Självständigt arbete på grundnivå

Independent degree project - first cycle

Datateknik

Computer engineering

Quantitative Comparison of SensibleThings and ThingWorx

Yun Yu
Quantitative Comparison of SensibleThings and ThingsWorx
Yun Yu
2016-05-26

MID SWEDEN UNIVERSITY
Department of information and communication systems

Examiner: Ulf Jennehag, ulf.jennehag@miun.se
Supervisor: Stefan Forsström, stefanforsstrom@miun.se
Author: Yun Yu, yuyu1500@student.miu.se
Main field of study: Computer engineering
Semester, year: Spring, 2016
Abstract

Internet of things is a technology innovation which connects thousands of millions of sensors and actuators. Platform is which provides both hardware and software service. There are so many platforms existed in the market. The objective of this study has been to answer the question which IoT platform fits bests for different scenarios by performing a quantitative comparison between SensibleThings and ThingWorx. The study has been conducted with the aid of my supervisor Stefan Forsström and professor TingTing Zhang. The study has shown that SensibleThings and ThingWorx both has its innovation features thus it fits to the different cases according to cases’ need. If the sensor need to update data quicker than 25 times/sec, it is better to choose ThingWorx. If the sensor doesn’t have good hardware it is better to use SensibleThings cause it is lightweight enough. For the case that more than one sensors’ data rush into one point simultaneously, ThingWorx has better performance. In this paper I use a health care system as an example. I think ThingWorx is more suitable for this case regard of its mature service on security and graphical visualization. Finally I presented my future work for platform evaluation.

Keywords: Internet of things, platform, SensibleThings, ThingWorx, evaluation.
Acknowledgments

First of all I would like to thank professor TingTing Zhang who gave me chance to study in MidSweden University and work in this project. What’s more she gave me guidance and direction on my research.

Next, I want to thank my supervisor Stefan Forsström. He gave me a lot of help on this thesis work especially on my thesis writing with great patient.

Last, I would like to thank my family and classmates who gave me support on my life and work.
Table of Contents

Abstract ......................................................................................................................... iii
Acknowledgments ........................................................................................................ iv
Terminology .................................................................................................................. vii

1 Introduction .............................................................................................................. 1
   1.1 Background and problem motivation ............................................................... 1
   1.2 High-level problem statement ...................................................................... 1
   1.3 Detailed problem statement ......................................................................... 1
   1.4 Scope ............................................................................................................... 2
   1.5 Outline ............................................................................................................ 3
   1.6 Contributions .................................................................................................. 3

2 Related Work ........................................................................................................... 4
   2.1 Internet of Things ........................................................................................... 4
   2.2 IoT platforms ................................................................................................ 4
      2.2.1 SensibleThings ....................................................................................... 5
      2.2.2 ThingWorx ........................................................................................... 5
   2.3 Related platform evaluation papers .................................................................. 6
      2.3.1 Proposed by Oleksiy Mazhelis and Pasi Tyrväinen ............................... 7
      2.3.2 Proposed by Bhumi Nakuva and Prof. Tushar Champaneria ............... 7
      2.3.3 Proposed by Hetal B Pandya and Tushar A Champaneria .................. 7
   2.4 Load-testing tool: Tsung ................................................................................... 7
   2.5 Case study: Health care applications ............................................................... 8

3 Methodology ............................................................................................................. 9
   3.1 Overall method for each goal .......................................................................... 9
   3.2 Concrete method for comparison .................................................................... 10

4 Implementation ....................................................................................................... 11
   4.1 Scenarios1: Test at different update rate ....................................................... 12
      4.1.1 ThingWorx .......................................................................................... 12
      4.1.2 SensibleThings ..................................................................................... 13
      4.1.3 Experiment Data Disposition ................................................................ 14
   4.2 Scenarios2: Test on different number of devices .......................................... 14
      4.2.1 SensibleThings ..................................................................................... 14
      4.2.1.1 One Sink node with several source node ....................................... 14
      4.2.1.2 One source node with several sink node ....................................... 15
      4.2.2 ThingWorx ......................................................................................... 15
   4.3 Scenario3: Load-Testing by Tsung ................................................................ 16
   4.4 Theoretic Evaluation works1: Implement an Evaluation Framework. 16
   4.5 Theoretic Evaluation works2: Compare the package and protocol that the platform used for sending data ......................................................... 17
      4.5.1 ThingWorx ......................................................................................... 17
      4.5.2 SensibleThings ..................................................................................... 17
   4.6 Theoretic Evaluation works3: Other Aspects Evaluation ............................. 18
4.7 Specific Case Study: health care system..............................................18

5 Results........................................................................................................20
5.1 Test environment......................................................................................20
5.2 Result for scenario1: Test on different update rate..................................20
5.2.1 SensibleThings performance.................................................................20
5.2.2 ThingWorx performance....................................................................24
5.2.3 Two platforms’ performance comparison.............................................26
5.3 Result for scenario2: Test on different number of devices.......................27
5.4 Result for scenario3: Using load-testing tool: Tsung............................30
5.5 Result for theoretic evaluation works1: Implement an evaluation framework........................................................................................................32
5.6 Result for theoretic evaluation works2: Comparison for the used protocol and packages..................................................................................33
5.6.1 Measurements on SensibleThings.......................................................33
5.6.2 Measurements on ThingWorx..............................................................34
5.6.3 Summary for two platforms’ comparison..............................................35
5.7 Result for theoretic evaluation works3: Other aspects comparison.........35
5.8 Result for Specific Case Evaluation: Health care application..............36

6 Conclusions.................................................................................................38
6.1 Ethical consideration..................................................................................39
6.2 Future work.............................................................................................40

References......................................................................................................41
## Terminology

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgment</td>
</tr>
<tr>
<td>EMS</td>
<td>Edge Micro Server</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>MBaas</td>
<td>Mobile Backend as a service</td>
</tr>
<tr>
<td>Paas</td>
<td>Platform as a service</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SICS</td>
<td>Swedish Institute of Computer Science</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>UCI</td>
<td>Universal Context identifiers</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
</tbody>
</table>
1 Introduction

My project is mainly about two Internet of things platforms. One is developed by MidSweden University, and another is held by PTC company. My aim is to find which platform suits best for different scenarios.

1.1 Background and problem motivation

In recent years, humankinds have witnessed millions of physical things connected with each other together with hundreds of millions of remote sensor data. Internet of things (IoT) will integrate the human beings and smart devices, which leads to highly distributed network via embedded system. Thanks to rapid development in underlying technologies, the internet of things is regarded as the opportunity and challenge in the information industry. IoT has gained much attention from professor and academic program as well as many companies.

The data gathered by millions of connected sensors must be managed securely, reliably and easy to access. A cloud computing and data storage platform is needed to aggregate IoT project with scalability and flexibility. Many commercial companies have built their own IoT platform to grab the opportunity created by IoT. "Everyone’s racing to build the ‘god platform’ for the Internet of Things: the highest, most generalized layer of intelligence and user interface that ties together connected devices and web services"[1]. And the customers have many choices to select. This study investigates an IoT platform’s performance, such as delay, scalability, and how easy to use it like so on.

1.2 High-level problem statement

When you are developing IoT applications, an IoT platform is which that allows you to develop applications, manage data storage and run server over cloud. An IoT platform can provide s set of functions to build an application. It simplify the communication among different types of devices by providing a standard working environment and therefore shorten development time. Most IoT platforms reside over cloud and hold servers for their guests at remote place. It will not only integrate different data entities but also supply numbers of analysis to data. So IoT platforms provide users methods to implement business usage.

There are many types of IoT platforms: centralized, semi-distributed, fully distributed systems[3]. Each type of IoT has its advantages and disadvantages. Therefore, the problem I will solve in this thesis is to determine which IoT platform fits best for different scenarios by performing a quantitative comparison of two platforms.

