choices exist at the different stages of the pipelines, as is clear from the discussion in this chapter. This has lead to a fertile area for research, which is likely to remain of great interest to multiple communities of researchers over the next few years.

References

[1] D. J. Abadi, A. Marcus, S. R. Madden, K. Hollenbach. Scalable Semantic Web Data Management using vertical partitioning. *VLDB Conference*, 2007.

[2] D. J. Abadi, S. R. Madden, M. C. Ferreira. Integrating Compression and Execution in Column Oriented Database Systems. *SIGMOD Conference*, 2006.

[3] D. J. Abadi, D. S. Myers, D. J. DeWitt, S. R. Madden. Materialization Strategies in a Column-Oriented DBMS. *ICDE Conference*, 2007.

[4] R. Acharya, K. Asha. Data integrity and intrusion detection in wireless sensor networks, *Proceedings of the IEEE ICON*, 2008.

[5] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[6] C. C. Aggarwal. On Unifying Privacy and Uncertain Data Models, *ICDE Conference*, 2008.

[7] C. C. Aggarwal. Managing and Mining Uncertain Data, *Springer*, 2009.

[8] C. C. Aggarwal, T. Abdelzaher. Social Sensing. *Managing and Mining Sensor Data*, Springer, 2013.

[9] C. C. Aggarwal, J. Han. A Survey of RFID Data Processing, *Managing and Mining Sensor Data*, Springer, 2013.

[10] C. C. Aggarwal, P. S. Yu. Privacy-Preserving Data Mining, *Springer*, 2008.

[11] S. Alexaki, V. Christophides, G. Karvounarakis, G. Plexsoukis. On Storing Voluminous RDF Descriptions: The Case of Web Portal Catalogs, *WebDB*, 2001.

[12] S. Alexaki, V. Christophides, G. Karvounarakis, G. Plexsoukis, K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, *SemWeb*, 2001.

[13] R. Angles, C. Gutierrez. Querying RDF Data from a Graph Database Perspective, *ESWC*, 2005.

[14] K. Ashton. That 'Internet of Things' Thing. In: *RFID Journal*, 22 July, 2009.

[15] M. Atre, V. Chaoji, M. J. Zaki, J. Hendler. Matrix "Bit" loaded: A Scalable Lightweight Join Query Processor for RDF Data, *WWW Conference*, 2010.

[16] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A Survey, *Computer networks*, 54(16), pp. 2787–2805, 2010.

[17] M. Balazinska et al. Data Management in the World Wide Sensor Web, *Pervasive Computing*, April–June, 2007.

[18] D. Beckett. The Design and Implementation of the Redland RDF Application Framework. *WWW Conference*, 2001.

[19] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, M. Szydylo. Security Analysis of a Cryptographically Enabled RFID Device, *USENIX Security*, 2005.

[20] V. Bonstrom, A. Hinze, H. Schweppe. Storing RDF as a graph. *LA-WEB*, 2003.

[21] A. Broring et al. New Generation Sensor Web Enablement, *Sensors*, 11(3), 2011.

[22] M. Buettner, B. Greenstein, A. Sample, J.R. Smith, D. Wetherall. Revisiting smart dust with RFID sensor networks, *Proceedings of ACM HotNets*, 2008.

[23] C. Bornhovd, T. Lin, S. Haller, J. Schaper. Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-Id Infrastructure, *VLDB Conference*, 2004.

[24] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. *ISWC*, 2002.

[25] V. Bychkovskiy, S. Megerian, D. Estrin, M. A. Potkonjak. collaborative approach to in-place sensor calibration. *IPSN Conference*, 2003.

[26] B. Calmels, S. Canard, M. Girault, H. Sibert. Low-cost cryptography for privacy in RFID systems, *Proceedings of IFIP CARIDS*, 2006.

[27] F. Chang et al. Bigtable: A Distributed Storage System for Structured Data, *OSDI*, 2006.

[28] H. Chen, W.-S. Ku, H. Wang, M.-T. Sun. Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing. *ACM SIGMOD Conference*, 2010.

[29] C.-Y. Chow, M. F. Mokbel. Privacy of Spatial Trajectories. *Computing with Spatial Trajectories*, pp. 109–141, 2011.

[30] I. Constandache, R. Choudhury, I. Rhee. Towards mobile phone localization without war-driving, *INFOCOM Conference*, 2010.

[31] D. Cook, L. Holder. Sensor selection to support practical use of health-monitoring smart environments. *Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery* 1(4): pp. 339–351, 2011.

