DYNAMIC SERVICE COMPOSITION FOR SERVICE ORIENTED WIRELESS SENSOR NETWORK

PROJECT REPORT

Submitted by

V.NITHYA KUMARI

Reg. No: 0920108014

In partial fulfillment for the award of the degree

of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 049

Department of Computer Science and Engineering

PROJECT WORK

APRIL 2011

This is to certify that the project entitled

DYNAMIC SERVICE COMPOSITION FOR SERVICE-ORIENTED WIRELESS SENSOR NETWORKS

is the bonafide record of project work done by

V.NITHYA KUMARI

Register No: 0920108014

of M.E. (Computer Science and Engineering) during the year 2010-2011.

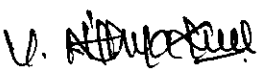
Project Guide

Head of the Department

Submitted for the Project Viva-Voce examination held on 25/04/2011

DECLARATION

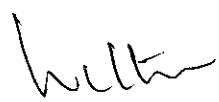
I affirm that the project work titled **DYNAMIC SERVICE COMPOSITION FOR SERVICE-ORIENTED WIRELESS SENSOR NETWORKS** being submitted in partial fulfillment for the award of **M.E** degree is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.



NITHYA KUMARI.V

Register No: 0920108014

I certify that the declaration made above by the candidate is true.



Mrs. **V.VANITHA M.E.,**

Associate Professor

Department of Computer Science and Engineering,
Kumaraguru College of Technology,
(An Autonomous Institution)
Coimbatore-641 049.

SRI SHAKTHI
INSTITUTE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE, New Delhi ■ Affiliated to Anna University of Technology

'Sri Shakthi Nagar', L&T By-Pass, Coimbatore - 641 062. Ph. : 0422 - 6450891 / 92 / 93 E-mail : youcanreachus@siet.ac.in Web : www.siet.ac.in

International Conference on Computer Communication and Informatics

It is certified that

Ms. Nithya Kumari

has presented a technical paper on

Network structure based routing Techniques in wireless Sensor Networks.

during the "International Conference on Computer Communication and Informatics" conducted during January 11th and 12th, 2011.

Organizing Secretary

Session Chair

Principal

Chairman

(Autonomous)

PERUNDURAI ERODE 638 052 TAMILNADU INDIA

School of Computer Technology and Applications

Department of Computer Technology - UG

KONGU
Assuring the Best

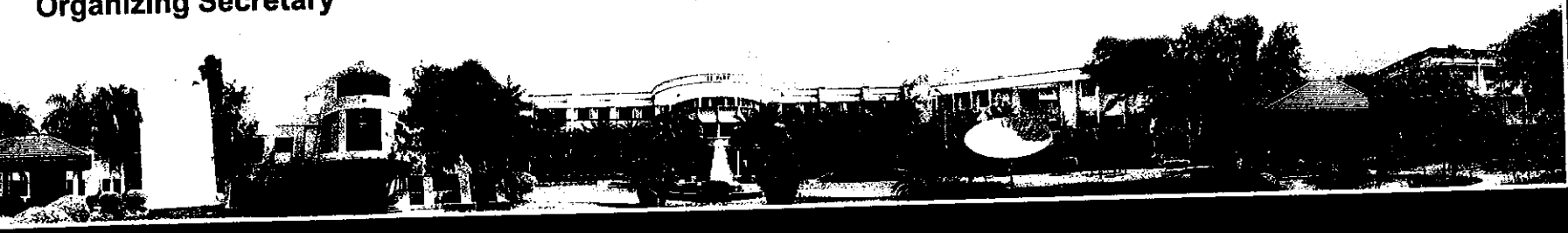
CERTIFICATE

This is to certify that ~~Mr.~~/Ms. V. NITHYAKUMARI of KUMARAGURU COLLEGE OF TECHNOLOGY has ~~Participated~~ / Presented paper entitled DYNAMIC SERVICE COMPOSITION FOR SERVICE ORIENTED WIRELESS SENSOR NETWORK in the National Conference held from 10th to 11th March 2011 on Emerging Computing and Communication Technologies (NECCT 2011).

Organizing Secretary

HOD

Principal



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 our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam B.Sc., F.I.E.,** for giving this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran Ph.D.,** *Principal* for providing the necessary facilities to complete my thesis.

I take this opportunity to thank **Mrs.P.Devaki M.E.,** *Head of the Department,* Computer Science and Engineering, for her precious suggestions.

I register my hearty appreciation to the Guide **Mrs.V.Vanitha M.E.,** *Associate Professor,* Department of Computer science and Engineering, my thesis advisor. I thank for her support, encouragement and ideas. I thank her for the countless hours she has spent with me, discussing everything from research to academic choices. Special thanks to her, for arranging the brain storming project review sessions. I thank all project committee members for their comments and advice during the reviews.

I would like to convey my honest thanks to all **Teaching** staff members and **Non Teaching** staffs of the department for their support. I would like to thank all my classmates who gave me a proper light moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this project work to my **parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't possible.

TABLE OF CONTENTS

ABSTRACT

The current sensor networks are assumed to be designed for specific applications, having strongly coupled data communication protocols. Service Oriented Architecture (SOA) for the design of sensor networks, in which sensor nodes are service providers and applications are clients of such services. Service-Oriented architectures allow us to modularize the business logic and to implement it in the form of services which are accessible in a network. Services are building blocks for service processes which represent the workflows of an enterprise. It is represented as sensor network applications and views them as a collection of component services. In this I used a service modeling approach that introduces a graph-based model for representing sensor network services that is intuitive and also amenable to analysis. Service composition is a fundamental method for offering advanced functionality by combining a set of primitive services provided by the system. Unlike in the case of web services for which there is an abundance of reliable resources, in sensor networks, the resources are constrained and communication among nodes is error-prone and unreliable. This project proposes two heuristic methods for its solution, the top-down and the bottom-up, and discusses their centralized and distributed implementations. Using simulations, I evaluate their performance.

ஆய்வுச்சுருக்கம்

இன்றைய கால கட்டத்தில் உணரித் தொழில்நுட்பமானது ஒரு குறிப்பிட்ட பயன்பாட்டிற்கு மட்டும் வலையமைக்கப்படுகிறது.மேலும் அவைகள் அதற்க்கென்றே உருவாக்கப்பட்ட வரைமுறைகளுக்குள் மட்டுமே செயல்படும்.இத்தகைய குறைபாட்டை நீக்குவதற்காக இந்த ஆய்வில் சேவை அடிப்படையிலான கட்டமைப்புடன் கூடிய உணரித்தொழில்நுல்பம் வலையமைக்கப்படுகிறது.இதில் உணரிகள் சேவை அளிப்பனாக கருதப்படுகிறது.

மேலும் அவைகளின் சேவைகள் தொகுப்புகளாக கருதப்படுகிறது.இந்த ஆய்வில் சேவை மாதிரியமைத்தல் வரைவி மூலமாக அறிமுகப்படுத்தப்பட்டு பகுப்பாய்வு செய்யப்படுகிறது.இதை அமுல்படுத்துவதற்க்கு சேவை கூட்டமைப்பு ஒரு அடிப்படை முறையாகும்.இணையதள சேவையில் இதை அமுல்படுத்தும்பொழுது அபரிமிதமான வள ஆதரங்கள் வீணாக்கப்படுகிறது.மேலும் இதில் இணைப்பு நம்பகமற்றதாகவும் பிழைஅனுமதிப்பானாகவும் செயல்படுகிறது.

இந்த ஆய்வில் இரண்டு வகையான நுட்ப விசாரனை ஊக்குவிக்கும் முறைகள் பயன்படுத்தப்பட்டு மேலே கூறப்பட்ட குறைகள் நீக்கப்படுகிறது.மேலும் இதன் செயல்திறன் வலையமைப்பு பாவனை மூலமாக மதிப்பீடு செய்யப்படுகிறது.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

WSN	Wireless Sensor Networks
SOA	Service-Oriented Architecture
QoS	Quality of Service
OS	Operating systems
NS	Network Simulator

LIST OF SYMBOLS

Si	**Service Input**
t	**Time**
m	**Metadata**
G_S	**Service Graph**
V	**Vertices**
E	**Edge**

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF WIRELESS SENSOR NETWORKS

1.1.1. Sensor network

A wireless sensor network is a collection of nodes organized in a network. Each node consists of one or more microcontrollers, CPUs or DSP chips, a memory and a RF transceiver, a power source such as batteries and accommodates various sensors and actuators. The nodes communicate wirelessly and often self-organize after being deployed in an ad hoc fashion.

