DETECTING AND FILTERING THE WEB SERVICES REDUNDANCY IN WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

MUTHULAKSHMI G.

in partial fulfillment for the requirement of award of the degree of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY,

COIMBATORE 641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2013

BONAFIDE CERTIFICATE

Certified that this project work titled **"DETECTING AND FILTERING THE WEB SERVICE REDUNDANCY IN WIRELESS SENSOR NETWORKS"** is the bonafide work of Ms. MUTHULAKSHMI G. (1120108009) who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other students.

Prof. N. JAYAPATHI, M.Tech.,	Dr.V.Vanitha M.E., P.hD.,
HEAD OF THE DEPARTMENT	SUPERVISOR
Professor	Senior Associate Professor
Dept. of Computer Science and Engineering	Dept. of Computer Science and Engineering
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore- 641 049	Coimbatore- 641 049

Submitted for the Project Viva-Voce examination held on _____

---------- ----------

Internal Examiner External Examiner

ABSTRACT

Wireless Sensor Networks (WSNs) are attractive for monitoring and gathering physical information (e.g. temperature) via lots of deployed sensors. For the applications in WSNs, web service is one of the recommended frameworks to publish, invoke, and manage services. However, the standard web service description language (WSDL), defines only the service input and output while ignoring the corresponding input-to-output mapping relationships. This presents a serious challenge in distinguishing services with similar input and output interface. To address this challenge the service policy is embedded into the traditional WSDL2.0 schema to describe the input-to-output mapping relationships. Furthermore, a new service redundancy detection approach is proposed based on this similarity. Finally, the case study and experimental analysis illustrate the applicability and capability of the proposed service redundancy detection approach.

ஆய்வுச்சுருக்கம்

அதிக இடங்களில் வைக்கப்பட்ட உணரி மூலமாக நேரடி மற்றும் மறைமுக இயற்பியல் தகவல்களை பெற கம்பியற்ற தகவல் தொடர்பு உணரி வலையமைப்பு பயன்படுகிறது. கம்பியற்ற தகவல் தொடர்பு உணரியின் வேலைகளைச் செய்வதற்கு வலை சேவை என்பவை தகவல்களை வெளியிட, செயலாக்க மற்றும் நிர்வகிக்க பரிந்துரைக்கப்பட்ட கட்டமைப்புகளில் ஒன்றாகும். இப்படி இருந்தபோதிலும் நிலையான வலை சேவை விளக்க மொழி உள்ளீடு மற்றும் வெளியீடுகளை வரையறுக்க மட்டுமே பயன்படுகிறது. இவை உள்ளீடு மற்றும் வெளியீடுகளுக்கு இடையே உள்ள தொடர்புகளை வரையறுக்க பயன்படுவதில்லை. மேற்கூறியவை ஒருமித்த உள்ளீடு மற்றும் வெளியீடுகளை பிரித்தெடுப்பதில் தீவிர சவாலாக இருக்கின்றது. இந்த சவாலை முறியடிக்க சேவை கொள்கையை வலை சேவை விளக்க மொழியுடன் உட்பொதியப்படுவதன் மூலம் உள்ளீடு மற்றும் வெளியீடுகளுக்கு இடையே உள்ள தொடர்பை விவரிக்க முடிகிறது. சேவை கொள்கையை, சேவை இரும மரமாக மாற்றப்படுவதன் மூலமாக வெவ்வேறு வலை சேவைகளுக்கு இடையே உள்ள ஒற்றுமைகளை மதிப்பீடு செய்யமுடிகிறது. இந்த ஒற்றுமையின் அடிப்படையில் ஒரு பணி நீக்க கண்டறிதல் அணுகுமுறை பரிந்துரைக்கப்படுகிறது. இறுதியாக வழக்கு ஆய்வு மற்றும் பரிசோதனை பகுப்பாய்வின் மூலமாக பணிநீக்கக் கண்டறிதல் அணுகுமுறையை விளக்குகின்றன.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project. I express my profound gratitude to **Padma Bhusan Arutselvar Dr.N.Mahalingam B.sc., F.I.E. Chairman, Dr.B.K.Krishnaraj Vanavarayar B.com., B.L., Co-Chairman, Mr.M.Balasubramaniam M.com., M.B.A., Correspondent Mr.K.Sankar Vanavarayar M.B.A., PGDIEM, Joint Correspondent** and **Dr.S.Ramachandran Ph.D., Principal** for providing the necessary facilities to complete my project.

I take this opportunity to thank **Prof.N.Jayapathi M.Tech.,** Head of the Department**,** Department of Computer Science and Engineering, for his support and timely motivation.

Special thanks to my Project Coordinator and Guide **Dr.V.Vanitha M.E., Ph.D.,** Senior Associate Professor, Department of Computer Science and Engineering, and project committee members for arranging brain storming project review sessions. I am grateful for her support, encouragement and ideas. I would like to convey my honest thanks to all **Teaching and Non Teaching staff** members of the department and my classmates for their support.

I dedicate this project work to **my Parents** for no reasons but feeling from bottom of my heart, without their love this work would not be possible.

- MUTHULAKSHMI G.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	LIST OF TABLES	**ix**
	LIST OF FIGURES	**x**
	LIST OF ABBREVIATIONS	**xi**
1	**INTRODUCTION**	**1**
	1.1 OVERVIEW OF SENSOR NETWORKS	1
	1.1.1 Sensor Nodes	2
	1.2 WIRELESS SENSOR NETWORK ARCHITECTURE	4
	1.2.1 Layered Architecture	4
	1.2.2 Clustered Architecture	6
	1.3 DESIGN CHALLENGES OF WSN	8
	1.4 WEB SERVICS IN WSN	12
	1.4.1 Benefits of Web Service	12
	1.5 WEB SERVICE ARCHITECTURE	13
	1.5.1 Working of Web Service	14
	1.5.2 SOAP (Simple Object Access Protocol)	15
	1.5.3 UDDI (Universal Description, Discovery and Integration)	15
	1.5.4 WSDL (Web Service Description Language)	16
	1.5.4.1 WSDL Elements	17
	1.6 LITERATURE SURVEY	19
	1.6.1 Using Web Service for Wireless Sensor Networks Applications	19

	1.6.2 The Design and Implementation of a Web Service Framework for Individual Nodes in Sinkless Wireless Sensor Networks	20
	1.6.3 A Model-Driven WSDL Extension for describing the Qos Of Web Services	20
	1.6.4 Web Services Policy Framework (Ws- Policy)	21
	1.6.5 Semantic Structure Matching for Assessing Web-Service Similarity	23
	1.6.6 An Extension of WSDL for Flexible Web Service Invocation with Large Data	24
	1.6.7 Mining Related Queries from Search Engine Query Logs	25
	1.7 PROBLEM DEFINITION	27
2	**IMPLEMENTATION**	**29**
	2.1 PROPOSED SYSTEM	29
	2.2 MODULE DESCRIPTION	33
	2.2.1 Web Service and Policy Description	33
	2.2.2 Service Redundancy Detection	34
	2.2.2.1 IO Similarity	34
	2.2.2.2 Policy Similarity	34
3	**RESULTS AND ANALYSIS**	**36**
	3.1 EXPERIMENTATION	36
	3.2 SYSTEM SPECIFICATION	36
	3.2.1 Hardware Requirements	36
	3.2.2 Software Requirements	37
	3.3 SNAPSHOTS	37

	3.4 DATABASE DESGIN	39
	3.5 SIMILARITY COMPUTATION	40
	3.6 CONCLUSION AND FUTURE WORK	42
	APPENDIX	**43**
	REFERENCES	**59**
	LIST OF PUBLICATIONS	**61**

LIST OF TABLES

Table No.	Caption	Page No.
1.1	WSDL Elements	18
3.1	Policy table	39

LIST OF FIGURES

FIGURE NO.	CAPTION	PAGE NO.
1.1	WSN Architecture	1
1.2	Components of a sensor node	2
1.3	Layered Architecture	4
1.4	Clustered Architecture	6
1.5	Service Oriented Architecture	14
1.6	WSDL 2.0 Structure	28
2.1	Extended WSDL 2.0 Structure	31
2.2	Service Redundancy Detection	32
3.1	Web Service and Policy Description	37
3.2	Update of services	38
3.3	Filtration of redundant services	39
3.4	Similarity Computation without using policy	40
3.5	Similarity Computation with using policy	41

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital Convertor
QOS	Quality of Services
RF	Radio Frequency
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
WSDL	Web Service Description Language
WSN	Wireless Sensor Network
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF SENSOR NETWORKS

A Wireless Sensor Network (WSN) is a self-configuring network of small sensor nodes communicating among themselves using radio signals, and deployed in quantity to sense, monitor and understand the physical world (I.F Akyildiz 2002). Wireless Sensor Network Architecture is shown in Fig. 1.1 which consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location.

