Building A Generic Architecture for the Internet of Things

Wei Wang, Kevin Lee, David Murray

School of Information Technology, Murdoch University,

Murdoch 6150, Western Australia, Australia

{W.Wang, Kevin.Lee, D.Murray}@murdoch.edu.au

Abstract—**The Internet of Things (IoT) allows physical objects to be connected on the Internet. Objects in the IoT have identities, attributes and personalities in the virtual world. These objects are integrated together using intelligent interfaces. The IoT has a lot of challenges and issues that require further research before achieving a global scale. This paper presents a generic IoT architecture to modularize physical objects into the digital world. It demonstrates that the future IoT can be designed based on component-based communication and existing communication standards. To achieve integration both on a device and semantic level, physical objects and services can be virtualised in stated middleware components. By building ontologies, third-parties can also customize objects and services.**

Keywords—**Internet of Things, Web of Things, Wireless Sensor Networks, WSNs, RFID, Integration.**

I. INTRODUCTION

The Internet is mainly used for people-to-people communication; Web pages, emails and online games. In sensor networks, people can also use the Internet to interact with the physical world. Environmental data can be collected by sensors and then transmitted via the Internet. However, these sensed physical parameters are isolated. The aim of Internet of Things (IoT) is to connect physical objects, sensors, actuators, and other technologies, to provide people-to-object and object-to-object communication [1].

The IoT enables physical objects to exchange and process information in a self-organized way. For example, self-driving cars can negotiate with other cars to co-design better travel paths. The U.S. National Intelligence Council lists the IoT as one of six technologies that will impact U.S. national power by 2025 [2]. The Chinese government is also promoting the IoT in Wuxi which was named as "The Smart City" in 2010 [2]. If each person owns 6 connected devices on the earth, there will be 36 billion of objects connected on the Internet. The numerous connections will require new abilities to create, process and exchange a large amount of real-time information.

One of major problems is to integrate heterogeneous sensing devices including Wireless Sensor Networks (WSNs) and Radiofrequency Identification (RFID). Some solutions try to define standards such as 6lowPAN to encapsulate sensed data over IPv6 communication. However, equipping sensors with IP stacks may not be suitable for resource-constrained devices such as passive RFID tags. Another solution is to interpret standards by different adapters. However, it is inefficient to design $O(N^{\wedge 2})$ adapters to support the integration among N standards. New standards are also incompatible with existing adapters. This paper proposes to abstract objects and services in a middleware layer, and shields the IoT system and users from the complexity of directly dealing with heterogeneous sensing devices, which can make it easier for users to compose services across different platforms.

Object-to-object communications are exchanging messages based on negotiated rules. For the IoT, the descriptions of objects' attributes and their relationships are called ontologies [3]. By building ontologies for virtual objects, the IoT system can understand the messages with different formats from different objects via semantic analysis.

This paper will address some key challenges of IoT, and then propose a generic IoT architecture to attempt to eliminate these issues. The proposed architecture can integrate any type of sensors/RFIDs by running their virtual components in middleware. Message size is minimized to support resource-constrained devices. Objects' virtual representation can either be customized by users or detected at runtime by systems.

The remainder of this paper is structured as follows. Section II introduces the background of IoT and analyses its key challenges. Section III proposes a generic framework to build the IoT. Section IV then extends the IoT to Web of Things (WoT) using a case study. Section V evaluates a prototype implementation based on the case study in Section IV. Finally, the conclusion is presented in section VI.

II. BACKGROUND OF INTERNET OF THINGS

The concept of IoT was first introduced by the MIT Auto-ID Centre in 1999 [3]. The aim of the Auto-ID Labs was to design a global network where all physical objects with RFID tags are connected, and each object has a globally unique ID. The IoT is emerging due to the developments of network bandwidth, Cloud Computing, hardware manufacture and the decrease of size and cost of chips in the last decade [1].

The IoT faces many issues and challenges across different domains. This paper tries to resolve the problems as follows:

1) Resource-constrained scenarios: Battery-constrained sensor devices can last from few days to years [4]. Energy can be saved in various ways: a) Moving processing ability from sensors to the Cloud; b) Using light-weight sensory devices. c) Reducing messages' size through energy-efficient encryption or caching; d) Improving batteries' capacity.