1.3 Detailed problem statement

The goal of this project is to perform a quantitative comparison of SensibleThings and the ThingWorx platform. This in order to evaluate both systems
to determine when they are most suitable. In particular, the scenario to be evaluated will include a raspberry pi device with a sensor that pushes updates more than 10 times per second. In detail the evaluation should determine how the platforms perform under stress and how it effects the scalability, response times, and overall quality of experience.

From this problem I have set the following goals:

1. Follow the suggested IoT learning path, finish Quick-start kits and read up Thingworx’s documents.

2. Read up all material that provided on the SensibleThings website and read some papers that I can found from Library relevant to SensibleThings.

3. Set up a Raspberry Pi to share sensor values to the different platforms.

4. Set up a client application to retrieve values and perform measurement.

5. Use three theoretic evaluation works measurements on ThingWorx and SensibleThings. In addition, Simulate a scenario that the sensor update data over 10 times per second and simulate another scenario that more than one sensor send data per second. What’s more, use a concrete case to evaluate two platforms.

6. Analysis the end results in terms of scalability and lightweight. Search papers to evaluate two platforms in terms of security and architecture, and then propose future work.

The challenge of this project is that how to make the comparison as fair as possible. Because the two platforms are different types of IoT platform. What’s more, the SensibleThings is running in Sweden, but ThingWorx’s server is in U.S.A. The influence made by objective difference should try to be avoided.

1.4 Scope

In this paper I take a slice of IoT. Since I don't have actual sensor, instead I simulate some data and send them. The study has its focus on comparison between the two IoT platforms’ performance. In other words the survey is not stressed on how the platform implements its capabilities. For this study I have chosen only to focus on Thingworx and SensibleThings, not focus on any other platforms. And more specific, I just used platforms’ interface from their libraries and modified provided example code. In other words I didn’t look deep into the underlying communication code I just invoked some functions in high level layer. What’s more, because my knowledge background as well as time is limited, I
only evaluated platforms in terms of scalability, used protocols and human friendly, not included field like data management.

1.5 Outline
Chapter two describes related work about IoT, IoT platform and some evaluation frameworks. Chapter three presents the methodology I use to achieve my goals. Chapter four provides the detailed design for my project. Chapter five shows the result that I get from experiment. Chapter six presents the conclusion I draw from the results as well as my future work.

1.6 Contributions
I develop an evaluation system integrated several fields evaluation. I think this system can be used to any other platforms’ comparison. What’s more, I develop some scenarios for evaluation which can be applied for other platform’s evaluation.
2 Related Work

This chapter is about the related work in IoT platform, SensibleThings and Thingworx. Besides there are also listed some related work on platform performance evaluation.

2.1 Internet of Things

The concept of the Internet of things originated in the United States at the Massachusetts Institute of Technology (MIT) in 1999 to establish an automatic identification center (Auto-ID Labs). A network radio frequency identification (RFID) system is proposed, which is connected with the Internet through the information sensing devices such as radio frequency identification and so on, to realize intelligent identification and management. “The IoT can be understood as a layer of digital information that covers the physical world” said by Ricardo Murer. It is now playing leading role in various area such as health care, smart home and smart factory. It reduces the time for troubleshooting and making decision thus it will increase in economy of industry[12].

![Figure 2-1: Internet of Things][11]

2.2 IoT platforms

In order to enable a widespread adoption of internet of things, there should be a platform to support development of applications, management of connected devices and services, see Figure 2-2. Instead of requiring companies to focus on the low levels of technology stack which is essential but not value-positive, an IoT platform allows application developer concentrate more on value-add applications.[8] Thus application can be put into market faster, with better service support. A number of IoT platform targeting IoT applications are available in the market. The many platforms found in the market can be categorized into three types: centralized, semi-distributed, fully-distributed systems.[2]
2.2.1 SensibleThings

The SensibleThings is a fully distributed open source architecture, which is aiming to share information fast and ubiquitously. The SensibleThings platform can run on many different devices such as smart phones, Raspberry Pi devices, and desktop computers. In order to enable a widespread proliferation of IoT services there must be a common platform for dissemination of sensor and actuator information on a global scale. The creators of this platform states a lots of requirements from underlying platform. The requirements include: quickly disseminate information to end points; stable and handle devices; lightweight enough; extensive and adaptive; easy to adopt and free to use. The SensibleThings platform is aiming to meet the addressed problems, which is based on MediaSense platform but significantly improved since then. The architecture of SensibleThings can be seen in Figure 2-3. There are five layer designed in this platform: Interface layer; Add-in layer; Dissemination layer; Networking layer; and Sensor& Actuator layer.

2.2.2 ThingWorx

ThingWorx is the first software enablement platform that provides a complete application design, runtime, and intelligence environment, this enables leading
companies to build innovative applications that connect, store, and relate the activities and data from people, systems, and things, including connected machines, devices, sensors, and equipment. ThingWorx is a JAVA application that running in the case of Tomcat JAVA network. It uses the most modern service based architecture. ThingWorx platform software architecture has four major components: server connectivity, communication, data management, application enablement (see Figure 2-4). Shown in Figure 2-5 ThingWorx has several important capabilities to help a developer build an IoT application. They include: the Composer, the Mashup Builder, storage, a search engine, collaboration, and connectivity to things. It has been widely used in smart application such as smart water, smart cities, smart buildings, smart agriculture. It has key features like: Faster deployment, Search-based Intelligence, Allows collaboration.[10]

Figure 2-4: Overview of the ThingWorx platform’s architecture[17]

Figure 2-5: ThingWorx IoT Platform Capabilities[9]

2.3 Related platform evaluation papers
There are various previous and related works to evaluate IoT platforms. Here
are some evaluation frameworks I used to evaluate platforms theoretically.

2.3.1 Proposed by Oleksiy Mazhelis and Pasi Tyrväinen

This evaluation framework is proposed by Oleksiy Mazhelis and Pasi Tyrväinen. An IoT application integrates multiple software elements that are distributed across several nodes and communicate with each other[13]. The IoT platforms need to implement commonly used functionality such as sensor or actuating devices management and web components. As is shown in the Figure 2-6: Oleksiy Mazhelis and Pasi Tyrväinen divided framework into two processes and each process has few elements from application provider viewpoint. Design and implementation process consists of device, gateway, UI devices, and web component. Operations include: fulfillment, assurance and billing[13]. I will compare the two platform through these aspects.

![Support for Design & Implementation Processes]

- Device
- GW
- Web
- UI

![Support for Operations Processes]

- Fulfillment
- Assurance
- Billing

Figure 2-6: IoT platform evaluation framework[13]

2.3.2 Proposed by Bhumi Nakhuva and Prof. Tushar Champaneria

Various IoT platforms nowadays are available that can be used for developing an IoT solution and in Bhumi Nakhuva’s paper it covers eleven popular platforms that are widely used for IoT solution building[10]. She discussed platforms in terms of scalability, availability, security and privacy, plug and play, support for millions of devices, real time data, storage of data, developer friendly, solution type.

2.3.3 Proposed by Hetal B Pandya and Tushar A Champaneria

IoT platform provides a skeleton for developing faster and better software application by providing support through programming scripts and code libraries[12]. There are many IoT platforms such as: IBM Bluemix which is cloud foundry based platform providing Paas and Mbaas capabilities[14]. ThingWorx makes developing faster application for any scenario of connectivity. In Hetal B Pandya and Tushar A Champaneria’ paper they compared many IoT platforms in terms of supportive software&hardware, scalability, features and their application areas.

2.4 Load-testing tool: Tsung

In paper “performance evaluation of an IoT platform” The author Konstantinos Vandikas and Vlasis Tsiatis used load-test tool: Tsung to do evaluation on a computational platform that they have designed and built in collaboration with the Swedish Institute of Computer Science (SICS) and Uppsala University. The
main goal of this evaluation is to better understand how the IoT-Framework behaves under heavy load with different amounts of producers and consumers[15]. They conducted two sets of tests using Tsung scripts. First set used a varying number of producers and no consumers utilizing; while the second set introduces a varying number of consumers[15].