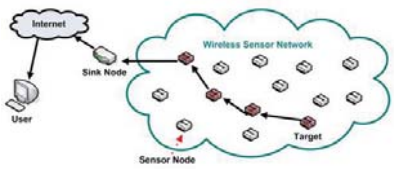

Fig. 1.1 WSN Architecture

The more modern networks are bi-directional, enabling also to control the activity of the sensors. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance, today such

Networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring and so on.

1.1.1 Sensor Nodes

The WSN is built of nodes from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts (Macros Augusto M. 2003), a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A wireless sensor node is composed of four basic components as shown in Fig. 1.2 a sensing unit, a processing unit (microcontroller), a transceiver unit and a power unit.

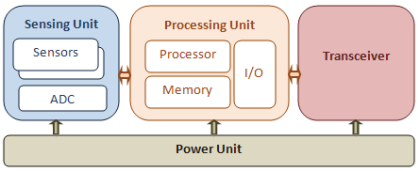

Fig. 1.2 Components of Sensor Node

In addition to the above units, a wireless sensor node may include a number of application-specific components, for example a location detection system or mobiliser, for this reason, many commercial sensor node products include expansion slots and support serial wired communication.

Sensing Unit

The main functionality of the sensing unit is to sense or measure physical data from the target area. The analog voltage or signal is generated by the sensor corresponding to the observed phenomenon. The continual waveform is digitized by an analog-to-digital converter (ADC) and then delivered to the processing unit for further analysis. The sensing unit is a current technology bottleneck because the sensing technologies are much slower than those of the semi-conductors.

Processing Unit

The processing unit which is generally associated with a small storage unit manages the procedures that make the sensor nodes collaborate with the other nodes to carry out the assigned sensing tasks.

Transceiver

There are three deploying communication schemes in sensors including optical communication (laser), infrared, and radio-frequency (RF). Laser consumes less energy than radio and provides high security, but requires line of sight and is sensitive to atmospheric conditions. Infrared, like laser, needs no antenna but is limited in its broadcasting capacity. RF is the most easy to use but requires antenna. Various energy consumption reduction strategies have been developed such as modulation, filtering, and demodulation. Amplitude and frequency modulation are standard mechanisms. Amplitude modulation is simple but susceptible to noise.

Power Unit

One of the most important components of a sensor node is the power unit. Every sensor node is equipped with a battery that supplies power to remain in

active mode. Power consumption is a major weakness of sensor networks. Any energy preservation schemes can help to extend sensor's lifetime. Batteries used in sensors can be categorized into two groups; rechargeable and non-rechargeable. Often in harsh environments, it is impossible to recharge or change a battery.

1.2 WIRELESS SENSOR NETWORKS ARCHITECTURE

The architecture of wireless sensor networks (David Culler 2004) is classified into two types.

- Layered Architecture
- Clustered Architecture

1.2.1 Layered Architecture

In this type of architecture there is a single powerful base station (BS) and layers of sensor nodes are formed around BS based on their hop count distance to reach BS. Therefore, in general layer i denote all nodes that are i-hop away from BS. Layered architecture is depicted in Fig. 1.3.

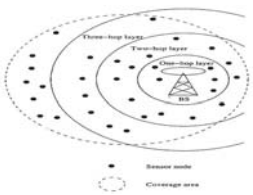

Fig. 1.3 Layered Architecture

Unified Network Protocol Framework (UNPF)

It is a type of layered architecture with a set of protocols that integrates the following operations such as Network Initialization and Maintenance Protocol, MAC Protocol and Routing Protocol

Network Initialization and Maintenance Protocol

BS broadcasts its ID using CDMA common control channel (BS reaches all nodes in one hop) .The nodes record BS ID & send beacon signal with their own IDs at their low default power levels. All nodes and the BS can hear are at 1-hop distance. The BS broadcasts a control packet with all layer one node IDs. All nodes send a beacon signal again. The layer one nodes record the IDs they hear-layer 2.The layer one nodes inform the BS of the layer 2.The BS broadcasts the layer2 nodes IDs .To maintain periodic beaconing updates are required.

MAC Protocol

A Time Division CDMA (TCDMA) protocol for spatial bandwidth reuses and ensures a scheduling scheme for fair access.

Routing Protocol

Downlink from the BS is by direct broadcast on the control channel. It enables multi-hop data forwarding to the BS. The remaining energy is considered when forwarding to the next hop (layer). Only the nodes of the next layer need to be maintained in the routing table.

1.2.2 Clustered Architecture

In this type of architecture sensor nodes are organized into clusters and each cluster is governed by a cluster-head. Only cluster heads send messages to a BS. This architecture is suitable for data fusion and is self-organizing in nature. This is depicted in Fig. 1.4.

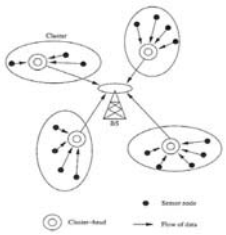

Fig. 1.4 Clustered Architecture

Low-Energy Adaptive Clustering Hierarchy (LEACH)

It is a self-organizing and adaptive clustering protocol which evenly distributes the energy expenditure among the sensors. It performs data aggregation where cluster heads act as aggregation points. There are two main phases in this architecture.

- Setup phase: organizing the clusters.
- Steady-state phase: deals with the actual data transfers to the BS.

Setup phase

Each sensor chooses a random number m between 0 and 1. From the equation 1.1, if $m < T(n)$ for node n, the node becomes a cluster-head where

$$T(n) = \begin{cases} \dfrac{P}{1 - P[r * \bmod(1/P)]} & \text{if } n \in G \\ 0 & \text{otherwise,} \end{cases} \qquad (1.1)$$

P: the desired percentage of cluster heads.

r: the round number.

G: the set of nodes that have not been cluster heads during the last $1 / P$ rounds.

A cluster head advertises its neighbors using a CSMA MAC. Surrounding nodes decide which cluster to join based on the signal strength of these messages. Cluster heads assign a TDMA schedule for their members.

Steady-state phase

All source nodes send their data to their cluster heads. Cluster heads perform data aggregation/fusion through local transmission. Then cluster heads send them back to the BS using a single direct transmission. After a certain period of time, cluster heads are selected again through the set-up phase.

Merits

It accounts for adaptive clusters and rotating cluster heads and provide opportunity to implement any aggregation function at the cluster heads.

Demerits

It provides highly dynamic environments and also has continuous updates.