[32] R. Cyganiak. A Relational Algebra for SPARQL, *HP-Labs Technical Report, HPL-2005-170.* http://www.hpl.hp.com/techreports/2005/HPL-2005-170.html.

[33] J. Dean, S. Ghemawat. MapReduce: A flexible data processing took, *Communication of the ACM*, Vol. 53, pp. 72–77, 2010.

[34] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong. Model-driven data acquisition in sensor networks. *VLDB*, 2004.

[35] L. Ding et al. Swoogle: A Semantic Web and Metadata Search Engine, *ACM CIKM Conference*, 2004.

[36] M. Schmitter-Edgecombe, P. Rashidi, D. Cook, L. Holder. Discovering and Tracking Activities for Assisted Living, *The American Journal of Geriatric Psychiatry*, In Press, 2011.

[37] M. Feldhofer, S. Dominikus, J. Wolkerstorfer. Strong authentication for RFID systems using AES algorithm, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, 2004.

[38] F. Gandon, N. Sadeh. Semantic Web Technologies to Reconcile Privacy and Context Awareness, *Web Semantics: Science, Services and Agents on the Worldwide Web*, 1(3), pp. 241–260, 2004.

[39] A. Gangemi. DOLCE UltraLite OWL Ontology, http://www.loa-cnr.it/ontologies/DUL.owl, 2007

[40] S. L. Garfinkel, A. Juels, R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions, *IEEE Security and Privacy*, 3(3), 2005.

[41] D. Giusto, A. Iera, G. Morabito, L. Atzori (Eds.), The Internet of Things, *Springer*, 2010.

[42] H. Gonzalez, J. Han, X. Li, D. Klabjan. Warehousing and Analyzing Massive RFID Data Sets. *ICDE Conference*, 2006.

[43] T. Gruber. A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 2(5), pp. 199–200, 1993.

[44] R. Guha, R. McCool, E. Miller. Semantic Search, *WWW Conference*, 2003.

[45] D. Guinard, V. Trifa. Towards the Web of Things: Web Mashups for Embedded Devices, *WWW Conference*, 2009.

[46] D. Guinard, V. Trifa, F. Mattern, E. Wilde. From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices, *Architecting the Internet of Things*, Springer, 2011.

[47] A. Gupta, M. Srivasatava. Developing auto-id solutions using sun java system rfid software, October 2004.
http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html

[48] J. Han, J.-G. Lee, H. Gonzalez, X. Li. Mining Massive RFID, Trajectory, and Traffic Data Sets (Tutorial). *ACM KDD Conference*, 2008.
Video of Tutoral Lecture at: http://videolectures.net/kdd08_han_mmrfid/

[49] S. Harris, N. Gibbins. Efficient Bulk RDF Storage. *PSSS*, 2003.

[50] S. Harris, N. Shadbolt. SPARQL query processing with conventional relational database systems. *SSWS*, 2005.

[51] A. Harth, S. Decker. Optimized index structures for querying RDF from the web. *LA-WEB*, 2005.

[52] O. Hassanzadeh, A. Kementsietsidis. Data Management Issues for the Semantic Web, *ICDE Conference*, 2012.

[53] J. Hayes, C. Gutierrez. Bipartite graphs as intermediate model for RDF. *ISWC*, 2004.

[54] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. A pipelined framework for online cleaning of sensor data streams. *ICDE Conference*, 2006.

[55] S. R. Jeffrey, M. Garofalakis, M. J. Franklin. Adaptive Cleaning for RFID Data Streams, *VLDB Conference*, 2006.

[56] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, J. Widom. Declarative Support for RFID Data Cleaning, *Pervasive*, 2006.

[57] S. Jirka, A. Broring, C. Stasch. Discovery Mechanisms for the Sensor Web,*Sensors*, 9, pp. 2661–2681, 2009.

[58] A. Juels. Minimalist Cryptography for RFID Tags. *Conference on Security in Communication Networks*, 2004.

[59] A. Juels, R. Rivest, M. Szydlo. The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy. *ACM Conference on Computer and Communication Security*, pp. 103–111, 2003.

[60] A. Juels, J. Brainard. Soft Blocking: Flexible Blocker Tags on the Cheap, *Workshop on Privacy in the Electronic Society (WPES 04)*, pp. 1–7, 2004.

[61] A. Juels. RFID Security and Privacy: A Research Survey, *IEEE Journal on Selected Areas in Communication*, vol. 24, pp. 381–394, Feb. 2006.

[62] A. Juels, R. Pappu. Squealing Euros: Privacy protection in RFIDenabled banknotes. *Proceedings of Financial Cryptography*, Springer–Verlag, 2003.