1.1.2. Components of sensor networks

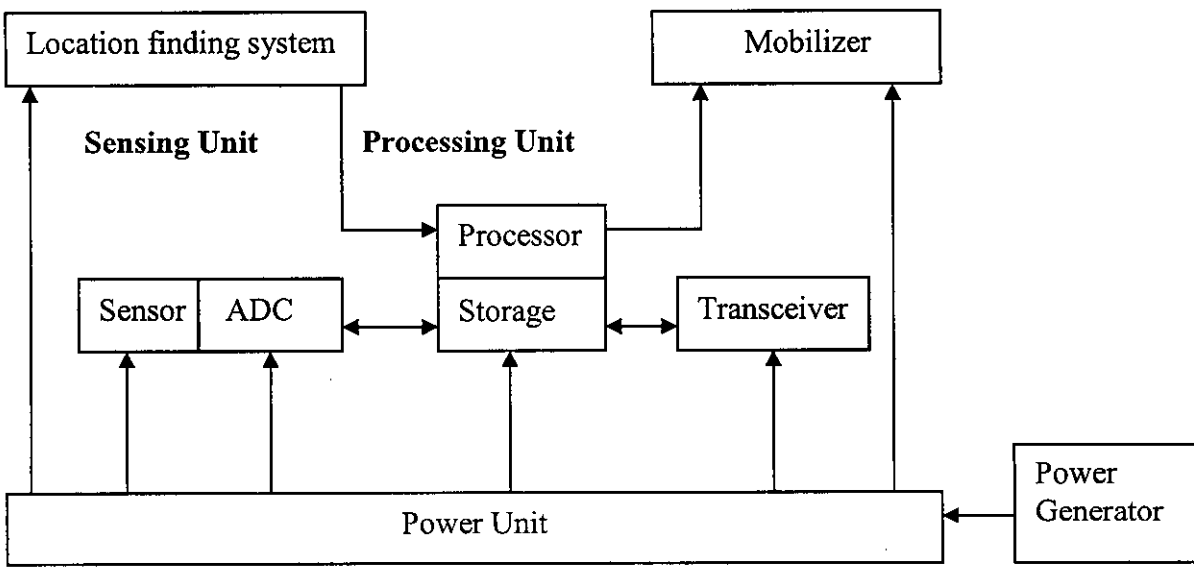


Fig 1.1.Components of Sensor Networks [1]

- **Sensing unit**

Sensing units are usually composed of two subunits: sensors and analog to digital converters (ADCs).The analog signals produced by the sensors are converted to digital signals by the ADC, and then fed into the processing unit.

- **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 Unit**

A transceiver unit connects the nodes to the networks.

- **Power Unit**

One of the most important components of a sensor node is the power unit. Power units may be supported by a power scanning unit such as solar cells.

1.2 APPLICATIONS OF SENSOR NETWORKS

Wireless Sensor Networks have a wide range of applications such as,

1. Military applications
    i. Monitoring friendly forces and equipment.
    ii. Battlefield surveillance.
    iii. Nuclear, biological and chemical attack detection.
2. Environmental Applications
    i. Forest fire detection.
    ii. Bio-complexity mapping of the environment.
    iii. Flood detection.
3. Health applications
    i. Tele-monitoring of human physiological data.
    ii. Tracking and monitoring doctors and patients inside a hospital.

iii. Drug administration in hospitals.

4. Home application

i. Home automation.

ii. Smart environment

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. The intrinsic properties of individual sensor nodes, pose additional challenges to the communication protocols in terms of energy consumption.

1.3 CHALLENGES OF WIRELESS SENSOR NETWORKS

A sensor network design is influenced by many factors, which include,

- **Energy efficiency/system lifetime**

As sensor nodes are battery-operated, protocols must be energy-efficient to maximize system life time. System life time can be measured such as the time until half of the nodes die or by application-directed metrics, such as when the network stops providing the application with the desired information about the phenomena.

- **Fault Tolerance**

Some sensor nodes may fail or be blocked due to lack of power, have physical damage or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. This is the reliability or fault tolerance issue. Fault tolerance is the ability to sustain sensor network functionalities without any interruption due to sensor node failures.

- **Scalability**

The number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or thousands. Depending on the application, the number may reach an extreme value of millions. The new schemes must be able to work with this number of nodes.

- **Production Costs**

Since the sensor networks consist of a large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the networks. If the cost of the network is more expensive than deploying traditional sensors, then the sensor network is not cost-justified. As a result, the cost of each sensor node has to be kept low.

- **Environment**

Sensor nodes are densely deployed either very close or directly inside the phenomenon to be observed. Therefore, they usually work unattended in remote geographic areas. They may be working in busy intersections, in the interior of large machinery, at the bottom of an ocean, inside a twister, on the surface of an ocean. They work under high pressure in the bottom of an ocean, in harsh environments such as debris or a battlefield, under extreme heat and cold such as in the nozzle of an aircraft engine or in arctic regions, and in an extremely noisy environment such as under intentional jamming.

- **Hardware Constraints**

A sensor node is made up of four basic components: a sensing unit, a processing unit, a transceiver unit and a power unit. Sensing units are usually composed of two subunits: sensors and analog to digital converters (ADCs). The analog signals produced by the sensors based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit. The processing unit, which is generally associated with a small storage unit, manages the

procedures that enable the sensor node collaborate with the other nodes to carry out the assigned sensing tasks. A transceiver unit connects the node to the network. One of the most important components of a sensor node is the power unit. Power units may be supported by a power scavenging unit such as solar cells. There are also other subunits, which are application dependent. Most of the sensor network routing techniques and sensing tasks require the knowledge of location with high accuracy.

- **Sensor Network Topology**

Sheer numbers of inaccessible and unattended sensor nodes, which are prone to frequent failures, make topology maintenance a challenging task. Hundreds to several thousands of nodes are deployed throughout the sensor field.

- **Pre-Deployment Phase**

Sensor nodes can be either thrown in mass or placed one by one in the sensor field. They can be deployed by dropping from a plane, delivering in an artillery shell, rocket or missile, throwing by a catapult , placing in factory, and placing one by one either by a human or a robot. Although the sheer number of sensors and their unattended deployment usually preclude placing them according to a carefully engineered deployment plan, the schemes for initial deployment must reduce the installation cost, eliminate the need for any pre-organization and preplanning, increase the flexibility of arrangement, and promote self-organization and fault tolerance.

- **Post-Deployment Phase**

After deployment, topology changes are due to change in sensor nodes position, reach ability (due to jamming, noise, moving obstacles, etc.), available energy, malfunctioning, and task details. Sensor nodes may be statically deployed. However, device failure is a regular or common event due to energy depletion or destruction. It is also possible to have sensor networks with highly mobile nodes. Besides, sensor nodes and the network experience varying task dynamics, and they may be a target for deliberate jamming. Therefore, sensor network topologies are prone to frequent changes after deployment.

- **Re-Deployment of Additional Nodes Phase**

Additional sensor nodes can be re-deployed at any time to replace the malfunctioning nodes or due to changes in task dynamics. Addition of new nodes poses a need to re-organize the network. Coping with frequent topology changes in an ad hoc network that has myriads of nodes and very stringent power consumption constraints requires special routing protocols.

- **Transmission Media**

In a multi-hop sensor network, communicating nodes are linked by a wireless medium. These links can be formed by radio, infrared or optical media. To enable global operation of these networks, the chosen transmission medium must be available worldwide. One option for radio links is the use of Industrial, Scientific and Medical (ISM) bands, which offer license free communication in most countries.

- **Power Consumption**

The wireless sensor node, being a microelectronic device, can only be equipped with a limited power source. In some application scenarios, replenishment of power resources might be impossible. Sensor node lifetime, therefore, shows a strong dependence on battery lifetime. The malfunctioning of few nodes can cause significant topological changes and might require rerouting of packets and re-organization of the network. Hence, power conservation and power management take on additional importance. It is for these reasons that researchers are currently focusing on the design of power aware protocols and algorithms for sensor networks. In sensor networks, power efficiency is an important performance metric, directly influencing the network lifetime. Application specific protocols can be designed by appropriately trading off other performance metrics such as delay and throughput with power efficiency. The main task of a sensor node in a sensor field is to detect events, perform quick local data processing, and then transmit the data. Power consumption can hence be divided into three domains: sensing, communication, and data processing.

1.4 INTRODUCTION TO SERVICE ORIENTED ARCHITECTURE

Service-oriented architecture presents an approach for building distributed systems that deliver application functionality as services to either end-user applications or other services. It is comprised of elements that can be categorized into functional and quality of service. It seems probable that eventually most software capabilities will be delivered and consumed as services. Likewise in sensor networks also the nodes capabilities to fulfill the requirements of the applications also assumed to be a service. So it need service based architecture to deploy the sensor networks effectively.

A service-oriented architecture (SOA) is the underlying structure supporting communications between services. SOA defines how two computing entities, such as programs, interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. Service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction. Service-oriented architecture allows us to manage the usage (data gathering, data dissemination, query routing, and query processing, and so on) of sensor networks in terms of, and in sets of, related services.