1.3 DESIGN CHALLENGES OF WSN

The unique network characteristics (Marcos Augusto M 2003) present many challenges in the design of sensor networks, which involve the following main aspects.

i. **Scalable and flexible architecture**

The network must preserve its stability. Introducing more nodes into the network means that additional communication messages will be exchanged, so that these nodes are integrated into the existing network.

ii. **Error-prone wireless medium**

The wireless medium can be greatly affected by noisy environments.

iii. **Fault tolerance and adaptability**

Fault tolerance means to maintain sensor network functionalities without any interruption due to failure of sensor node because in sensor network every node have limited power of energy so the failure of single node doesn't affect the overall task of the sensor network.

iv. **Infrastructure**

Sensors network are infrastructure less in which nodes can communicate directly with base station.

v. **Node Deployment**

Sensor network can be deployed randomly in geographical area. After deployment, they can be maintained automatically without human presence.

vi. **Real–time**

Achieving Real-time in WSN is difficult to maintain. It must support maximum bandwidth, minimum delay and several QOS parameters.

vii. **Dynamic changes**

As in sensor network nodes are deployed without any topology and they are adaptable to changes due to addition of new nodes or failure of nodes.

viii. **Power Consumption**

Wireless sensor node is microelectronic device means it is equipped with a limited number of power source. Nodes are dependent on battery for their power. Hence power conservation and power management is an important issue in wireless sensor network.

ix. Production cost

As the name suggests production cost, it is known that in the sensor network, large no of nodes can be deployed, so if a single node will be very high then the cost of overall network will be very high.

x. Short Range Transmission

In WSNs, the transmission range should be considered as short in order to reduce the possibility of being eavesdropped.

xi. Hardware design

While designing any hardware of sensor network, it should be energy-efficient.

xii. **Limited computational power and memory size**

It is another factor that affects WSN in the sense that each node stores the data individually and sometime more than one node stored same data and transferred to the base station which wastes the power and storing capacity of nodes so must develop effective routing schemes and protocols to minimize the redundancy in the network.

xiii. **Quality of Service**

It means data should be delivered within time period.

xiv. **Security**

Security is very important parameter in sensor network since sensor networks are data centric so there is no particular id associated with sensor nodes and attacker can easily inserted himself into the network and stole the important data by becoming the part of network without the knowledge of sensor nodes of the network. So it is difficult to identify whether the information is authenticated or not.

For many applications in wireless sensor networks, users may want to continuously extract data from the networks for analysis later. However, accurate data extraction is difficult it is often too costly to obtain all sensor readings, as well as not necessary in the sense that the readings themselves only represent samples of the true state of the world. In order to enable reliable and efficient observation and initiate right actions, physical phenomenon features should be reliably detected/estimated from the collective information provided by sensor nodes. Moreover, instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. Hence, these properties of WSN impose unique challenges for development of communication protocols in such architecture.

1.4 WEB SERVICES IN WSN

For Service-Oriented Architectures, web services are claimed as state of the art to connect business execution layers as well as networking devices. Additionally, the deployment of Wireless Sensor Networks became applicable over the last years. The usage of application layer gateways and proxy concepts allow the integration of these sensor networks into real world scenarios and existing networks that make use of web services.

Web services (Cubera 2005) is a technology that allows applications to communicate with each other in platform- and programming language-independent manner. A Web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another web service.

1.4.1 Benefits of Web Services

i. Loosely Coupled

Each service exists independently of the other services that make up the application. Individual pieces of the application will be modified without impacting unrelated areas.

ii. Ease of Integration

Data is isolated between applications creating silos (a structure for storing bulk data). Web services act as glue between these and enable easier communications within and across organizations.

iii. Service Reuse

Takes code reuse a step further. A specific function within the domain is only ever coded once and used over and over again by consuming applications.

iv. **Interoperability**

Web services allow different applications to talk to each other and share data and services among themselves. Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, a web service is used to make the application platform and technology independent.

v. Low Cost of communication

Web services uses SOAP over HTTP protocol for the communication, so can use existing low cost internet for implementing Web services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Beside SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP etc.

1.5 WEB SERVICES ARCHITECTURE

The simplest web service system has two participants

- A service producer (provider)
- A service consumer (requester).

The provider presents the interface and implementation of the service and the requester uses the web service. The service oriented architecture is show in Fig. 1.5 which describes about the relationship among the web services registry, services provider and the web services consumer.

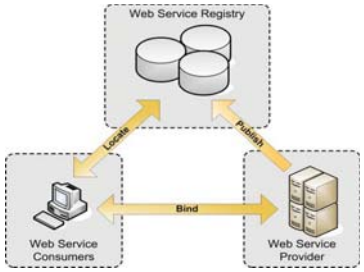

Fig. 1.5 Services Oriented Architecture

A more sophisticated system provides a registry which acts as a broker for web services, a provider which can publish services to the registry and a consumer who can then discover services in the registry.

1.5.1. Working of Web Service

The basic web services platform is XML + HTTP. XML provides a language which can be used between different platforms and programming languages and still express complex messages and functions. The HTTP protocol is the most used Internet protocol.

Web services platform elements:

- Simple Object Access Protocol (SOAP)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)

1.5.2 Simple Object Access Protocol (SOAP)

SOAP stands for "Simple Object Access Protocol", a relatively new protocol for distributed applications developed by Microsoft, IBM, DevelopMentor, and UserLand. SOAP is highly flexible and can support different applications however; the most important one is to enable remote procedure calls (RPC) over HTTP using XML. SOAP is an XML-based object invocation protocol and was originally developed for distributed applications to communicate over HTTP and through corporate firewalls and it was meant to access services, objects and servers in a platform-independent manner.

1.5.3 Universal Description, Discovery and Integration (UDDI)

The UDDI specification describes a registry of web services and its programmatic interfaces. UDDI itself is a set of web services. The UDDI specification defines services that support the description and discovery of Businesses, organizations and other providers of web services. The technical interfaces which may be used to access and manage those services.

Conceptually, a business can register three types of information into a UDDI registry. The specification does not call out these types specifically, but they provide a good summary of what UDDI can store for a business:

i. **White pages**

Basic contact information and identifiers about a company, including business name, address, contact information and unique identifiers such as D-U-N-S (Data Universal Numbering System) numbers or tax IDs. This information allows others to discover web service based upon business identification.

ii. **Yellow pages**

The Information describes a web service using different categorizations (taxonomies). This information allows others to discover web service based upon its categorization (such as being in the manufacturing or car sales business).

iii. **Green pages**

Technical information describes the behaviors and supported functions of a web service that hosted by business. This information includes pointers to the grouping information of web services and where the Web services are located.

1.5.4 Web Services Description Language (WSDL)

WSDL is an XML-based language for locating and describing web services. WSDL stands for Web Services Description Language. It is based on XML.WSDL is a W3C standard. It specifies the location of the service and the operations (or methods) the service exposes.

A WSDL definition is an XML document with a root definition element namespace. The definitions element may contain several other elements including types, message, portType, binding, and service, all of which come from the wsdl namespace.

The following illustrates the basic structure of a WSDL definition:

```
<!-- WSDL definition structure -->
<definitions
name="MathService"
targetNamespace="http://example.org/math/"
xmlns="http://schemas.xmlsoap.org/wsdl/" >

  <!-- abstract definitions -->
   <types> ...
   <message> ...
  <portType> ...

  <!-- concrete definitions -->
  <binding> ...
  <service> ...
</definition>
```

1.5.4.1 WSDL Elements

WSDL2.0 has four key elements such as types, interface, binding and service which are described in Table 1.1. The types element defines the data types used by the exchanged messages. The interface element encloses a set of abstract operations and messages. The binding element defines the underlying format for the transportation of messages, and each binding in the WSDL refers to an interface.

The service element describes a set of endpoints that point out the network address for each binding element.

WSDL Extension is based on the four elements involved in the service input, output, and service function. The mapping relations from the elements to service input, output and service function can be analyzed. The types and interface elements mainly describe the service input and output, including data types, message names, and message exchange sequences. The service element specifies an access address for the service function.

Table 1.1 WSDL Elements

Element Name	Description
Types	A container for abstract type definitions defined using XML Schema
Message	A definition of an abstract message that may consist of multiple parts, each part may be of a different type
portType	An abstract set of operations supported by one or more endpoints (commonly known as an interface); operations are defined by an exchange of messages
Binding	A concrete protocol and data format specification for a particular portType
Service	A collection of related endpoints, where an endpoint is defined as a combination of a binding and an address (URI)

1.6 LITERATURE SURVEY

The literature survey is done related to bridging up end-user applications and wireless sensor networks using web services. From that, redundancy in web service using redundancy detection and filtering technique were studied.

