SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

P.NITHYANAYAKI	**0710108033**
P.PAVITHRA	**0710108035**
E.SANDHYA	**0710108041**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology,Coimbatore)

COIMBATORE 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORKS"** is the bonafide work of **"NITHYA NAYAKI.P, PAVITHRA.P and SANDHYA.E"** who carried out the project work under my supervision.

SIGNATURE

Prof.P.Devaki,

Head of the Department.

SIGNATURE

Mrs.V.Vanitha,

Supervisor,

Associate Professor.

Department of Computer science & Engineering,
Kumaraguru College of Technology,
Coimbatore - 641 006.

The candidates with University Register Nos. **0710108033, 0710108035, 0710108041** were examined by us in the project viva-voice examination held on ....20/4/..........

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled **"SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK"** is a record of original work done by us and to the best of our knowledge; a similar work has not been submitted to Anna University or any institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment for the award of the degree of bachelor of computer science and engineering of Kumaraguru College of Technology affiliated to Anna University, Coimbatore.

Place: COIMBATORE

Date: 18/04/11

P. Nithyanayaki

(P.Nithyanayaki)

Pavithra

(P.Pavithra)

Sandhya

(E.Sandhya)

ABSTRACT

Wireless Sensor Networks (WSN) is a large heterogeneous Sensor devices spread over a large field. The task of WSN is to perform wireless sensing and data networking. It is a group of sensors linked by wireless media to perform distributed sensing tasks .Each node in a sensor network is typically equipped with a radio transceiver and microcontroller, and an energy source, usually a battery. WSN are basically used for critical monitoring and control applications. WSN is applicable to a wide range of industrial and health applications.

With the enlargement of WSN, number of service applications increases and the number of users also increases. Also, it is necessary to interconnect several networks. Current architecture for WSN has certain limitations like tight coupling between WSNs and applications, Costly optimization or suboptimal efficiency, Limited reusability. These limitations are overcome by Service Oriented Architecture which led to our project to discovery and composes the available services in the network.

Service Composition is to combine two or more services based on the request. Composition means taking advantage of currently existing web services to provide a new service that does not exist on its own. One of the part of Service Composition is to discover a service. Service Discovery is to find particular services that match with the request from the available services in the network. It enables service users and services to dynamically find available services in a network.

In this project, we propose a method to perform service discovery and composition. Here, the main focus is on the performance of service discovery and composition. We have proposed an algorithm which minimizes the search time to discover a service by choosing an optimal path . Our experimental results show the evidence for the proposed method.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

WSN – Wireless Sensor Network

SOWSN – Service Oriented Wireless Sensor Network

WSCR – Web Services Composition Result

WS – Web Service

BAN – Body Area Network

IC – Integrated Chip

SOA – Service Oriented Architecture

TWSC – Tree-based Web Service Composition

PWSC – Process-based Web Service Composition

SIF – Service Inverted File

PIF – Parameter Inverted File

HB – Heart Beat

BP – Blood Pressure

RBC – Red Blood Cells

WBC – White Blood Cells

ECG – Echo Cardio Graphy

EEG – Electro Encephalo Gram

SQL – Structured Query Language

BPEL – Business Process Execution Language for web services

FSM – Finite State Machine

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 OVERVIEW

Sensors are an integral part of our environment. Smaller chips, emerging wireless communication standards, and more capable sensors and actuators are pushing the development towards wireless sensor networks. Wireless sensor networks are composed of multifunctional miniature sensor devices with limited processing and storage capabilities, often battery operated, and interconnected with each other over wireless links. Over the past years, networking sensors have received more attention in various sectors. One application area is a health Body Area Network (BAN), consisting of a network of sensors carried by, and moving around with, the patient. The benefits from mobile (wireless) monitoring in out-of-hospital environments have been the focus in several studies and a number of prototypes have been developed. These systems incorporate a range of wireless devices with varying capabilities. Even though a small number of nodes might be carried at the same time, the network of sensors may be configured quite differently over time (e.g. depending on how the medical condition of a patient progresses).

In this project, we review a number of well-known established service discovery architectures, which could be used to manage the complexity of sensor networks. Our target platform is a resource constrained sensor node, hence we seek a lightweight scheme that enables devices and their services to auto-configure, cooperate, adapt to changes, and to dynamically find available services in a sensor network.

In this project, we are interested in studying how services can be composed to provide more complicated features. There might be frequently the case that a service does not provide a requested service on its own, but delegates parts of the execution to other services and receives the results from them to perform the whole service. In this case, the involved services together can be considered a composite service. Those services that perform the requested tasks individually are called simple services. The main advantage of the work presented in this project is providing concrete approaches in service discovery and composition.

1.2 WIRELESS SENSOR NETWORK

Sensors integrated into structures, machinery, and the environment, coupled with the efficient delivery of sensed information, could provide tremendous benefits to society.

The ideal wireless sensor is networked and scalable, consumes very little power, is smart and software programmable, capable of fast data acquisition, reliable and accurate over the long term, costs little to purchase and install, and requires no real maintenance. Selecting the optimum sensors and wireless communications link requires knowledge of the application and problem definition. Battery life, sensor update rates, and size are all major design considerations.

Recent advances have resulted in the ability to integrate sensors, radio communications and digital electronics into a single integrated circuit (IC) package. This capability is enabling networks of very low cost sensors that are able to communicate with each other using low power wireless data routing protocols. A wireless sensor network (WSN) generally consists of a basestation (or "gateway") that can communicate with a number of wireless sensors via a radio link. Data is collected at the wireless sensor node, compressed, and transmitted to the gateway directly or, if required, uses other wireless sensor nodes to forward data to the gateway. The transmitted data is then presented to the system by the gateway connection.

1.3 LIMITATIONS OF WSN

Current architecture for WSN has inherent limitations that may be summarized as follows:

A.Tight coupling between WSNs and applications

A tight network-application coupling characterizes most current sensor network and WSN architectures. This coupling usually takes one of two forms:

- Network-dependent application development: Most current sensor applications are designed and implemented to be used on a specific sensor querying interface. Often, application developers need to know network-specific information such as network topology, nodes, transmission range and processing/memory capabilities, etc.
- Application-dependent network design and deployment: In some cases, a decision is first made to use a specific software system to build a sensor application. The requirements of the software must then be taken into consideration when designing and deploying the sensor network supporting that system.

B. Costly optimization or suboptimal efficiency

In current WSNs, application-dependent optimization makes the sensor network unable to provide the same levels of performance to other applications.

Also, several rounds of optimization may be needed as the application evolves or is replaced. The alternative of application independent optimization is often too generic and does not exploit optimization opportunities that a specific class of applications may offer. Typically, this leads to suboptimal efficiency.

C. Limited reusability

Ideally, for a WSN to be cost effective, it would be necessary to amortize its deployment and maintenance cost by sharing its functionalities amongst a large group of users and applications. This reusability is generally not easily achievable in current sensor infrastructures due to the tight coupling between networks and applications.

D. Low return on investment

This drawback follows from the previous one. The monolithic, application-specific design of current WSN makes it difficult to reuse most of an application's modules in developing another application. Often, intensive programming is required each time a new application has to be developed.

E. Non-scalability

Most of today's WSN applications are designed and optimized for light loads, i.e., destined to be used by a small group of users. As a result , current WSNs often do not scale to support large numbers of simultaneous users.

1.4 SERVICE ORIENTED ARCHITECTURE FOR WSN

Service-oriented computing is gaining wider acceptance. For Web services to become practical, an infrastructure needs to be supported that allows users and applications to discover, deploy, compose and synthesize services automatically. Recently, the Service Oriented Architecture (SOA) has been considered as a good candidate to develop open, efficient, inter-operable, scalable and customizable WSN applications.

In Service Oriented WSNs, node's sensing capability is exposed in the form of in-network services. Application development is simplified by providing standards for data representation, service interface description, and service discovery facilitation. By wrapping application functionality into a set of modular services, a programmer can then specify execution flow by simply connecting the appropriate services together.

1.5 ADVANTAGES OF SOWSN

Advantages of service-based architecture approach are attractive for three key reasons. First, it enables existing applications to be used in new contexts. We describe a real-world scenario where customer data from a mainframe application is made available to newer applications. Second, this approach enables services to be partitioned or replicated, to improve both availability and scalability. For example, Amazon uses Dynamo, a partitioned and replicated data storage service, to maintain its customers' shopping cart data. Third, service-based systems enable components to evolve independently, reducing the cost and availability impact of upgrades. This style of evolution is practiced by most large web applications, allowing new functionality to be delivered while providing continuous availability. It is also used to evolve the software in an enterprise: legacy systems can be incrementally replaced without disrupting the normal operation of a company.

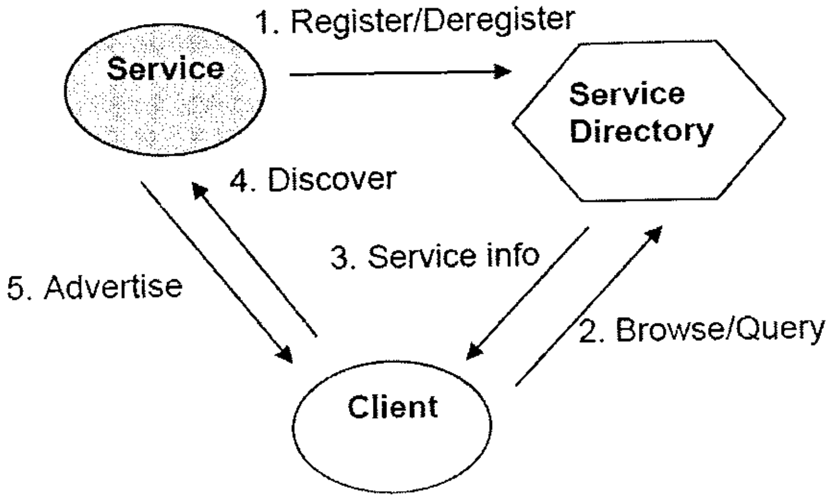


Figure 1: Service Oriented Architecture

1.6 SERVICES

A service is a mechanism that provides a well-defined functionality for applications or other services through a standardized interface. Web services are defined as self-contained, modular units of application logic which provide business functionality to other applications via an Internet connection. Web services support the interaction of business partners and their processes by providing a stateless model of atomic synchronous or asynchronous message exchanges. These atomic message exchanges can be composed into longer business interactions by providing message exchange protocols that show the mutually visible message exchange behavior of each of the partners involved.

The growing trend in software architecture is to build platform-independent software components, called Web services, which are available in the distributed environment of the Internet. Applications are to be assembled from a set of appropriate Web services and no longer be written manually. Seamless composition of Web services has enormous potential in streamlining business-to-business or enterprise application integration.

The SOA defines an interaction between software agents as an exchange of messages between service requesters (clients) and service providers. Clients are software agents that request the execution of a service. Providers are software agents that provide the service. Agents can be simultaneously both service clients and providers. Providers are responsible for publishing a description of the service(s) they provide. Clients must able to find the description(s) of the services they require and must be able to bind to them.

The SOA is not architecture only about services; it is a relationship of three kinds of participants: the *service provider*, the *service discovery agency*, and the *service requestor* (*client*). The interactions involve *publish, find* and *bind* operations.

1.6.1 SERVICE DISCOVERY

Service discovery is a crucial component of any service oriented network. Discovery criteria depend on the underlying network and the application. Since we are in the context of WSNs, service discovery must be performed in an efficient way, i.e., with constant storage load on each node and with no global computation.

Service discovery provide the necessary means to describe services so that the service users can determine if a discovered service matches its requirements as well as utilize this service.

1.6.2 SERVICE COMPOSITION

Composition is useful when we are looking for a service with specific inputs and outputs and there is no single service satisfying the request. Given a repository of service descriptions, and a query with the requirements of the requested service, in case a matching service is not found, the composition problem involves automatically finding a chain of services that can be put together in correct order of execution to obtain the desired service.

1.7 SURVEY ON WSN

Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces and their components. They usually consist of a processing unit with limited computational power and limited memory, sensors or MEMS (including specific conditioning circuitry), a communication device (usually radio transceivers or alternatively optical), and a power source usually in the form of a battery. Other possible inclusions are energy harvesting modules, secondary ASICs, and possibly secondary communication devices (e.g. RS-232 or USB)[4].

The base stations are one or more distinguished components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user as they typically forward data from the WSN on to a server. Other special components in routing based networks are routers, designed to compute, calculate and distribute the routing tables. Many techniques are used to connect to the outside world including mobile phone networks, satellite phones, radio modems, high power Wi-Fi links etc.

The main characteristics of a WSN include:

- Power consumption constrains for nodes using batteries or energy harvesting
- Ability to cope with node failures
- Mobility of nodes
- Dynamic network topology
- Communication failures
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions
- Ease of use
- Unattended operation

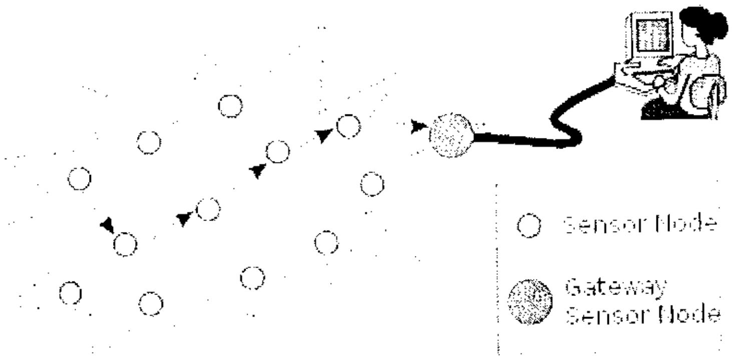


Figure 2: Wireless Sensor Network

1.8 USAGE SCENARIOS OF WSN

1.8.1 AREA MONITORING

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors to detect enemy intrusion; a civilian example is the geo-fencing of gas or oil pipelines.

When the sensors detect the event being monitored (heat, pressure), the event is reported to one of the base stations, which then takes appropriate action (e.g., send a message on the internet or to a satellite)[5]. Similarly, wireless sensor networks can use a range of sensors to detect the presence of vehicles ranging from motorcycles to train cars.

1.8.2 ENVIRONMENTAL MONITORING

A number of WSNs have been deployed for environmental monitoring.

1.8.2.1 AIR POLLUTION MONITORING

Wireless sensor networks have been deployed in several cities (Stockholm, London or Brisbane) to monitor the concentration of dangerous gases for citizens. The sensor nodes can control important parameters like CO, CO2, NO2 or CH4, which are generated by vehicles and industry, and have a severe impact on the human health. This way, the public institutions have a good tool to design plans to reduce pollution, improve the air quality and ensure the compliance with current legislation.

1.8.2.2 FOREST FIRES DETECTION

A network of Sensor Nodes can be installed in a forest to control when a fire has started. The nodes will be equipped with sensors to control temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

1.8.2.3 GREENHOUSE MONITORING

Wireless sensor networks are also used to control the temperature and humidity levels inside commercial greenhouses. When the temperature and humidity drops below specific levels, the greenhouse manager must be notified via e-mail or cell phone text message, or host systems can trigger misting systems, open vents, turn on fans, or control a wide variety of system responses.

1.8.2.4 LANDSLIDE DETECTION

A landslide detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. And through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.

1.8.3 INDUSTRIAL MONITORING

1.8.3.1 MACHINE HEALTH MONITORING

Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionalities. In wired systems, the installation of enough sensors is often limited by the cost of wiring. Previously inaccessible locations, rotating machinery, hazardous or restricted areas, and mobile assets can now be reached with wireless sensors.