In recent years the need for decoupling has led to the emergence of generic specific, an alternative design model where an application - independent query system is developed on the network. In application specific deployments, the application consists of a distributed code installed on some or all of the nodes of the network. In simple applications, the some code is installed on all nodes. In more complex applications, different code modules are installed on different nodes. Generic specific are not intended to be used by a specific application. The author usually require that a generic code (i.e., the query processing system) to be installed on all nodes of the network. Current design models for sensor systems seem increasingly unable to cope with the requirements of the next generation of open, ubiquitous interoperable, multipurpose.

Architectures for future sensor systems will have to be able to serve different applications and adopt to different post deployment query patterns. To enable next-generation sensor systems, new customizable architectures are needed. Customizable networks are readily configurable to serve different types of applications with arbitrary query patterns. Customizable networks would provide developers the flexibility to combine the resources provided by nodes in one or more networks to meet the requirements of new applications and yet expect the some level of performance that would result from an application - specific deployment. A possible alternative to building customizable networks is to use generic as their backbone and develop additional software layers that customize the functionalities of generic networks to satisfy the requirements of the given application. This leads to further lower performance and memory availability.

A service-oriented architecture (SOA) is an architectural style encompassing a set of services for building complex systems of systems. It can be regarded as a model in which a system is decomposed into smaller, distinct units that are able to provide certain functionality. SOA offers flexibility in the design of WSN applications since it provides accepted standards for representing and packaging data, describing the functionality of services, and facilitating the search for available services which can be invoked to meet application requirements.

Service oriented architecture is a software architecture model, not a technology, and it is becoming popular in business and information technology communities. The reason that it is becoming popular is because it interoperates between heterogeneous systems, allows reuse of components and it makes it possible to have flexible and efficient business processes. SOA make it possible to share resources, data and functionality via digital networks. Because with this approach, it share, integrate and cooperate heterogeneous hardware and software components, distributed applications can be achieved with lower cost, better overall system utilization and performance. This makes it possible for individual users to obtain shared data, resources and functionalities which are not locally available.

The main goal is to partition business functionality in a way that it can be orchestrated in a loosely-coupled and re-usable fashion. It also presents an approach for building distributed systems that deliver application functionality as services to either end-user applications or other services. A **service** is generally implemented as a coarse-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model.

Service - oriented architecture is a novel approach to building customizable networks. In this, nodes reusing and actuation capabilities are exposed to applications in the form of a collection of programmatic abstractions called services. A service deployed on a node is a lightweight code that provides some functionality supported by the node. These services may be individually invoked or combined in complex ways to form a virtual network with far reaches sensing and actuation capabilities. Services are deployed directly on a top of the operating systems, and they are accessible directly by applications.

In service - oriented query model for networks, nodes have heterogeneous sensing and actuation capabilities. Each node exposes its capabilities as services. A service is a computational component that:

a) has a unique network-wide identifier,

b) may be invoked asynchronously,

c) may have one or more parameters,

d) produces one or more values as a result of invocation.

Sensor nodes may have different and varying capabilities, be manufactured and operated by different vendors, and be accessed by multiple clients exercising different functionalities.

1.4.1 Advantages of Service-oriented Architectures

(1) Platform and Programming Language Independence - Since services can be published and consumed across development and operating platforms, an application can leverage existing legacy components that reside on different types of servers and were built using different technologies. The main challenge is to integrate heterogeneous components into a larger application. This applies for wireless sensor networks if standardized protocols for communication between services exist.

(2) Focused Developer Roles - Since a service is a discrete implementation independent of other services, developers in charge of a service can focus completely on implementing and maintaining that services without having to worry about other services as long as the pre-defined contract is honored. However, the semantics and orchestrability of such a service must be made explicit and must be documented and managed accordingly. For WSN-services this information should be stored on an external server in order to save resources.

(3) Location Transparency - Services are often published to a directory where consumers can look them up. The advantage of this approach is that the service can change its location where it is executed at any time. Consumers of the service will be able to locate the service through the directory. The underlying traditional assumption that communication is basically for free does not hold for WSN. Communication costs energy.

(4) Code Reuse - Since SOA breaks down an application into small independent pieces of functionality, the services can be reused in multiple applications, thereby bringing down the cost of development. This demands that the principle of stateless business logic encapsulated into a small and precisely defined service is honored at all times. Distributing functionality can save memory provided energy consumption is taken into account carefully.

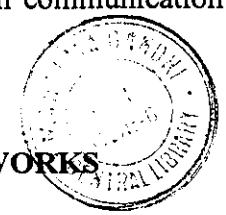
(5) Greater Testability - Small, independent services are easier to test and debug than monolithic applications. This leads to more reliable software. In WSN this benefit cannot be leveraged in general, since distributed functionality is severely restricted by the unreliable wireless communication.

(6) Parallel Development - Since the services are independent of each other and contracts between services are pre-defined, the services can be developed in parallel this shortens the software development life cycle considerably. This holds for services in WSN as well.

(7) Better scalability - In principle loosely coupled and distributed systems do scale better. There can be multiple instances of a service running on different servers. This increases scalability. However, the impact of communication, load balancing, and management may interfere with scalability. This is particularly important for WSN with constrained resources.

(8) Higher availability - The same argument as above applies to a possible high availability. Services in industrial WSN are usually location based. This limits the possibility for redundant designs and thus higher availability may not be given.

(9) Dynamic deployment - New services can be deployed on demand or existing ones can be updated. In WSN this is accompanied by high communication costs and should thus be the exception rather than the rule.



1.5 SERVICE COMPOSITION IN WIRELESS SENSOR NETWORKS

In a service oriented WSN, a typical application requires several different services like data aggregation, data processing, decoding etc.., which are provided by service providers that are also sensors. However, it might happen that there is no single service satisfying the requirement of the end user. So it is requires to form a composite service by combining one or more existing services.

Service composition is a key technology for building service oriented, loosely coupled integrated applications. Service Composition involves the development of customized services often by discovering, integrating, and executing existing services. This can be done in such a way that already existing services are combined into one or more new services that fit better to composite service.

For supporting the lifecycle of service compositions several aspects have to be addressed. Synthesis of service compositions deals with creation of service compositions which can happen both at design-time and run-time.

Synthesis of service compositions is done in three different ways

1. Automated Service Composition

   It is a family of approaches to service composition that aim at full or partial automation of the composition process, in order to enable handling of higher levels of complexity. Automated Service Composition can be achieved using either Workflow-based or planning techniques in order to create the Composition Schema.

2. Model-driven Service Composition

   Model-Driven Service Composition is a service composition that generates service orchestrations from a more general or abstract model.

3. Quality of Service-Aware Service Composition

   QoS-aware service composition is a form of service composition that is based on and attempts to improve the overall Quality of Service (QoS) of the composite service, such as execution time, reliability, availability or cost.

1.5.1 Dynamic service composition

The dynamic service composition has the potential to realize flexible and adaptable applications by properly selecting and combining components based on the user request and context. The dynamic service composition may also elicit a number of useful applications that are not envisioned at the design time. Therefore, the dynamic service composition is suitable for end-user applications in ubiquitous, mobile environments where available components are dynamic and expected users may vary.

Dynamic composition of complex services from primitive components brings flexibility and adaptability to future applications. By properly selecting and combining components on demand, applications would adapt to individual user preference and would consider available context information. Existing service composition systems often require users to request services in strict syntax formats, such as data types, service templates or logic formulas. This requirement may become an obstacle for end-users to use such systems. Instead, service composition should be semantics - based so that a service is requested and composed not by its syntax but by its semantics.

The core of the dynamic service composition is the ability to identify, select, and integrate appropriate component services from a potentially large set of services. That is, it requires an effective mechanism to find a composition of individual services that satisfies a variety of requirements. Second, another fundamental problem in service composition is to specify individual services at an appropriate level that enable precise understanding of cross-enterprise and intra-enterprise application integration. Because it can be time and cost-prohibitive to identify and compose services manually, automatic or semi-automatic composition of existing services to achieve new functionality has become the center of current attention.

CHAPTER 2

LITERATURE REVIEW

2.1 Service Oriented Software Architecture for Sensor Networks.

In [5] the author describes the concept of service oriented software architecture for sensor networks. By the use of this architecture, a flexible, scalable programming of applications based on an adaptive middleware is possible. The middleware supports mechanisms for cooperative data mining, self-organization, networking, and energy optimization to build higher-level service structures. The development of applications speeds up and is simplified by the use of optional administration components.

In this paper the author also explain the needs of services in sensor networks, such as routing and packet forwarding, future service architectures are required enabling location and utilization of services. A service is a program which can be accessed about standardized functions over a network. Services allow a cascading without previous knowledge of each other, and thus enable the solution of complex tasks. A typical service used during the initialization of a node is the localization of a data sink for sensor data. Gateways or neighboring nodes can provide this service. To find this service, the node uses a service discovery protocol.