2) Observation and data measurement: The IoT will use different type of sensors/RFIDs, requiring a mechanism to observe and measure data from heterogeneous APIs. Some

models such as [5] use different adaptors to communicate with different sensing devices. Users require technical API details specified in the Device Ontology to access their services, which adds too much complexity. Via reasoning for heterogeneous APIs, observation and data measurement can be automatically implemented by coordinating a group of adaptors [6]. However, this approach lacks scalability, as the device ontology cannot recognize new APIs. This paper proposes an effective approach using middleware, shielding users from the complexity of devices' heterogeneous APIs.

3) Customization: Personalization can meet user demands and improve service experience to deliver customized services [7]. The future IoT consists of billions of objects connected to the Internet. It is difficult to predesign all object and service by large organizations. It should follow Web 2.0 principles to allow users to create contents and discover services easily [8]. Some ontology models are only designed for a specific area and users are not allowed to customize domain knowledge. For example, OBOE [9] is only limited for coastal ecosystems. The proposed architecture gives users much flexibility to extend domain knowledge and ontologies.

4) Inefficient runtime discovery: Because a large amount of sensing devices are deployed or deactivated on runtime, they keep running in dynamical states. It is inefficient to preload all sensor states from a central knowledge database. The IoT models [6] send event notifications to update changes in a knowledge base. These updates are queued for processing. However, some time-sensitive services require an immediate response. It is necessary to design an IoT model that can efficiently discover objects' dynamic states on runtime. This paper proposes to use component-based communication for the IoT. By retrieving objects' real-time states directly from running components, it can perform better than indirectly retrieving from a central knowledge database.

III. A Generic Architecture of the Global IoT

This paper proposes a generic architecture to integrate physical objects into the IoT systems. This architecture is designed based on component-based communication. It can integrate any type of sensors/RFIDs by running their virtual components in middleware, which is especially suitable for resources-constrained devices. By building ontologies for objects and services, the physical objects in different domains can be described and modularized in the IoT systems.

A. Methodology

The design of global IoT systems is challenging as many heterogeneous components are involved. Web-based model and component-based model can be compared as follows.

a)Performance: Component-based communication performs better than Web-based communication [10]. Component-based communication uses proprietary protocols with a lower overhead and does not need to parse data. Comparatively, web-based communication partly uses other web services that result in adding much overhead for data encapsulation [11]. The global IoT will involve billions of objects and each object is modularized as a fine-grained component. Distributing so many fine-grained objects would cause a non-negligible impact to the whole system [10]. For example, the RESTful IoT architecture requires users to indirectly access a central server to retrieve tremendous fine-grained data encapsulated by 6lowPAN [12]. This paper proposes to use component-based model for the IoT to improve the overall performance.

b) Loose coupling: The global IoT systems will involve many primitive services. With loose coupling, these service components can be combined to create new services. Both web-based architecture and component-based architecture can provide loose coupling. However, the services in web-based model cannot reach a long link, because the accumulated high latency is intolerable in some real-time scenarios [10]. From this angle, component-based model is also suitable for the IoT.

c) Provide and discover services: The most remarkable advantage of web-based communication is that third parties can provide and discover services on the Internet, such as the applications of Web 2.0 [10]. The Web of Things (WoT) can be extended to create and discover services for the IoT.

d) Integration: To achieve integration, web-based model uses HTTP or 6lowPAN to encapsulate sensed data [11]. The component-based model exchanges data via proprietary APIs, which impedes the integration of heterogeneous devices. For example, NesC [13] runs on TinyOS. This paper extends previous work [14] that can integrate any type of WSN.