2.5 Case study: Health care applications

Nicola Bui and Michele Zorzi are the authors for this case study. This paper focuses on how the IoT can be a suitable framework for e-health communication and especially how low-cost and low-power devices can enhance the quality of life of people suffering from chronic diseases or during emergencies[6]. Their investigation took a look into a real scene of a man. Thus, they came up with the main communication requirements for those processes.
3 Methodology

In this chapter I will state my methodology from two parts, sub-chapter 3.1 is about the overall method for each goal, chapter 3.2 is the concrete method on how I will do my comparison work.

3.1 Overall method for each goal

For first goal, I want to figure out how to use Thingworx platform. I will achieve this goal by following the learning path. I read up all the listed material, watch up the provided video and build up my simple weather application step by step following tutorial. Besides I will finish Devices QuickStart and Application QuickStart Kits so that I have general idea on how to build an application and bind data on ThingWorx platform.

Meanwhile I will complete my second goal: study on how to use SensibleThings platform. I will achieve this goal by searching on SensibleThings website, reading all the posted documents and downloading their example code. Besides I read up some papers that I can found about SensibleThings.

After figuring out how do the two platforms work, I will set about to finish my third goal: set up a raspberry pi to share sensor values to the different platforms. The raspberry pi has already been set up and I use Bitvise SSH to connect to my pi.

For fourth goal to set up a client application to retrieve values and perform measurements, I will modify some example code and use platforms' library to build my own program.

Next goal is to evaluate two platforms using three theoretic evaluation works and simulate a scenario that the sensor update data over 10 times per second and simulate another scenario that more than one sensor send data per second. For accomplish five evaluation works, I will mainly by searching on the internet and reading relevant paper. By changing the parameters of my client application which I accomplished in fourth goal, I can finish two scenarios and also get some metric data such as: Round-trip time, CPU usage...for each platform which is the raw data for goal 6. Last I will use a concrete case study to evaluate two platforms.

The last goal is to analyze the test data and do evaluation on two platforms. I will use some evaluation frameworks to compare two platforms from many aspects by discovering their paper. Then I will use the data I got from goal 5 and draw some graphs to illustrate the results. Last I will get some conclusions and propose future work.
3.2 **Concrete method for comparison**

My task is to figure out which platform fits best for different scenarios by performing a quantitative comparison of two platforms. In order to measure two platforms comprehensively, I plan to evaluate them from three main aspects: first file quantified experiment test; second file is theoretically discussion about two platforms’ architecture and many other aspects; Last evaluation is to use a concrete case to show each platform’s advantage and disadvantage and determine which platform is more suitable for this case.

For quantified experiment test part, I will use two metric to measure two platforms. I use round-trip time to measure two platforms’ scalability and I use CPU usage to see its lightweight. After completing my test, I will get many RTT at different situation. I will draw three kinds of figures, first I use sequential number as x-axis and RTT as y-axis to see each platforms performance. For another kind of graph I use different rate as x-axis and RTT/CPU usage as y-axis. Last I will integrate two platforms’ test data into one graph to see the comparison between them. By using load test Tsung, I will definitely get some graphs and I can draw some conclusions from them.

For theoretic evaluation part, I found some papers are about platforms’ evaluation, I chose three of them as my comparison’s foundation which is aforementioned in chapter 2.

Not only abstract comparing two platforms’ technique aspects, I will also use a specific case to illustrate each platforms’ features. The case I chose is a health care system. And in order to find out what’s the system’s requirement is I use an assumed scenario to illustrate it.
4 Implementation

In order to compare two platforms comprehensively, I draw the following implementation architecture, see Figure 4-1.

![Figure 4-1: overall implementation design](image)

From Figure 4-1, I will measure two platforms from three fields: theoretic, experiment and specific case evaluation which I called for “Evaluation systems”. It lists three theoretic evaluation works and three scenarios as well as a specific case study to do evaluation. The evaluation system uses two platforms as input and through my evaluation system it will get evaluation result as output. This evaluation system can be implemented to any other two platforms for their performance evaluation. In my project, these two platforms are ThingWorx and SensibleThings. I put platform 1 and platform 2 into some evaluation scenarios. After some cases comparison, I will get an output as an evaluation result. Currently I only set up these situations to do evaluation work, but there can be more scenarios added to this framework by future work. So the following chapters is the details about these evaluation works.

Chapter 4.1 is about a scenario that I record the round-trip time and CPU usage at different update rate. The scenario 2: Test on different number of devices will be represented in detail in chapter 4.2. Chapter 4.3 is about a scenario using a load testing tool. And chapter 4.4 is about theoretic evaluation works1, implementing an evaluation framework I found from previous paper. While chapter 4.5 is in order to compare platforms’ protocols that these two platforms use. Chapter 4.6 is about other aspects that isn’t included in the previous evaluation but I found in other papers. For chapter 4.7, there will be a description on an e-healthcare system case study to discuss which platform is more suitable for this case.
4.1 Scenarios1: Test at different update rate

In order to evaluate the scalability of two platforms, I will change the data update rate to see each platform’s performance and whether each platform works well under great pressure. Figure 4-2 is an overall design for this test scenario.

![Figure 4-2: overall design for scenario1](image)

4.1.1 ThingWorx

First I would like to introduce EMS(edge micro server). This server is only a Java program running on my devices. Since some sensors have limitation in hardware, so it first need to send data to EMS, and then through EMS the data will be sent to cloud server. But in my case, there is no sensor connecting to my EMS, and my EMS generates random data, so the EMS is acting like a sensor in my project.

See Figure 4-3 to get acknowledge of ThingWorx deployment for this scenario. For ThingWorx, I create a unique application key for EMS(edge micro server) using ThingWorx’s composer, which is needed when EMS want to transmit data to my cloud server. I also set up an entity instance for this simulated sensor(EMS) at the server side. The properties and value type defined in the entity is corresponding to the properties which are hard coded in the EMS. At the EMS side, there is a variable named "POLLING_INTERVAL_MILLISECONDS". I can change update speed by changing the value of this variable. The EMS consists of three Java class: "Main" class for read arguments from console and create an instance for class "BaseEdgeServer"; "BaseEdgeServer" is responsible for sending data periodically and create class "TempAndHumidity"; class "TempAndHumidity" encapsulates two properties and get these two properties value from sensors, in my project, these value are randomly generated.

![Figure 4-3: scenario deployment structure for ThingWorx](image)

The speeds I chose are: 5second/time, 1second/time, 0.8second/time, 0.5second/time, 0.4second/time, 0.2second/time, 0.1second/time, 0.08second/time, 0.05second/time, 0.04second/time, 0.02second/time,
0.01second/time, 0.008second/time, 0.005second/time, 0.004second/time, 0.002second/time, 0.001second/time.

I can see debug message from the Figure 4-4 that, for each data there will print out a timestamp when the data is sent and will print out another timestamp when received response attached with a RequestId. In order to use these debug messages, I write a Java program to pick up these timestamps and use these two timestamp to calculate and get round-trip time. At same time I will record the CPU usage when the EMS is sending data periodically.

I will get RTT and CPU usage from this test and I can draw some graphs to see ThingWorx’s performance at different speed.

![Figure 4-4: Debug message printed for ThingWorx](image)

### 4.1.2 SensibleThings

Figure 4-5 is a workflow structure for SensibleThings. At source node side, I set a timer called "POLLING_INTERVAL_MILLISECONDS", it will control the update speed for sending data. Once the source node is triggered by "GET" it will send data at set speed and attach the current timestamp as payload. When sink node get response, it will send this response back to source node. After source node get response from sink node, it will get a timestamp from response’s payload and compare with current time. The round-trip time is current time minus the previous timestamp. I set the source node to send data 1000 times in total at different speed.