1.6.1 USING WEB SERVICES FOR WIRELESS SENSOR NETWORKS APPLICATIONS

The authors (Othman 2006) devoted to the use of Web services for bridging end-user applications and wireless sensors networks. Its first contribution is a systematic evaluation of the current frameworks for the interactions between end-user applications and wireless sensors networks. The evaluation shows the potential of web services, compared to the other frameworks and motivates their usage in the case study. The second contribution is the definition of web services for the I-centric telecommunication services, and their implementation in a wireless sensor network that does not support web services. The trend in WSN is towards allowing applications from different domains to make use of sensor data. Developers are not expected to be sensor experts. It is foreseen that there will be numerous sensor networks, offering a variety of services, the owners of which may choose to charge for them, directly or via service providers. Database, Mobile Code and web services fully provide high level of abstraction while, as expected, low level APIs do not. Web services and Database approach are very familiar to most developers and can easily be integrated into applications. Mobile Code is a

paradigm not widely adopted, and, the APIs generally require knowledge of sensors and/or learning a specialized language.

Web Services is easy to integrate with different applications. Web Services suffer from performance problems as they introduce significant overhead due to SOAP processing and XML encodings.

1.6.2 THE DESIGN AND IMPLEMENTATION OF A WEB SERVICE FRAMEWORK FOR INDIVIDUAL NODES IN SINKLESS WIRELESS SENSOR NETWORKS

The authors (Glitho 2007) presented an embedded web services platform residing on the sensor nodes, allowing direct interaction between applications and individual sensor nodes. In a web services based sinkless architecture, the sensor nodes hosts the web services and the UDDI can be a separate standalone entity or distributed over several nodes as an overlaid entity. TinyWS is a small web service platform that resides on the sensor nodes. It hosts the web services and has SOAP processing engine. Thus it can receive a SOAP message, extract the actual request from it and invoke the relevant web service that can act upon the request. It can also receive response from the invoked web service, package the response as a SOAP message and send it back to the original requesting application. SOAP is carried over HTTP, an application layer protocol carried on the TCP/IP stack. This implies that a web services platform, such as TinyWS, should be built over TCP/IP and should handle HTTP as a transport protocol for SOAP. It provides high level of abstraction and support for heterogeneity. Web Services performances is an issue.

1.6.3 A MODEL-DRIVEN WSDL EXTENSION FOR DESCRIBING THE QOS OF WEB SERVICES

The authors (D'Ambrogio 2006) proposed Q-WSDL extension can effectively be used to specify QoS requirements, to establish service level agreements (SLA), to add QoS-oriented characteristics when querying registries of web services and to support the automated mapping from WSDL documents to Q-WSDL ones and from UML models to Q-WSDL web services. The quality of service (QoS) characteristics of web service introduces a lightweight WSDL extension to transform the WSDL metamodel into the QoS-enabled WSDL (Q-WSDL) metamodel. The WSDL metamodel is derived from the WSDL XML Schema, and then transforming it into a QoS-enabled WSDL (Q-WSDL) metamodel, from which the Q-WSDL XML Schema is derived. Indeed, an XML Schema defines the WSDL language in the same respect as a metamodel is used to define a model. Representing the WSDL grammar in terms of a metamodel allows enhancing its comprehensibility and facilitating its extension. The WSDL and Q-WSDL metamodels are defined by use of the Meta Object Facility *(*MOF*)*, the Object Management Group's (OMG) standard for specifying technology neutral metamodels, or models used to describe other models. A WSDL document consists of a set of definitions that describe what a service does (the operation it provides), how a service is acceded (data formats and protocols) and where it is located (network address). Such definitions are specified in the WSDL XML Schema which has been used to identify the classes and associations of the WSDL metamodel.

It supports the automated mapping from UML models to services in Q-WSDL. It specifies QoS requirements of web services and supports the automated mapping from WSDL documents to Q-WSDLs.

1.6.4 WEB SERVICES POLICY FRAMEWORK (WS-POLICY)

The authors (Bajaj 2004) proposed a WS-Policy which defines a framework for allowing web services to express their constraints and requirements.

Such constraints and requirements are expressed as policy assertions. This document defines a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in WSS: SOAP Message Security, WS-Trust and WS-SecureConversation. This document takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages. WS-Policy is emerging as a standard way to describe the properties of a web service. Based on this specification, the functional aspects (e.g., constraints on exchanged data) and non-functional aspects (e.g., security, message reliability) of a service can be tied to a set of assertions. An assertion is defined as ‘‘an individual preference, requirement, capacity or other properties’’, and the WS-Policy document is responsible for composing such assertions to identify how a web service should work. The WS-Policy aims to increase the functional or non-functional property information. The goal of WS-Policy is to provide the mechanisms needed to enable web services applications to specify policy information. Specifically, this specification defines the following

- An XML Infoset called a policy expression that contains domain specific, web service policy information.
- A core set of constructs to indicate how choices and/or combinations of domain specific policy assertions apply in a web services environment.

WS-Policy is designed to work with the general web services framework, including WSDL service descriptions and UDDI service registrations. Security is a major issue in web service.

1.6.5 SEMANTIC STRUCTURE MATCHING FOR ASSESSING WEB-SERVICE SIMILARITY

The authors (Wang 2003) described and experimentally evaluated service- similarity assessment methods in WSDL. A set of WSDL similarity-assessment methods, which can be used, in conjunction with the current UDDI API, to support a more automated service-discovery process, by distinguishing among the potentially useful and the likely irrelevant services and by ordering the candidates in the first category according to their relevance to the task at hand. This method utilizes both the semantics of the identifiers of WSDL descriptions and the structures of their operations, messages and types to assess the similarity of two WSDL files. Given only a textual description of the desired service, a semantic information-retrieval method can be used to identify and order the most similar service-description files. This step assesses the similarity of the provided desired-service description, extended to include semantically similar words according to WordNet, with the available services. If a (potentially partial) specification of the desired service behavior is also available, this set of likely candidates can be further refined by a semantic structure matching step assessing the structure and semantic similarity of the desired vs. the retrieved services. The WordNet-powered vector-space model extension involves the maintenance of three sub-vectors for each document and query: stems of original words in a document, stems of words' synonyms for all word senses, and stems of words' direct hypernyms, hyponyms and siblings for all word senses. All document terms' word senses are included, and therefore bypass the problem of lacking effective automated word sense disambiguation techniques. The WSDL syntax allows textual descriptions for services, their types and operations, grouped under <documentation> tags. Given a natural language description of the desired service, the WordNet-powered vector

space model to retrieve published WSDL services that are most similar to the input description on the respective vectors. Corresponding sub-vectors from documents and queries are matched and obtain three similarity scores accordingly. Different weights are assigned to sub-vector matching scores: matching scores of original word stems (first sub-vectors) are assigned twice the weight assigned to matching scores of synonyms (second sub-vectors), hypernyms, hyponyms, and siblings (third subvectors).A higher overall score indicates a closer similarity between the source and target specifications. The semantic WSDL structure matching algorithm is an extension to it: it also tries to find an optimal mapping between source and target service components based both on the similarity of their syntactic structures and also the semantic similarity between the identifiers of data types, operations and services to assess service similarities. The identifier-matching process is similar to that of the original WSDL structure matching. It starts by comparing the names of the data types (identifiers) involved in the two WSDL specifications. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target data-types. The next step in the process is the matching of the service operations. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target operations. The degree to which two operations are similar is decided on the semantic distance between operations' names and how similar their parameter lists are, in terms of the identifiers they contain. Finally, the overall score for how well the two services match is computed by matching the services' names and by identifying the pair-wise correspondence of their operations that maximizes the sum total of the matching scores of the individual pairs. This is a very blunt and imprecise service-discovery mechanism.