1.8.3.2 WATER/WASTEWATER MONITORING

There are many opportunities for using wireless sensor networks within the water/wastewater industries. Facilities not wired for power or data transmission can be monitored using industrial wireless I/O devices and sensors powered using solar panels or battery packs.

1.8.4 LANDFILL GROUND WELL LEVEL MONITORING AND PUMP COUNTER

Wireless sensor networks can be used to measure and monitor the water levels within all ground wells in the landfill site and monitor leachate accumulation and removal. A wireless device and submersible pressure transmitter monitors the leachate level. The sensor information is wirelessly transmitted to a central data logging system to store the level data, perform calculations, or notify personnel when a service vehicle is needed at a specific well.

1.8.5 AGRICULTURE

Using wireless sensor networks within the agricultural industry is increasingly common; using a wireless network frees the farmer from the maintenance of wiring in a difficult environment. Gravity feed water systems can be monitored using pressure transmitters to monitor water tank levels, pumps can be controlled using wireless I/O devices, and water use can be measured and wirelessly transmitted back to a central control center for billing. Irrigation automation enables more efficient water use and reduces waste.

1.8.6 FLEET MONITORING

It is possible to put a node on-board of each vehicle of a fleet. The node gathers its position via the GPS module, and reports its coordinates so that the location is tracked in real-time. The nodes can be connected to temperature sensors to avoid any disruption of the cold chain, helping to ensure the safety of food, pharmaceutical and chemical shipments. In situations where there is not reliable GPS coverage, like inside buildings, garages and tunnels, using information from GSM cells is an alternative for to GPS localization.

2. LITERATURE SURVEY

2.1 RELATED WORK ON SERVICE DISCOVERY

The description files of Web Services are widely distributed over the Internet where it may be centralized service registry or decentralized servers of service providers. If single service can satisfy the service request fully, it can be discovered from the Internet quickly[8]. On the other hand, it is need to compose some basic services to satisfy the complicated service request. It is impossible to compose services while discover them from the network because there are many factors prevent this such as network load and response time. Thus, Web Services composition process should be divided into three steps as same as that of web search engine, for example, Google, that is (1) WSDL (Web Service Description Language) files deployed are crawled to local system; (2) Preprocess these WSDL files and store them with an appropriate data structure; (3) Use local services information to provide composition service[7].

In the post-query model, each server posts its service(s) to a set of nodes according to the posting protocol and each client queries its desired services according to the querying protocol. A pair of Posting and Querying protocols forms a post-query protocol. In order to adapt to topological changes over time in an ad hoc network, these protocols are executed in rounds. Barbeau and Kranakis [14] propose various post-query strategies. These include:

- **Greedy** - all nodes post and query all other nodes.
- **Incremental** - all servers and clients post to and query. A small set of nodes in the first round and increase the size of these sets in subsequent rounds.

- **Uniform memory less** - servers post to a random set of nodes and clients query a random set of all nodes.
- **Uniform with memory** - in each round a new set of nodes are posted to or queried. Even though the above strategies work well and result in nodes successfully finding the services they need, all of them tend to consume high bandwidth. Multiple rounds in each strategy tend to consume a lot of resources in terms of power, bandwidth of the network, and memory in individual nodes.

For discovery and binding, the designer replaces the destination node's ID with a special constant. The binding is persistent and future requests are sent to the same node[10]. The proposed design is generic to wireless sensor networks; it does not cater to a lot of issues a service discovery protocol may have. Persistent bindings are used, which may not be ideal for wireless sensor networks. Components in a Service Discovery Protocol

- **Service Description** - Defines a service profile or template.
- **Service Query** - Defines a language and mechanism by which services can be discovered.
- **Service Acquisition and Use** - Defines a mechanism by which the service is delivered.
- **Framework** - Defines the components and interaction among those components in order to discover, acquires, and uses the service.
- **Leasing (not really a key component)** - Lifetime of a service

2.2 RELATED WORK ON SERVICE COMPOSITION

Service Composition consists of several approaches which are studied and described below. These approaches range from those aspiring to become industry standards (BPEL and OWL-S) and more abstract methods.

Business Process Execution Language for Web Services (BPEL) supports process-oriented service composition. In BPEL the result of composition is called process, participation services are partners and message exchange is called an activity[15]. A process consists of a set of activities. Among design goals associated with BPEL is to define business processes using an XML based language and to define a set of web service orchestration concepts to be used by external and internal views of business process.

Semantic web service approach aims to provide an ontology that allows software agents to discover, execute and compose web services automatically[12]. OWL-S defines an upper ontology for services which consists of a service profile for advertising and discovering services, a process model which gives a detailed description of a service's operation and a service grounding which provides details on how to interoperate with a service via message exchange.

In the Web Component approach[11] services are treated as components where the composite-logic information is encapsulated inside a class definition. A public interface of such web component can be published and used for discovery and reuse. This approach supports sequential and parallel composition constructs.

In the Algebraic Service Composition[14] approaches services are modeled as mobile processes. This approach is based on π-calculus theory and aims to provide simple descriptions which can be efficiently verified.

Services can be modeled also as Petri Nets. A Petri Net is a directed, connected and bipartite graph where nodes represent places and transitions and token occupy places. In this approach each service has a petri net where transitions are assigned to methods and places to states. A petri net describes service behavior and has one port for input and one port for output. When a petri net definition for each service is completed, a composition operator performs the composition. As a result of composition process new service is created. An advantage of this approach is that petri nets can be used to prove the absence of deadlocks and correct termination.

In Conversation specification approach individual services[3] (peers) communicate through asynchronous messages. Each individual service has a queue for incoming messages and there is a global "watcher" to keep track on messages. The conversation is a sequence of messages. This conversation among the peers models the global behavior of the service composition. In this approach services are modeled as Mealy machines which are finite-state machines with input and output. A finite state machine (FSM) or finite-state machines is a model of behavior composed of states, transitions and actions. In this approach correctness of a web service composition (properties such as data consistency, deadlock avoidance, and business-constraint satisfaction) can be verified by checking correctness inside a workflow specification.

3. PROPOSED METHOD

3.1 PROBLEM DESCRIPTION AND ASSUMPTION

The study on the theory and implementation techniques of Web Services becomes an important research issue. In general, one Web Service can only implement one basic function, therefore in order to get a more powerful Web Service it is necessary to discover and compose known Web Services.

Under the assumption that Services are provided in large scale in wireless sensor network, we implemented how to store, discover and compose them efficiently and effectively.

Based on the known references on the issue, it seems that current solutions for Web Services composition can be divided into two categories, i.e.

(1) Tree(graph)-based Web Services Composition (TWSC): There is no fixed or predefined process model and a service requester only knows what he can provide and what he need but does not know the details of composition process. TWSC is implemented automatically without the support from domain experts. In which, uses graph-based approach and interface automata for services composition. Every service is defined as ws = (A^I_{ws} , A^O_{ws}, K_{ws}) where A^I_{ws} is inputs, A^O_{ws} is outputs and K_{ws} is dependency between A^I_{ws} and A^O_{ws}. Dependency Graph is used for storing all services and their dependencies. A Breadth First Search (BFS) algorithm is constructed to get the WSCR. However, it does not introduce how to construct the graph efficiently.

(2)Process-based Web Services Composition (PWSC): There are predefined process models and descriptions for the activities in details beforehand. PWSC is carried out based on the support from domain experts, where describes the task process model and the specific activities in it with UML. Every activity matches with services according to the text descriptions of them. If there is no matched service, new one is created.[1]

3.2. THE DISCOVERY PROBLEM

Given a repository of Web services, and a query requesting a service (we refer to it as the query service in the rest of the text), automatically finding a service from the repository that matches these requirements is the Web service Discovery problem[3].

3.3. THE COMPOSITION PROBLEM

Given a repository of service descriptions, and a query with the requirements of the requested service, if a matching service is not found, then the composition task can be performed. The composition problem involves automatically finding a directed acyclic graph of services that can be composed to obtain the desired service.

3.4 SEARCH OPERATION

This section presents the formal descriptions of mutual search operations among service operations (Op), inputs (I) and outputs (O) which are necessary in the process of Web Services composition. We divide them into three categories and give the formal descriptions as follows:

(1) I -> Ops, O⇒ Ops. The symbol I ⇒ Ops denotes the search operation, i.e., for a given input I, to get all service operations Ops whose inputs contain I. And, O⇒ Ops denotes the search operation, i.e., for a given output O, to get all service operations Ops whose outputs contain O.

Table 1: The General Process of Service Composition

<table>
<tr><th>Front to Back Web Service Composition</th><th>Back to Front Web Service Composition</th></tr>
<tr><td colspan="2"><b>Input:</b> request(input, output), web services set to be composed
<b>Output:</b> service composition result</td></tr>
<tr><td colspan="2"><b>Step 1</b>. Find the order of service composition by the following operations:</td></tr>
<tr><td>1)Search service operations whose inputs contain the given input</td><td>1)Search service operations whose outputs contain the given output</td></tr>
<tr><td>2)Search outputs of the given service operation</td><td>2)Search inputs of the given service operation</td></tr>
<tr><td>3)Search input matched to the given output</td><td>3)Search output matched to the given input</td></tr>
<tr><td colspan="2"><b>Step 2</b>. Execute the order to obtain the optimal WSCR</td></tr>
</table>

(2) Op ⇒ Iop, Op ⇒ Oop. The symbol Op ⇒ Iop denotes the search operation, i.e., for a given service operation Op, to get its inputs Iop. And, Op ⇒ Oop denotes the search operation, i.e., for a given service operation Op, to get its outputs Oop.

(3) I ⇔ O denotes the mutually match operation between input I and output O.(1) - (3) present the mutual search operations among service operations, inputs and outputs which are iterated to build the InvertedFile[1].

Table 2: Search Operations Mapping

Op, Input, Output	**Op, Para**
I => Ops, O => Ops	Para => Ops
Op => I_{op} ,Op => O_{op}	Op => Paras
Input ⇔ Output	$Para_I$ ⇔ $Para_O$

3.5 INVERTED FILE

In this section, we propose the Inverted File to implement the mutual search operations in services composition process based on Op and Para shown in Table 2. InvertedFile uses Para as index. There are two kinds of objects ParaObj and OpObj in InvertedFile whose definitions are as follows:

Definition 1 (Para Object, ParaObj): ParaObj= {paraName, inOpList, outOpList}.

paraName is the name of this Para. inOpList is the pointer list of all OpObjs whose inputs contain Para; outOpList is the pointer list of all OpObjs whose outputs contain Para.

Definition 2 (Op Object, OpObj): OpObj={IDws:IDop, inParaList, outParaList}.

IDws:IDop is the unique identifier of this Op in WSSet. inParaList and outParaList are, respectively, the pointer list of input Para object ParaObjs and output Para object ParaObjs contained by this Op.

Although algorithm 1 costs time slightly, it only prepares for services composition and can run offline. We can use Inverted File to provide real time composition service for millions of service requests. Therefore, Algorithm 1 has no effect on the real time performance of services composition.

ALGORITHM 1: Construction of Inverted File

```
INPUT  : ServiceSet
OUTPUT: InvertedFile
for each Service in ServiceSet
    Create serviceobject
    for each Parameter
        if ParaObj not exists
            Create ParaObj
        else
            get ParaObj in the InvFile
        add ParaObj to IF in the ascending order of ParameterName
    if ParaObj is input
        add it in the input list of the Service
        add the Service in the input list of the Parameter
    else
        add it in the output list of the Service
        add the Service in the output list of the Parameter
    return InvertedFile;
```

Based on InvertedFile, it requires O(logn)(n is the number of ParaObjs in InvertedFile) time to find any ParaObj according to the given ParaName; requires O(1) time to find all operations OpObjs whose inputs or outputs contain this ParaObj, that is, in O(1) time to complete Para ⇒ Ops; requires O(1) time to find all inputs or outputs ParaObjs of a certain OpObj, that is, in O(1) time to complete Op ⇒ Paras; and requires O(1) time to match between input and output mutually, that is, in O(1) time to complete ParaI⇔ ParaO. In short, composition time will be greatly reduced.

Table 3: Web Services, Inputs and Outputs

Web Services ID	Input Name	Output Name
WS_1	I_5	O_2
WS_2	I_3	O_7
WS_3	I_3	O_5
WS_4	I_4	O_2
WS_5	I_6	O_4
WS_6	I_8	O_2
WS_7	I_1	O_4
WS_8	I_1	O_6
WS_9	I_8	O_6

In Table 3, I and O with the same subscript denote one and the same. For example, I1 and O1 is the same.Table 4 shows the mapping among Input, Output andPara according to definition 1.

Table 4: Mapping among I, O and Para

I	O	Para
I_1		P_1
	O_2	P_2
I_3		P_3
I_4	O_4	P_4
I_5	O_5	P_5
I_6	O_6	P_6
	O_7	P_7
I_8		P_8

Figure 1 shows ServiceInvertedFile constructed from Table 3 by algorithm 1. Pi(i=1…8) presented. As is shown in Figure 1, it needs O(log8) time to discover any Para from Figure 1.a) (e.g. find P5).Only in O(1) time all Ops whose inputs or outputs contain this Para can be searched (e.g. find WS1:Op1whose inputs contain P5 or WS3:Op1 whose outputs contain P5). All inputs and outputs of any given Op can be searched from Figure 1.b) in O(1) time (e.g. find input P5 and output P2 of the given WS1:Op1). And in O(1) time to complete ParaI ⇔ ParaO (e.g. for the given input P5, the matched output is itself because P5.OutOpList is not null. while for the given output P2, there is no matched input because P2.inOpList is null). It is clear that InvertedFile builds these search operations among operations, inputs and outputs so well that makes them search each other quickly.

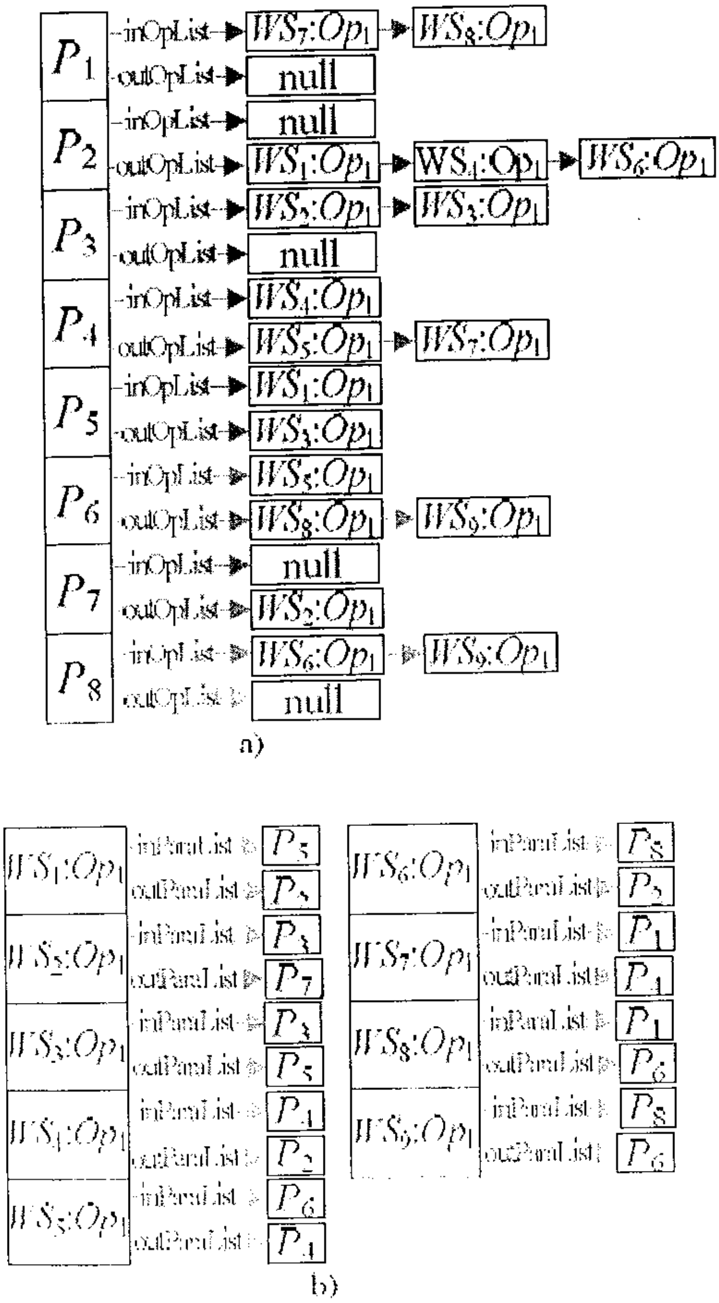


Figure 3: a)Parameter Inverted File b)Service Inverted File

3.6 SERVICE COMPOSITION BASED ON INVERTED FILE

This section defines a search operation that needs to be carried out in the Inverted file which determines all possible services for composition. Here, algorithm 2 performs a back to front composition.