[63] L. Kagal, T. Finin, A. Joshi. A Policy-based Approach to Security for the Semantic Web, *ISWC*, 2003.

[64] L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T. Finin, K. Sycara. Authorization and Privacy for Semantic Web Services, *IEEE Intelligent Systems*, 19(4), 2004.

[65] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, V. Terziyan. Smart Semantic Middleware for the Internet of Things, *ICINCO*, 2008.

[66] N. Khoussainova, M. Balazinska, D. Suciu. Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. *Fifth International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06)*, 2006.

[67] C. Kiefer, A. Bernstein, M. Stocker. The fundamentals of iSPARQL – a virtual triple approach for similarity-based Semantic Web tasks. *ISWC*, 2007.

[68] Y. Kim, B. Kim, J. Lee, H. Lim. The path index for query processing on RDF and RDF Schema. *ICACT*, 2005.

[69] S. Kinoshita, F. Hoshino, T. Komuro, A. Fujimura, M. Ohkubo. Low-cost RFID privacy protection scheme. *IPS Journal*, 45(8), 2004.

[70] R. Kumar, E. Kohler, M. Srivastava. Harbor: software-based memory protection for sensor nodes, *IPSN Conference*, 2007.

[71] M. Langheinrich. A Survey of RFID Privacy Approaches. *Personal and Ubiquitous Computing*, Springer, 2008.

[72] Y. Lei, V. Uren, E. Motta. SemSearch: A Search Engine for the Semantic Web, *Managing Knowledge in a World of Networks*, 2006.

[73] T. B. Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientific American*, 2001.

[74] A. Madan, S. Moturu, D. Lazer, A. Pentland. Social Sensing: Obsesity, Healthy Eating and Exercise in Face-to-Face Networks, *Wireless Health*, 2010.

[75] A. Matono, T. Amagasa, M. Yoshikawa, S. Uemura. A path-based relational RDF database. *ADC*, 2005.

[76] E. Miller. An Introduction to the Resource Description Framework, *D-Lib Magazine*, 4(5), 1998.

[77] P. Mika, G. Tummarello. Web semantics in the clouds. *IEEE Intelligent Systems*, 23(5), pp. 82–87, 2008.

[78] D. Molnar, D. Wagner. Privacy and Security in Library RFID: Issues, Practices and Architectures, *CCS*, 2004.

[79] T. Neumann, G. Weikum. Scalable Join Processing on Very Large RDF Graphs, *ACM SIGMOD Conference*, 2009.

[80] T. Neumann, G. Weikum. The RDF-3X Engine for Scalable Management of RDF Data, *VLDB Journal*, 19(1), 2010.

[81] M. Ohkubo, K. Suzuki, S. Kinoshita. RFID Privacy Issues and Technical Challenges, *Communications of the ACM*, 48(9), 2005.

[82] M. Ohkubo, K. Suzuki, S. Kinoshita. A cryptographic approach to "privacy-friendly" tags. *RFID Privacy Workshop*, 2003.

[83] C. Olston, B. Reed, U. Srivastava, R. Kumar. PigLatin: A Not so Foreign Language for Data Processing, *ACM SIGMOD Conference*, 2008.

[84] J. Paek, J. Kim, R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones, *MobiSys*, 2010.

[85] Z. Pan, J. Heflin. DLDB: Extending relational databases to support Semantic Web queries. *PSSS*, 2003.

[86] J. Perez, M. Arenas, C. Gutierrez. Semantics and Complexity of SPARQL, *ISWC*, 2006.

[87] R. A. Popa, H. Balakrishnan, A. Blumberg. VPriv: Protecting Privacy in Location-Based Vehicular Services. *USENIX Security Symposium*, 2008.

[88] E. Prud'hommeaux, A. Seaborne. SPARQL Query Language for RDF (Working Draft), http://www.w3.org/TR/2007/WD-rdf-sparql-query-20070326/, 2007

[89] D. Anicic, P. Fodor, S. Rudolph, N. Stojanovic. EP-SPARQL: a unified language for event processing and stream reasoning, *WWW Conference*, 2011.

[90] B. Quilitz, U. Leser. Querying Distributed RDF Data Sources with SPARQL, *The Semantic Web: Research and Applications*, 2008.

[91] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, M. J. Neely. Energy-delay tradeoffs in smartphone applications, *MobiSys*, 2010.

[92] R. Raskin, M. Pan, Semantic Web for Earth and Environmental Terminology (SWEET, *Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[93] S. E. Sarma, S. A. Weis, D.W. Engels. Radio-frequency identification systems. *CHES*, pp. 454–469, 2002.