1.6.6 AN EXTENSION OF WSDL FOR FLEXIBLE WEB SERVICE INVOCATION WITH LARGE DATA

In (Wiley 2008), a traditional web services are limited in the request-reply framework, where service invocation can only be bundled together with all

relevant data in a single message. However, more and more services require large datasets (e.g., multimedia and scientific data). Under the WSDL standards, these datasets must be ported to an appropriate message format and transferred in their entirety upon each service invocation or response. A strategy to solve this problem which is called WSDL-D is to separate invocation messages from their datasets. Such a separation ultimately grants service consumers the ability to pass parameter datasets from third party hosts, to maintain dataset parameters on the service provider host for use with future service invocations, and to provide datasets in a variety of different formats. Approach of WSDL-D makes significant improvements in service flexibility and performance stand simply by separating invocation and response messages from their respective datasets. Such a separation not only grants service consumers the ability to pass parameter datasets from third party hosts but also to maintain dataset parameters on the service provider host for use with future service invocations, and to provide datasets in a variety of different formats. The standard of Web Service Description Language can support service invocation mechanism of WSDL-D which separates service invocation and dataset transferring between service caller and service provider. In WSDL-D, input parameters (datasets) of a service invocation is not required to be sent with the invocation message, instead, an input dataset can be fetched by a service provider, or sent later by the requester, or simply the dataset used in previous invocation requests. Similarly, an output dataset can be pushed to the requester asynchronously, or fetched by the requester. However, there is no mechanism to support complex input and output parameters required in WSDL-D. WSDL-D extension is used for handling large dataset.

1.6.7 MINING RELATED QUERIES FROM SEARCH ENGINE QUERY LOGS

The authors (Shi 2006) proposed a method that retrieves a list of related queries given an initial input query. The related queries are based on the query log

of previously issued queries by human users, which can be discovered using improved association rule mining model. Users can use the suggested related queries to tune or redirect the search process. This method not only discovers the related queries, but also ranks them according to the degree of their relatedness. Unlike many other rival techniques, it exploits only limited query log information and performs relatively better on queries in all frequency divisions. This method is to use an improved association rule mining model to mine related queries from query transactions in query logs. A simple but effective segmentation algorithm that segments user sessions into query transactions is also proposed. Because search engine users often reformulate their input queries by adding, deleting or changing some words of the original query string, Levenshtein distance similarity is used which is a special type of edit distance to measure the degree of matching between query strings. It defines a set of edit operations, such as insertion or deletion of a word, together with a cost for each operation. The distance between two query strings then is defined to be the sum of the costs in the cheapest chain of edit operations transforming one query string into the other. For example, the Levenshtein distance between "adobe photoshop" and "photoshop" is 1. Hence the equation 1.2 for the similarity between two queries can be measured by the Levenshtein distance similarity between them and defined as:

$$similarity_{Levenshtein}(q_1, q_2) = 1 - \frac{Levenshtein\_distance(q_1, q_2)}{max(wn(q_1), wn(q_2))} \quad (1.2)$$

Where wn() is the number of words (or characters for Chinese queries) in a query. The Levenshtein distance similarity is seldom applied to finding related queries because it retrieves only highly matching queries and thus fails to discover those related queries that are dissimilar in their terms, e.g. "search engine" and "google". The proposed model is based on the traditional association rule mining technique. For mining association rules of queries, statistically measure is needed for the co-occurrences between queries in query transactions. So the quality of segmenting user sessions into query transactions is critical for mining related

queries. A dynamic sliding window segmentation algorithm is developed that adopts three time interval constraints, i.e.

- The maximum interval length allowed between adjacent query records in a same query transaction (α),
- The maximum interval length of the period during which the user is allowed to be inactive (β), and
- The maximum length of the time window the query transaction is allowed to span (γ) ($\alpha \le \gamma \le \beta$).

It also sets a lower bound for the Levenshtein distance similarity between adjacent queries, i.e. θ, to justify the borders of query transactions. This method will improve the efficiency and accuracy of the search based on the querying related mining.

1.7 PROBLEM DEFINITION

For the applications in WSNs, web service is one of the recommended frameworks to publish, invoke, and manage services. However, the standard Web service description language (WSDL), defines only the service input and output while ignoring the corresponding input-to-output mapping relationships. The Fig. 1.6 shows structure of traditional WSDL 2.0.

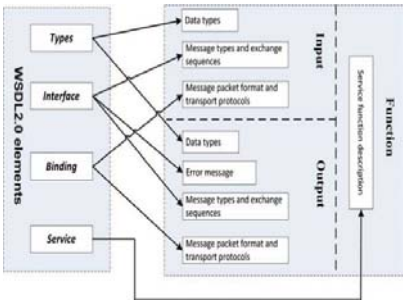

Fig. 1.6 WSDL 2.0 Structure

This presents a serious challenge in distinguishing services with similar input and output interface. In order to overcome this problem, the WSDL is extended through input-to-output mapping relationships.

CHAPTER 2

IMPLEMETATION

The WSDL finds difficulty for distinguishing services with similar input and output. The challenge is addressed by embedding the service policy into the traditional WSDL2.0 schema to describe the input-to-output mapping relationships. Furthermore, a new web service redundancy detection approach is proposed based on this similarity.

2.1 PROSPOSED SYSTEM

The objective is to improve the efficiency of services retrieval and to achieve more reliable WSN's Services. The efficiency of services retrieval can be achieved by service redundancy detection and filtering algorithm. The web service redundancy is detected by extending the WSDL. The input-to-output mapping relationship is used to find the redundancy in efficient manner. Firstly web services that need redundancy detection are retrieved via the web service API. Through analyzing the WSDL documents regarding these services, the service IO and the service policy can be abstracted and obtained. Secondly, the policy is used to represent input-to-output mapping relationships for finding similarity. After that, the service similarity can be calculated. If the similarity exceeds a pre-set threshold, which shows that the redundancy exists and will mark the corresponding services as redundant and undertake post processing according to different roles. For users, if the redundancy exists, which means these services are similar, they need to choose the most appropriate service according to the policy description. For the service

providers, if the newly published service has redundancy with the existing ones, they can delete the redundant service if necessary.

The services such as A, B, C, D, E.., for illumination and temperature are given below, where service A, B, C are used for light control and service D, E are used for temperature control. From the services, service A, Service B and Service C are similar to each other, similarly the services D, E and F are similar.

Service A

Input: illumination

Output: switch the lights control button

Policy: If the illumination<5 lux, or the illumination>80 lux, switch the lights control button

Service B

Input: illumination

Output: turn on the lights

Policy: If the illumination<5 lux, turn on the lights

Service C

Input: illumination

Output: turn on the lights

Policy: If the illumination<10 lux, turn on the lights

Service D

Input: temperature

Output: turn on the air conditions

Policy: If the temperature>26 ^{0}C, turn on the air conditions

Service E

Input: temperature

Output: turn on the air conditions

Policy: If the temperature>15 ^{0}C, turn on the air conditions

Service F

Input: temperature

Output: turn off the air conditions

Policy: If the temperature>15 ^{0}C, and the temperature<26 ^{0}C, turn off the air conditions

In the Policy Based WSDL Extension as shown in Fig. 2.1 defines the service policy to represent Input-Output mapping relationships. The service policy describes the conditions that need to be satisfied by the service input before the service can be executed. The service policy is usually specified by the if-then pattern, where the if-part includes a set of logic expressions for conditions reflecting the service Input–Output mapping relationships. Therefore, the if-part can be embedded as a sub-element of the service element in a WSDL2.0 document.