ALGORITHM 2: All Possible Service for Composition

INPUT : Service Inverted File(SIF),Input I_o and Output OP parameters.

OUTPUT : All possible services available for composition.

We used Back to Front Service Composition

```
Service Composition(SIF , $I_o$ ,OP)
for each service in the SIF
    for each OP in the service
        Compare the service output of SIF with the OP
        If it matches
            Save the Input parameter($I_1$) of the service
            Save the name of the service
            Set a flag to 1
If flag is set to 1 and $I_1$ not equal to null then
    Compare $I_1$ with $I_0$
    If it does not match
        Call Service Composition with $I_1$ as OP
    else if it matches
        Return the names of all possible services for composition
else
    return null;
```

3.7 OPTIMAL COMPOSITION

Currently, a lot of research has been done on the QoS of service composition result. However, most of evaluate factors for QoS (e.g., response time, reliability and security) are too abstract to be obtained. And, most of these systems to compute QoS are too complicated to satisfy the real time properties of services composition. Toward this problem, we proposed a succinct and effective method to ensure the QoS of Web Service composition result.

(1) The number of Web Service operations in WSCR. Generally, if a WSCR has more Web Service operations, its run time may be longer and its price may be higher. In addition, there inevitably have message interactions between different operations in WSCR. The more the message interactions is, the less the performance time of WSCR is. Furthermore, if a piece of message is exchanged among different Web Service operations which are provided by different servers, they need to be transmitted by network. It is obvious that has disadvantageous effect on the performance of WSCR. Therefore, the number of Web Service operations of WSCR is in inverse proposition to the QoS.

(2) The number of level in WSCR. Web Services in the same level (i.e. node) of WSCR can run in parallel, however, these in the different level must run sequentially. The more the number of levels in WSCR is, the more the running time is needed. In a word, the number of level in WSCR is in inverse proposition to the QoS.

(3) The number of request inputs. If the WSCR needs more request input, the requester should provide more information and the run time of system may be longer. So, the number of request input in WSCR is still in inverse proposition to the QoS.

ALGORITHM 3: Optimal Composition

```
INPUT    : ParaInvertedFile(PIF),ServiceInvertedFile(SIF),Input(IP)and
Output(OP) parameters.
OUTPUT :Best service for composition
BestComposition(PIF,SIF,IP,OP)
Initialize the number of operation(Nop) to some infinite value
do
        for each possible parameter(P) in PIF
                Compare the parameter name with the OP if matches
                        Save the service name(S) of  PIF
        for each S in the SIF
                Compare S with the service name in SIF
                        If it matches
Check whether the operation(OPs) of the output service is less than the (Nop)
                                If it is less
                                        Assign the  Nop to OPs
                                        Save the OP1 of SIF to  OP
                                        Save the service name
Until OP matches to the given IP parameter
return the best service
```

4. IMPLEMENTATION DETAILS

4.1 IMPLEMENTATION SPECIFICATIONS

Language	: C# & .Net
Platform	: Microsoft Windows
Implementation Environment	: Visual Studio 2008
Framework	: Net framework3.5
Operating System used	: Windows 7 Ultimate
Database	: MySQL
Server	: ASP.Net Development Server

4.2 MySQL

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. It is named after developer Michael Widenius' daughter, My. The SQL phrase stands for Structured Query Language.

The official MySQL Workbench is a free integrated environment , that enables users to graphically administer MySQL databases and visually design database structure. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL frontend, MySQL Workbench lets users manage the following:

- Database design & modeling
- SQL development – replacing MySQL Query Browser

- Database administration – replacing MySQL Administrator

MySQL Workbench is available in two editions, the regular free and open source *Community Edition* which may be downloaded from the MySQL website, and the proprietary *Standard Edition* which extends and improves the feature set of the Community Edition.

4.3 C#

C# is a multi-purpose computer programming language suitable for all development needs. C# is derived from the C programming language, it has features such as garbage collection that allow beginners to become proficient in C# more quickly than in C or C++. Similar to Java, it is object-oriented, comes with an extensive class library, and supports exception handling, multiple types of polymorphism, and separation of interfaces from implementations. Those features, combined with its powerful development tools, multi-platform support, and generics, make C# a good choice for many types of software development projects: rapid application development projects, projects implemented by individuals or large or small teams, Internet applications, and projects with strict reliability requirements. Testing frameworks such as N Unit make C# amenable to test-driven development and thus a good language for use with Extreme Programming (XP). Its strong typing helps to prevent many programming errors that are common in weakly typed languages.

4.4 SENSOR NETWORK

Sensor network is abstracted as database in our implementation. We used Benchmark data values of Temperature, Heartbeat, Blood Pressure, ECG, etc.. in health care domain for our implementation.

4.5 IMPLEMENTATION CONSTRAINTS

- ✓ To enter multiple input or output parameters we used comma to separate the parameters.
- ✓ Data can be refreshed through removing the currently running queries in the presentation layer and hence latest reading can be obtained.

4.6 MODULES DESCRIPTION

4.6.1 SERVICE CREATION

In this module, we have consider the services available in the wireless sensor network and we have a created 80 such services in ASP.net environment.

4.6.2 SERVICE DISCOVERY (SD)

This module is required for the proper functioning of the service composition module. An important component of the SD function is to match and recommend appropriate services among a possibly large set of service instances, taking benefits from context-information. When a client requests a service it may be necessary to compose a complex service out of the registered basic services.

4.6.3 SERVICE COMPOSITION

This module is responsible for carrying out the process on managing the discovery of services to yield a composite service. It further refines the set of selected services using the remaining context parameters that are relevant for the composition rather than for the discovery.

4.6.4 SERVICE EXECUTION

This module is responsible for carrying out the execution of the composite service. Prior to this the Service Composition module provides a feasible order in which these services can be executed. Mainly two types of execution exist according to the nature of the service, which can be either an electronic service (e-service) or a mobile service (m-service). For m-services execution is always local to the device of execution since the service itself has been transported to the device. However for e-services the execution can be remote or local. In the remote invocation the client sends remotely a request to a provider asking the execution of a service. The execution takes place in the provider platform. In local invocation the client asks remotely to transfer a copy of the service to its site. After being transferred, the execution takes place in the client site.

5. RESULTS AND EVALUATION

We implemented above algorithms consisting of 80 Web Services crawled from [16-18]etc. We carried out a series of experiments on an 2.10 GHz 3 GB Intel(R) Core (TM) 2 Duo machine running Windows 7 ultimate to study the performance of WS composition approach. From the graph we infer that our algorithm reduces the search time compared to the previous algorithm. In addition to that service composition reduces the energy consumption by executing only the required data services. Obviously, the performance gain of the proposed method will increase as the number of services increase.

First, we compare the approach based on Process Model with this based on InvertedFile. As is shown in Figure 4, it is clear that the performance of the latter is higher than that of the former. And besides, the time growth rate of the latter is much lower than that of the former. In short, the approach in this project is not only with high performance but also with low time growth rate.

Time of web services composition based on Process Model versus that based on InvertedFile .Then, Figure 5 shows the performance based on Process Model versus that based on InvertedFile as the level of WSCR grows from 1 to 4. Here, we fix the number of Web Services to be 80. As expected, the latter is better than the former, and, the latter almost keeps steady when the level of WSCR grows.

Figure 4: Time of web service composition based on InvertedFile versus Process Model.

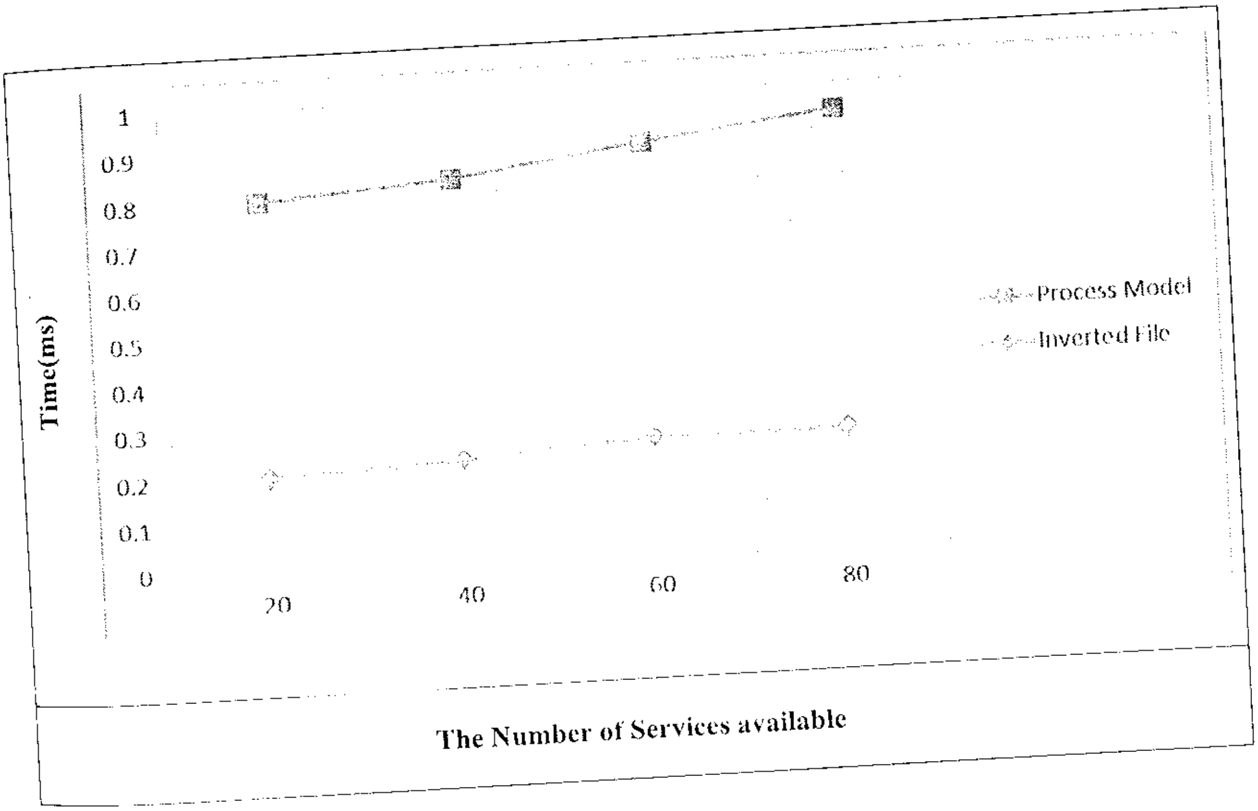


Figure 5: Time of web services composition as the level of WSCR grows from 1 to 4.

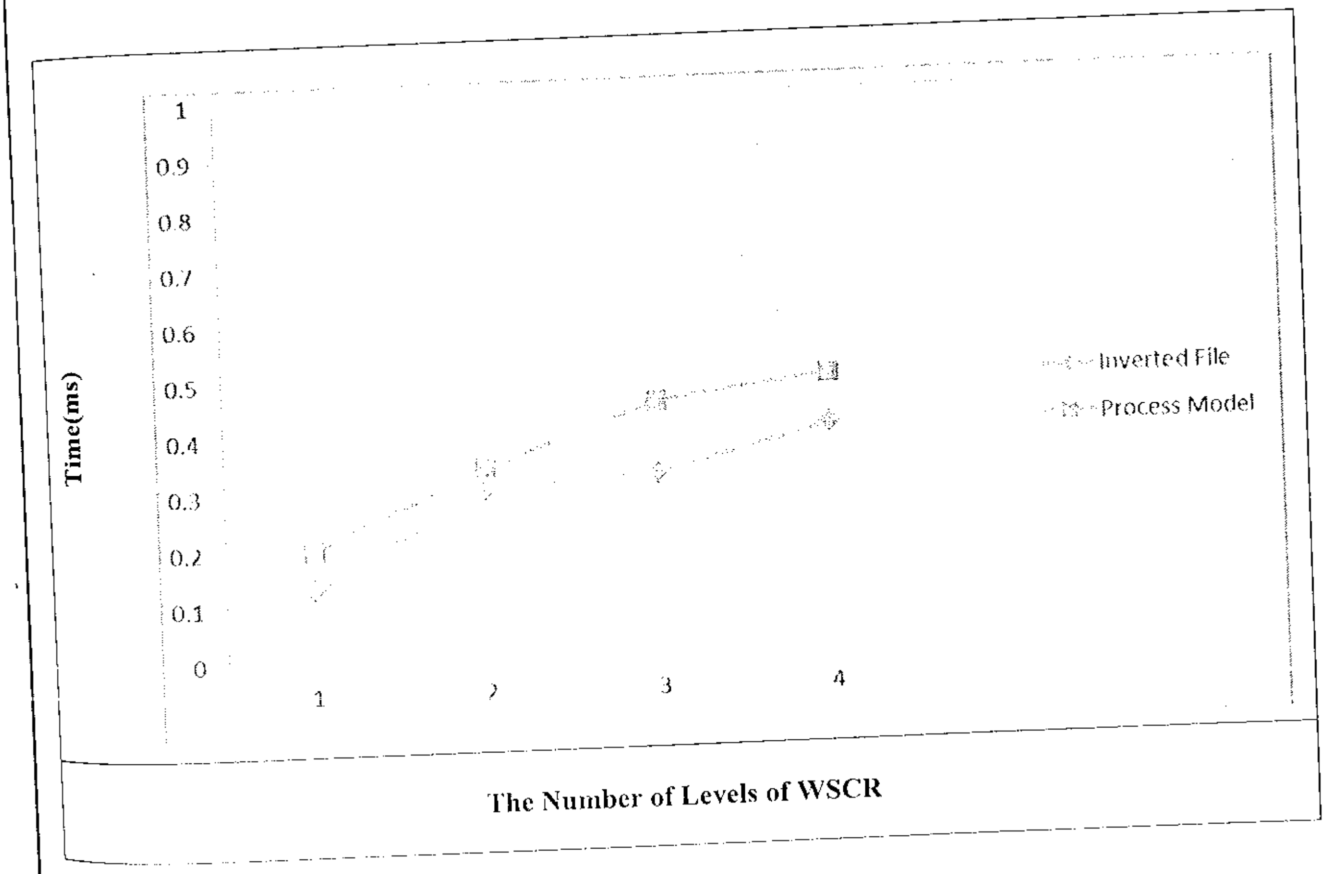


Table 5: DATASET

SERVICES	INPUT	OUTPUT
Room	RoomNo	PatientID
PatientLocation	PatientID	PatientLocation
HBandBP	PatientID	HB,BP
Level	HB,BP	Level
DischargeOK	Level	DischargeOK
Doctor	PatientID	DoctorID
Location	StaffID	S_Location
PatientDetails	PatientID	Disease,DoctorID
DoctorName	DoctorID	D_Name,D_Dept
Staff	PatientID	StaffID
StaffName	StaffID	StaffName,S_Location
PatDocStaDetails	PatientID	DoctorID,StaffID
Ambulance	AmbulanceID	A_Loc,DrID,DrName
Status	StaffID	Present/Absent
PatientLocation	PatientID	PatientLocation
DoctorLocation	DoctorID	DoctorLocation
Machine	MachineID	ON/OFF
Test	PatientID	Temp,HB,BP
BGroupAvail	BGroup	Available