![Figure 4-5: scenario workflow structure for SensibleThings](image)

I designed the Source Node update data at speed: 1second/time, 0.5second/time, 0.4second/time, 0.2second/time, 0.1second/time, 0.08second/time, 0.05second/time, 0.04second/time, 0.025second/time, 0.02second/time,
0.01 second/time. And 1000 times for each speed in total. Dare not to damage the system, I chose 100 times/second as fastest test speed.

I will get 1000 RTTs for each speed and I will draw some graphs to see its performance. Meanwhile I can also get CPU usage for different speed.

4.1.3 Experiment Data Disposition

I use sequential number as x-axis and RTT as y-axis to see each platforms performance as time went by. For another kind of graph I use different rate as x-axis and RTT/CPU usage as y-axis. Last I will integrate two platforms’ test data into one graph to see the comparison between them.

4.2 Scenarios2: Test on different number of devices

The overall design for this scenario can be seen from Figure 4-6. The model I used is try to send data to one node at several source node simultaneously. But there will be different deployment for ThingWorx and SensibleThings. Because sensor can only "push" data to server and server can’t "pull" data from sensor on ThingWorx. While on SensibleThings the sink node can use "GET" function to trigger source node sending data towards sink node. In order to avoid other factors influence the result, the update speed will keep at 10 times/second in this scenario test.

4.2.1 SensibleThings

There are two kinds of situations for SensibleThings in this scenario. So chapter 4.2.1.1 will present minutely in first situation: one sink node with several source nodes. While chapter 4.2.1.2 will present another situation: one source node with several sink nodes.

4.2.1.1 One Sink node with several source node

As is shown in Figure 4-7, in this scenario I deploy one sink node and try to retrieve data from several different source node simultaneously. The source node used the same code in previous scenario test but with different UCI. And I achieve sink node by invoking "resolve" function with sources’ UCI many times. The sink node is running on my laptop, and the source node is running on
Quantitative Comparison of SensibleThings and ThingsWorx
Yun Yu

raspberry Pi. The number of source node is increased from 1 to 7. For each test I will record the RTT for one source node by the same way as in the scenario 1 used.

4.2.1.2 One source node with several sink node

For another scenario as shown in Figure 4-8, I will deploy one source node and several sink nodes. These sink nodes want to get data from this only one source node at same time. The number of sink node will increased from 1 to 7. For each different number of sink node I will record the RTT from source node side as I do in previous test. After I have done these tests, I can draw some graph using this test data.

In scenario 1, I set sink node to use function “GET” once and when the Source node is triggered by “GET” function it will send data at a certain speed. But if there are several sink node try to get data at same time, the source node can only send data to one sink node randomly each time, so it will not be a successful load test for source node. Instead I will set sink node to use function “GET” at 10 times/second, and once the source node is triggered by “GET” it will send out data only once. For example, if I use previous way to do test and there are five sink node, the source node update speed is 10 times/second and each sink node will get response less than 10 times/second. But for second method, the source node will update data at 50 times/second and each sink node can get response at 10 times/second.

4.2.2 ThingWorx

For ThingWorx, I set several EMS sending simulated data to server at 10 times/second simultaneously and record each test’s RTT. Figure 4-9 is a sketch map for this deployment. The number of EMS will increase from 1 to 7. I use same application key and same Java code to run EMS as I build in scenario 1.
4.3 Scenario3: Load-Testing by Tsung

From paper "performance evaluation of an IoT platform", I found that Tsung can be used for load test. I download and install the Tsung following Tsung document. A lot of Tsung script are available on github and stackoverflow, so next I will configure my tsung.xml file refer to these script. The test create a total number of 10000 request using Tsung scripts "<client host="localhost" use_controller_vm="true" maxusers="10000"/>" and the client’s speed is set to 1 request/second using scripts "<users maxnumber="1" arrivalrate=1" unit="second"/>".

For ThingWorx I set server host to be "52.48.69.37:80". Because I was behind a firewall, so I was poxied via the bootstrap. So the test scripts’ for SensibleThings is set to be "193.10.119.42:14525". After finishing configuring the tsung.xml I will get some test result for each platform.

4.4 Theoretic Evaluation works1: Implement an Evaluation Framework

An IoT platform can integrate multiple software components. As is shown from the Figure 4-10, the evaluation framework will measure platform from two points: design&implementation process and operations processes. The design&implementation process include these elements: sensing or actuating device, a gateway, a user interface and a web component responsible for executing the application data storage and service. Operation processes include: fulfillment is the configuration and activation of application software, proactive and reactive assurance and billing of services and applications[12].

But not all these listed elements are required to make up a platform. So there can be some missing information for ThingWorx and SensibleThings measurements.

![Figure 4-10: structure for an evaluation framework](image-url)
4.5 **Theoretic Evaluation works**: Compare the package and protocol that the platform used for sending data

In order to figure out how many packages each platform need and which protocol is used for communication, I use wireshark to capture packages and analyze them. By comparison the protocol and package, I can see which platform is more safer and which one is at less message cost.

4.5.1 **ThingWorx**

I draw a Figure 4-11 about how does ThingWorx work by observing the package that captured by wireshark. As is shown in the Figure 4-11, I measure the platform from two directions: Edge server to cloud server and cloud server to data consumers.

![Figure 4-11: ThingWorx communication workflow](image)

4.5.2 **SensibleThings**

For SensibleThings I used the Figure 4-12 which I found in the slide tutorial. The whole process includes four phases: resolve, get or set and notify. So I will measure SensibleThings from these four aspects.

First I will run a sink node which resolve an UCI periodically at low speed and capture all the packages and record the number of packages. Next I will run a sink node which resolve a source node’s UCI and invokes “GET” function periodically, then I capture all packages by wireshark again. Next I modify the code to let sink node invoke “Set” instead “Get” periodically. Last I run a sink node to resolve a source node and only use “GET” function once(not use periodically). Once the source is triggered by this sink node, it will send data periodically at low speed. I can observe the package for one process from both source node side and sink node side. Thus I can figure out the packages and protocols usage in processes: resolve, get, set and notify.
In addition, there are some other aspects which do not look deep into technology details to evaluate two platforms such as: programming language, platform support, application area, special feature, developer friendly and security. By searching on the internet and reading two platforms’ paper I will generate a table to compare them in these different aspects. After done measurements I will full in the Table 4-1 as my evaluation result.

<table>
<thead>
<tr>
<th>platform</th>
<th>programming language</th>
<th>platform support</th>
<th>application area</th>
<th>special feature</th>
<th>developer friendly</th>
<th>security</th>
</tr>
</thead>
<tbody>
<tr>
<td>SensibleThings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ThingWorx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-1: designed table for some aspects measurements

**4.7 Specific Case Study: health care system**

IoT platform has been widely used to develop many kinds of applications, some of which have special requirements. So I will study on a health care system and find out what kind of support it needed. Using the data I have already gotten from previous scenario tests, I can judge that which platform suits this health system well.