2.2 Applying the Service Oriented Paradigm to Develop Sensor/Actuator Networks

In [6], the author present a model driven approach based on the service oriented paradigm to support the different expert involved in the development, namely platform experts, domain experts and end users. The goal of this approach is to enable the use of pre-implemented services in a potentially heterogeneous sensor/actuator

network. The execution of these components is performed by a middleware. To address the resource constraints, this middleware is tailored for each application using a domain specific development tool. The platform experts can expand the code generator to support further platforms. Domain experts provide services and describe a potential interaction between different services. The end users can select, configure and combine adequate services to form a running application.

But the future goals are to support dynamic instantiation of services to increase flexibility and to address fault-tolerance issues as well as system triggered optimizations at run-time. The code implementing the service instances is independent of the concrete service interaction. Thus, the Service instance needs not to have knowledge about the location of the interconnected services and nodes. The data processing in a service is triggered by incoming data at an input port; after successful processing, the output data is sent to an output port. For basic services, processing could also be triggered by hardware events. Output events of a service are observed by the Service Broker which routes those messages to all connected input ports. The service even does not know where his input came from and where his output will be sent to by the Service Broker.

In this paper, they proposed an approach using domain specific languages and a template-based code generator to accelerate the development of sensor actuator network applications and to increase reusability. For the domain specific language, the authors are using a service-oriented approach. The sensor network application is interpreted as a set of services that interact via an event based push model. Basic services are used to access hardware devices and offer an abstraction of the low-level implementation. Logic services in combination with the service composition done in the Service Brokers realize the actual application logic and can be implemented independent of the used hardware and operating system.

2.3 Toward Effective Service Composition for Real-Time SOA-Based Systems

In [7], the author explains about the many application domains are increasingly leveraging service-oriented architecture (SOA) techniques to facilitate rapid system deployment. Many of these applications are time-critical and, hence, real-time assurance is an essential step in the service composition process. However, there are gaps in existing service composition techniques for real-time systems. First, admission control is an essential technique to assure the time bound for service execution, but most of the service composition techniques for real-time systems do not take admission control into account.

In this paper, they proposed a three-phase composition approach to address the above issues. In this approach, it first use a highly efficient but moderately accurate algorithm to eliminate most of the candidate compositions based on estimated communication latencies and assured service response latency. Then, a more accurate timing prediction is performed on a small number of selected compositions in the second phase based on confirmed admission and actual communication latency. In the third phase, specific concrete services are selected for grounding, and admissions are actually performed. The approach is scalable and can effectively achieve service composition for satisfying real-time requirements. In order to support the new composition approach, it is necessary to effectively specify the needed information.

The proposed timing specification model provides the capability to specify the timing requirements and service composition of the real-time systems and timing properties of Web services. It is comprehensive in providing the information required for timing analysis. It has also proposed a novel three-phase composition technique that takes admission control into account at the composition time.

2.4 Service Composition in Service Oriented Wireless Sensor Networks with Persistent Queries

In [8], the service-oriented wireless sensor network (WSN) has been recently proposed as an architecture to rapidly develop applications in WSNs. In WSNs, a query task may require a set of services and may be carried out repetitively with a given frequency during its lifetime. A service composition solution shall be provided for each execution of such a persistent query task. Due to the energy saving strategy, some sensors may be scheduled to be in sleep mode periodically. Thus, a service composition solution may not always be valid during the lifetime of a persistent query. When a query task needs to be conducted over a new service composition solution, a routing update procedure is involved which consumes energy.

This paper, aims to use the minimum number of service composition solutions during a persistent query's lifetime such that the energy consumption caused by repetitively conducting query routing procedure is minimized. Once the minimum number of required service composition solutions is derived, it then selects the service composition solutions with minimum transmission cost.

2.5 Dynamis: Dynamic Overlay Service Composition for Distributed Stream Processing

In [9], the author addresses the problem of mapping software services onto an overlay network, specifically, the probing to locate suitable nodes on which to instantiate or configure data processing operators. The service composition problem arises in distributed environments where the system needs to set up and bind a number of entities in order to realize services. With the emergence of overlay networks and adaptive middleware technologies, dynamic composition of overlay services has recently attracted considerable attention.

In this paper, the author focus on the problem of mapping distributed services onto an overlay network. This functionality is an integral part of any overlay-based streaming framework and typically requires a probing mechanism to locate suitable nodes on which to instantiate new data processing operators , or to reconfigure and possibly share existing operators .The probing protocol should incur minimal traffic overhead while producing a high-quality mapping of services onto the overlay infrastructure. The quality can be measured in terms of metrics such as end-to-end delay, load balance, security, and cost.

These studies and others have significantly advanced the areas of service composition and data stream processing. Dynamis complements the above approaches by realizing a generic optimization technique based on distributed selection. As it has been a distributed selection can be applied to existing probing protocols. A service cloud can be viewed as a collection of hosts whose resources are available to enhance services (e.g., in terms of fault tolerance and quality of service) transparently to the endpoints. Effectively, overlay nodes provide a "blank computational canvas" on which services can be instantiated and reconfigured as needed. Here, it uses different probing algorithms in Service Clouds and evaluates the differences in the resulting service mappings.

2.6 Qos-Guaranteed Path Selection Algorithm for Service Composition

In [10], authors proposed a depth first search heuristic algorithm K-Closest Pruning (KCP) to solve the problem of multi-constraint service path selection for Service Overlay Network in polynomial time. The main feature of this algorithm is that the path selected by this algorithm meets all the QoS requirements specified by the user/application. The main objective is to find a service-path, which is defined as sequence of nodes providing such that the request QoS constraints such as end-to-end loss, delay and available bandwidth are satisfied.

2.7 A QoS Aware Service Composition Protocol in Mobile Ad Hoc Networks

In [11], authors proposed a distributed QoS-based service composition protocol for MANETs. The protocol uses flooding-based source route techniques to broadcast service composition messages in search of the optimal service composition path which meets QoS constraints and minimizes resource consumption. As a flooding-based protocol may create a large number of control messages, some measures must be taken to reduce the control message overhead and eliminate the broadcast storm problem. However due to the energy constraints of WSN this approach may not be suitable for service composition in WSN.

2.8 An Approach for Dynamic Composition of Services in Distributed Systems.

In [12], the author proposed an approach allows the client to benefit from the functionalities of more than one service and to combine them at run-time to get novel services which provide novel functionalities. It can describe the service behavior which enables the client to use the service discovered and to understand how the invocation of the service operations needs to be performed. Based on this description, it has presented in this paper an approach that allows the dynamic composition of services in distributed systems. This approach is a powerful one since it allows the client to benefit from the functionalities of more than one service to get novel functionalities and novel services. This allows the client to get its appropriate results by avoiding each time to enter its traditional requirements as criteria in its requests.

CHAPTER 3

PROPOSED METHOD

3.1. Sensor Service Modeling and Composition

A graph-based model for representing sensor network services that is intuitive and also amenable to analysis. A service s_i in a sensor network is defined by the input data that it accepts, the transformation function that it applies to its input, the output data that it produces, as well as metadata that provides additional information that characterizes the service and its outputs:

$$s_i = \{input_i = (input_i,1, ..., input_i,m),$$
$$output_i = (output_i,1, ..., output_i,k),$$
$$f_i(input_i) \rightarrow (output_i), metadata_i(t)\}$$

A service implemented in a sensor network may be just a source service, which does not receive any input and only outputs data. Similarly, it can also have sink services, which do not produce any outputs (but may take an action instead). In the above service definition, metadata carries information about the data and the services that process it, following the approach. Metadata may contain properties of the data such as levels of reliability, as well as cost information and certain characteristics of the service itself, such as energy consumption per output data produced, processing delays, number of other services that make use of its outputs, etc. The service metadata depends on time (t) and each service has different types of metadata that is transmitted to other services offered by the sensor network. Metadata information is used to find which services are most cost efficient to use for a given composite service requested by an end user.

3.2 Service Graph of a Sensor Network

Service graph of a sensor network, G_S, consists of vertices representing services and directional edges representing the potential data flows between services. The edge directed from the vertex of service A to the vertex of service B is created if and only if the output of A and input of B intersect in some fields (informally, A can provide some of its outputs to B). Additionally, it requires that each input of B is connected to at least one output of some service (assuming that by definition, a service requires all its input data to produce any output). A formal definition of the service graph is given below:

$G_S = \{V, E\}$ where.

$V = \{s_i\}$ (one vertex per service) and

$$E \subseteq VxV, \text{ where } e_{i,j} = \begin{cases} 1 & \text{if } output_i \cap input_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Although not stated explicitly in the graph definition, it requires matching properties of the metadata of the service providing some data as output with those of the service that is requesting this data as input. For example, if a service requires input with reliability of at least a certain value, only the services that can provide this type of data at the requested reliability level can potentially be connected to the service description with a directed edge. While the above definition of the service graph requires exact match between input and output fields of two services to create an edge between them, this requirement can be relaxed by using type hierarchies. If a service A's output field is a subtype of a service B's input field, then A can provide the information that B requires, hence the edge between them should be allowed.