B. Self-described Messaging for Component-based Model

Standard resource representation formats are beneficial for decentralized systems in which clients and servers can interact without individual negotiations. For example, a XML file describes a carton of milk in Fig. 1. Its attribute values are concluded in standard tags which are understandable for XML parser. Fig. 2 describes the milk based on JSON which is a lightweight format used to describe objects. However, it still adds much overhead for resource-constrained sensory devices.

```
<Milk>
  <TypeId>FOOD6812</TypeId>
  <Brand>HarveyFresh</Brand>
  <Product>FreshMilk</Product>
- <Volume>
    <Value>2</Value>
    <Unit>Liter</Unit>
  </Volume>
- <TotalFat>
    <Value>9.5</Value>
    <Unit>Gram</Unit>
  </TotalFat>
  <Expiration>09-06-2012</Expiration>
</Milk>
```

Fig. 1 Milk is described in XML

```
{
  "TypeId": "FOOD6812",
  "Brand": "HarveyFresh",
  "Product": "FreshMilk",
  "Volume":
  { "Value": "2", "unit": "Liter" },
  "TotalFat":
  { "Value": "9.5", "unit": "Gram", },
    "Expiration": "09-06-2012",
}
```

Fig. 2 Milk is described in JSON

Reducing messages' size can save energy for sensory devices. An improvement is replacing the attributes by a URI. Fig.3 describes a compressed message. "FOOD6862" is a URI that links to an object template on web servers. The template describes the milk's schema and attributes.

FOOD6862 HarveyFresh FreshMilk 2 9.5 11.5 09-06-2012

Fig. 3 A compressed message with a URI

C. A Message-oriented, Component-based IoT Model

If sensor components run in middleware, the integration for heterogeneous sensors/RFIDs can be handled via the inherent APIs of the middleware. Self-described messages contain objects' template URI, which is also benefit for semantic integration. Considering these virtues, this paper introduces a message-oriented, component-based (MOCB) architecture for the future IoT. Fig.4 demonstrates the overall architecture.

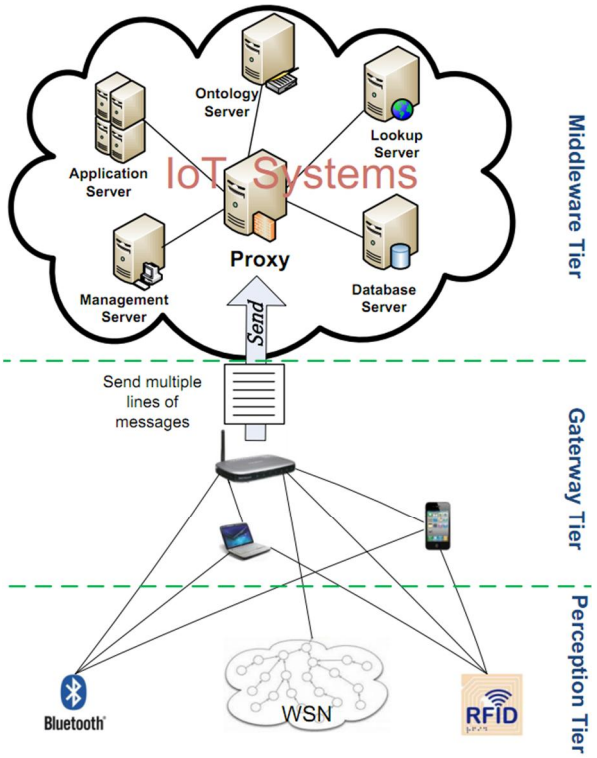

Fig. 4 The overall architecture

Perception Tier: On this layer, various types of sensors and RFIDs are used to gather raw values from distributed objects. The IoT system cannot distinguish a message is received from a passive RFID tag or from a heavyweight Java-based sensors, as these messages do not contain any data used to describe sensors/RFIDs. The benefit is that the complexity of sensory devices can be shield from end users without Device Ontology.

Gateway Tier: The purpose of the gateway layer is to establish communication channels for heterogeneous sensors and RFIDs. A dedicated gateway can discover and connect to sensors/RFIDs with open wireless standards such as 802.15.4, Wi-Fi and Bluetooth. If the sensory devices use proprietary protocols, proxies can be used to read and route messages. With the popularity of wireless communication, mobile phone, laptop, or other mobile devices can also play a role of gateway. In this way, any sensory devices can connect to the IoT system anywhere and anytime. The aggregated messages from distributed objects are routed to the IoT system finally.