[94] S. E. Sarma, S. A. Weis, D.W. Engels. RFID systems, security and privacy implications. *Technical Report MIT–AUTOID–WH–014*, AutoID Center, MIT, 2002.

[95] M. Schmidt, M. Meier, G. Lausen. Foundations of SPARQL Query Optimization, *ICDT Conference*, 2010.

[96] A. Sheth, C. Henson, S. Sahoo. Semantic Sensor Web, *IEEE Internet Computing*, 2008.

[97] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, D. Reynolds. SPARQL basic graph pattern optimization using selectivity estimation. *WWW Conference*, 2008.

[98] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madde, E. O'Neil, P O'Neil, A. Rasin, N. Tran, S. Zdonik. C-Store: A Column-Oriented DBMS, *VLDB Conference*, 2005.

[99] B. Sterling. Shaping Things – Mediawork Pamphlets, *The MIT Press*, 2005.

[100] W. J. Tu, W. Zhou, S. Piramuthu. Identifying RFID-embedded objects in Pervasive Healthcare Applications, *Decision Support Systems*, 46(2), 2009.

[101] T. Tudorache, J. Vendetti, N. F. Noy. Web-Protege: A Lightweight OWL Ontology Editor for the Web, *OWLED*, 2008.

[102] G. Tummarello, R. Delbru, E. Oren. Sindice.com: Weaving the Open Linked Data, *The Semantic Web*, 2007.

[103] O. Udrea, A. Pugliese, V. Subrahmanian. GRIN: A Graph Based RDF Index. *AAAI*, 2007.

[104] J. Urbani, S. Kotoulas, E. Oren, F. van Harmelen. Scalable Distributed Reasoning using MapReduce, *The Semantic Web - ISWC*, 2009.

[105] J. Urbani, J. Maassen, H. Bal. Massive Semantic Web data compression using MapReduce, *HPDC*, 2010.

[106] R. Want. An Introduction to RFID Technology, *IEEE Pervasive*, 2006.

[107] R. Want. Enabling Ubiquitous Sensing with RFID, *Computer*, 37(4), pp. 84–86, 2004.

[108] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, G. Borriello. Building the Internet of Things Using RFID. *IEEE Internet Computing*, 13(3), May–June, 2009.

[109] N. Wiegand, G. Berg-Cross, D. Varanka. Proceedings of the 2011 International Workshop on Spatial Semantics and Ontologies, *SSO* 2011.

[110] C. Weiss, P. Karras, A. Bernstein. Hexastore: Sextuple Indexing for Semantic Web Data Management, *VLDB Conference*, 2008.

[111] S. A. Weis, S. Sarma, R. Rivest, D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. *First International Conference on Security in Pervasive Computing*, 2003.

[112] J. Wickramasuriya, M. Datt, S. Mehrotra, N. Venkatasubramanian. Privacy-protecting data collection in Media Spaces, *ACM Multimedia Conference*, 2004.

[113] K. Wilkinson. Jena property table implementation. *SSWS*, 2006.

[114] K. Wilkinson, C. Sayers, H. A. Kuno, D. Reynolds. Efficient RDF storage and retrieval in Jena2. *SWDB*, 2003.

[115] T. White. Hadoop: The Definitive Guide. *Yahoo! Press*, 2011.

[116] D. Wood, P. Gearon, T. Adams. Kowari: A platform for Semantic Web storage and analysis. *XTech*, 2005.

[117] Z. Zhuang, K.-H. Kim, J. P. Singh. Improving energy efficiency of location sensing on smartphones, *MobiSys*, 2010.

[118] The EPCglobal Architecture Framework, March 2009. http://www.epcglobalinc.org

[119] Semantic Sensor Network XG Final Report, http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/, 2011.

[120] iAnyWhere Solutions Inc Whitepaper: *Manage Data Successfully with RFID Anywhere Edge Processing*, http://www.sybase.com/files/White_Papers/SybaseRFID_edgepro-053107-wp.pdf.

[121] http://seattle.intel-research.net/wisp/

[122] http://www.ipso-alliance.org/

[123] http://www.w3.org/TR/rdf-primer/#rdfmodel

[124] http://www.ted.com/talks/sebastian_thrun_google_s_driverless_car.html

[125] http://www.w3.org/TR/P3P11/

[126] http://hadoop.apache.org/hbase

[127] http://www.sindice.com

[128] http://sw.deri.org/2007/07/sitemapextension

[129] http://swoogle.umbc.edu/