3.3 Heuristics for Dynamic Service Composition in Sensor Networks

The algorithms that here present in this section aim at achieving cost optimization across the sensor network, while reacting to changing network conditions by recomposing the service. Here I use a two approaches termed top-down and bottom-up, with the latter one further divided into two variants, which differ in the amount of information used to find the composition. To ensure the heuristics by considering only service graphs those are acyclic and directed.

3.3.1 Top-down Approach

The top-down algorithm starts when the user-request (which is represented by a sink service) is received. It first finds a set of services that satisfy its inputs and minimize the local cost, which is the sum of the cost of services chosen and communication costs between those services and user request. Then, the services that were selected choose their input providers and so on. A key requirement in this scheme is that all services at the current level are composed before any services on the next level are considered. It is easy to see that this approach is indeed a breadth-first traversal in the service graph, G_S, which provides us with the ability to identify, which services are already used for composition. However, this breadth-first traversal requirement also makes the top-down approach difficult to implement in a distributed way, since it requires synchronization among sensor nodes involved in the composition process.

At each level, first, it select the critical services, which are those that exclusively provide input fields of a service that are not provided by any other service. Since these services have to be included in any feasible set of services, they may cover some inputs that could also be provided by otherwise non-critical services. Then, a set of services is chosen such that among all sets that can supply the inputs to the service under consideration, the chosen set exhibits the smallest cost.

A drawback of the top-down approach is that, at any level, it may choose a set of input providers that cannot be further decomposed (their inputs cannot be satisfied), since it makes local decisions without knowledge of available services at the lower levels of the graph, as each service knows only its immediate neighbors.

3.3.2. Bottom-up Approach

The bottom-up approach sorts topologically the directed and acyclic service graph G_S. At each service, assuming the possible input providers (neighbors) have composed themselves, a subset of neighbors is chosen to satisfy input fields. A filtering function can be applied at this stage to filter out neighbors with unwanted properties (e.g. high cost, low reliability etc.).

Algorithm for Choosing Services

Method ***find comp(input,output_sets,compositions,S)***
Remaining_in = input
Remaining_out = output_sets
Composition_graph = vertex(service S)
While remaining_in ! = Ø **do**
    If £ S_j *Є remaining_out* is a critical service **Then**
        $S_{min} == S_j$
Else
    For each service S_j in *remaining_out* do
    Find S_{min} where is smallest
    End for
End if
Remaining_in - = remining_in ∩ S_{min}
Composition_graph + = Compositions[S_{min} *] + Edge (* S_{min} → *S)*
Remaining_out - = S_{min}
For each service S_k in remaining_out do
Composition[S_k]- = compositions [S_k] ∩ compositions[S_{min}]
End for
If remaining_out ==Ø then
Break
End if
End while
Return Composition_graph

The bottom-up method (Algorithm 1) is designed following the heuristic for the set cover. At each step, it attempts to find the neighboring service that has the smallest cost per new input field that it can provide. The cost is calculated by the composition graph of the neighbor node and communication cost between the neighbor and the service that is being composed. Note that, when a service is selected, its composition graph (compositions [Smin]) is subtracted from the graphs of all services that haven't been used yet. This is done in order to prevent duplicate additions of services and links between services. When a service A is chosen to provide input to another service B, all services and their links that denote information flow are already used in the composition. As in the top-down approach, critical services are always included first due to the input fields that exclusively provide.

The bottom-up method, as described above, incurs significant communication overhead when implemented in a distributed manner, due to transferring complete composition sub graphs among the neighbors of a service. An alternative approach transmits upstream only the composition cost of the sub graph. When a service S chooses a set of services to utilize in order to satisfy its input, the cost of this service is sent upstream to services that may utilize S's output. Sending the collective cost information incurs a lighter load on the system. However, less information is also made available about the composition of a service's possible input providers and hence may result in a less cost efficient composition. When a service does not know which services will be utilized by its input providers, it is not possible for it to reuse the services potentially increasing the composition's cost.

3.4. Implementing the Composition Selection Algorithm

The selection of composition can be made either in a centralized way, at a single node that receives information from all services, or distributive, wherein each node with a service assigns to it chooses which services it will use to satisfy inputs of its service. In the following, the details of these two approaches are further discussed.

3.4.1 Centralized Implementation: uses a single node on which metadata of each needed service is first collected. Then, the node runs either the top-down or bottom-up algorithm and selects the component services to be activated. In the top-down approach it gives a lower cost solution than the bottom-up one, while in some others cases, the opposite is true. The service graph can be generated at the central node based on the sensor network topology, which is discovered through the information collected from every sensor node. This information includes the set of services (and their metadata) that the reporting node offers and the communication costs from that node to its neighboring.

3.4.2 Distributed Implementation: makes each node with a service allocated to it to select, independently of others, services that it needs to receive inputs for its service. The advantage of this scheme is its robustness to network faults and the quick reaction to changes in the network conditions. Additionally, no single node is selecting the entire composition, avoiding a single point of failure and bottleneck. The composition process proceeds from bottom to top. In the distributed implementation, a node selecting services has only information about its own neighbors, i.e. the services which could provide input to its service. This information is included in the metadata that the node receives from its potential input providers. When a composition process starts, messages are sent from the node at which the end-user request for service originated (considered to be at the top level of the service graph) to the nodes capable to provide lower level services.

User-requested data has certain properties that are provided at the time of the request. According to these properties, a set of nodes with services whose outputs can satisfy the request will be considered neighbors of the node from which the user-request originates. This process repeats until the necessary information is disseminated downstream to all the nodes with source services, which do not have any incoming edges, i.e. any inputs. At that time, the reverse dissemination process takes

place, in which, at each stage, the service composition algorithm is executed. Once every node with the service is aware of its smallest composition cost, backward messaging takes place, where services on certain nodes are activated. This generates the composition graph.

3.5. Dynamic Composition

In a sensor network environment, the conditions of service can change fairly frequently. Hence, it is important to be able to dynamically change service composition. Metadata information exchanged throughout the network can be used for this purpose. For the centralized implementation, dynamic composition is similar to generating the initial one: for each update in the system, the composition process is re-executed, triggered by the new costs. In the distributed case, each node with a service will decide on its new composition graph based on the updates of its neighbors. When a node with a service updates its own information, it also notifies its neighbors so that they can, if needed, change their respective compositions. If a service (or a link of a service) is not used anymore, the node running will deactivate its corresponding output links. If all links of the service are deactivated, the node will stop the service itself and will send stop signals to every service that provided input to stopped service.

Finally, an important issue that needs to be considered in dynamic composition is the frequency of updates. Some of the metrics, such as residual energy on the nodes that the services are implemented on, change continuously. Therefore a dynamic composition solution would be triggered constantly by the change, leading to high overhead. Such a situation can be prevented by setting up a composition update period, or an updating scheme. A node running a service can wait for significant changes in service conditions to notify the central node or the nodes with neighboring services. Example of such significant changes is threshold on reliability, critical low battery level of the sensor node, etc.

CHAPTER 4

IMPLEMENTATION DETAIL

4.1 SIMULATION ENVIRONMENT

4.1.1 Network Simulator (NS)

Network simulator (NS) is an object–oriented, discrete event simulator for networking research. NS provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks. The simulator is a result of an ongoing effort of research and developed. Even though there is a considerable confidence in NS, it is not a polished product yet and bugs are being discovered and corrected continuously.

NS is written in C++, with an OTcl1 interpreter as a command and configuration interface. The C++ part, which is fast to run but slower to change, is used for detailed protocol implementation. The OTcl part, on the other hand, which runs much slower but can be changed very fast quickly, is used for simulation configuration. One of the advantages of this split-language program approach is that it allows for fast generation of large scenarios. To simply use the simulator, it is sufficient to know OTcl. On the other hand, one disadvantage is that modifying and extending the simulator requires programming and debugging in both languages.

4.1.2 Define Node Configuration Parameters

Following is a list of commands used in wireless simulations:

Parameter	Value
Network size	1500*1500 (m)
Number of sensor nodes	100
Initial energy	50 J
Data packet size	1024 bytes
Transmission range	250 m
Protocol	AODV
Propagation	Two Ray Ground
Flat Grid	1000 *100

Table 1.1 Node Configuration parameters

CHAPTER 5

RESULTS AND DISCUSSIONS

The experiments are performed on a Pentium 2.4 GHz with 128MB of RAM on Fedora 8.0 with C++ & TCL Scripting

5.1. TOP- DOWN AND BOTTOM UP APPROACH

In top-down and bottom-up approach I constructed a graph based topology for 10 nodes. Based on the user request it may select either of that approach and satisfy the service of the user. Initially its displays the detail of service that are provided by each nodes and type of approaches that used to select the request. Based on the service requested node it's satisfied the user request by choosing the less cost when compared to other nodes and display its metadata information to the user. It also displays the nodes details such that satisfied service, node id and communication cost, and filters the non- participated nodes to the user.