Middleware Tier: The IoT systems run on the middleware tier. To modularise the physical objects, the Proxy can map objects' messages to their logical components in middleware. These running components are the virtual representations for physical objects and services. They are dynamically generated in the Proxy and then interact with other components via inherited APIs of the middleware. The Proxy can run on local machines or in the cloud if elastic resources are required. The Ontology Server contains the templates used to describe objects, services and relations. The Lookup Server provides a public platform to discover objects and services.

D. IoT Systems

The framework of IoT systems is illustrated in Fig. 5. It includes the following components.

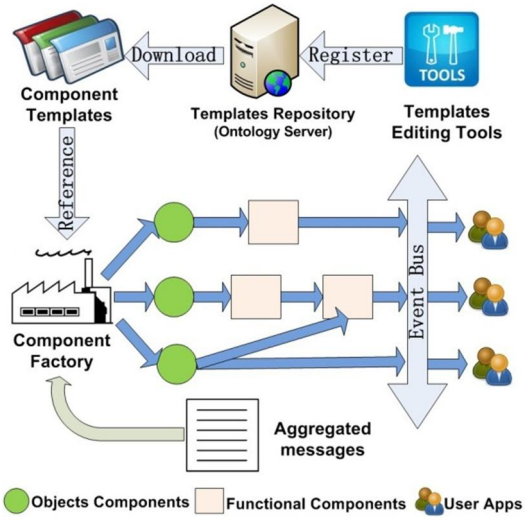

Fig.5 The framework of IoT Systems

Component Factory: The Component Factory is used to parameterize the gathered messages from objects and then dynamically generate logical components for the objects and services in middleware [14]. The gathered messages contain multiple-lines of text, and each message is received from a sensory device attached on an object. Each message consists of two types of data: 1) raw values that collected from objects; and 2) a URL links to an object template on web servers.

The Component Factory can read received real-time messages one at a time. If a message from a new object emerged, the Component Factory dynamically generates a component for the object. If a message is from existing objects in the IoT systems, the messages will be routed to its logical component. By mapping objects' messages to their logical components in middleware, physical objects can be virtualized and modularized on an abstracted level. Moreover, heterogeneous sensory devices are also effectively integrated by shielding their proprietary APIs in the IoT systems.

Service: The raw data collected from sensors/RFIDs can be filtered, aggregated, and converted to deliver customized services. The Functional Components are the primitive elements of services. They can process data and then generate outputs for further process. As this architecture is component-based, these Functional Components can reach a long link to combine various services with lower latency. The Functional Components are also dynamically generated by the Component Factory to meet users' customization. Both object components and functional components exist in intermediate state. The benefit is that the unused system resource can be released when objects are removed or services are terminated.

Templates Repository: To save energy, the messages do not contain any information used to describe objects' scheme, attributes and supported services. The ontologies are stored as Templates in the Template Repository. The templates are the primitive units to form the Ontologies. Each template describes one object or service. Similar with the URIs in RESTful architecture, the Template ID in each message links to a global unique template. By matching objects' messages with the templates in ontologies, the messages' semantic meaning can be precisely interpreted by the IoT system.

Customization: In the physical world, each object can be classified into multiple categories when it shares some common attributes with other objects. The Lookup Server allows users to discover objects' generic templates based on objects' classification. The users can add objects' private attributes to the generic templates via Templates Editing Tools, which result in generating a new customized sub-template. The new sub-template then is registered in the Templates Repository, and a new URI is provided to access it. Similarly, services can also be customized by editing their templates. For example, Temperature Filter is a generic service used to filter temperature data within predefined range. With the Template Editing Tools, users can configure temperature's thresholds as required or design other new services.

Public services and private services: To reuse the Component Templates, any users can discover, download and edit generic templates on Lookup Servers. These templates are considered as public services. Oppositely, users can also own their private templates which cannot be accessed by others. These services are considered as private services.

Global connectivity: To connect billions of objects, each object component in middleware is supposed to have its global connectivity. IP connectivity will enable the object components to communicate with each other globally. As the IPv4 address is almost exhausted but it is still dominating computer networks, a combination of "object ID + IPv4 address" can also represent a global unique address. In this architecture, each object component is allocated a global object ID by the system. The object components in the same network can share an IP address.