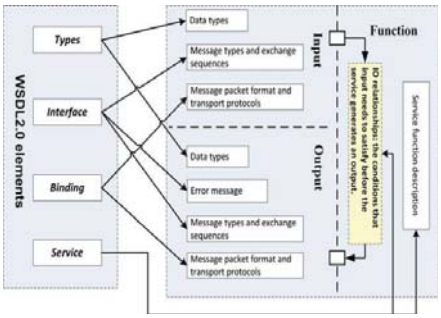

Fig. 2.1 Extended WSDL 2.0 Structure

The service redundancy detection approach is based on the policy-based WSDL extension. There are different stages in detecting the web services. First stage is to retrieve web services from web services API which need redundancy detection. Then in second stage, analyzing the WSDL documents regarding these two services. In third stage, i.e. pre-processing stage, each leaf node in the policy binary tree is transformed as a vector. Based on analyses and pre-processing stage, the web services similarity can be computed. The similarity computation considers two parts such as IO similarity and policy similarity.

The overall flowchart of the proposed approach is shown in Fig. 2.2. Firstly, two web services that need redundancy detection are retrieved via the web service API. Through analyzing the WSDL documents regarding these two services, the service IO and the service policy can be abstracted and obtained.

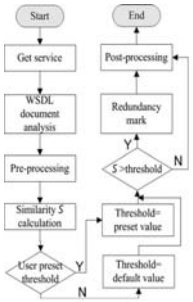

Fig. 2.2 Service redundancy detection

Then based on Input-Output mapping relationship, similarity is calculated. It can accurately reflect the function of different services and describe the redundant information. Secondly, service policy is used for finding similarity between two web services. After that, the service similarity can be calculated. If the similarity exceeds a pre-set threshold, which shows that the redundancy exists, will mark the corresponding services as redundant. For users, if the redundancy exists, which means these services are similar, they need to choose the most appropriate service according to the policy description. For the service providers, if the newly published service has redundancy with the existing ones, they can delete the redundant service if necessary. For the UDDI administrator, they can make use of the redundancy and cluster similar services together to improve the performance of the service search engine.

2.2 MODULE DESCRIPTION

The similarity computation between different web services is based on two parts: the IO similarity and the policy similarity. The modules of the project are

- Web service and Policy Description
- Service Redundancy Detection
- Reporting

2.2.1 WEB SERVICE AND POLICY DESCRIPTION

In web service description, the services are temperature control, light control, humidity control etc. The policy-based WSDL is used to describe the services. The service policy is used to represent Input-Output mapping relationships and it describes the conditions that need to be satisfied by the service input before the service can be executed. It is usually specified by the if-then pattern, where the

if-part includes a set of logic expressions for conditions reflecting the service Input–Output mapping relationships.

2.2.2 SERVICE REDUNDANCY DETECTION

The web services that need redundancy detection are retrieved via the web API. Through analyzing the WSDL documents regarding these two services, the service IO and the service policy can be abstracted and obtained. Based on the similarity between two services, the redundancy is computed. The similarity computation (Wang 2003) considers two parts: the IO similarity and the policy similarity.

2.2.2.1. IO Similarity

Let I = {i1, i2, …} be the input variable set, and O = {o1, o2, …} be the output variable set. Assuming that there are two web services W1 and W2, the IO similarity of W1 and W2 is then calculated as shown in equation 2.1.

$$Sim_{IO}(W_1,W_2)=\frac{2*(|BI1|+|BO|)}{|I1|+|O1|+|I2|+|O2|} \quad (2.1)$$

where BI refers to the intersection set between I1 and I2, while BO refers to the intersection set between O1 and O2. The operator | | refers to the cardinality.

2.2.2.2. Policy Similarity

The policy similarity SimPo(W1,W2) between W1 and W2 is calculated as shown in equation 2.2.

$$Sim_{PO}(W_1,W_2)=0.5*\frac{2*|Fs|}{|F1|+|F2|}+0.5*\left(\left(\frac{1}{N}\right)*Sim\,Leven(t1i,t2j)\right) \quad (2.2)$$

where STr1, STr2 are the pre-order traversals of Tr1, Tr2; N is the sum of the number of elements in STr1 and STr2; F1;F2 are the logic operator subsets in the STr1; STr2; Fs is the logic operator intersection set that includes the same operators at the same position in the STr1 and STr2. Finally, the overall similarity between two web services is the combination of the IO similarity and policy similarity where k is the weight of the IO similarity. The range of k should be limited as (0, 1).

CHAPTER 3

RESULTS AND ANAYLSIS

3.1 EXPERIMENTATION

The proposed work is implemented using .Net Framework. For the evaluation purpose, 100 services are created. Among each service, the redundancies are found using service redundancy detection algorithm. The ASP website and web services are designed for creating a website and web service. The services are stored in SQL server database. The code is written in C# and forms are designed using ASP.

3.2 SYSTEM SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS

Processor	: Pentium III and above
Clock speed	: 550MHz
Hard Disk	: 20GB
RAM	: 128MB or above
Cache Memory	: 512KB
Monitor	: Color Monitor
Keyboard	: 104 Keys
Mouse	: 3 Buttons

3.2.2 SOFTWARE REQUIREMENTS

Operating System	: Windows XP
Language	: Microsoft visual studio .net
Database	: SQL Server 2005

3.3 SNAPSHOT

The Fig. 3.1 shows the details of services for WSN application.

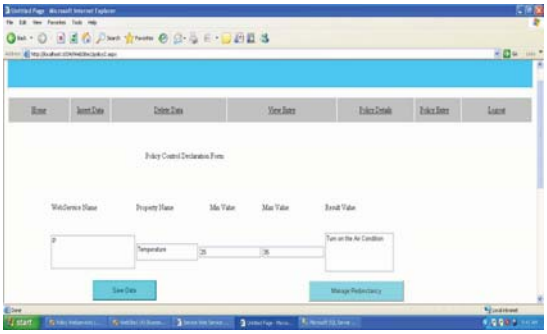

Fig. 3.1 Web Service and Policy Description

The services details are entered for different WSN's application such as temperature control services, light control services, pressure control service and humidity control service. It provides web service name, wireless sensor network parameter name, min and max value and finally conditions for the services.

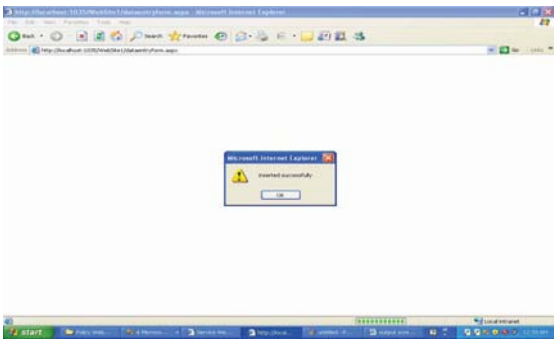

Fig. 3.2 Update of services

The Fig. 3.2 shows the update of services in the database. The Fig. 3.3 shows the filtered services among the provided services. The redundancy is computed based on the policy.

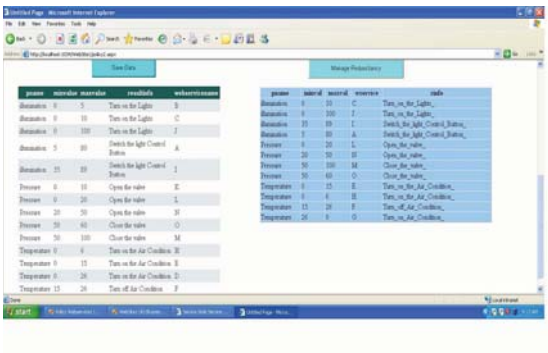

Fig. 3.3 Filtration of redundant services

3.4 DATABASE DESIGN

The Policy Table 3.1 is used for storing the services data such as service name, parameter name, min value, max value, result information.

Table 3.1 Policy table

wservice	pname	min	max	rinfo
A	Illumination	0	10	Turn on the light
B	Illumination	0	15	Turn on the light
C	Temperature	10	25	Turn on the AC
D	Temperature	11	24	Turn on the AC
E	Illumination	0	26	Switch the lights control button

3.5 SIMILARITY COMPUTATION

The services similarity is combination of the IO similarity and the policy similarity. The IO similarity evaluates the similarity of services i.e input and output while the policy similarity reflects the similarity in the services functions. The Fig. 3.4 shows the similarity computation for the given services without finding policy similarity.