BGroupBranch	Available	Branch
BGroupAvailBranch	BGroup	Branch,Available
BGroupDono	BottleNo	Donotor
BottleTempGlad	BottleNo	Temp,gladrate
GetBottleNo	PatientID	BottleNo
GetBloodetails	BottleNo	HB,WBC,RBC,Platlets
GetHB	BottleNo	HB
GetWBC	BottleNo	WBC
GetRBC	BottleNo	RBC
GetPlatlets	BottleNo	Platlets
MedicineAvail	MedicineName	Available
GetMedicineName	MedicineID	Medicine_Name
GetShelfNo	MedicineID	ShelfNo
Domain	MedicineID	Domain
Medical_shop_Loc	MedicineID	Location,Specification
Medicine_Price	MedicineID	Price
Medicine_stockLevel	MedicineID	Stock_level
Balance	PatientID	Balance
GetGlucoseCount	PatientID	Glucose_Count
EmergencyAlarm	Oxygen_Level	Alarm ON/OFF
GetECG	PatientID	ECG
GetEEG	PatientID	EEG
GetUrineTest	PatientID	UrineTest
GetUltraSona	PatientID	UltraSonagraphy
GetXray	PatientID	X-ray
GetScan	PatientID	Scan

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This project gives a full review on current solutions of Web Services composition and groups them into two categories: TWSC and PWSC. So, from different previous study, this project shows how to discover and compose Web Services according to the input and output of service request when there are a large number of services available. We believe that building the InvertedFile is the key of Web Services composition. Based on this, the preparatory work is to carry out the mutual search operations among service operation, inputs and outputs in the InvertedFile. For any given service request, InvertedFile is searched with the given Parameters. And then, the best WSCR in terms of QoS is obtained from that search output. In fact, other Web Service composition algorithms, e.g. those based on graph or process, can also achieve good performance based on InvertedFile.

6.2 FUTURE WORK

In this project, we only discuss the exact matching between input and output by the name, so, the next step is to implement the fuzzy matching. In addition, another future work is to include the service semantics in the matching process.