Cloud Layer: The global IoT system is required to process large amount of real-time information to guarantee the Quality of Service (QoS). Dynamically generating components in middleware also needs processing power to compile components [14]. Previous work [15] demonstrates that Cloud Computing can provide elastic resource to process real-time data from sensory devices.

IV. A CASE STUDY OF THE WEB OF THINGS

The IoT can be considered as a connectivity enabler to integrate heterogeneous sensory devices. The aim of WoT is to provide a public platform to create and discover services on the Internet. To explain the concept of WoT and ontology, this section uses a case study to validate the proposed architecture.

A. Entity Ontology

The Entity Ontology or object ontology is used to describe physical objects in the IoT system. Different physical objects have different attributes such as length, weight, or temperature. Objects can be categorized into same group as they share a set of common attributes. For example, refrigerators can be classified into appliances. If the objects' attributes are the same, an identical template can be used.

In this case, a refrigerator and a bottle of milk are connected to the IoT system. Both of them are attached with sensors/RFIDs to sense environmental change. The sensors in the refrigerator can sense two types of data: temperature and voltage. The RFID tag on the milk can only sense temperature data. Fig.6 shows the two messages received from two objects.

A Message Received From the Milk

FOOD6862 HarveyFresh FreshMilk 2 9.5 11.5 09-06-2012

A Message Received From the Refrigerator

APP3468 Mistral BarFridge 240 -5 90 http://iot.com/photoes/3465.jpg 08-05-2012

Fig.6 The received messages from two objects

These two messages only contain the raw values measured from objects. At the first field of messages, FOOD6862 and APP3486 are the Template ID links to two object templates in the Template Repository. The objects' schemas, attributes and measure units are described in the object templates. Fig. 7 illustrates the hierarchical structure of Entity Ontology.

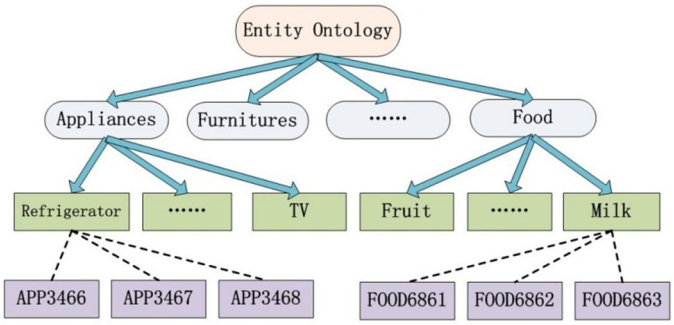

Fig. 7 the Entity Ontology

The Refrigerator and Chicken both have a generic object template which can be discovered in the Templates Repository. As the user adds their private attributes into these templates, new sub-templates FOOD6862 and APP3468 are generated to expand the object ontology. Object templates are the primitive elements to form the Entity Ontology. Each object template describes one type of object, and provides a unique Template ID used to access it. When the IoT system is interpreting the received messages, these object templates are referenced to understand each field's semantic meaning. Fig. 8 shows how a message from a carton of milk is interpreted by a template.

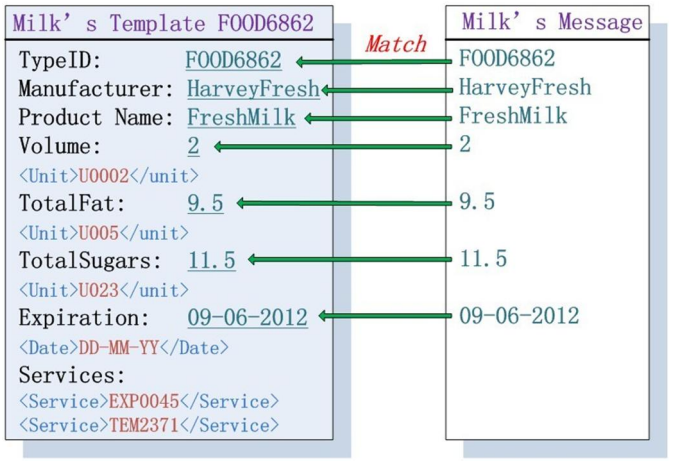

Fig. 8 The milk's object templates

Striping objects' attributes from messages has three virtues:

1) By reducing messages' size to a maximum, lightweight sensors or passive RFIDs can also be supported.