In order to understand the practical needs for this system, considering this scenario[6], a man who has diabetes, got a blood sugar sensor to continuously monitor his health condition along with other sensors such as heart beat rate and breathing rate. On one occasion, he ate a piece of cake without doctor’s permission. After few minutes the sensor noticed his blood sugar increased out of normal range and immediately sent out an alert to talk to him to inject himself insulin within the next 30 minutes. What’s more his doctor can check his blood sugar levels. On another occasion this man slipped on the stairway and hurt his leg. His two sensor agreed that he was in danger and released an emergency message along with his location data to emergency centre. The nearest emergency centre sent ambulance to this man and when ambulance took this man back to hospital his doctor has got the man’s information beforehand.
In the aforementioned scenarios, the IoT has to meet these requirements: Interoperability, several types of sensor or devices need to cooperate with each other; bounded latency and reliability, can act immediately with little delay in emergency case; retrieve data on demand, the doctor can query interested data; alert management, when some data is out of normal range it can send alert to its user; last privacy, authentication are essential when sensitive data are exchange across network.
5 Results

This chapter will present the test result I got from previous tests which have already been explained minutely in chapter 4. This chapter is organized as follows: chapter 5.1 states my test environment; chapter 5.2, chapter 5.3 and chapter 5.4 present the result for scenario 1, 2 and 3; the rest of the chapter is about result for each evaluation works; the last part will present the concrete case study.

5.1 Test environment

I use my laptop and raspberry Pi to do experiments. Hardware-wise, my laptop is an Intel Core i5 at 1.80GHz with 4.00GB of RAM. And raspberry Pi is ARMv6-compatible processor rev 7 (v6l). All application programs are written in java and use MyEclipse J2EE platform.

5.2 Result for scenario1: Test on different update rate

For this scenario I will record two metric: round-trip time and CPU usage for measurements. The CPU usage will immediately increase as high as 98%, after connection is set up and send data periodically the CPU usage will stabilize around a certain value. The shape of the chart is like Figure 5-1.

![Figure 5-1: CPU performance](image)

I draw graph use the CPU usage when sending data periodically. I will detailed describe performance for each platform in sub-chapter 5.2.1 and 5.2.2. Last, the comparison for two platforms will be presented in sub-chapter 5.2.3.

5.2.1 SensibleThings performance

I changed source node’s speed from 1 times/second to 100 times/second, and for each rate I send data 1000 times in total and thus I got 1000 RTT time. I use these RTT time as Y-axis and use sequence number as X-axis. The following graphs show the result for different speed.

Figure 5-2 is a graph that shows 1000 RTT at speed 1 time/second. I also use “xmeans” cluster to analysis these RTT data: 61% instances are around 321ms, and 39% instances are around 231ms.
Figure 5-2: test result for 1 time/second

Figure 5-3 is SensibleThings performance at speed 20 times/second. Applying “Xmeans” cluster to test data, I found: 70% instances are around 176 ms, 30% instances are around 259ms. Shown from the figures, I can see the RTT did not change too much when the update rate change from 1 time/second to 20 times/second.

For next testing at 25 times/second, I can clearly see that: as the time went by, RTT becomes bigger and bigger. The value becomes numerical continuous and almost connects like a curve, see Figure 5-4.
I test many times and the graph for 40 times/second is like a line, shown in Figure 5-5. With more and more data being sent, RTT increases in O(n). The red line is a trend line and the formula is attached near the line. The Figure 5-6 for 100 times/second has the same shape as 40 times/second. The rake ratio for 40 times/second is 22.78 while the rake ratio for 100 times/second is 31.84, so when the update speed increased the growth rate for RTT also increased.
By observing the RTT graph of different rate, I divided the rate into three stages. First stage: from 1 time/second to 20 times/second, most of the RTT gathered in a few near value, the value are stable on the whole; the second stage: from 20 times/s to 25 times/s, RTT’s value increased at small-scale; the third stage, more than 25 times/second the RTT and the number of send data has linear relationship.

I integrate the result for different speed into one table. Table 5-1 shows the result.

<table>
<thead>
<tr>
<th>Polling Rate</th>
<th>Worst case (ms)</th>
<th>Average (ms)</th>
<th>Standard deviation</th>
<th>CPU usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>602</td>
<td>286</td>
<td>56</td>
<td>0.3%</td>
</tr>
<tr>
<td>2</td>
<td>522</td>
<td>214</td>
<td>26</td>
<td>0.7%</td>
</tr>
<tr>
<td>2.5</td>
<td>594</td>
<td>215</td>
<td>26</td>
<td>0.7%</td>
</tr>
<tr>
<td>5</td>
<td>516</td>
<td>222</td>
<td>16</td>
<td>1.0%</td>
</tr>
<tr>
<td>10</td>
<td>338</td>
<td>229</td>
<td>19</td>
<td>1.6%</td>
</tr>
<tr>
<td>12.5</td>
<td>476</td>
<td>212</td>
<td>46</td>
<td>2.0%</td>
</tr>
<tr>
<td>20</td>
<td>453</td>
<td>199</td>
<td>47</td>
<td>3.0%</td>
</tr>
<tr>
<td>25</td>
<td>1322</td>
<td>1024</td>
<td>232</td>
<td>3.6%</td>
</tr>
<tr>
<td>40</td>
<td>23225</td>
<td>12076</td>
<td>6555</td>
<td>5.3%</td>
</tr>
<tr>
<td>50</td>
<td>18248</td>
<td>9603</td>
<td>5112</td>
<td>6.6%</td>
</tr>
<tr>
<td>100</td>
<td>32345</td>
<td>16657</td>
<td>9199</td>
<td>11.2%</td>
</tr>
</tbody>
</table>

Table 5-1: Integrated result for SensibleThings in scenario 1

Figure 5-7 and Figure 5-8 are graphs using Table 5-1’s data. Not only observed RTT performance as time went by, but also observed by comparison with other update rate, the update rate from 1 time/s to 20 times/s has better stability than other two stages. When the speed reaches 50 times/s has strange performance. I tested many times but it still performs like this. When the speed is more than 25 times/second, the worst case RTT and average RTT both increased sharply. When observing CPU usage graph, I can see that SensibleThings is a lightweight platform since it uses little CPU usage and the growth rate is not so high when the update speed increased.
5.2.2 ThingWorx performance

I changed EMS’s update speed from 0.2 times/s to 1000 times/second and I got some test results. For each rate I send data several times and thus I draw the graphs as I did for SensibleThings. The Figure 5-9 shows the result for different speed. From the figures, I can see that there is little difference for different speed. In other word, the update speed doesn’t influence performance on ThingWorx.
I integrate the result for different speed into one Table. Table 5-2 shows the result.