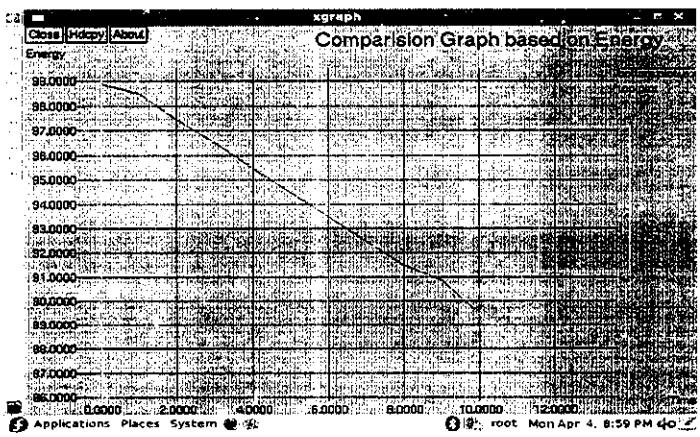


Fig 1.2 Energy Comparisons Graph for Top- Down and Bottom-Up Approach

Here I compare both the approaches of top down and bottom up based on energy and the performance is high to bottom up when compare to top-down.

5.2 CENTRALIZED APPROACH AND DISTRIBUTED APPROACH

In Centralized and Distributed approached I constructed a topology such that one node is assumed to centralized node to start its transmissions. Here the user is request to provide information of request and response node details. For Centralized case it starts requesting from the central node and the response is provided based on its centralized node to the requested node and for distributed case the centralized node will broadcast its detail to all nodes. Based on the request, it starts the transmission to it specified response node and it response to its requested node Here it also displays the metadata information of the user selected node and displayed to the user. It also displays the nodes details such that satisfied service, node id and communication cost, and filters the non- participated nodes to the users.

Fig 1.3 Energy Comparison graph for Centralized and Distributed approach

Thus performance comparison graph is drawn for both the Centralized and Distributed Approach and found that distributed is high when comparing to the centralized.

5.3 DYNAMIC COMPOSITION

In Dynamic composition I construct a topology such that three node act as special node to satisfy the request. This node acts as critical service such that it provides as an exclusively service that are not provided by other nodes. Based on the user request it maps the request and response through the special nodes. Finally it displays the detail of the satisfied node and metadata information.

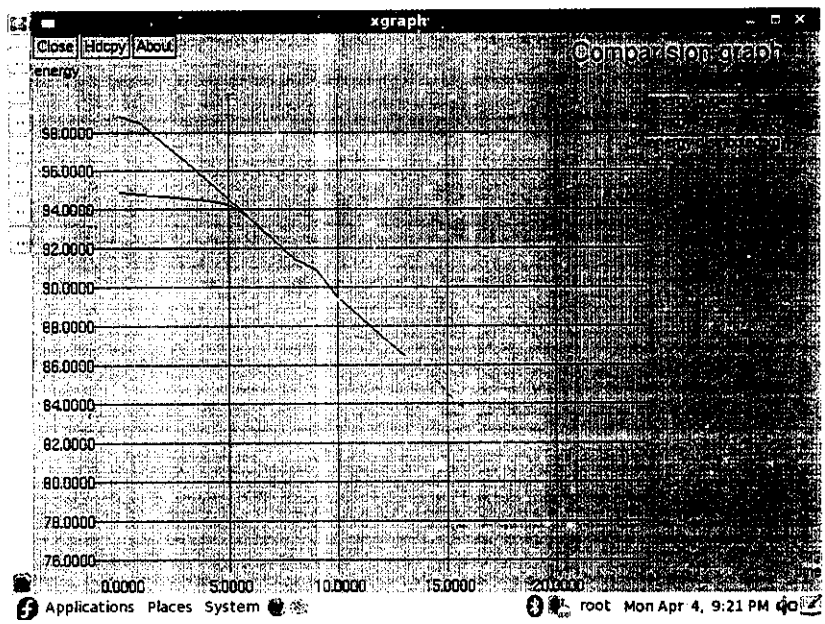


Fig 1.4 Energy Comparison Graph for Dynamic Service, Centralized and Distributed approaches

Thus performance comparison graph is drawn for the centralized, distributed and dynamic service composition. Hence distributed provide high when compare to others.

CHAPTER 6

CONCLUSION AND FUTURE OUTLOOK

In this method of service modeling and dynamic composition is described, which is suitable to the unreliable and dynamic sensor network environments. Two heuristic algorithms for composition of sensor services are introduced that differ in the direction of traversing the service graph during the composition process. Centralized and distributed implementations of these algorithms are also described and evaluated through simulations, along with the associated trade-off between overhead of performing the composition and the cost of the final composition result.

As a future work this can be extended for providing a quality of service.