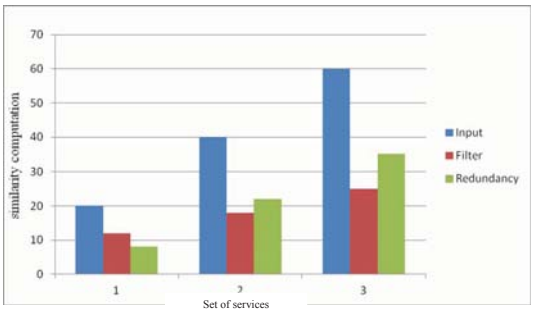

Fig. 3.4 Similarity Computation without using policy

The Fig. 3.5 shows the similarity computation with policy similarity. It will help to improve the redundancy detection efficiently and also search retrieval.

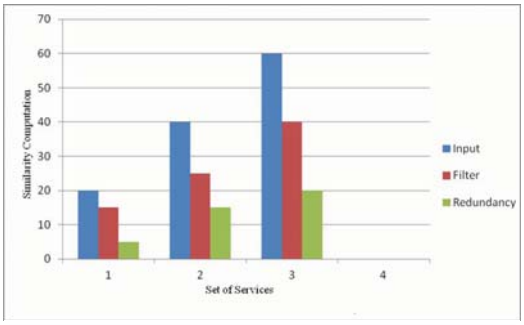

Fig. 3.5 Similarity Computation using policy

3.6 CONCLUSION AND FUTURE WORK

WSNs composed of a large number of sensor nodes which are attractive for information gathering in numerous real world applications. As one of the recommended frameworks to publish, invoke, and manage services, Web Service has become a trend in developing WSNs' applications. However, the standard web service description language, WSDL, lacks a satisfactory mechanism to distinguish services with similar input and output. This makes it impractical for many WSNs' applications, where service redundancy exists and many services are in a constant state of change.

A policy based WSDL extension is used for characterizing Input– Output mapping relationship. Based on such an extension, the similarity between Web services is then defined based on the IO similarity and policy similarity, and a novel web service redundancy detection approach is further proposed. As for future work, a plan is to improve redundancy detection efficiency. Moreover this leads to potentials for further improving service retrieval efficiency.

APPENDIX

SOURCE CODE

Web Service

```
using System;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;
using System.Data.SqlClient;
using System.IO;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
// [System.Web.Script.Services.ScriptService]
public class Service : System.Web.Services.WebService
{
public Service () {
 //Uncomment the following line if using designed components
 //InitializeComponent();
 }
[WebMethod]
```



```
public string TempEntry(string locationName, string pnmae, string value, string Edate, string cperson, string country)
{
SqlConnection con = new
SqlConnection("server=HOME\\SQLEXPRESS;;integrated
security=true;database=policy;");
string retn = "";
try
{
 con.Open();
 SqlCommand cmd = new SqlCommand("insert into sourcetb values('" +
locationName + "','"+ pnmae +"'," + value + ",'" + Edate + "','" + cperson + "','" +
country + "')", con);
cmd.ExecuteNonQuery();
 retn = "0";
}
Catch
{
 retn = "1";
 }
con.Close();
 return retn;
}
[WebMethod]
public string policyEntry(string pname, string minval, string maxval, string resultinfo,string webservicename)
{
```



```
SqlConnection con = new
SqlConnection("server=HOME\\SQLEXPRESS;;integrated
security=true;database=policy;");
string retn = "";
try
{
 con.Open();
 SqlCommand cmd = new SqlCommand("insert into policy values('" + pname + "'," + minval + "," + maxval + ",'" + resultinfo + "','"+ webservicename +"')", con);
 cmd.ExecuteNonQuery();
 retn = "0";
 }
catch
{
 retn = "1";
}
con.Close();
return retn;
}

[WebMethod]

 public string UpdateEntry(string locationName, string Temperature, string Humidity, string Edate, string cperson, string country, int tid)
{
string retn = "";
return retn;
}
```

```
[WebMethod]
public string DeleteEntry(string locationname)
{
 string retn = "";
 SqlConnection con =
newSqlConnection("server=HOME\\SQLEXPRESS;;integrated
security=true;database=policy;")
try
{
 con.Open();
 SqlCommand cmd = new SqlCommand("delete from sourcetb where
locationname='" + locationname+ "'", con);
cmd.ExecuteNonQuery();
 retn = "0";
}
catch
{
 retn = "1";
      }
con.Close();
return retn;
}
[WebMethod]
 public string ViewEntry(string locationname, int tid)
{
 SqlConnection con = new
SqlConnection("server=HOME\\SQLEXPRESS;;integrated
security=true;database=policy;");
```



```
string retn = "";
try
{
con.Open();
SqlCommand cmd = new SqlCommand("select * from sourcetb", con);
retn = "0";
}
Catch
{
 retn = "1";
}
con.Close();
///string retn = "";
 return retn;
}
[WebMethod]
public string ViewPolicy(string locationName)
{
string retn = "";
 SqlConnection con =
newSqlConnection("server=HOME\\SQLEXPRESS;;integrated
security=true;database=policy;");
try
{
con.Open();
SqlCommand cmd = new SqlCommand("select * from policy", con);
//SqlDataAdapter da = new SqlDataAdapter(cmd);
// da.Fill(ds, "a");
```



```
 SqlDataReader dr;
 dr = cmd.ExecuteReader();
 while (dr.Read())
{
 retn = dr[0].ToString();
}
con.Close();
}
catch
{
 retn = "1";
}
con.Close();
 return retn;
}
}
```

Policy website

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
```



```
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Text;
//using webservice1;
using System.Data.SqlClient;
public partial class policy2 : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
 SqlConnection con = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
con.Open();
SqlDataAdapter da = new SqlDataAdapter("select * from policy", con);
DataSet ds = new DataSet();
da.Fill(ds);
gdView.DataSource = ds.Tables[0];
gdView.DataBind();
con.Close()
 }

protected void Button1_Click(object sender, EventArgs e)
{
SqlConnection con = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
 SqlCommand cmd;
 try
 {
cmd = new SqlCommand("select * from policy", con);
```

```
con.Open();
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
da.Fill(ds, "policy");
gdView.DataSource = ds;
gdView.DataBind();
con.Close();
}
catch (Exception ex)
{
}
}

protected void Button1_Click1(object sender, EventArgs e)
{
webservice1.Service obj = new webservice1.Service();

string retnval = "";

retnval = obj.policyEntry(TextBox1.Text, TextBox2.Text, TextBox3.Text,
TextBox4.Text,TextBox5.Text);

 if (retnval == "0")
{
Response.Write("<scriptlanguage='javascript'>alert('inserted
successfully')</script>");
}
else if (retnval == "1")
```



```
{
Response.Write("<script language='javascript'>alert('Some Error in data
insertion')</script>");

}

SqlConnection con = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
con.Open();
SqlDataAdapter da1 = new SqlDataAdapter("select * from policy",con);
DataSet ds = new DataSet();
da1.Fill(ds);
gdView.DataSource = ds.Tables[0];
gdView.DataBind();
con.Close();
}
int minvali,maxvali;
string[] minvalseq = new string[100];
string[] maxvalseq = new string[100];
string pname;
string rs,wserve;
string mval, maxval;
int d7;
protected void Button2_Click(object sender, EventArgs e)
{
 Response.Redirect("Report Page.aspx");
}
protected void AutoRefreshTimer_Tick(object sender, EventArgs e)
```



```
{
 bindcustomers();
}
private void bindcustomers()
{
 SqlConnection con = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
SqlConnection con1 = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
SqlConnection con6 = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
SqlConnection con7 = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
con7.Open();
SqlCommand cmd7 = new SqlCommand("select distinct * from temp", con7);
SqlDataReader dr7 = cmd7.ExecuteReader();
while (dr7.Read())
{
 try
{
con.Open();
SqlCommand cmd8 = new SqlCommand("drop table " + dr7[0].ToString(), con);
cmd8.ExecuteNonQuery();
con.Close();
}
catch
{
}
```