7. APPENDIX

7.1 CODING

```
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 MySql.Data.MySqlClient;
using System.Xml.Linq;
using System.Diagnostics;

public partial class  Default : System.Web.UI.Page
{
  string[] xy2 = new string[10];
  string[] xy = new string[10];
  ss[] service = new ss[39];
  ss[] para = new ss[50];
  static int b = 0;
  DataSet ds = new DataSet();

  public struct ss
  {
    public string name;
    public string[][] pa;

    public ss(string p1, string[][] p2)
    {
      name = p1;
      pa = p2;
    }
```

```
public string[] best(ss[] p, ss[] ser, string[] ip, string[] op)
{
  string[] wsn = new string[10];
  string[] wsb = new string[10];
  int l = 455;
  int j = 0, d = 0,t=0;
  do
  {
    l = 455;
    for (int x = 1; x < p.Length; x++)
    {
      for (int h = 0; h < op.Length; h++)
      {
        if (p[x].name.CompareTo(op[h]) == 0)
        {
          for (int i = 0; i < p[x].pa[1].Length; i++)
          {
            wsn[i] = p[x].pa[1][i];
          }
        }
      }
    }

    for (int y = 0; y < wsn.Length; y++)
    {
      if (wsn[y] != null)
      {
        for (j = 0; j < ser.Length; j++)
        {
          if (ser[j].name.CompareTo(wsn[y]) == 0)
          {
            if (ser[j].pa[1].Length < l)
            {
              l = ser[j].pa[1].Length;
              if (ser[j].pa[0][0] != null)
              {
                op[d] = ser[j].pa[0][0];
                op[1] = " ";
                op[2] = " ";
                wsb[t] = ser[j].name;
```

```
                                t++;
                            }
                        }
                    }
                }
            }
        }
    } while (op[d].CompareTo(ip[0]) != 0);
    return wsb;
}

public string[] composition(string[] output, ss[] ser, string[] name, string[] inp)
{
    string s = string.Empty;
    int flag = 0, k = 0, f = 0;
    string[] input = new string[10];

    for (int x = 0; x < ser.Length; x++)
    {
        for (int i = 0; i < ser[x].pa[1].Length; i++)
        {
            for (int j = 0; j < output.Length; j++)
            {
                if (ser[x].pa[1][i].CompareTo(output[j]) == 0)
                {
                    input[k] = ser[x].pa[0][k];
                    for (int p = 0; p < name.Length; p++)
                    {
                        if (name[p] == ser[x].name)
                            f = 1;
                    }
                    if (f == 0)
                    {
                        name[b] = ser[x].name;
                        if (b == 9)
                        {
                            b = 0;
                        }
                        b++;
                        flag = 1;
                    }
                }
             }
          }
       }

       if (flag == 1 && input != null)
       {
          for (int i = 0; i < input.Length; i++)
          {
             if (input[i] != inp[i])
             {
                 composition(input, ser, name, inp);
             }
             else if (input[i] == inp[i])
             {
                Console.WriteLine(input[i]);
                Console.WriteLine(inp[i]);
                return name;
             }
          }
       }
       for (int k1 = 0; k1 < name.Length; k1++)
       {
          name[k1] = null;
       }
       return name;
    }
    public string invoke(string ws,string t,string t1)
    {
       wsn.Service ss = new wsn.Service();
       string s=string.Empty;
       if (ws == "Getlevel")
       {
          s = ss.GetLevel(t, t1);
       }
       return s;
    }
   public DataSet invoke(string ws,string t)
    {
       DataSet ds = new DataSet();
```

```
wsn.Service ss = new wsn.Service();
if (ws == "GetPatientLocation")
{
   ds = ss.GetPatientLocation(t);
}
else if (ws == "Ambulance")
{
   ds = ss.Ambulance(t);
}
else if (ws == "GetDoctorDetails")
{
   ds = ss.GetDoctorDetails(t);
}
else if (ws == "Discharge")
{
   ds = ss.DischargeOK(t);
}
else if (ws == "Machine")
{
   ds = ss.Machine(t);
}
else if (ws == "GetDepartment")
{
   ds = ss.GetDepartment(t);
}
else if (ws == "GetDiseaseDetails")
{
   ds = ss.GetDiseaseDetails(t);
}
else if (ws == "GetHBandBP")
{
   ds = ss.GetHBandBP(t);
}
else if (ws == "GetDoctorDepartment")
{
   ds = ss.GetDoctorDepartment(t);
}
else if (ws == "GetPatientID")
{
   ds = ss.GetPatientID(t);
}
```

```
else if (ws == "GetDoctorID")
{
   ds = ss.GetDoctorID(t);
}
else if (ws == "GetDoctorLocation")
{
   ds = ss.GetDoctorLocation(t);
}
else if (ws == "GetDoctorName")
{
   ds = ss.GetDoctorName(t);
}
else if (ws == "GetPatientDetails")
{
   ds = ss.GetPatientDetails(t);
}
else if (ws == "GetPatientName")
{
   ds = ss.GetPatientName(t);
}
else if (ws == "GetPatientTestDetails")
{
   ds = ss.GetPatientTestDetails(t);
}
else if (ws == "GetStaffDetails")
{
   ds = ss.GetStaffDetails(t);
}
else if (ws == "GetStaffID")
{
   ds = ss.GetStaffID(t);
}
else if (ws == "GetStaffLocation")
{
   ds = ss.GetStaffLocation(t);
}
else if (ws == "GetStaffName")
{
   ds = ss.GetStaffName(t);
}
return ds;
```

```
        }

public void ssss(string[] output, ss[] ser, string[] name, string[] inp)
        {
        }

        public string search(string[] r1, string[] r2, ss[] ser)
        {
            int flag = 0, flag1 = 0, c = 0,v=0,j=0;

            for (int x = 0; x < ser.Length; x++)
            {
                c = 0; v = 0;
                for (int k = 0; k < r1.Length; k++)
                {
                    if (r1[k] != null)
                    {
                        v++;
                    }
                }
                if (v == ser[x].pa[0].Length)
                {
                    for ( j = 0; j < ser[x].pa[0].Length; j++)
                    {
                        if (ser[x].pa[0][j] == r1[j])
                        { }
                    }
                }
                if (j == v && ser[x].pa[0][j-1].CompareTo(r1[j-1])==0)
                    flag = 1;

                if (flag == 1)
                {
                    for (int l = 0; l < r2.Length; l++)
                    {
                        if (r2[l] != null)
                        {
                            c++;
                        }
                    }
                if (c == ser[x].pa[1].Length)
                {
                    for (int l = 0; l < c; l++)
                    {
                        if (ser[x].pa[1][l].CompareTo(r2[l]) == 0)
                        {
                            flag1++;
                        }
                    }
                }
            }

            if (flag1 == c && flag == 1)
            {
                return ser[x].name;
            }
            flag = 0;
            flag1 = 0;
        }
        return null;
    }
}

protected void Page_Load(object sender, EventArgs e)
{
    //SERVICES
    string[][] p10 = new string[2][];
    p10[0] = new string[] { "pid" };
    p10[1] = new string[] { "did", "disease" };
    service[1].name = "GetDiseaseDetails";
    service[1].pa = p10;

    string[][] p1 = new string[2][];
    p1[0] = new string[] { "aid" };
    p1[1] = new string[] { "aloc", "drid", "drname" };
    service[0].name = "Ambulance";
    service[0].pa = p1;

    string[][] p2 = new string[2][];
    p2[0] = new string[] { "pid" };
```

```
p2[1] = new string[] { "ploc" };
service[9].name = "GetPatientLocation";
service[9].pa = p2;

string[][] p3 = new string[2][];
p3[0] = new string[] { "level" };
p3[1] = new string[] { "DischargeOK" };
service[2].name = "Discharge";
service[2].pa = p3;

string[][] p4 = new string[2][];
p4[0] = new string[] { "pid" };
p4[1] = new string[] { "hb", "bp" };
service[3].name = "GetHBandBP";
service[3].pa = p4;

string[][] p5 = new string[2][];
p5[0] = new string[] { "hb", "bp" };
p5[1] = new string[] { "level" };
service[4].name = "Getlevel";
service[4].pa = p5;

string[][] p6 = new string[2][];
p6[0] = new string[] { "roomNo" };
p6[1] = new string[] { "pid" };
service[5].name = "GetPatientID";
service[5].pa = p6;

string[][] p7 = new string[2][];
p7[0] = new string[] { "mid" };
p7[1] = new string[] { "mstatus" };
service[6].name = "Machine";
service[6].pa = p7;

string[][] p8 = new string[2][];
p8[0] = new string[] { "did" };
p8[1] = new string[] { "dname","dloc","dept" };
service[7].name = "GetDoctorDetails";
service[7].pa = p8;

string[][] p9 = new string[2][];
```

```
p9[0] = new string[] { "dept" };
p9[1] = new string[] { "deptloc" };
service[8].name = "GetDepartment";
service[8].pa = p9;

string[][] p11 = new string[2][];
p11[0] = new string[] { "did" };
p11[1] = new string[] {  "dept" };
service[10].name = "GetDoctorDepartment";
service[10].pa = p11;

string[][] p12 = new string[2][];
p12[0] = new string[] { "pid" };
p12[1] = new string[] { "did" };
service[11].name = "GetDoctorID";
service[11].pa = p12;

string[][] p13 = new string[2][];
p13[0] = new string[] { "did" };
p13[1] = new string[] { "dloc" };
service[12].name = "GetDoctorLocation";
service[12].pa = p13;

string[][] p14 = new string[2][];
p14[0] = new string[] { "did" };
p14[1] = new string[] { "dname" };
service[13].name = "GetDoctorName";
service[13].pa = p14;

string[][] p15 = new string[2][];
p15[0] = new string[] { "pid" };
p15[1] = new string[] { "did","sid" };
service[14].name = "GetPatientDetails";
service[14].pa = p15;

string[][] p18 = new string[2][];
p18[0] = new string[] { "sid" };
p18[1] = new string[] { "sname", "sloc", "status" };
service[15].name = "GetStaffDetails";
service[15].pa = p18;
```

```
string[][] p17 = new string[2][];
p17[0] = new string[] { "pid" };
p17[1] = new string[] { "hb", "bp", "temp" };
service[16].name = "GetPatientTestDetails";
service[16].pa = p17;

string[][] p19 = new string[2][];
p19[0] = new string[] { "pid" };
p19[1] = new string[] { "sid" };
service[17].name = "GetStaffID";
service[17].pa = p19;

string[][] p20 = new string[2][];
p20[0] = new string[] { "sid" };
p20[1] = new string[] { "sloc" };
service[18].name = "GetStaffLocation";
service[18].pa = p20;

string[][] p21 = new string[2][];
p21[0] = new string[] { "sid" };
p21[1] = new string[] { "sname" };
service[19].name = "GetStaffName";
service[19].pa = p21;

string[][] p16 = new string[2][];
p16[0] = new string[] { "pid" };
p16[1] = new string[] { "pname" };
service[20].name = "GetPatientName";
service[20].pa = p16;

string[][] p22 = new string[2][];
p22[0] = new string[] { "BGroup" };
p22[1] = new string[] { "avail" };
service[22].name = "GetAvailability";
service[22].pa = p22;

string[][] p23 = new string[2][];
p23[0] = new string[] { "BGroup", "Branch" };
p23[1] = new string[] { "avail" };
service[23].name = "GetAvailabilityBranch";
service[23].pa = p23;
```

```
string[][] p24 = new string[2][];
p24[0] = new string[] { "BNo" };
p24[1] = new string[] { "Donotor" };
service[24].name = "GetDonotor";
service[24].pa = p24;

string[][] p25 = new string[2][];
p25[0] = new string[] { "BNo" };
p25[1] = new string[] { "Temp", "GlatRate" };
service[25].name = "GetTempGlatrate";
service[25].pa = p25;


string[][] p26 = new string[2][];
p26[0] = new string[] { "Temp", "GlatRate" };
p26[1] = new string[] { "Usable" };
service[26].name = "GetUsable";
service[26].pa = p26;

string[][] p27 = new string[2][];
p27[0] = new string[] { "BNo" };
p27[1] = new string[] { "BGroup" };
service[27].name = "GetBGroup";
service[27].pa = p27;

string[][] p28 = new string[2][];
p28[0] = new string[] { "pid" };
p28[1] = new string[] { "BNo" };
service[28].name = "GetBNo";
service[28].pa = p28;

string[][] p29 = new string[2][];
p29[0] = new string[] { "BNo" };
p29[1] = new string[] { "HG" };
service[29].name = "GetHG";
service[29].pa = p29;

string[][] p30 = new string[2][];
p30[0] = new string[] { "BNo" };
p30[1] = new string[] { "WBC", "RBC", "Platlets" };
```

```
service[30].name = "GetWbcRbcPlat";
service[30].pa = p30;

string[][] p31 = new string[2][];
p31[0] = new string[] { "MedicineName" };
p31[1] = new string[] { "MedicineAvail" };
service[31].name = "GetMedicineAvail";
service[31].pa = p31;


string[][] p32 = new string[2][];
p32[0] = new string[] { "MedicineID" };
p32[1] = new string[] { "MedicineName" };
service[32].name = "GetMedicineName";
service[32].pa = p32;

string[][] p33 = new string[2][];
p33[0] = new string[] { "MedicineID" };
p33[1] = new string[] { "ShelfNo" };
service[33].name = "GetShelfNo";
service[33].pa = p33;

string[][] p34 = new string[2][];
p34[0] = new string[] { "MedicineID" };
p34[1] = new string[] { "Domain" };
service[34].name = "GetDomain";
service[34].pa = p34;

string[][] p35 = new string[2][];
p35[0] = new string[] { "MedicineShopID" };
p35[1] = new string[] { "ShopLocation", "Specialization" };
service[35].name = "GetLocSpecial";
service[35].pa = p35;

string[][] p36 = new string[2][];
p36[0] = new string[] { "MedicineID" };
p36[1] = new string[] { "Price" };
service[36].name = "GetPrice";
service[36].pa = p36;

string[][] p37 = new string[2][];
```

```
p37[0] = new string[] { "MedicineID" };
p37[1] = new string[] { "StockLevel" };
service[37].name = "GetStockLevel";
service[37].pa = p37;

string[][] p38 = new string[2][];
p38[0] = new string[] { "pid" };
p38[1] = new string[] { "Balance" };
service[38].name = "GetBalance";
service[38].pa = p38;

string[][] p39 = new string[2][];
p39[0] = new string[] { "pid" };
p39[1] = new string[] { "GlucloseNo" };
service[21].name = "GetGulcloseNo";
service[21].pa = p39;

string in1, in2;
in1 = TextBox1.Text.ToString();
in2 = TextBox2.Text.ToString();
char[] ch = new char[10];
char[] ch1 = new char[10];
int x = 0, x1 = 0;

for (int i = 0; i < in1.Length; i++)
{
   if (in1[i] == ',')
   {
      x = 1;
   }
}

for (int k = 0; k < in2.Length; k++)
{
   if (in2[k] == '.')
      x1 = 1;
}

if (x1 == 0)
{
   xy2[0] = in2;
```

```
}

if (x1 == 1)
{
    int v = 0, u = 0;
    for (int y = 0; y < in2.Length; y++)
    {

        if (in2[y] == ',')
        {
            ch[u] = '\0';
            xy2[v] = new string(ch);
            v++;
            u = 0;
        }
        else
        {
            ch[u] = in2[y];
            u++;
            if (y == in2.Length - 1)
            {
                ch[u] = '\0';
                xy2[v] = new string(ch);
            }
        }
    }
}
if (x == 0)
{
    xy[0] = in1;
}
if (x == 1)
{
    int v = 0, u = 0;
    for (int y = 0; y < in1.Length; y++)
    {
        if (in1[y] == ',')
        {
            ch1[u] = '\0';
            xy[v] = new string(ch1);
            v++;
```

```
                u = 0;
            }
            else
            {
                ch1[u] = in1[y];
                u++;
                if (y == in1.Length - 1)
                {
                    ch1[u] = '\0';
                    xy[v] = new string(ch1);
                }
            }
        }
    }
    string s = string.Empty;
}

protected void Button1_Click(object sender, EventArgs e)
{
    Stopwatch sw = new Stopwatch();
    sw.Start();
    ss ser = new ss();

    string[] nam = new string[10];
    string k = ser.search(xy, xy2, service);
    if (k != null)
    {
        Label4.Visible = true;
        TextBox3.Visible = true;
        TextBox3.Text = k;
        Label5.Visible = true;
        TextBox5.Visible = true;
        Button2.Visible = true;
        if (TextBox3.Text == "Getlevel")
        {
            Label6.Visible = true;
            TextBox5.Visible = true;
            TextBox4.Visible = true;
        }
    }
    else
    {
      string[] nm = new string[10];
      ser.ssss(xy2, service, nam, xy);
      nm = ser.composition(xy2, service, nam, xy);
      for (int i = 0; i < nm.Length; i++)
      {
        if (nm[i] != null)
        {
          ListBox1.Items.Add(nm[i]);
        }
      }
      if (ListBox1.Items.Count == 0)
      {
        Label7.Visible = true;
      }
      else
      {
        Label3.Visible = true;
        Label12.Visible = true;
        ListBox1.Visible = true;
        Button4.Visible = true;
      }
    }
    sw.Stop();
    Response.Write(sw.Elapsed.TotalSeconds);
  }
  protected void Button2_Click(object sender, EventArgs e)
  {
    Label8.Visible = true;
    wsn.Service ss = new wsn.Service ();
    string ws= TextBox3.Text;
    if (ws =="GetPatientLocation" )
    {
      ds = ss.GetPatientLocation(TextBox5.Text);
    }
    else if (ws == "Ambulance")
    {
      ds = ss.Ambulance(TextBox5.Text);
    }
    else if(ws =="GetDoctorDetails")
    {
```

```
  ds = ss.GetDoctorDetails(TextBox5.Text);
}
else if(ws=="Discharge")
{
  ds = ss.DischargeOK(TextBox5.Text);
}
else if (ws == "Machine")
{
  ds = ss.Machine(TextBox5.Text);
}
else if (ws == "Getlevel")
{
  string s = ss.GetLevel(TextBox5.Text, TextBox4.Text);
  Label9.Visible = true;
  Label9.Text = s;
}
else if (ws == "GetDepartment")
{
  ds = ss.GetDepartment(TextBox5.Text);
}
else if (ws == "GetDiseaseDetails")
{
  ds = ss.GetDiseaseDetails(TextBox5.Text);
}
else if (ws == "GetHBandBP")
{
  ds = ss.GetHBandBP(TextBox5.Text);
}
else if (ws == "GetDoctorDepartment")
{
  ds = ss.GetDoctorDepartment(TextBox5.Text);
}
else if (ws == "GetPatientID")
{
  Response.Write(ds.Tables.Count);
  ds = ss.GetPatientID(TextBox5.Text);
}
else if (ws == "GetDoctorID")
{
  ds = ss.GetDoctorID(TextBox5.Text);
}
```

```
else if (ws == "GetDoctorLocation")
{
   ds = ss.GetDoctorLocation(TextBox5.Text);
}
else if (ws == "GetDoctorName")
{
   ds = ss.GetDoctorName(TextBox5.Text);
}
else if (ws == "GetPatientDetails")
{
   ds = ss.GetPatientDetails(TextBox5.Text);
}
else if (ws == "GetPatientName")
{
   ds = ss.GetPatientName(TextBox5.Text);
}
else if (ws == "GetPatientTestDetails")
{
   ds = ss.GetPatientTestDetails(TextBox5.Text);
}
else if (ws == "GetStaffDetails")
{
   ds = ss.GetStaffDetails(TextBox5.Text);
}
else if (ws == "GetStaffID")
{
   ds = ss.GetStaffID(TextBox5.Text);
}
else if (ws == "GetStaffLocation")
{
   ds = ss.GetStaffLocation(TextBox5.Text);
}
else if (ws == "GetStaffName")
{
   ds = ss.GetStaffName(TextBox5.Text);
}
if (ws != "Getlevel")
{
   GridView1.DataSource = ds;
   GridView1.DataBind();
}
```

```
}

protected void Button3_Click(object sender, EventArgs e)
{
  ss ser = new ss();
  int c = ListBox2.Items.Count;
  int j=c-1;
  string[] wsc=new string[5];
  string[] s = new string[5];
  Label8.Visible = true;
  for (int i =0;i<c;i++,j--)
  {
    wsc[j] = ListBox2.Items[i].ToString();
  }
  ds = ser.invoke(wsc[0],TextBox5.Text);
  DataRow dr = ds.Tables[0].Rows[0];
  string s1 = string.Empty;
  string s2 = string.Empty;
  string s3 = string.Empty;
  if (ds.Tables[0].Columns.Count ==2)
  {
    s1 = dr.ItemArray.GetValue(0).ToString();
    s2 = dr.ItemArray.GetValue(1).ToString();
  }
  else
  {
    s1 = dr.ItemArray.GetValue(0).ToString();
    s2 = null;
  }
   for (int j1 = 1; j1 < c; j1++)
    {
      if (s1 != null && s2 != null)
      {
        s1 = ser.invoke(wsc[j1], s1, s2);
        s2 = null;
      }
      else
      {
        ds = ser.invoke(wsc[j1], s1);
        dr = ds.Tables[0].Rows[0];
        if (ds.Tables[0].Columns.Count ==2)
```

```
                {
                  s1 = dr.ItemArray.GetValue(0).ToString();
                  s2 = dr.ItemArray.GetValue(1).ToString();
                  Label10.Visible = true;
                  Label9.Visible = true;
                  Label9.Text = s1;
                  Label10.Text = s2;
                }
                else
                {
                  s1= dr.ItemArray.GetValue(0).ToString();
                  Label9.Visible = true;
                  Label10.Visible = false;
                  Label9.Text = s1;

                }
              }
            }
    }

   protected void Button4_Click(object sender, EventArgs e)
    {
      ListBox2.Visible = true;
      Label5.Visible = true;
      TextBox5.Visible = true;
      Button3.Visible = true;
      Label13.Visible = true;

      Stopwatch sw = new Stopwatch();

      ss ser = new ss();
      string[] nm = new string[10];
      string[] nam = new string[10];
      DataSet ds = new DataSet();
      ss[] para=new ss[50];

      string[][] s1 = new string[2][];
      s1[0] = new string[] { "Ambulance" };
      s1[1] = new string[] { null };
      para[1].name = "aid";
      para[1].pa = s1;
```

```
string[][] s2 = new string[2][];
s2[0] = new string[] { null };
s2[1] = new string[] { "Ambulance" };
para[2].name = "aloc";
para[2].pa = s2;

string[][] s3 = new string[2][];
s3[0] = new string[] { null };
s3[1] = new string[] { "Ambulance" };
para[3].name = "drid";
para[3].pa = s3;

string[][] s4 = new string[2][];
s4[0] = new string[] { null };
s4[1] = new string[] { "Ambulance" };
para[4].name = "drname";
para[4].pa = s4;

string[][] s5 = new string[2][];
s5[0] = new string[] { "Discharge" };
s5[1] = new string[] { "Getlevel" };
para[5].name = "level";
para[5].pa = s5;

string[][] s6 = new string[2][];
s6[0] = new string[] { null  };
s6[1] = new string[] { "Discharge" };
para[6].name = "DischargeOK";
para[6].pa = s6;

string[][] s7 = new string[2][];
s7[0] = new string[] { "GetDepartment" };
s7[1] = new string[] { "GetDoctorDepartment" };
para[7].name = "dept";
para[7].pa = s7;

string[][] s8 = new string[2][];
s8[0] = new string[] { null };
s8[1] = new string[] { "GetDepartment" };
para[8].name = "deptloc";
```

```
    para[8].pa = s8;

    string[][] s9 = new string[2][];
    s9[0] = new string[] { "GetDiseaseDetails", "GetDoctorID","GetHBandBP",
"GetPatientDetails","GetPatientLocation","GetPatientName","GetPatientTestDetails","GetStaffI
D"};
    s9[1] = new string[] { "GetPatientID" };
    para[9].name = "pid";
    para[9].pa = s9;

    string[][] s10 = new string[2][];
    s10[0] = new string[] { null };
    s10[1] = new string[] { "GetDiseaseDetails" };
    para[10].name = "disease";
    para[10].pa = s10;

    string[][] s11 = new string[2][];
    s11[0] = new string[] { null };
    s11[1] = new string[] {"GetDoctorName", "GetDoctorDetails" };
    para[11].name = "dname";
    para[11].pa = s11;

    string[][] s12 = new string[2][];
    s12[0] = new string[] {
"GetDoctorDepartment","GetDoctorDetails","GetDoctorLocation","GetDoctorName" };
    s12[1] = new string[] { "GetDoctorID","GetDiseaseDetails","GetPatientDetails", };
    para[12].name = "did";
    para[12].pa = s12;

    string[][] s13 = new string[2][];
    s13[0] = new string[] { null };
    s13[1] = new string[] { "GetDoctorLocation" };
    para[13].name = "dloc";
    para[13].pa = s13;


    string[][] s14 = new string[2][];
    s14[0] = new string[] { "Getlevel" };
    s14[1] = new string[] {"GetHBandBP", "GetPatientTestDetails" };
    para[14].name = "hb";
    para[14].pa = s14;
```

```
string[][] s15 = new string[2][];
s15[0] = new string[] { "Getlevel" };
s15[1] = new string[] { "GetHBandBP", "GetPatientTestDetails" };
para[15].name = "bp";
para[15].pa = s15;

string[][] s16 = new string[2][];
s16[0] = new string[] { "GetStaffDetails","GetStaffLocation","GetStaffName" };
s16[1] = new string[] { "GetStaffID" ,"GetPatientDetails"};
para[16].name = "sid";
para[16].pa = s16;


string[][] s17 = new string[2][];
s17[0] = new string[] { "GetPatientID" };
s17[1] = new string[] { null };
para[17].name = "roomNo";
para[17].pa = s17;

string[][] s18 = new string[2][];
s18[0] = new string[] { null };
s18[1] = new string[] { "GetPatientLocation" };
para[18].name = "ploc";
para[18].pa = s18;

string[][] s19 = new string[2][];
s19[0] = new string[] { null };
s19[1] = new string[] { "GetPatientName" };
para[19].name = "pname";
para[19].pa = s19;

string[][] s20 = new string[2][];
s20[0] = new string[] { null };
s20[1] = new string[] { "GetPatientTestDetails" };
para[20].name = "temp";
para[20].pa = s20;

string[][] s21 = new string[2][];
s21[0] = new string[] { null };
s21[1] = new string[] { "GetStaffDetails","GetStaffName" };
```

```
para[21].name = "sname";
para[21].pa = s21;

string[][] s22 = new string[2][];
s22[0] = new string[] { null };
s22[1] = new string[] { "GetStaffLocation", "GetStaffDetails"};
para[22].name = "sloc";
para[22].pa = s22;

string[][] s23 = new string[2][];
s23[0] = new string[] { null };
s23[1] = new string[] { "GetStaffDetails"};
para[23].name = "status";
para[23].pa = s23;

string[][] s24 = new string[2][];
s24[0] = new string[] { null };
s24[1] = new string[] { "Machine" };
para[24].name = "mid";
para[24].pa = s24;

string[][] s25 = new string[2][];
s25[0] = new string[] { null };
s25[1] = new string[] { "Machine" };
para[25].name = "mstatus";
para[25].pa = s25;

string[][] s26 = new string[2][];
s26[0] = new string[] { "GetMedicineAvailability", "GetAvailabilityBranch" };
s26[1] = new string[] { "GetBGroup" };
para[26].name = "BGroup";
para[26].pa = s26;

string[][] s27 = new string[2][];
s27[0] = new string[] { null };
s27[1] = new string[] { "GetMedicineAvailability", "GetAvailabilityBranch" };
para[27].name = "avail";
para[27].pa = s27;


string[][] s28 = new string[2][];
```