<table>
<thead>
<tr>
<th>Update rate (times/second)</th>
<th>Worst Case RTT (ms)</th>
<th>Average RTT (ms)</th>
<th>Standard Deviation</th>
<th>CPU usage(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>625</td>
<td>72</td>
<td>54</td>
<td>2%</td>
</tr>
<tr>
<td>1</td>
<td>625</td>
<td>69</td>
<td>63</td>
<td>5%</td>
</tr>
<tr>
<td>1.25</td>
<td>750</td>
<td>64</td>
<td>57</td>
<td>5.50%</td>
</tr>
<tr>
<td>2</td>
<td>734</td>
<td>66</td>
<td>62</td>
<td>0%</td>
</tr>
<tr>
<td>2.5</td>
<td>609</td>
<td>63</td>
<td>54</td>
<td>9.20%</td>
</tr>
<tr>
<td>5</td>
<td>677</td>
<td>63</td>
<td>65</td>
<td>15.20%</td>
</tr>
<tr>
<td>10</td>
<td>672</td>
<td>59</td>
<td>51</td>
<td>24%</td>
</tr>
<tr>
<td>12.5</td>
<td>672</td>
<td>62</td>
<td>57</td>
<td>26%</td>
</tr>
<tr>
<td>20</td>
<td>688</td>
<td>65</td>
<td>62</td>
<td>32%</td>
</tr>
<tr>
<td>25</td>
<td>656</td>
<td>56</td>
<td>43</td>
<td>34%</td>
</tr>
<tr>
<td>50</td>
<td>945</td>
<td>63</td>
<td>70</td>
<td>40%</td>
</tr>
<tr>
<td>100</td>
<td>673</td>
<td>56</td>
<td>40</td>
<td>45%</td>
</tr>
<tr>
<td>125</td>
<td>1000</td>
<td>63</td>
<td>65</td>
<td>46%</td>
</tr>
<tr>
<td>200</td>
<td>908</td>
<td>71</td>
<td>80</td>
<td>47%</td>
</tr>
<tr>
<td>250</td>
<td>375</td>
<td>55</td>
<td>36</td>
<td>48%</td>
</tr>
<tr>
<td>500</td>
<td>969</td>
<td>61</td>
<td>61</td>
<td>47%</td>
</tr>
<tr>
<td>1000</td>
<td>232</td>
<td>57</td>
<td>44</td>
<td>47%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Update rate (times/second)</th>
<th>Worst Case RTT (ms)</th>
<th>Average RTT (ms)</th>
<th>Standard Deviation</th>
<th>CPU usage(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>625</td>
<td>72</td>
<td>54</td>
<td>2%</td>
</tr>
<tr>
<td>1</td>
<td>625</td>
<td>69</td>
<td>63</td>
<td>5%</td>
</tr>
<tr>
<td>1.25</td>
<td>750</td>
<td>64</td>
<td>57</td>
<td>5.50%</td>
</tr>
<tr>
<td>2</td>
<td>734</td>
<td>66</td>
<td>62</td>
<td>0%</td>
</tr>
<tr>
<td>2.5</td>
<td>609</td>
<td>63</td>
<td>54</td>
<td>9.20%</td>
</tr>
<tr>
<td>5</td>
<td>677</td>
<td>63</td>
<td>65</td>
<td>15.20%</td>
</tr>
<tr>
<td>10</td>
<td>672</td>
<td>59</td>
<td>51</td>
<td>24%</td>
</tr>
<tr>
<td>12.5</td>
<td>672</td>
<td>62</td>
<td>57</td>
<td>26%</td>
</tr>
<tr>
<td>20</td>
<td>688</td>
<td>65</td>
<td>62</td>
<td>32%</td>
</tr>
<tr>
<td>25</td>
<td>656</td>
<td>56</td>
<td>43</td>
<td>34%</td>
</tr>
<tr>
<td>50</td>
<td>945</td>
<td>63</td>
<td>70</td>
<td>40%</td>
</tr>
<tr>
<td>100</td>
<td>673</td>
<td>56</td>
<td>40</td>
<td>45%</td>
</tr>
<tr>
<td>125</td>
<td>1000</td>
<td>63</td>
<td>65</td>
<td>46%</td>
</tr>
<tr>
<td>200</td>
<td>908</td>
<td>71</td>
<td>80</td>
<td>47%</td>
</tr>
<tr>
<td>250</td>
<td>375</td>
<td>55</td>
<td>36</td>
<td>48%</td>
</tr>
<tr>
<td>500</td>
<td>969</td>
<td>61</td>
<td>61</td>
<td>47%</td>
</tr>
<tr>
<td>1000</td>
<td>232</td>
<td>57</td>
<td>44</td>
<td>47%</td>
</tr>
</tbody>
</table>

Table 5-2: Integrated result for ThingWorx in scenario 1

From the Figure 5-10, we can see ThingWorx has good performance from low speed to high speed. Although the worst RTT changed greatly when the speed is high, the average RTT and standard deviation still keep stable at small value.
See Figure 5-11, the CPU usage increased when the update rate increased. But the marginal growth rate decreased when the speed became higher and higher.

### 5.2.3 Two platforms' performance comparison

Figure 5-12-A shows the comparison between two platforms in terms of worst case RTT, while chart B states the comparison in terms of average RTT. According to the Figure 5-12, It is clear that ThingWorx and SensibleThings all has good stability when the update rate is less than 25 times per second. In terms of the value of Round-trip time, the ThingWorx has better performance than SensibleThings and If update data more than 25 times per second the ThingWorx has better performance.
As is clearly described in the Figure 5-13, the SensibleThings has less CPU usage than ThingWorx. What’s more the SensibleThings growth rate is less than ThingWorx. From these two aspects we can see SensibleThings is more lightweight than ThingWorx, which is a main feature stated in related paper[3].

5.3 Result for scenario2: Test on different number of devices

When look at ThingWorx’s performance at different number of EMS, the RTT’s fluctuation range did not change too much when the node of sensor increased according to Figure 5-14. The RTT’s fluctuation range is about 100ms to 450 ms. Besides, the Figure 5-15 illustrates that the value of Worst RTT and average RTT are not influenced when the number of sensor increased. To sum up, the ThingWorx performed well at different number of sensor.
Because there are two kinds of deployment for SensibleThings in this scenario, so I will first state the result for situation: one source node with several sink nodes. In this situation, Several Sink Nodes try to get data from one source node at 10 times/second simultaneously. The number of node change from 1 to 7. As is shown from Figure 5-16: the value of RTT increased when number of sink node increased but the RTT fluctuation is not so big. From Figure 5-17, I can see that the value of worst case and average RTT acted as linear regression when number of sink node increased. While the standard deviation did not change greatly.
For another situation: Sink Node get data from several Source Nodes. And each Source Node will send data at speed 10 times/second to the Sink Node. Unfortunately, there are some problems when the Source Node number become bigger than 4. For example, I observed that the Sink Node failed to resolve all UCI, in other word It failed to receive resolveResponse for some Source Node. And there is a case that some Source Node can't get "GetEvent", so these Source Node failed to send data to Sink Node. The successful cases number of Source Node I set is from 1 to 3. Figure 5-18 is the result for this situation.
When the data consumer need to "pull" some data from several sensor, the SensibleThings works better, because the ThingWorx provided the sensor with capability of "push", the server can't "pull" data from specific sensor. If we suppose that several Sink Node get data simultaneously from one Source. From the previous graph I can see the round-trip time will increased as O(n) as the number of Sink Node increased. When the situation is that several sensors need to send data to one sink(consumer) node simultaneously, ThingWorx works better, cause there will be some problems occurred in SensibleThings' communication between sink and source. What's more, The RTT is not greatly influenced by the number of sensor node on ThingWorx.

5.4 Result for scenario3: Using load-testing tool: Tsung

Configuring “tsung.xml” and using this configuration, I got load-test results for both SensibleThings and ThingWorx. Table 5-3 is a main statistics about connect, session and request for SensibleThings. The network throughput for SensibleThings is 72.80Kb/sec shown in Table 5-4. There are many graphs generated by Tsung, I only picked some interested ones from them. Figure 5-19 display requests and connection establishment for SensibleThings. The “connect” means duration of the connection establishment only and “request” means mean duration of requests. Figure 5-19 uses response time as metric while Figure 5-20 uses throughput as metric. In Figure 5-20 I can know requests throughput for SensibleThings. See Figure 5-21: users is the number of simultaneous users whose session has started, but not yet finished; connected is the number of users with an opened TCP/UDP connection. Unfortunately the load-test for ThingWorx failed because for some unknown reason, see Figure 5-22. The reason may be the authenticated denied but on the other hand it shows that the ThingWorx has good safety. So it is not possible to compare two platforms.