CHAPTER 7

APPENDIX

7.1 SOURCE CODE

Coding for Top Down and Bottom Up Approach

/* IMPLEMENTATION OF TOP DOWN AND BOTTOM UP APPRAOCH */

/* ---------- */

```
# Top Down and Bottom Up Approach
=================================================================
===

# Define Node Configuration paramaters

#=================================================================
====

set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-
propagation model
set val(netif)          Phy/WirelessPhy            ;# network
interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface
queue type
set val(ll)             LL                         ;# link layer
type
set val(ant)            Antenna/OmniAntenna        ;# antenna
model
set val(ifqlen)         50                         ;# max packet
in ifq
set val(nn)             11                        ;# number of
mobilenodes
set val(rp)             AODV                       ;# routing
protocol
set val(x)              1000                        ;
set val(y)              1000                       ;
#set opt(energymodel)    EnergyModel                ;
#set opt(radiomodel)     RadioModel                 ;
#set opt(initialenergy)  100                        ;# Initial
energy in Joules
Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.2818     ;# for 250.0
Phy/WirelessPhy set freq_ 914e+6
```

```
Mac/802_11 set dataRate_  2.0e6            ;# 1Mbps
# Initialize Global Variables
set ns_          [new Simulator]

#=================================================================
=====
# Initialize trace file desctiptors

#=================================================================
=====
set tracefd     [open dist2.tr w]
$ns_ trace-all $tracefd

set namtrace [open dist.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

# Create God
create-god $val(nn)

# Create channel #1 and #2
set chan_1_ [new $val(chan)]
set chan_2_ [new $val(chan)]


# Create node(0) "attached" to channel #1
# configure node, please note the change below.
$ns_ node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace ON \
            -idlePower 1.0 \
            -rxPower 1.0 \
            -txPower 1.0 \
            -sleepPower 0.001 \
            -transitionPower 0. \
            -transitionTime 0.003 \
            -channel $chan_1_
```

```
#$ns_ node-config \              channel $chan_2_
set node_(0) [$ns_ node]
set node_(1) [$ns_ node]
set node_(2) [$ns_ node]
set node_(3) [$ns_ node]
set node_(4) [$ns_ node]
set node_(5) [$ns_ node]
set node_(6) [$ns_ node]
set node_(7) [$ns_ node]
set node_(8) [$ns_ node]
set node_(9) [$ns_ node]


set node_(10) [$ns_ node]
$node_(3) random-motion 0
$node_(4) random-motion 0
$node_(5) random-motion 0
$node_(6) random-motion 0
$node_(7) random-motion 0
$node_(8) random-motion 0
$node_(9) random-motion 0
$node_(10) random-motion 0

for {set i 0} {$i < $val(nn)} {incr i} {
      $node_($i) set X_ 0.0
      $node_($i) set Y_ 0.0
      $node_($i) set Z_ 0.0
      $ns_ initial_node_pos $node_($i) 20
}

# Now produce some simple node movements


$node_(0) color brown
$node_(0) color green
$node_(1) color blue
$node_(7) color orange

$node_(2) color red
$node_(3) color red
$node_(4) color red
$node_(5) color red
$node_(6) color red


$node_(8) color chocolate
$node_(9) color chocolate
$node_(10) color chocolate


$ns_ at 0 "$node_(0) setdest 301.688 605.809 3000"
$ns_ at 0 "$node_(1) setdest 164.895 498.596 3000"
$ns_ at 0 "$node_(2) setdest 440.065 498.596 3000"
```

```
$ns_ at 0 "$node_(3) setdest 88.825 364.286 3000"
$ns_ at 0 "$node_(4) setdest 239.049 364.286 3000"
$ns_ at 0 "$node_(5) setdest 388.663 364.286 3000"
$ns_ at 0 "$node_(6) setdest 529.265 364.286 3000"

$ns_ at 0 "$node_(7) setdest 88.825 246.555 3000"
$ns_ at 0 "$node_(8) setdest 239.049 246.555 3000"
$ns_ at 0 "$node_(9) setdest 388.663 246.555 3000"
$ns_ at 0 "$node_(10) setdest 529.265 246.555 3000"

$ns_ at 0 "$node_(0) add-mark c4 black circle"
$ns_ at 0 "$node_(0) label DS0"


$ns_ at 0 "$node_(1) add-mark c4 blue circle"
$ns_ at 0 "$node_(1) label DS1"

$ns_ at 0 "$node_(2) add-mark c4 blue circle"
$ns_ at 0 "$node_(2) label DS1"


$ns_ at 0 "$node_(3) add-mark c4 yellow circle"
$ns_ at 0 "$node_(3) label DS2"


$ns_ at 0 "$node_(4) add-mark c4 yellow circle"
$ns_ at 0 "$node_(4) label DS2"



$ns_ at 0 "$node_(5) add-mark c4 yellow circle"
$ns_ at 0 "$node_(5) label DS2"


$ns_ at 0 "$node_(6) add-mark c4 yellow circle"
$ns_ at 0 "$node_(6) label DS2"


$ns_ at 0 "$node_(7) add-mark c4 red circle"
$ns_ at 0 "$node_(7) label BS0"


$ns_ at 0 "$node_(8) add-mark c4 red circle"
$ns_ at 0 "$node_(8) label BS1"


$ns_ at 0 "$node_(9) add-mark c4 red circle"
$ns_ at 0 "$node_(9) label BS2"


$ns_ at 0 "$node_(10) add-mark c4 red circle"
$ns_ at 0 "$node_(10) label BS3"
```

```
#file

puts "Enter the Service that to be provide"
puts "SERVICE 0 & 10 ---> LIGHT"
puts "SERVICE 1 & 4 & 6---> TEMPERATURE"
puts "SERVICE 2 & 5 & 8 & 9---> PRESSURE"
puts "SERVICE 3 & 7 ---> WEATHER"
puts "(0/1/2) ********************* TOP-DOWN
APPROACH***************"
puts "(7/8/9/10) ****************** BOTTOM-UP APPRAOCH
*************"
puts "Enter the approches node ( 0/1/2/7/8/9/10))"
gets stdin r
if { ($r == 0) || ($r == 1) || ($r == 2) } {
puts "The SOURCE SERVICE node-$r for TOP DOWN approach"
}
if { ($r == 7) || ($r == 8) || ($r == 9) || ($r == 10) } {
puts "The SINK SERVICE node-$r for BOTTOM UP approach"
}
if { ($r != 7) && ($r != 8)  &&  ($r != 9)  && ($r != 10)  &&
($r!= 0)  && ($r != 1)  && ($r != 2) } {
puts "Enter the Valid node"
exit 0
}
puts "Enter the service that to be requested"

gets stdin cost
set f 1
for {set node 1 } {$node < $val(nn)} {incr node } {
set req [open cost($node) r]
gets $req cost1
close $req
if { $cost == $cost1 } {
puts "service-$node is having service of $cost"
set nnode $node
if { ($node == 7 || $node == 6 || $node == 9 ||$node == 10)} {
puts "This service of -$node is having service of $cost with low
cost"
}
set f 1
}
}
############################

puts "Nodes that are not Serviced"
for {set i 0} {$i < $val(nn)} {incr i} {
if { ($i!=$nnode) && ($i!=$r ) } {
puts "node-($i) \n"
}
}

puts "Serviced nodes"
puts "node-($r) \n node-($nnode) "
for {set i 0} {$i < $val(nn)} {incr i} {
```

```
set sink($i) [new Agent/LossMonitor]
$ns_ attach-agent $node_($i) $sink($i)

}
proc attach-CBR-traffic { node sink size interval } {
   #Get an instance of the simulator
   set ns [Simulator instance]
   #Create a CBR  agent and attach it to the node
   set cbr [new Agent/CBR]
   $ns attach-agent $node $cbr
   $cbr set packetSize_ $size
   $cbr set interval_ $interval
   #Attach CBR source to sink;
   $ns connect $cbr $sink
   return $cbr
  }

set cbr1 [attach-CBR-traffic $node_($r)  $sink($nnode)  512
0.048]

$ns_ at 1.0 "$cbr1 start"
$ns_ at 3.0 "$cbr1 stop"

if { ($r == 0) || ($r == 1) || ($r == 2) } {
$ns_ at 1.3 "$ns_ trace-annotate \"  Requesting node-$r is
getting the service using TOP DOWN APPROACH from node-$nnode
which staisfy its service \""
}
if { ($r == 7) || ($r == 8) || ($r == 9) || ($r == 10) } {
$ns_ at 1.3 "$ns_ trace-annotate \"  Requesting node-$r is
getting the service using BOTTOM UP APPROACH from node-$nnode
which staisfy its service \""
}
set nbr [open Neighbor w]
puts $nbr
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
puts $nbr "\tsource-Node\tNeighbor-Node"
puts $nbr
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
close $nbr
#~~~~~~~~~~~~~~~~ For Calculation of Euclidean
distance~~~~~~~~~~~~~~~~~~~
proc distance { n1 n2 nd1 nd2 fl} {
global r
set nbr [open Neighbor a]
set x1 [expr int([$n1 set X_])]
set y1 [expr int([$n1 set Y_])]
set x2 [expr int([$n2 set X_])]
set y2 [expr int([$n2 set Y_])]
set d [expr int(sqrt(pow(($x2-$x1),2)+pow(($y2-$y1),2)))]
```

```
if {$nd2>=$nd1} {
if {$fl == 14} {
#puts $dis "\t$nd1\t\t$nd2\t\t$x1\t$y1\t$d"
}
}
if {$d<250} {
if {$nd2!=$nd1} {
puts $nbr "\t$nd1\t\t$nd2"
}
}
#close $dis
close $nbr
}
for {set i 0} {$i<=10} {incr i} {
for {set j 0} {$j<=10} {incr j} {
$ns_ at 3.5 "distance $node_($i) $node_($j) $i $j 10"
}
}
#**********************************************************************
***************
#**********************************************************************
***************
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 20.0 "$node_($i) reset";
}

$ns_ at 35.0 "stop"
$ns_ at 35.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
    #exec awk -f dbserver.awk ds-server-db &
    #exec awk -f reqst.awk rst ds-server-db &
      puts "running nam..."
      exec nam dist.nam &
}
puts "Starting Simulation..."
$ns_ run
```