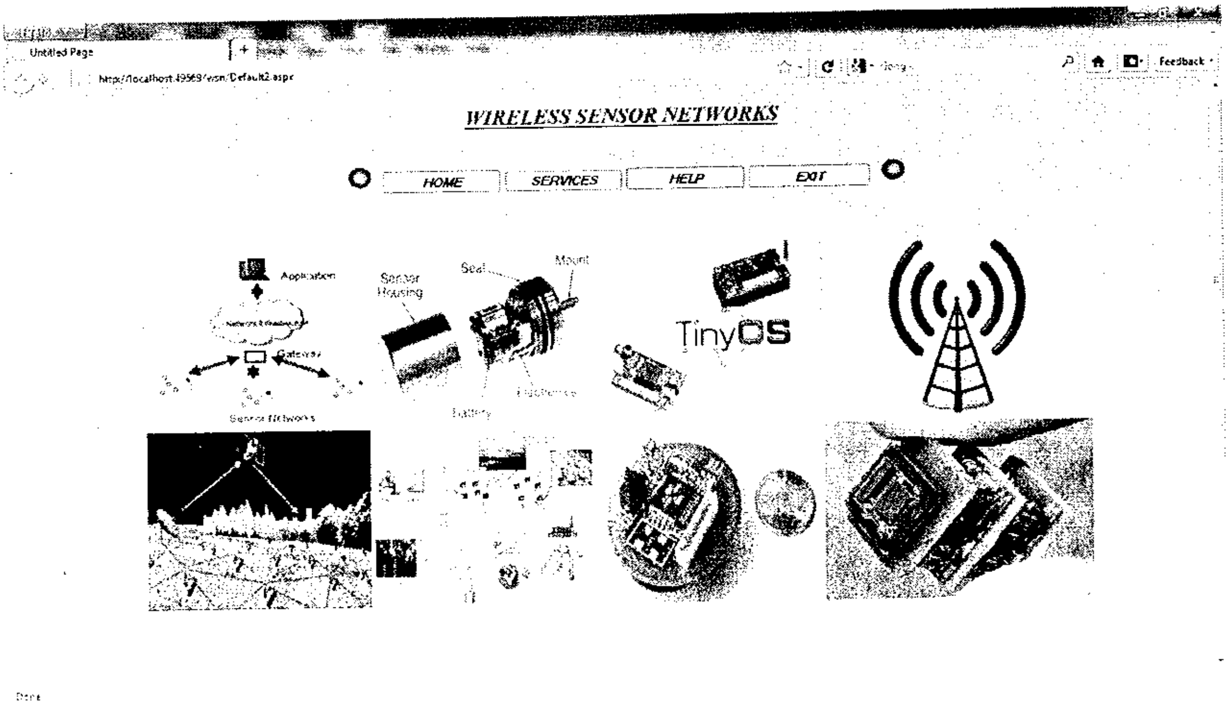
```
    s28[0] = new string[] { "GetAvailabilityBranch" };
    s28[1] = new string[] { null };
    para[28].name = "Branch";
    para[28].pa = s28;

    string[][] s29 = new string[2][];
    s29[0] = new string[] { "GetDonotor", "GetBGroup", "GetTempGlatrate",
"GetWbcRbcPlat", "GetHG" };
    s29[1] = new string[] { "GetBNo" };
    para[29].name = "BNo";
    para[29].pa = s29;

    string[][] s30 = new string[2][];
    s30[0] = new string[] { null };
    s30[1] = new string[] { "GetDonotor" };
    para[30].name = "Donotor";
    para[30].pa = s30;

    string[][] s31 = new string[2][];
    s31[0] = new string[] { "GetUsable" };
    s31[1] = new string[] { "GetTempGlatrate" };
    para[31].name = "Temp";
    para[31].pa = s31;

    string[][] s32 = new string[2][];
    s32[0] = new string[] { "GetUsable" };
    s32[1] = new string[] { "GetTempGlatrate" };
    para[32].name = "GlatRate";
    para[32].pa = s32;

    string[][] s33 = new string[2][];
    s33[0] = new string[] { null };
    s33[1] = new string[] { "GetUsable" };
    para[33].name = "Usable";
    para[33].pa = s33;

    string[][] s34 = new string[2][];
    s34[0] = new string[] { null };
    s34[1] = new string[] { "GetHG" };
    para[34].name = "HG";
    para[34].pa = s34;
```

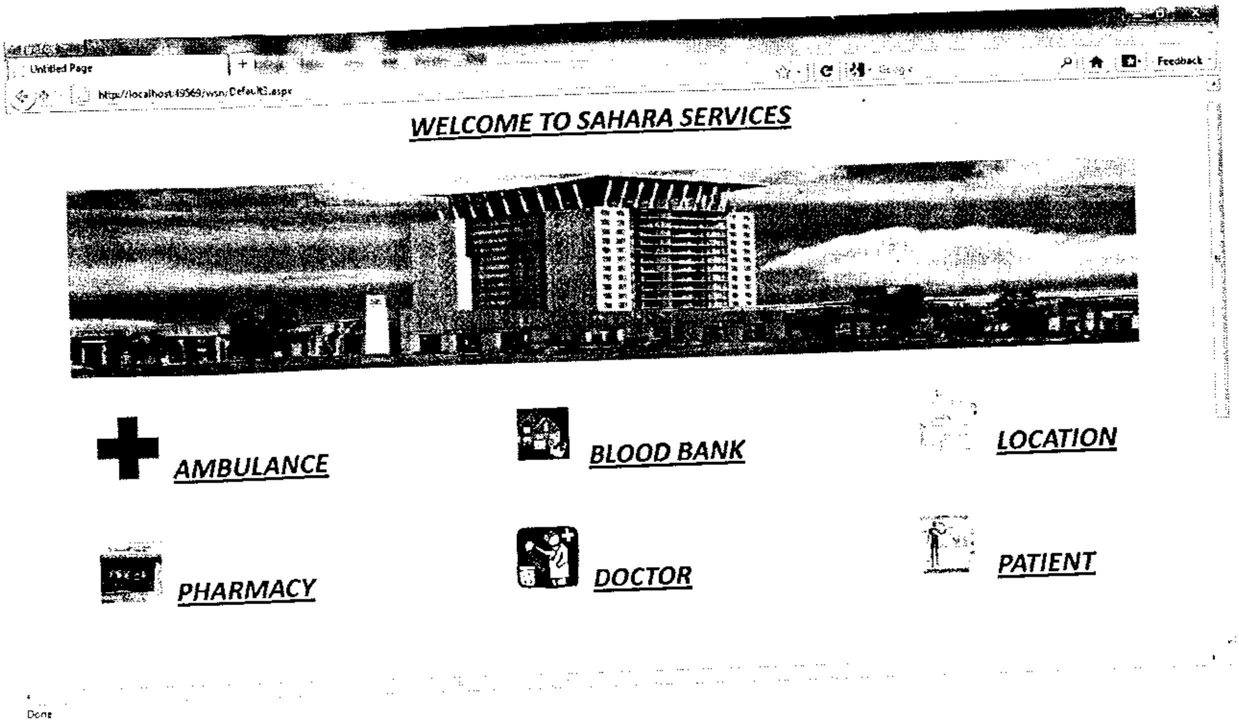
```
      string[][] s35 = new string[2][];
      s35[0] = new string[] { null };
      s35[1] = new string[] { "GetWbcRbcPlat" };
      para[35].name = "WBC";
      para[35].pa = s35;


      string[][] s36 = new string[2][];
      s36[0] = new string[] { null };
      s36[1] = new string[] { "GetWbcRbcPlat" };
      para[36].name = "RBC";
      para[36].pa = s36;

      string[][] s37 = new string[2][];
      s37[0] = new string[] { null };
      s37[1] = new string[] { "GetWbcRbcPlat" };
      para[37].name = "Platlets";
      para[37].pa = s37;

      string[][] s38 = new string[2][];
      s38[0] = new string[] { "GetMedicineAvail" };
      s38[1] = new string[] { "GetMedicineName" };
      para[38].name = "MedicineName";
      para[38].pa = s38;

      string[][] s39 = new string[2][];
      s39[0] = new string[] { "GetMedicineName", "GetShelfNo", "GetDomain", "GetPrice",
"GetStockLevel" };
      s39[1] = new string[] { null };
      para[39].name = "MedicineID";
      para[39].pa = s39;

      string[][] s40 = new string[2][];
      s40[0] = new string[] { null };
      s40[1] = new string[] { "GetMedicineAvail" };
      para[40].name = "MedicineAvail";
      para[40].pa = s40;

      string[][] s41 = new string[2][];
      s41[0] = new string[] { null };
```

```
s41[1] = new string[] { "GetShelfNo" };
para[41].name = "ShelfNo";
para[41].pa = s41;

string[][] s42 = new string[2][];
s42[0] = new string[] { null };
s42[1] = new string[] { "GetDomain" };
para[42].name = "Domain";
para[42].pa = s42;

string[][] s43 = new string[2][];
s43[0] = new string[] { "GetLocSpecial" };
s43[1] = new string[] { null };
para[43].name = "MedicineShopID";
para[43].pa = s43;

string[][] s44 = new string[2][];
s44[0] = new string[] { null };
s44[1] = new string[] { "GetLocSpecial" };
para[44].name = "ShopLocation";
para[44].pa = s44;

string[][] s45 = new string[2][];
s45[0] = new string[] { null };
s45[1] = new string[] { "GetLocSpecial" };
para[45].name = "Specialization";
para[45].pa = s45;

string[][] s46 = new string[2][];
s46[0] = new string[] { null };
s46[1] = new string[] { "GetPrice" };
para[46].name = "Price";
para[46].pa = s46;

string[][] s47 = new string[2][];
s47[0] = new string[] { null };
s47[1] = new string[] { "GetStockLevel" };
para[47].name = "StockLevel";
para[47].pa = s47;

string[][] s48 = new string[2][];
```

```
        s48[0] = new string[] { null };
        s48[1] = new string[] { "GetBalance" };
        para[48].name = "Balance";
        para[48].pa = s48;

        string[][] s49 = new string[2][];
        s49[0] = new string[] { null };
        s49[1] = new string[] { "GetGlucloseNo" };
        para[49].name = "GlucloseNo";
        para[49].pa = s49;

        sw.Start();
        nm = ser.best(para, service, xy, xy2);
        sw.Stop();
        Response.Write(sw.Elapsed.TotalMilliseconds);

        for (int i = 0; i < nm.Length; i++)
        {
            if (nm[i] != null)
            {

                ListBox2.Items.Add(nm[i]);
            }
        }
    }
    protected void TextBox4_TextChanged(object sender, EventArgs e)
    {
    }
}
```

7.2 SNAPSHOTS





WELCOME TO SAHARA SERVICES

AMBULANCE

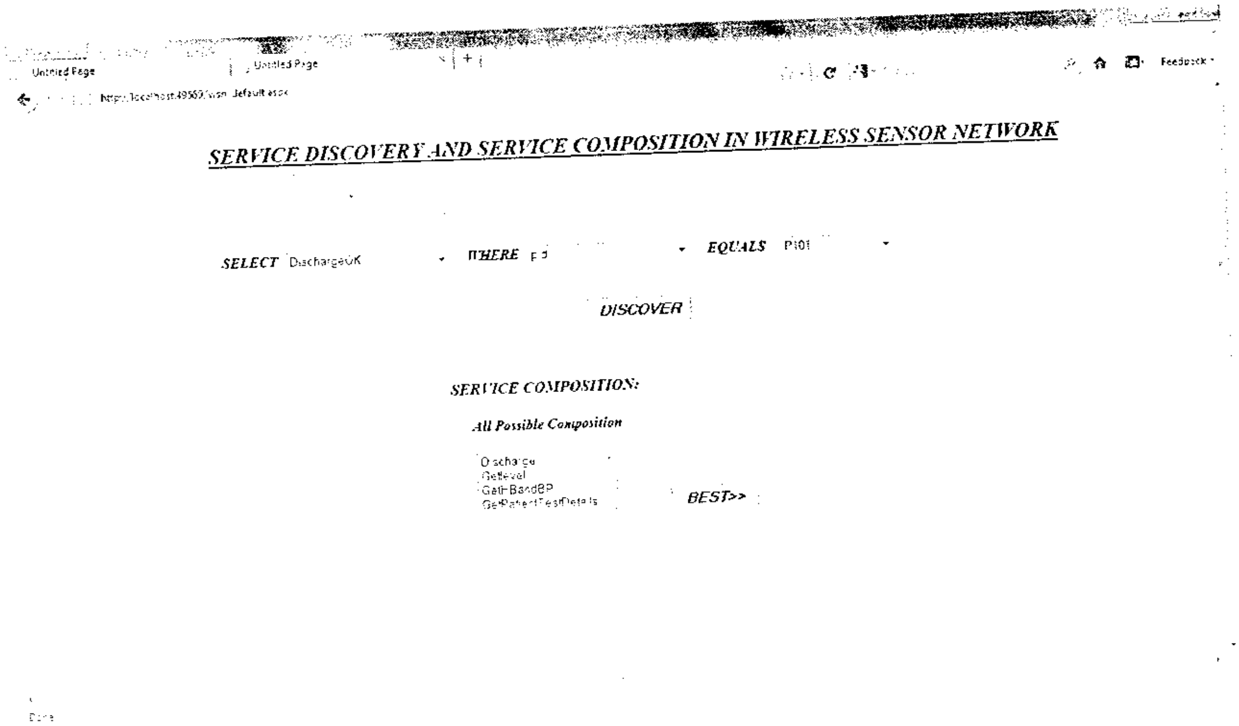
BLOOD BANK

LOCATION

PHARMACY

DOCTOR

PATIENT



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT DischargeOK WHERE Pid EQUALS P101

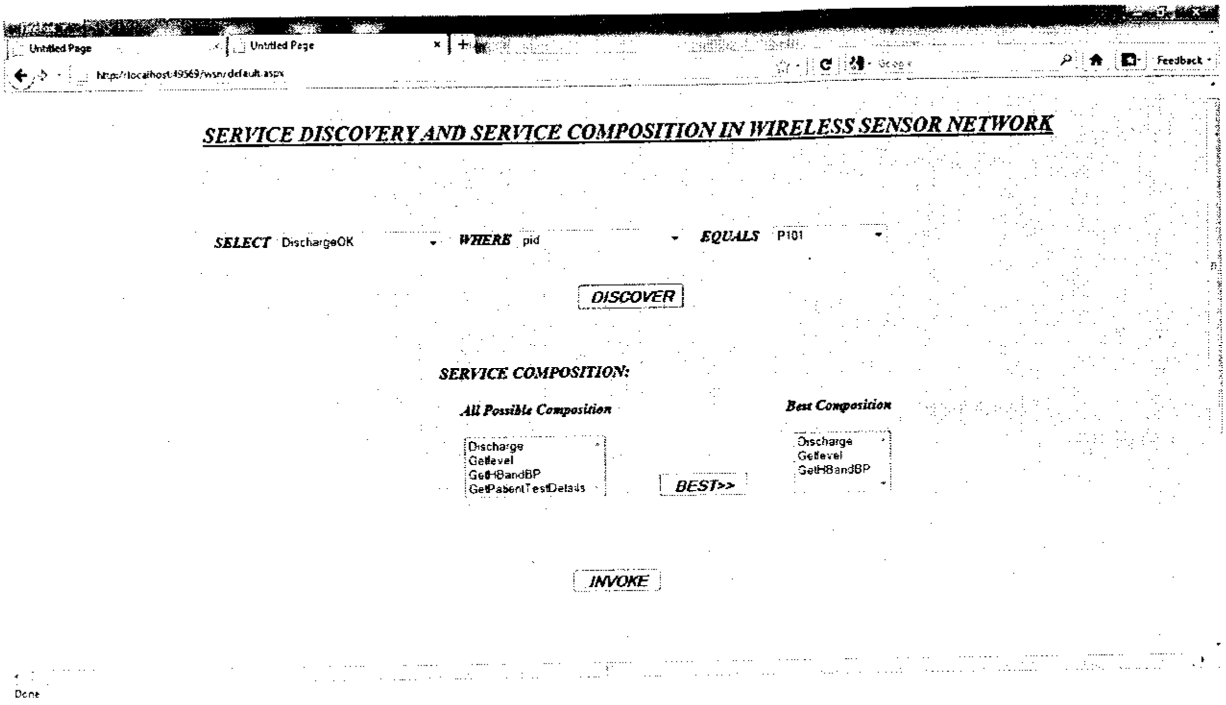
DISCOVER

SERVICE COMPOSITION:

All Possible Composition

Discharge
Getlevel
GetHBandBP
GetPatientTestDetails

BEST>>



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT DischargeOK WHERE pid EQUALS P101

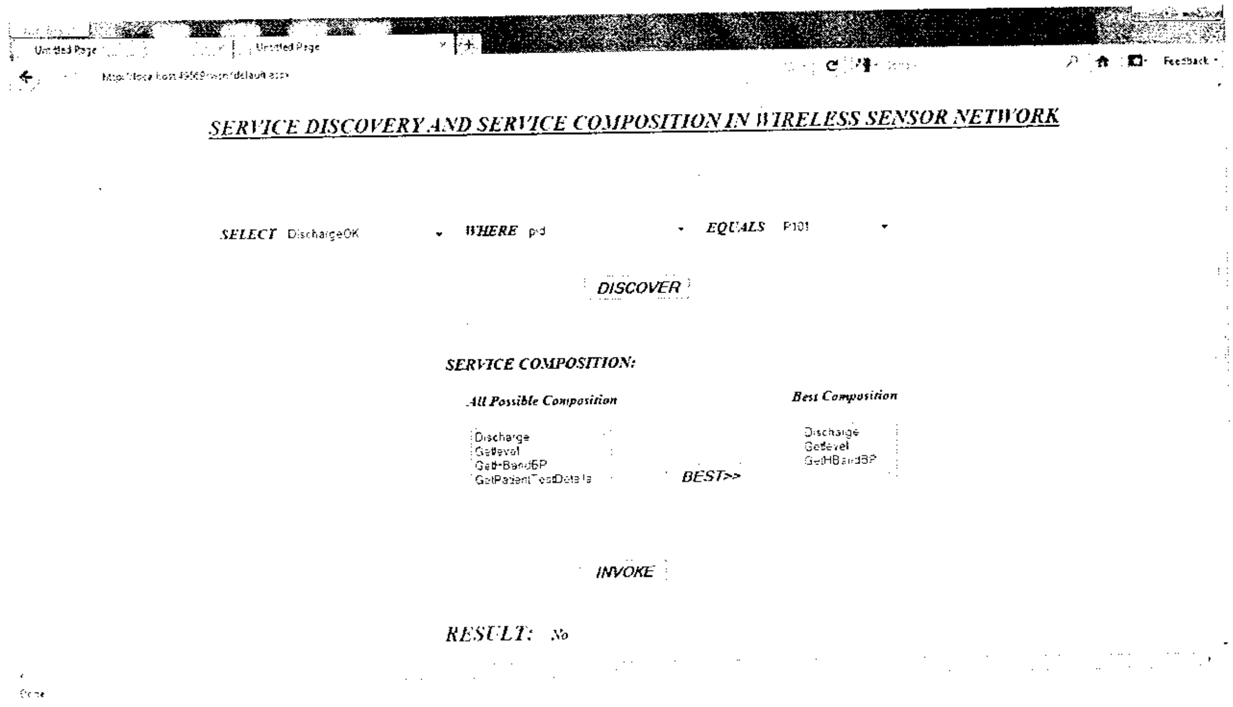
SERVICE COMPOSITION:

All Possible Composition

Discharge
Getlevel
GetHBandBP
GetPatientTestDetails

Best Composition

Discharge
Getlevel
GetHBandBP



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT DischargeOK WHERE pid EQUALS P101

SERVICE COMPOSITION:

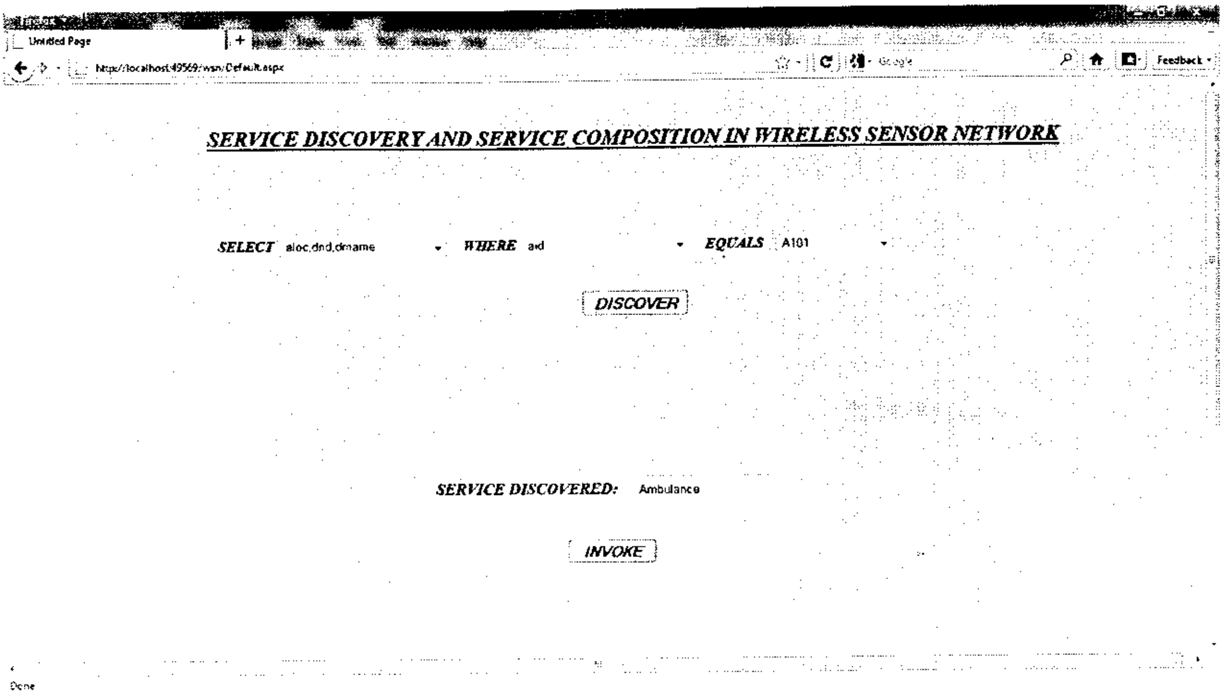
All Possible Composition

Discharge
Getlevel
GetHBandBP
GetPatientTestDetails

Best Composition

Discharge
Getlevel
GetHBandBP

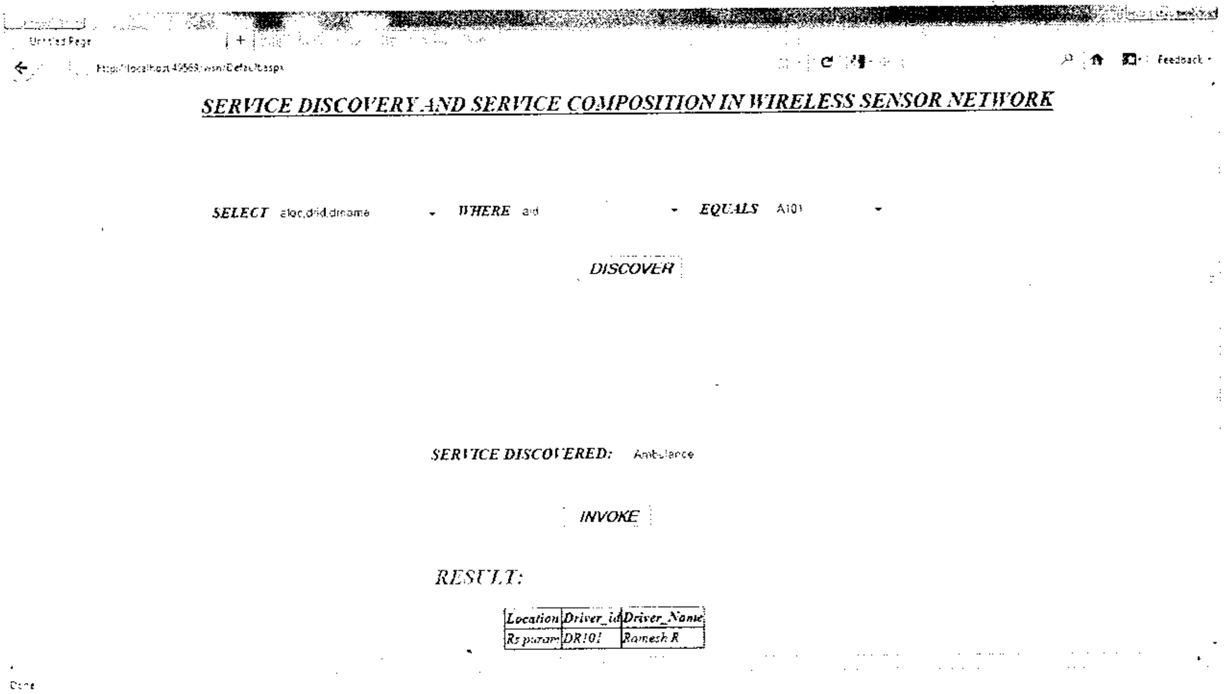
RESULT: No



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT aloc,drid,drname WHERE aid EQUALS A101

SERVICE DISCOVERED: Ambulance



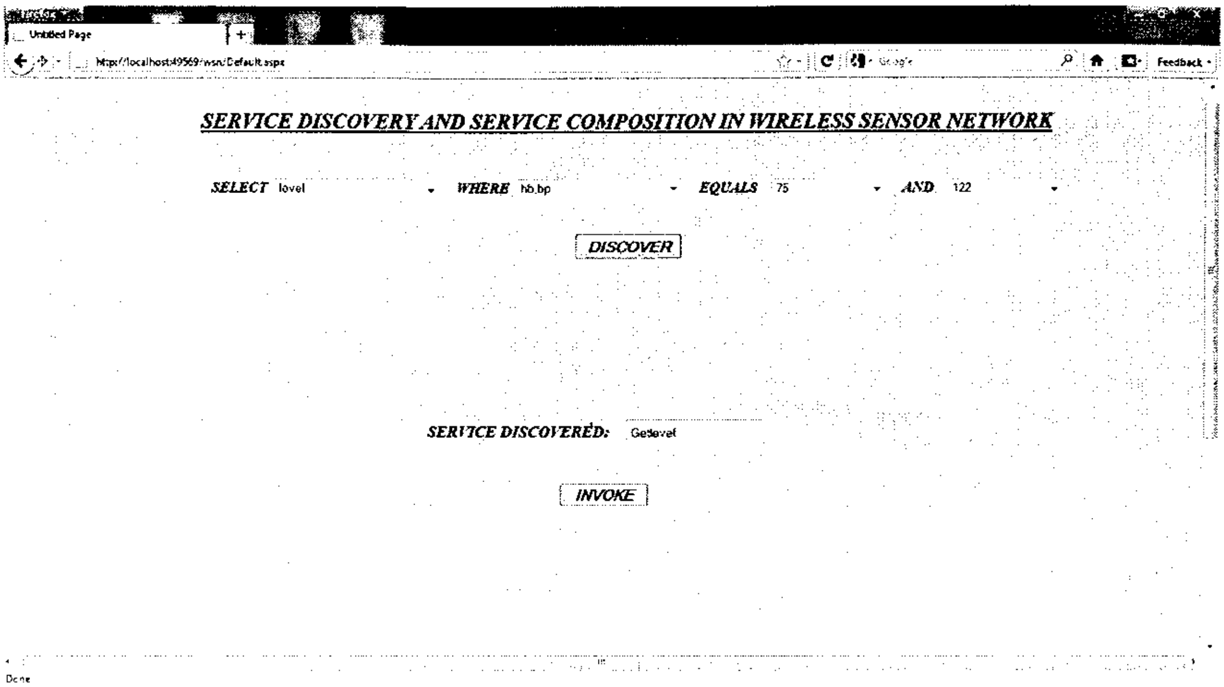
SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT aloc,drid,drname WHERE aid EQUALS A101

SERVICE DISCOVERED: Ambulance

RESULT:

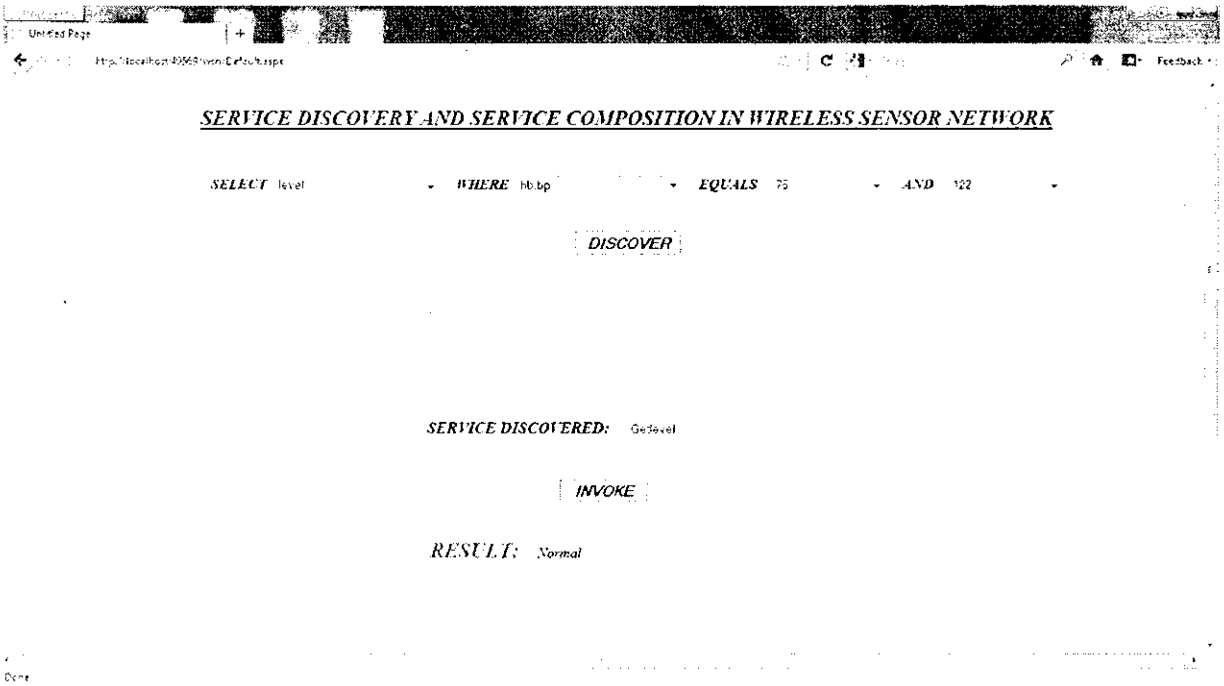
Location	Driver_id	Driver_Name
Rs puram	DR101	Ramesh R