```
}
con.Close();
con.Open();
SqlCommand cmd16 = new SqlCommand("delete from temp", con);
cmd16.ExecuteNonQuery();
con.Close();
con.Open();
SqlCommand cmd = new SqlCommand("select distinct resultinfo from policy",
      con);
SqlDataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
con1.Open();
SqlCommand cmd1 = new SqlCommand("select
pname,minvalue,maxvalue,resultinfo,webservicename from policy where
resultinfo='" + dr[0].ToString() + "'", con1);
SqlDataReader dr1 = cmd1.ExecuteReader();
while (dr1.Read())
{
minvali += 1;
maxvali += 1;
pname = dr1[0].ToString();
mval = dr1[1].ToString();
maxval = dr1[2].ToString();
rs = dr1[3].ToString();
wserve = dr1[4].ToString();
minvalseq[minvali] = mval;
maxvalseq[maxvali] = maxval;
```

```
}
StringBuilder ww = new StringBuilder();
string[] ars = rs.Split(' ');
int gg = ars.Count();
if (gg > 1)
{
int m;
for (m = 0; m <= gg - 1; m++)
{
ww = ww.Append(ars[m] + "_");
}
}
SqlConnection con3 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
SqlConnection con4 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
SqlConnection con5 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
con6.Open();
SqlCommand cmd6 = new SqlCommand("insert into temp values('" + pname + "')", con6);
cmd6.ExecuteNonQuery();
con6.Close();
con5.Open();
SqlCommand cmd5 = new SqlCommand("select * from sysobjects where type='u' and name='" + pname + "'", con5);
SqlDataReader dr5 = cmd5.ExecuteReader();
if (dr5.Read())
```



```
{
 con4.Open();
 SqlCommand cmd4 = new SqlCommand("insert into " + pname + " values('" + pname + "'," + mval + "," + maxval + ",'" + wserve + "','" + ww + "')", con4);
cmd4.ExecuteNonQuery();
con4.Close();
}
else
{
con3.Open();
SqlCommand cmd3 = new SqlCommand("create table " + pname + "(pname varchar(50),minval int,maxval int," + wserve + " varchar(50),resultinfo varchar(200) )", con3);
cmd3.ExecuteNonQuery();
con3.Close();
con3.Open();
SqlCommand cmd4 = new SqlCommand("insert into " + pname + " values('" + pname + "'," + mval + "," + maxval + ",'" + wserve + "','" + ww + "')", con3);
cmd4.ExecuteNonQuery();
con3.Close();
}
con5.Close();
con1.Close();
SqlConnection con15 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
con15.Open();
SqlCommand cmd9 = new SqlCommand("select distinct * from temp", con15);
SqlDataReader dr9 = cmd9.ExecuteReader();
```



```
 while (dr9.Read())
{
 SqlConnection con10 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
 con10.Open();
 SqlCommand cmd10 = new SqlCommand("select distinct resultinfo from " + dr9[0].ToString(), con10);
SqlDataReader dr10 = cmd10.ExecuteReader();
while (dr10.Read())
{
SqlConnection con11 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
con11.Open();
 string qry;
//qry="select * from "+dr9[0].ToString()
SqlCommand cmd11 = new SqlCommand("select * from " + dr9[0].ToString() + " where resultinfo= '" + dr10[0].ToString() + "'", con11);
SqlDataReader dr11 = cmd11.ExecuteReader();
if (dr11.Read())
{
SqlConnection con12 = new SqlConnection("server=.\\sqlexpress;integrated security=true;database=policy;");
con12.Open();
SqlCommand cmd12 = new SqlCommand("insert into  result values('" + dr11[0].ToString() + "','" + dr11[1].ToString() + "','" + dr11[2].ToString() + "','" + dr11[3].ToString() + "','" + dr11[4].ToString() + "')", con12);
 cmd12.ExecuteNonQuery();
con12.Close();
```



```
}
con11.Close();
}
con10.Close();
}
con15.Close();

}
con.Close();
con.Open();
SqlDataAdapter da = new SqlDataAdapter("select min(minval), max(maxval),rinfo from result group by rinfo", con);
DataSet ds = new DataSet();
da.Fill(ds);
GridView1.DataSource = ds.Tables[0];
GridView1.DataBind();
 con.Close();

string maxrws, minrws;
int minresult, maxresult;
con.Open();

 SqlCommand rcmd = new SqlCommand("select  min(minval),wservice,rinfo from result group by wservice,rinfo having rinfo='Turn_on_the_Lights_'", con);
 SqlDataReader rreader = rcmd.ExecuteReader();
if (rreader.Read())
{
minresult = Convert.ToInt32(rreader[0].ToString());
```

```
minrws = rreader[1].ToString();
SqlConnection con3 = new SqlConnection("server=.\\sqlexpress;integrated
security=true;database=policy;");
 con3.Open();
}
```


REFERENCES

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey.", Comput. Networks 38, pp: 393–422, 2002.
2. D'Ambrogio, A. "A model-driven WSDL extension for describing the QoS of web services." In: Proceedings of the IEEE International Conference on Web Services, pp. 789–796. IEEE Computer Society ,2006.
3. Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C.,Langworthy, D., Malhotra, A., et al. "Web services policy framework (WS-policy)." Specification,IBM, BEA, Microsoft, SAP AG, Sonic Software, VeriSign ,2004.
4. Curbera, F., Leymann, F., Storey, T., Ferguson, D., Weerawarana, S. "Web Services Platform Architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable Messaging and More." Prentice Hall PTR , 2005.
5. David Culler, Deborah Estrin and Mani Srivastava, "Overview of Sensor Networks," IEEE Computer, Vol. 37, No. 8, August 2004.
6. Glitho, R., Khendek, F., Othman, N., Chebbine, S. "Web services-based architecture for the interactions between end-user applications and sink-less wireless sensor networks." In: 4th IEEE Consumer Communications and Networking Conference, 2007 CCNC 2007, pp. 865–869 ,2007.
7. Marcos Augusto M. Vieiral Claudionor N. Coelho. Jr. Diogenes Cecilio da Silva Junior Jose M. da Mata: "Survey on Wireless Sensor Network Devices," Proc. of IEEE Conference on Emerging Technologies and Factory Automation, Vol. 1, pp. 537-44, 2003.
8. Shi, X., Yang, C. "Mining related queries from search engine query logs." In: Proceedings of the 15th International Conference on World Wide Web, pp. 943–944. ACM 2006.
9. Ta, T., Othman, N., Glitho, R., Khendek, F. "Using web services for bridging end-user applications and wireless sensor networks". In: Proceedings of 11th IEEE Symposium on Computers and Communications, 2006 (ISCC'06), pp. 347–352 ,2006.

10. Wang, Y., Stroulia, E. "Semantic structure matching for assessing web-service similarity." Service-Oriented Computing-ICSOC 2003, pp. 194–207, 2003.
11. "Web services description language (WSDL)." http://www.w3.org/TR/wsdl.
12. Wiley, M., Wu, A., Su, J. "WSDL-D: A flexible web service invocation mechanism for large datasets." In: 10th IEEE Conference on E-commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-commerce and E-services, pp. 157–164. 2008.

LIST OF PUBLICATIONS

1. G.Muthulakshmi, Dr.V.Vanitha, "Detecting and Filtering the Web Service Redundancy in Wireless Sensor Networks", International Conference, Advance in Computer and Communication, Sun College of Engineering and Technology Nagercoil, 18th February 2013.

2. G.Muthulakshmi, Dr.V.Vanitha, "Web Service Redundancy Detection in Wireless Sensor Networks", National Conference, Innovation on Information Technology, Bannari Amman Institute of Technology, Coimbatore, 22nd February 2013.