<table>
<thead>
<tr>
<th>Name</th>
<th>highest 10sec mean</th>
<th>lowest 10sec mean</th>
<th>Highest Rate</th>
<th>Mean Rate</th>
<th>Mean</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect</td>
<td>0.48 sec</td>
<td>0.43 sec</td>
<td>50.7 / sec</td>
<td>47.00 / sec</td>
<td>0.47 sec</td>
<td>3057</td>
</tr>
<tr>
<td>page</td>
<td>2.40 sec</td>
<td>2.20 sec</td>
<td>20.3 / sec</td>
<td>18.71 / sec</td>
<td>2.32 sec</td>
<td>1172</td>
</tr>
<tr>
<td>request</td>
<td>0.94 sec</td>
<td>0.87 sec</td>
<td>51.3 / sec</td>
<td>47.04 / sec</td>
<td>0.92 sec</td>
<td>3020</td>
</tr>
<tr>
<td>session</td>
<td>6.07 sec</td>
<td>4.98 sec</td>
<td>10 / sec</td>
<td>9.14 / sec</td>
<td>5.57 sec</td>
<td>576</td>
</tr>
</tbody>
</table>

Table 5-3: Main Statistics for SensibleThings using Tsung
Table 5-4: Network Throughput for SensibleThings using Tsung

<table>
<thead>
<tr>
<th>Name</th>
<th>Highest Rate</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>size_rev</td>
<td>0/sec</td>
<td>0</td>
</tr>
<tr>
<td>size_sent</td>
<td>72.80 Kbits/sec</td>
<td>636.48 KB</td>
</tr>
</tbody>
</table>

Figure 5-19: Requests and connection establishment for SensibleThings

Figure 5-20: Requests throughput for SensibleThings

Figure 5-21: Simultaneous Users for SensibleThings
5.5 Result for theoretic evaluation works1: Implement an evaluation framework

I measure ThingWorx and SensibleThings using an evaluation framework proposed by Oleksiy Mazhelis and Pasi Tyrväinen. The following Table 5-5 is my implemented measurement.

<table>
<thead>
<tr>
<th>Elements</th>
<th>ThingWorx</th>
<th>SensibleThings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design and implementation</strong></td>
<td><strong>A sensing or actuating device</strong> AlwaysOn Connectivity (e.g. via REST API or MQTT) for two-way “non-polling” communications is supported[13].</td>
<td>The sensor and actuator layer enables different sensors and actuators to connect into the platform. The sensors and actuators can vary greatly and the platform therefore offers two options to connect them[3].</td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>No information has been found.</td>
<td>The sensors and actuators can connect through the sensor and actuator abstraction. The abstraction enables connectivity either directly to wireless sensor networks or via more powerful gateways[3].</td>
</tr>
<tr>
<td><strong>User interface (UI)</strong></td>
<td>“drag and drop” mashup builder, with a set of widgets for collaboration (blogs, forums, wiki), further expandable via a script engine. The APIs are RESTful, public and open[9].</td>
<td>Without UI, but you can build your application UI by using Java code.</td>
</tr>
<tr>
<td><strong>Web component</strong></td>
<td>A model-based development environment, with integrated search-query-analysis capabilities. Different deployment alternatives[13].</td>
<td>In SensibleThings, there is dissemination layer:The dissemination layer enables dissemination of information between all entities that participate in the system and are connected to the platform[3].</td>
</tr>
</tbody>
</table>

**Operations**

| **Fulfillment**               | The platform offers some support for remote software installation and file transfer. However, neither application discovery nor application purchasing are supported[13]. | The SensibleThings offers some example code to build nodes. However, neither application discovery nor application purchasing are supported[3]. |
| **Assurance**                 | No information about the support of monitoring functions.                | No information has been found.                      |
| **Billing**                   | No information has been found. But it will supply more services if you are their customer. | The SensibleThings platform is open source and licensed under the LGPL license[16]. |

Table 5-5: Implement for an evaluation framework
5.6 Result for theoretic evaluation works2: Comparison for the used protocol and packages

In order to make the results more clearly to understand, the result for SensibleThings will be stated in chapter 5.6.1 and chapter 5.6.2 for ThingWorx, finally there will be a summary for two platforms’ comparison in chapter 5.6.3.

5.6.1 Measurements on SensibleThings

In order to observe nodes’ packages easily, I make the program execute “resolve”, “Set”, “Get”, “Notify” periodically at very low speed. As is shown from Figure 5-23, it needs 11 message to finish a resolve process. And see from Figure 5-24, GET process needs 4 messages. while Figure 5-25 illustrates the process of Notify needs 4 messages and Set also need 4 messages (see Figure 5-26). The protocol it used is TCP, actually it is a UDP package wrapped in TCP to pass through the firewall. The process for “Set”, “Get” and “Notify” went as Figure 5-27.

Figure 5-23: Resolve process

Figure 5-24: Get process

Figure 5-25: Notify process

Figure 5-26: Set process

Then I set the situation like this: after sink node received response it would send response back to source node. In other words, there are two “Notify” in this process, but I only observe 7 packages. However, each “Notify” needs 4 package. Let’s see what happen by using wirshark. As is shown in the Figure 5-28, The forth package is the last ACK for first Notify process, meanwhile it is second Notify’s first package.
5.6.2 Measurements on ThingWorx

For ThingWorx, I will observe its package using wireshark from two directions which I have discussed minutely in chapter 4.5.1. About direction from sensor to server, I found that after starting my program(edge micro server) the EMS need 4 requests to set up connection: “AuthRequest”; “BindRequest”; “GetPropertySubscriptions”; “GetEventSubscriptions”. After four websockets requests all connected successfully, the EMS will send the “UpdateSubscribedPropertyValues” which means sending data to server. So the first RTT I got is always started with "RequestID=5". For each sending process, there will be two websockets packages and two ACK packages, see Figure 5-29. The first websocket is the request to send data and the second websocket is a respond websocket sent from server. The second websocket’s RTT is very little around 0.1 ms, so this time can ignored regard of first websocket’s round-trip time.

With a view to another direction: retrieve data from server, I found that the consumer use HTTP: POST request to get data and the server will send response HTTP package to consumer. Because ThingWorx is a web application, no wonder it use HTTP for communication. See Figure 5-30 which is captured by wireshark for each data retrieve process. The number of messages to get data depends on the service I use, for example in this case I use two service: GetNamedProperties and QueryPropertyHistory. Except HTTP, I observe there are 29 messages in total to retrieve data each time. Figure 5-31 is a whole communication workflow for ThingWorx.
5.6.3 Summary for two platforms’ comparison

By observing the captured package, I compared two platforms in terms of package number, protocols and generated a Table 5-6 to clearly illustrate my work. If I want to use the simple function to get properties value, SensibleThings only need 4 messages but ThingWorx need more than 6 messages basically (for only the simplest function). So SensibleThings can disseminate information to end devices with low overhead.

<table>
<thead>
<tr>
<th>Package number used for communication</th>
<th>SensibleThings</th>
<th>ThingWorx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolve process needs 11 messages; Get process needs 4 messages; Notify needs 4 messages; Set needs 4 messages.</td>
<td>From sensor to send data, it needs 4 messages; from server to consumer, it needs 2 HTTP package for each service and some more TCP packages.</td>
<td></td>
</tr>
</tbody>
</table>

| Protocol used for communication | TCP From paper [3], SensibleThings use RUDP but was not observed in my test; support SSL encryption | websocket, HTTP, TCP; not be observed in my test but according to some paper ThingWorx use XMPP for real-time interaction[2]. All packages are sent compressed and encrypted using Secure Sockets Layer (SSL) technology[4] |

| Features consider of used protocols | RUDP: Reliable but with less overhead, retransmission of lost packets. SSL support security support. | websocket: full-duplex communication, only need one hand-shake. XMPP: slow but having good scalability, addressing and security capabilities[5] SSL: ensure user and data security. |

Table 5-6: comparison for two platforms in terms of protocol

5.7 Result for theoretic evaluation works3: Other aspects comparison

I measured two platforms from the aspects which I have mentioned in chapter 4. I fill in a table to clearly illustrate my findings.
Table 5-7: Other aspects comparison