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT level WHERE hb.bp EQUALS 75 AND 122

SERVICE DISCOVERED: Getlevel

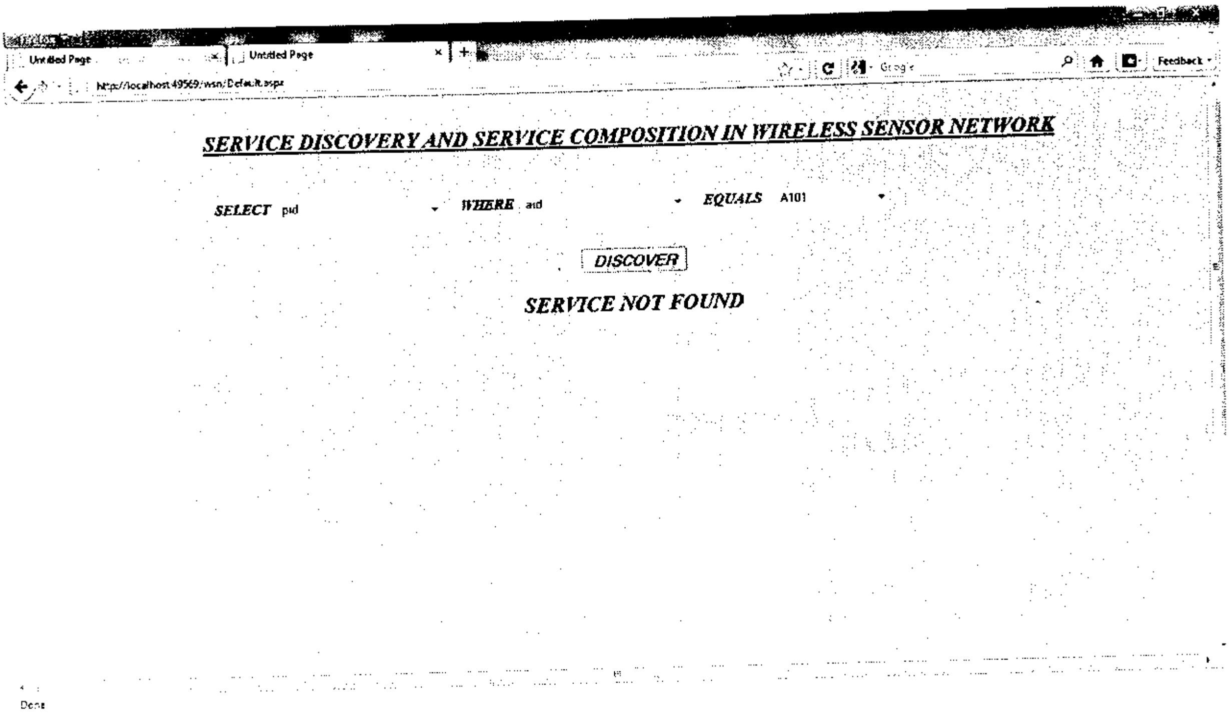


SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT level WHERE hb.bp EQUALS 75 AND 122

SERVICE DISCOVERED: Getlevel

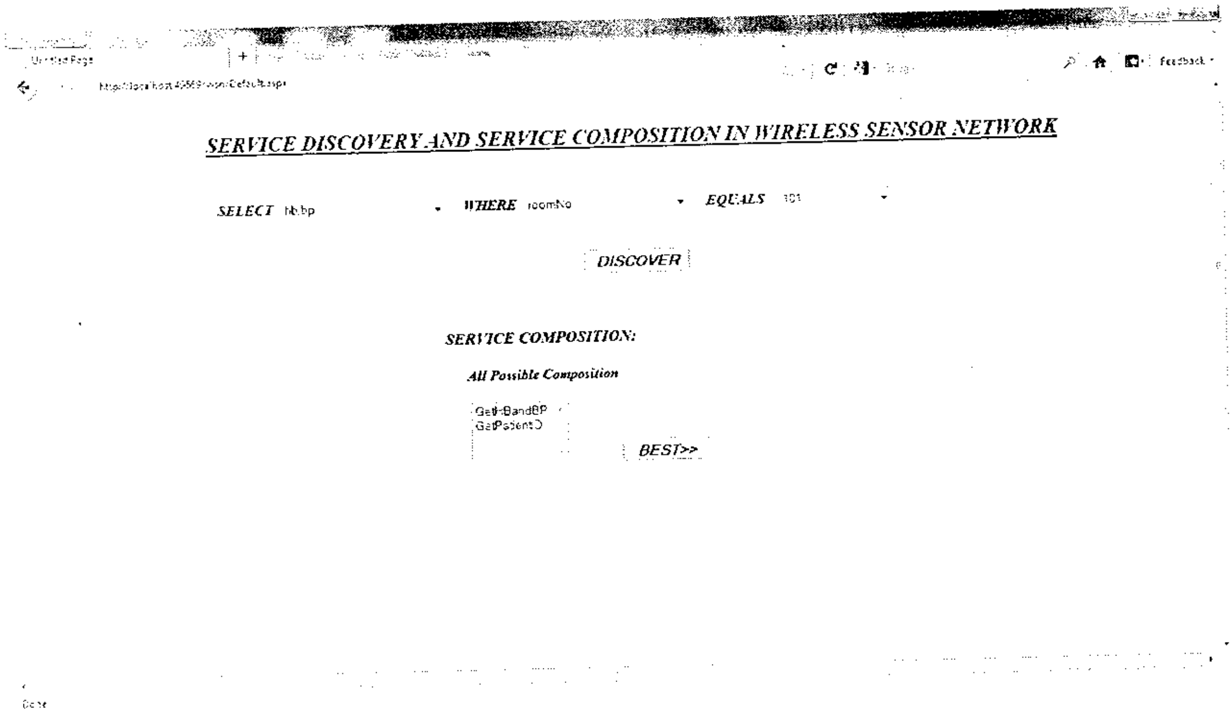
RESULT: Normal



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT pid WHERE aid EQUALS A101

SERVICE NOT FOUND



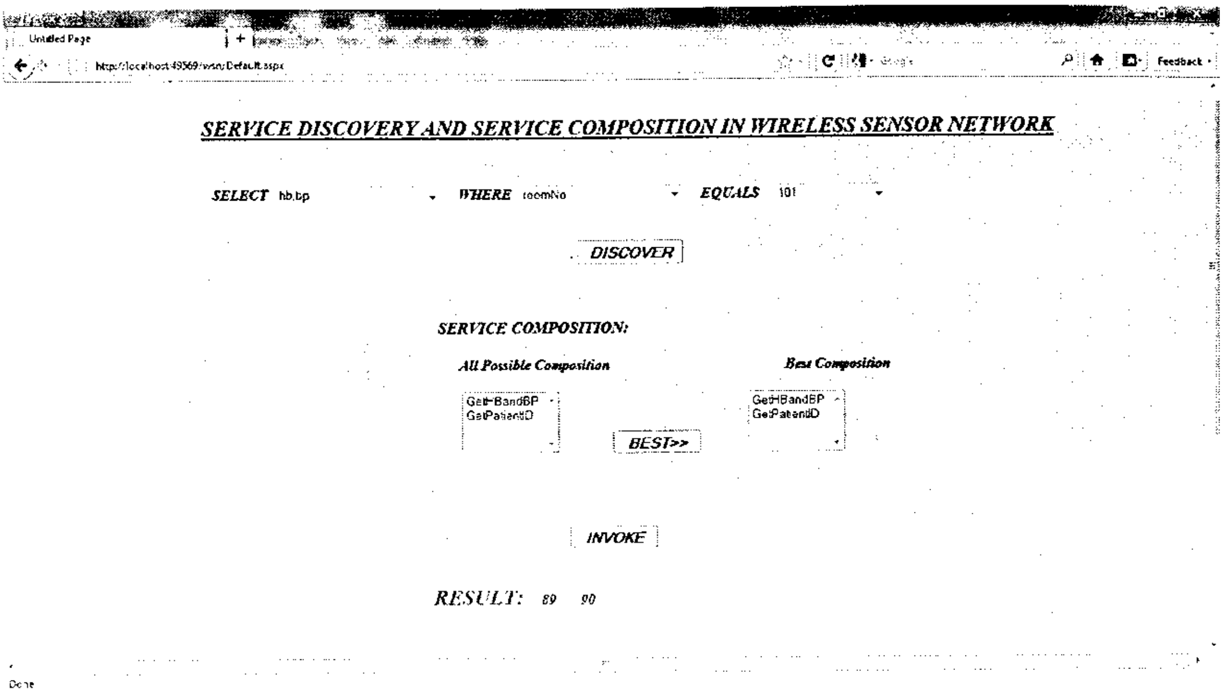
SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT hb.bp WHERE roomNo EQUALS 101

SERVICE COMPOSITION:

All Possible Composition

GetHBandBP
GetPatientID



SERVICE DISCOVERY AND SERVICE COMPOSITION IN WIRELESS SENSOR NETWORK

SELECT hb,bp WHERE roomNo EQUALS 101

DISCOVER

SERVICE COMPOSITION:

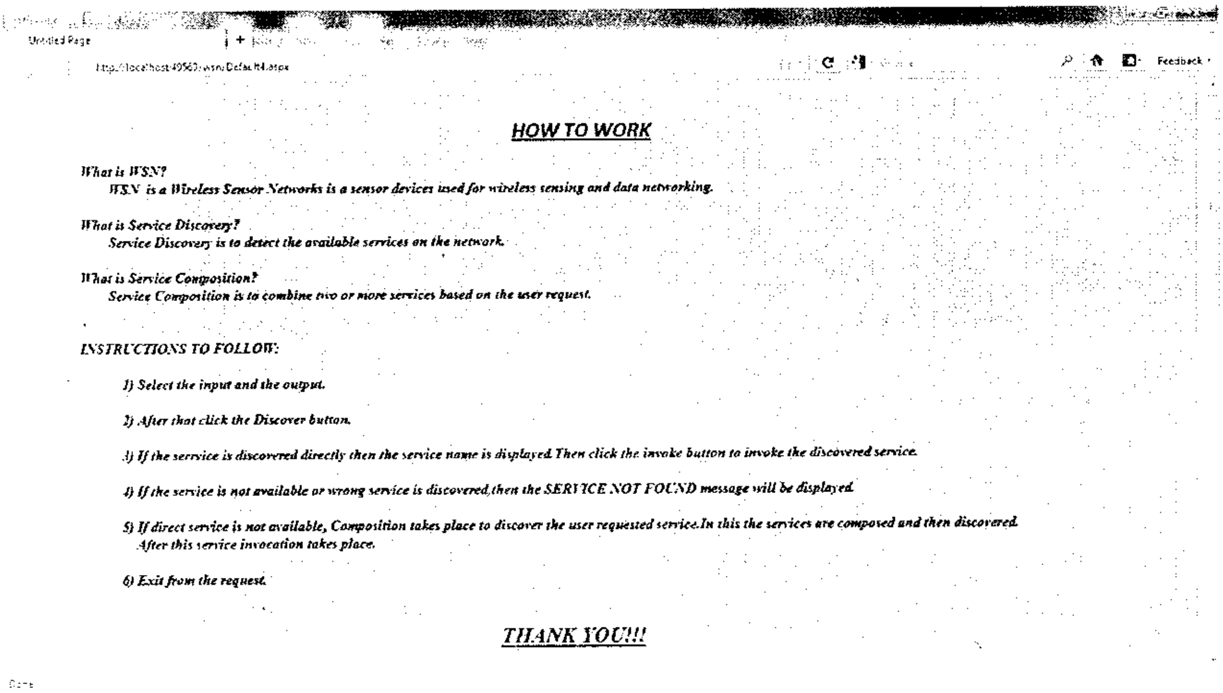
All Possible Composition: GetHBandBP, GetPatientID

Best Composition: GetHBandBP, GetPatientID

BEST>>

INVOKE

RESULT: 89 90



HOW TO WORK

What is WSN?
WSN is a Wireless Sensor Networks is a sensor devices used for wireless sensing and data networking.

What is Service Discovery?
Service Discovery is to detect the available services on the network.

What is Service Composition?
Service Composition is to combine two or more services based on the user request.

INSTRUCTIONS TO FOLLOW:

1) Select the input and the output.

2) After that click the Discover button.

3) If the service is discovered directly then the service name is displayed.Then click the invoke button to invoke the discovered service.

4) If the service is not available or wrong service is discovered,then the SERVICE NOT FOUND message will be displayed.

5) If direct service is not available, Composition takes place to discover the user requested service.In this the services are composed and then discovered. After this service invocation takes place.

6) Exit from the request.

THANK YOU!!!

8. REFERENCES

[1] "An Efficient Approach to Web Services Discovery and Composition when Large Scale Services are Available", Zhumin Chen, Jun Ma, Ling Song, Li Lian, School of Computer Science & Technology, Shandong University, Jinan, 250061, China.

[2] " An Efficient Protocol for Service Discovery in Wireless Sensor Networks", Mandeep Kaur Shilpa Bhatt Loren Schwiebert Golden G. Richard III .

[3] "A Flexible Service Discovery Protocol for Dynamic and Heterogeneous Wireless Sensor
Networks" ,Aleksandar Kovacevic, Junaid Ansari, and Petri M¨ah¨onen .

[4] " Localized Distance-Sensitive Service Discovery in Wireless Sensor Networks" ,
Xu Li SCS, Nicola SantoroSCS.

[5] "Model Driven Service Composition"Bart Orri¨ens, Jian Yang, and Mike. P. Papazoglou.

[6] "Efficient Web Service Discovery and Composition using Constraint Logic Programming"
Srividya Kona, Ajay Bansal, Gopal Gupta1and Thomas D. Hite.

[7] "Semantics-based EfficientWeb Service Discovery and Composition"
Srividya Kona, Ajay Bansal, and Gopal Gupt and Thomas D. Hite.

[8] "Differences and Commonalities of Service-Oriented Device Architectures,Wireless Sensor Networks and Networks-On-Chip"
Guido Moritz1, Claas Cornelius1, Frank Golatowski2, Dirk Timmermann1, Regina Stoll3.

[9] "LiteOS based Extended Service Oriented Architecture for Wireless Sensor Networks"
V.Vanitha, Dr.V.Palanisamy, N.Johnson and G.Aravindhbabu.

[10] "A Graph-Based Approach to Web Services Composition"
Seyyed Vahid Hashemian Farhad Mavaddat

[11] "Web Services Discovery and Constraints Composition"
Debmalya Biswas.

[12] "Towards a Service Oriented Architecture for Wireless Sensor Networks in Industrial
Applications"
Rudolf Sollacher, Christoph Niedermeier, Norbert Vicari,Maxim Osipov.

[13] "Web Service Composition - Current Solutions and Open Problems"
Biplav Srivastava, Jana Koehler

[14] " A Global QoS-Aware Service Composition in Wireless Sensor Networks"
Vanitha ,Dr.V.Palanisamy

[15] "A Semantic Framework for Analyzing Web Services Composition"
F. Latreche ,F. Belala

[16] XMethods. http://www.xmethods.net/ve2/index.po.

[17] Web Service List. http://www.webservicelist.com/.

[18] WebserviceX.NET. http://www.webservicex.net/WS/default.aspx.