2) Energy saving for resource-constrained sensory devices.

3) Describing objects in shared templates rather than in self-described messages, the objects' descriptive vocabularies are consistent globally. It is benefit for interpreting objects' semantic meaning correctly and precisely.

B. Pattern Marching

The Lookup Server can provide a tool to help users to design objects' messages. The users do not need to know messages' patterns and object's templates. When users input a set of attributes, the Lookup Server can intelligently find the templates most suitable for the objects, and then dynamically generate a message pattern. The proposed architecture also has the intelligence to reason messages' pattern. For example, if the IoT system receives a message without "Template ID", it can intelligently match the message to an appropriate object template by analysing the message's attributes.

C. Unit Ontology

```
- <Temperature>
  - <Unit>
      <UnitId>U004</UnitId>
      <Name>Centrigrade</Name>
    </Unit>
  - <Unit>
      <UnitId>U005</UnitId>
      <Name>Fahrenheit</Name>
    </Unit>
</Temperature>
```

Fig. 9 Unit Ontology

```
Temperature Monitor
Service Type:   TEM2371
Input Format:   U0005
Output Format:  Strings
Threshold:      -5
<Unit>U0005</Unit>
Constrains:     ……
Description:    ……
```

Fig. 10 Temperature Monitor's template

The field "<unit>"in the Fig. 8 refers to Unit Ontology which is used to describe measure units for object's attributes. By removing measure units from messages, messages' size can be reduced further, and to guarantee the consistency of units' semantic meaning as well. The Fig. 9 demonstrates a XML schema to describe the Unit Ontology for temperature.

D. Service Ontology

The field "Services" in the Fig. 8 contains the Services IDs which are linked to Service Ontology. The purpose of Service Ontology is to describe and constrain the service types for different objects. The relationship between Entity Ontology and Service Ontology is many to many. One type of object can use multiple services, and one service can also support multiple types of objects. However, one service can only support limited types of objects. For example, the service of Speed Monitor used for vehicles cannot be utilized for milk.

The refrigerator supports three services: Temperature monitor will send a notification to users if the refrigerator's temperature is below a defined threshold; Voltage Monitor can watch if the refrigerator is working within a normal voltage range. Expiration manager can check the expiration date of the food in the refrigerator, and then send notifications to users if any food is expired. The milk supports two types of services: Temperature Monitor and Expiration Manager. Fig. 10 shows the service template for Temperature Monitor.

Users are also allowed to customize services. The milk and refrigerator both support the service of Temperature Monitor. By editing the threshold value on their service templates, the Temperature Monitor can be reused by different objects.

If objects are added with new attributes, the new generated sub-templates can also support the services inherited from their parent-templates. In this case, the services for the object FOOD6862 and APP3468 are inherited from the generic object templates used to describe the refrigerator and milk. Therefore, in this architecture, the Entity Ontology can be expanded indefinitely without losing the inherited services.

E. Service Compositor

The Service Compositor is used to generate and deliver combined services based on users' requirements. As the proposed architecture is loosely-coupled, it is able to create various new services by combining existing services.

In this case, the component of Expiration Manager can check if the milk is expired by comparing the expiration date with the refrigerator's local data. To deliver the combined service, the Service Compositor can intelligently bind the components of refrigerator and milk with the component of Expiration Manager. By building the connections among three components, the Expiration Manager can receive the messages from the milk and the refrigerator. Then the two dates from the two objects can be compared by interpreting the date fields in two messages. The Fig.11 demonstrates the process.