<table>
<thead>
<tr>
<th>platforms</th>
<th>Programming language</th>
<th>platform support</th>
<th>application area</th>
<th>special features</th>
<th>developer friendly (from my point of view)</th>
<th>security</th>
</tr>
</thead>
<tbody>
<tr>
<td>SensibleThings</td>
<td>Java</td>
<td>Fully distributed communication</td>
<td>M2M/IoT application</td>
<td>1. The platform can disseminate information to end devices quickly with low overhead. 2. It is stable and operates without any central point of failure. 3. It is lightweight and can run in mobile devices. 4. It is extensible. 5. Fully distributed IoT platform with chord structure[3]</td>
<td>SensibleThings' Interface layer's code is reusable and easy to invoke these functions, but it need developer have basic knowledge of Java.</td>
<td>Encrypted using secure socket layer (SSL) technology</td>
</tr>
<tr>
<td>ThingWorx</td>
<td>Since ThingWorx is a web application, it is written in Java script; For edge server to send data, it can be written in many language, C, Java as long as it use ThingWorx’s SDK</td>
<td>Various type of sensors and cloud devices[5]</td>
<td>M2M/IoT application</td>
<td>1. Event driven execution support for 3D storage. 2. Provide developer with codeless mashup builder. 3. Search-based intelligence. 4. Flexible connectivity options.</td>
<td>It is easy to build graphical application on ThingWorx, and it's requirement on programming is not so high. If I have some problem I can post question in their online community and get answer quickly.</td>
<td>Since ThingWorx is a web application, in its base/simple form, it will be as secure as your Tomcat is. Support User Authentication and other protections for sensitive data[7]</td>
</tr>
</tbody>
</table>

5.8 Result for Specific Case Evaluation: Health care application

Since I aforementioned in chapter 4, this e-health care application need to meet requirements: interoperability, low latency and good reliability, retrieve data on demand, alert management and privacy.

As for interoperability, ThingWorx can achieve this by deploying logistic analysis in cloud server, SensibleThings can achieve this by using a sink node to manage several source nodes and coding some functions in Java. In terms of second requirement, low latency and reliability, Since in my previous test there is case that sink node can’t resolve all UCI, further more ThingWorx has low latency. So I think ThingWorx is suit for developing this e-health application. When comes to requirement: retrieve data on-demand, SensibleThings is a good choice than ThingWorx because it can “poll” data and as well as “push” data. As far as I know, ThingWorx can deploy alert management easily using graphical composer see Figure 5-32. What’s more, according to ThingWorx white paper, It uses encrypted storage for sensible data and has an extremely granular security model to enable data isolation[7]. Unfortunately, SensibleThings lack alert design, maybe It can extended by more coding. Compared with ThingWorx, It’s design on privacy is less than ThingWorx. In addition, ThingWorx support graphical visualization which is more easily to see data fluctuation. To sum up, although ThingWorx lack capability to “poll” data and SensibleThings
can improve its function by more coding, I think using ThingWorx to develop application is more quicker and has better performance.

Figure 5-32: alert management by ThingWorx Composer.
6 Conclusions

An IoT platform makes it easier to build IoT application and integrates great number of capabilities such as: data control, rock-bottom communication, cloud computing and devices management. There are many types of platforms which have different solution types and architecture. In my project work, I only focus two platforms: SensibleThings which is fully distributed system and another is ThingWorx which is centralized cloud system.

For first goal to follow the suggested IoT learning path, finish QuickStart kits and read up ThingWorx’s documents. I have done all these works in order to figure out how to use this platform, and I have general idea on how to build application using its composer and bind data between devices and application.

As for second goal: read up all materials on SensibleThings website and read some papers about how does this platform work. By done these task I know some basic functions I can invoke for communication in interface layer.

Set up a raspberry Pi is my third goal, I finish this by using SSH application and modifying platforms’ example code. I deployed this program to raspberry Pi and ran it to send out simulated data.

Meanwhile I finished my fourth goal: set up a client application at my laptop side to retrieve data and recorded some metric data.

For fifth goal: use three evaluation works, simulate two scenarios and a case study. I finished theoretic evaluation works and a case study by reading paper and searching on the internet and integrated my answer into a table. By modifying example code and invoking specific function every 5 second, I can clearly see the performance for this function through wireshark. I configured Tsung file to do my load test. Last, for other two scenarios I changed some parameters of program and recorded needed data for further analysis.

The Last goal is to analysis data and evaluate two platforms in terms of scala-bility and stability. I write some java program to pick up needed data and put it into consolidated format. I compare the data not only with it self at different parameter and time but also with another platform. According to graphs, It will illustrate the result vividly.

My problem aforementioned in the beginning is to determine which IoT platforms fits best for different scenarios. From first scenario, changing update rate, I think if the update rate no more than 25 times/sec its no great different to choose any platform, but it is better to choose ThingWorx if the update rate goed higher than 25 times/sec. However if the sensor’s hardware is not so good it is better to use SensibleThings since it is lightweight than ThingWorx. For the case that no more one sensors’ data rush into one point simultaneously, ThingWorx has better performance. ThingWorx and SensibleThings have its innova-
Quantitative Comparison of SensibleThings and ThingsWorx
Yun Yu

6.1 Ethical consideration

The thesis I submitted is independently done by me under supervisor’s serious guidance. Except the mentioned contents of references, this paper does not use other people’s academic achievements. The research is not a destructive experiment to both ThingWorx and SensibleThings. The findings of this paper should not be used in the harmful aspects of human beings. Any fabrication or exaggeration of test data is avoided in the research. I kept objectivity in both test and analysis through whole project.

Although IoT is developing rapidly, some ethical considerations about this industry are raised, for example: Signal leakage and interference; because lack human’s control there will be some problem on node security; the sensitive data needed to be transfer around network so it will be some problem on sensitive data security. This is because the network is composed of a large number of machines, lack of people efficient monitoring of equipment, equipment cluster and other related characteristics. The special security problems can mainly be the following aspects.

Networking machine/sensing node's local security problems. Because the IoT application can replace people to finish some complex, dangerous and mechanical work. So most of IoT machine/sensing nodes were deployed in unattended monitoring scene. Then the attacker can easily access to these devices, resulting in destructing them, even can replace the local hardware and software.

Perception of the network transmission and the problem of information security. Sensor nodes usually have simple function (such as automatic thermometer), carry less energy (battery), making them unable to have complex security protection. what's more, sensing network is variety from temperature measurements to hydrological monitoring, from the road navigation to automatic control, their data transmission and message lack specific standard, so there is no way to provide a unified security system.

Core network transmission and information security problems. Core network has relatively complete safety protection ability, but due to the large number of nodes in the Internet of things and exists in a cluster, thus lead to mass data transmission network congestion, resulting in a denial of service attack. In addition, existing communication network security architecture are designed from the perspective of human’s communication, does not apply to machine communication. Using the existing security mechanism will fragment IoT machine logic relationship.
The IoT development is still in the primary stage, more times it is only a concept. So IoT security research still has long way to go.

6.2 Future work

I think Tsung is a very useful tool for load test, so I want to use it successfully on ThingWorx and SensibleThings. But for some unknown reason it failed, if I have more time I will look deep into how it failed and try to understand meaning of some configurations. There are lots of configuration scripts posted on GitHub and stackoverflow, so it can be useful if someone do this measurements. What’s more, since in the paper “Performance Evaluation of an IoT Platform” authors do the successful test, I can try to contact to them to get their configuration code or ask them for help.

The aspects measured in this project is only a piece of platform evaluation, I think it should look deep into other aspects like data storage and synchronization in the future work. In the chapter 4, I mentioned that my evaluation system can be expanded by more scenarios, such as comparing underlying layer.
References


[9] Introduction to ThingWorx, spimeSense-PTC-ThingWorxver.12, published in 2014.09.26


Quantitative Comparison of SensibleThings and ThingsWorx
Yun Yu

https://www.researchgate.net/publication/267035168, CONFERENCE PAPER · MARCH 2014 DOI: 10.1109/WF-IoT.2014.6803137


[15] SensibleThings Documentation

[16] SensibleThings Tutorial, 2013-09-18,
http://www.sensiblethings.se/wiki/index.php/Tutorials/PresentationSlides


[18] Iot.knowledge-bytes.com Internet of Things Rundtable: IoT platform