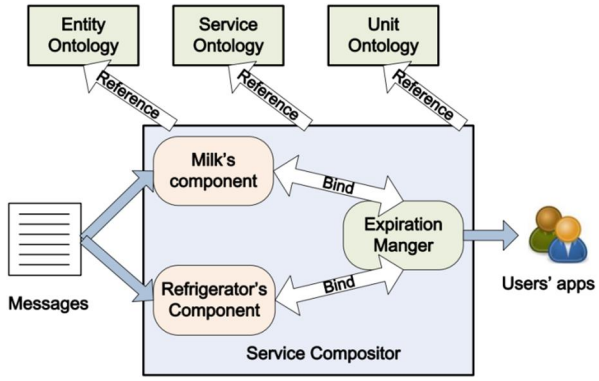

Fig. 11 The process of service composition

V. EVALUATION

The proposed generic architecture is suitable for many domains. As this IoT architecture is device-independent, any type of sensory device can support it. This section outlines the implementation details of the proof-of-concept prototype based on the case study in Section IV. All resources in this evaluation are open-source and representative.

A. Experiment Setup

An Arduino UNO, a Zigbee Xbee and several electronics were combined together to simulate a generic wireless sensor or a RFID tag. A ZigBee USB explorer was used to simulate a gateway. It can collect messages and then route the aggregated messages to the IoT. LooCI-OSGi v1.0 was selected as the middleware. The LooCI components can be reconfigured on runtime, which is especially suitable for the dynamic states of the IoT. Compared with other middleware such as OpenCOM, LooCI supports multiple threads among multiple components. To provide combined services, the proposed architecture also needs to support multiple components communication.

A standard dual-core PC with Windows XP was used as the Proxy, and a USB port was connected to the gateway to route aggregated messages. Openjdk 1.7 was installed to design the Component Factory and Service Compositor.

XML was used to describe the ontologies and templates. In this implementation, four templates were designed including two object templates for the refrigerator and the milk, and two service templates for Expiration Manager and Temperature monitor. The templates are illustrated in the Fig. 8 and Fig. 10.

The ontology was developed based on the core concept of proposed architecture. It uses a hierarchal structure links to its sub-ontologies including Object Ontology, Service Ontology and Unit Ontology. These ontologies are inter-connected by referencing Template IDs to form a mesh network.

B. Building Ontologies

Previous work [14] has demonstrated dynamic generation of sensor components in proxies to integrate heterogeneous sensors/RFIDs. The main purpose of this evaluation is to investigate the feasibility of building ontologies for the IoT. The process of this evaluation was based on the case study in section IV. Two messages in Fig. 6 were written into sensors to describe the refrigerator and the milk. These messages were sent to the gateway every two seconds. In the LooCI middleware, two object components were generated to map the messages from two objects, and two service components were deployed to simulate the service of Expiration Manager and Temperature Monitor. The Service Compositor bound these components together to deliver the combined services.

C. Evaluation Results

The first evaluation is to test the Expiration Manager. The milk's expiration date was configured as "09-06-2012" in messages, while the local date of refrigerator was set as "16-06-2012". As " 09-06-2012" is before "16-06-2012", the screen displayed a notification of "Your milk is still fresh". By changing the milk's expired data to "18-06-2012", and then the notification turned into "Your milk is expired".

The second evaluation is used to validate the Temperature Monitor. If the refrigerator's temperature is above the predefined threshold "-2", a notification of "Your refrigerator is too hot" will be sent. Otherwise, the notification is "Your refrigerator works well". By adjusting the temperature value on the sensor, the evaluation result also turned as expected.

C. Reducing Messages' Size

Reducing messages' size can achieve energy saving for battery-driven devices. The proposed IoT architecture can minimize messages' size to fit lightweight sensors/RFIDs. Assuming the messages from the milk and refrigerator are encoded based on ASCII, the messages' size can be measured by using a ASCII calculator [16]. The Fig 12 shows the proposed architecture (MOCB) can dramatically reduce messages' size compared with using XML and JSON to describe objects.

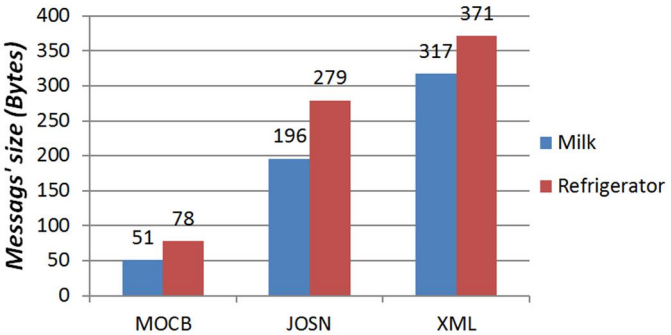

Fig. 12 Messages' size in different micro formats

VI. CONCLUSION

This paper proposes a generic architecture for building ontologies for the IoT. It can integrate any type of sensory device in a component-based middleware. The minimized messages can fit resource-constrained devices, such as passive RFID tags. By building the ontologies in the IoT system, the semantic meaning of received messages can be interpreted, and generate logical components for physical objects and services. These components are bound together on event bus to deliver various combined services. The proposed architecture provides a mechanism for third-parties to customize objects and services. The evaluation demonstrates that it is feasible to design such a generic IoT architecture based on existing infrastructure and communication standards.

REFERENCE

[1] R. B. Kranenburg, D. Caprio, E. Anzelmo, S. Dodson, A. Bassi, and M. Ratto, "The Internet of Things " presented at the 1st Berlin Symposium on Internet and Society, Berlin, 2011.

[2] A. Iera, C. Floerkemeiea, J. Mitsugi, and G. Morabito, "The Internet of Things," *IEEE Wireless Communications,* vol. 17, 2010.

[3] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and Challenges for realising the Internet of Things* European Commission - Information Society and Media DG, 2010.

[4] F. N. Eduardo, A. F. L. Antonio, and C. F. Alejandro, "Information fusion for wireless sensor networks:Methods, models, and classifications," *ACM Computing Surveys,* vol. 39, 2007.

[5] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the Internet of Things," in *Computer Science and Information Systems (FedCSIS),* 2011, pp. 949-955.

[6] C. Barbero, P. D. Zovo, and B. Gobbi, "A Flexible Context Aware Reasoning Approach for IoT Applications," in *Mobile Data Management (MDM),* 2011, pp. 266-275.

[7] G. M. Lee and N. Crespi, "Shaping future service environments with the cloud and internet of things: networking challenges and service evolution," Leveraging applications of formal methods, verification, and validation, vol.1, Heraklion, Crete, Greece, 2010.

[8] D. Uckelmann and M. Harrison, *Architecting the Internet of Things*. Heidelberg, Germany: Springer, 2011.

[9] S. Bowers, J. S. Madin, and M. P. Schildhauer, "A Conceptual Modeling Framework for Expressing Observational Data Semantics," Conceptual Modeling, Barcelona, Spain, 2008.

[10] H. Petritsch., 2005, Service-Oriented Architecture (SOA) vs. Component Based Architecture. Available: http://worldcolleges.info/worldcolleges_new/sites/default/files/SOA_vs_component_based.pdf

[11] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks," Embedded network sensor systems, Raleigh, NC, USA, 2008.

[12] J. W. Hui and D. E. Culler, "IP is dead, long live IP for wireless sensor networks," Embedded network sensor systems, 2008.

[13] P. Costa, G. Coulson, R. Gold, M. Lad, C. Mascolo, L. Mottola, G. Picco, T. Sivaharan, N. Weerasinghe, and S. Zachariadis, "The RUNES Middleware for Networked Embedded Systems and its Application in a Disaster Management Scenario," in *Pervasive Computing and Communications,* 2007, pp. 69-78.

[14] W. Wang, K. Lee, and D. Murray, " Integrating Sensors with the Cloud using Dynamic Proxies," presented at the 23rd Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2012), Sydney, 2012.

[15] K. Lee, D. Murray, D. Hughes, and W. Joosen, "Extending sensor networks into the Cloud using Amazon Web Services," in *Networked Embedded Systems for Enterprise Applications*, 2010.

[16] D. Hughes, P. Greenwood, G. Blair, G. Coulson, P. Grace, F. Pappenberger, P. Smith, and K. Beven, "An experiment with reflective middleware to support grid-based flood monitoring," *Concurr. Comput. : Pract. Exper.,* vol. 20, pp. 1303-1316, 2008.