Coding for Dynamic Service Composition

/* IMPLEMENTATION OF DYNAMIC SERVICE COMPOSITION */

/* ---------- */

```
# Dynamic services composition
      ================================================================
========

# Define Node Configuration paramaters

#================================================================
====

set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-
propagation model
set val(netif)          Phy/WirelessPhy            ;# network
interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface
queue type
set val(ll)             LL                         ;# link layer
type
set val(ant)            Antenna/OmniAntenna        ;# antenna
model
set val(ifqlen)         50                         ;# max packet
in ifq
set val(nn)             11                         ;# number of
mobilenodes
set val(rp)             AODV                       ;# routing
protocol
set val(x)             1000                     ;
set val(y)             1000                    ;


Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.2818    ;# for 250.0
Phy/WirelessPhy set freq_ 914e+6
Mac/802_11 set dataRate_  2.0e6           ;# 1Mbps

# Initialize Global Variables
set ns_          [new Simulator]

#================================================================
=====

# Initialize trace file desctiptors

#================================================================
=====
set tracefd     [open dist.tr w]
$ns_ trace-all $tracefd
```

```
set namtrace [open dist.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)
# Create God
create-god $val(nn)

# Create channel #1 and #2
set chan_1_ [new $val(chan)]
set chan_2_ [new $val(chan)]


# Create node(0) "attached" to channel #1
# configure node, please note the change below.
$ns_ node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace ON \
            -idlePower 1.0 \
            -rxPower 1.0 \
            -txPower 1.0 \
            -sleepPower 0.001 \
            -transitionPower 0.1 \
            -transitionTime 0.003 \
            -channel $chan_1_


# node_(1) can also be created with the same configuration, or
with a different
# channel specified.
# Uncomment below two lines will create node_(1) with a different
channel.

#$ns_ node-config \  channel $chan_2_
set node_(0) [$ns_ node]
set node_(1) [$ns_ node]
set node_(2) [$ns_ node]
set node_(3) [$ns_ node]
set node_(4) [$ns_ node]
set node_(5) [$ns_ node]
```

```
set node_(6) [$ns_ node]
set node_(7) [$ns_ node]
set node_(8) [$ns_ node]
set node_(9) [$ns_ node]
set node_(10) [$ns_ node]
$node_(3) random-motion 0
$node_(4) random-motion 0
$node_(5) random-motion 0
$node_(6) random-motion 0
$node_(7) random-motion 0
$node_(8) random-motion 0
$node_(9) random-motion 0
$node_(10) random-motion 0

for {set i 0} {$i < $val(nn)} {incr i} {
      $node_($i) set X_ 0.0
      $node_($i) set Y_ 0.0
      $node_($i) set Z_ 0.0
      $ns_ initial_node_pos $node_($i) 20
}

puts "**********************Dynamic-Composition
service************************"

# Now produce some simple node movements


$node_(0) color brown
$node_(0) color green
$node_(1) color blue
$node_(7) color orange

$node_(2) color red
$node_(3) color red
$node_(4) color red
$node_(5) color red
$node_(6) color red


$node_(8) color chocolate
$node_(9) color chocolate
$node_(10) color chocolate


$ns_ at 0 "$node_(0) setdest 151.39 555.166 3000"
$ns_ at 0 "$node_(1) setdest 260.529 593.2 3000"
$ns_ at 0 "$node_(2) setdest 260.619 512.661 3000"
$ns_ at 0 "$node_(3) setdest 370.67 624.244 3000"
$ns_ at 0 "$node_(4) setdest 370.723 538.777 3000"
$ns_ at 0 "$node_(5) setdest 374.406 462.052 3000"
$ns_ at 0 "$node_(6) setdest 465.991 596.768 3000"
```

```
$ns_ at 0 "$node_(7) setdest 462.486 508.259 3000"
$ns_ at 0 "$node_(8) setdest 549.879 573.948 3000"
$ns_ at 0 "$node_(9) setdest 555.047 513.201 3000"
$ns_ at 0 "$node_(10) setdest 611.556 552.202 3000"


$ns_ at 0 "$node_(0) add-mark c4 blue circle"
$ns_ at 0 "$node_(0) label DS0"


$ns_ at 0 "$node_(1) add-mark c4 blue circle"
$ns_ at 0 "$node_(1) label DS1"

$ns_ at 0 "$node_(2) add-mark c4 blue circle"
$ns_ at 0 "$node_(2) label DS2"



$ns_ at 0 "$node_(3) add-mark c4 blue hexagon"
$ns_ at 0 "$node_(3) color red"
$ns_ at 0 "$node_(3) label DS3-SN"


$ns_ at 0 "$node_(4) add-mark c4 blue hexagon"
$ns_ at 0 "$node_(4) color red"
$ns_ at 0 "$node_(4) label DS4-SN"



$ns_ at 0 "$node_(5) add-mark c4 blue hexagon"
$ns_ at 0 "$node_(5) color red"
$ns_ at 0 "$node_(5) label DS5-SN"


$ns_ at 0 "$node_(6) add-mark c4 blue circle"
$ns_ at 0 "$node_(6) label DS6"


$ns_ at 0 "$node_(7) add-mark c4 blue circle"
$ns_ at 0 "$node_(7) label DS7"


$ns_ at 0 "$node_(8) add-mark c4 blue circle"
$ns_ at 0 "$node_(8) label DS8"


$ns_ at 0 "$node_(9) add-mark c4 blue circle"
$ns_ at 0 "$node_(9) label DS9"


$ns_ at 0 "$node_(10) add-mark c4 blue circle"
$ns_ at 0 "$node_(10) label DS10"
```

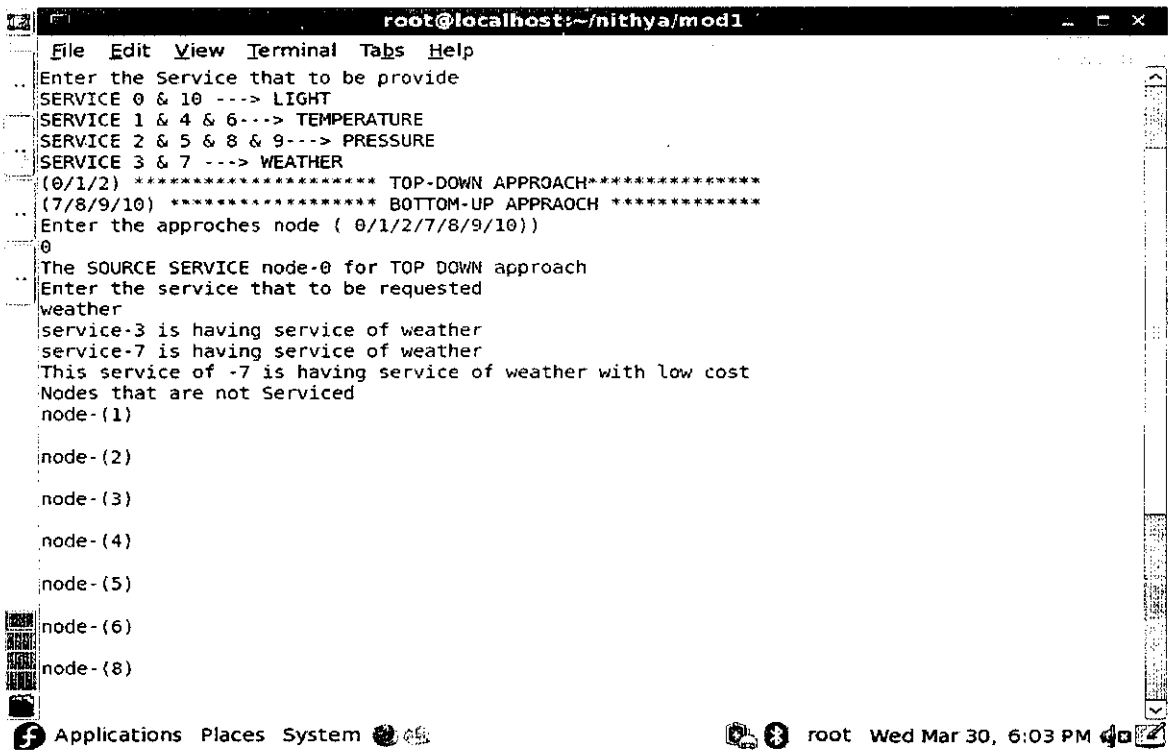
```
##########################
####################
    proc attach-CBR-traffic { node sink size interval } {
   #Get an instance of the simulator
   set ns [Simulator instance]
   #Create a CBR  agent and attach it to the node
   set cbr [new Agent/CBR]
   $ns attach-agent $node $cbr
   $cbr set packetSize_ $size
   $cbr set interval_ $interval
   #Attach CBR source to sink;
   $ns connect $cbr $sink
   return $cbr
  }

  #file
puts "Enter the Service that to be provide"
puts "SERVICE 0 ---> LIGHT"
puts "SERVICE 1 ---> TEMPERATURE"
puts "SERVICE 2 ---> PRESSURE"
puts "CRITICAL SERVICE 3 ---> TEMPERATURE & PRESSURE & MOISTURE"
puts "CRITICAL SERVICE 4 ---> WEATHER & MOISTURE & HUMIDITY"
puts "CRITICAL SERVICE 5 ---> SOUND & TEMPERATURE & VIBRATION"
puts "SERVICE 6 ---> GPS"
puts "SERVICE 7 ---> WEATHER"
puts "SERVICE 8 ---> MOISTURE"
puts "SERVICE 9 ---> TIMER"
puts "SERVICE 10 ---> SOUND"
set req [open rst w]
close $req
set res [open rsp1 w]
close $res
set cs [open sn1 w]
close $cs
puts "Enter the request node"
gets stdin r
puts "Enter the response node (any node other than 3/4/5)"
gets stdin rsp
puts "Enter the special node (3/4/5)"
gets stdin sn
set res [open rsp1 a]
puts $res "$r"
close $res
set req [open rst a]
puts $req "$rsp"
close $req
set cs [open sn1 a]
puts $cs "$sn"
close $cs
```

```
##2 module-cache-discovery

#*******Requesting*********

Agent/TCP set packetSize_    800

set tcp10 [new Agent/TCP]
$tcp10 set class_ 2
set sink11 [new Agent/TCPSink]
$ns_ attach-agent $node_($r) $tcp10
$ns_ attach-agent $node_($sn) $sink11
$ns_ connect $tcp10 $sink11
set ftp10 [new Application/FTP]
$ftp10 attach-agent $tcp10

$ns_ at 5.0 "$ftp10 start"
$ns_ at 9.0 "$ftp10 stop"

$ns_ at 5.0 "$ns_ trace-annotate \" Dynamic composition service
started from requesting node-$r to special servicing-node-$sn
through dynamic-composition \""


#*******responsing*********

Agent/TCP set packetSize_    800

set tcp11 [new Agent/TCP]
$tcp11 set class_ 2
set sink12 [new Agent/TCPSink]
$ns_ attach-agent $node_($sn) $tcp11
$ns_ attach-agent $node_($rsp) $sink12
$ns_ connect $tcp11 $sink12
set ftp11 [new Application/FTP]
$ftp11 attach-agent $tcp11
$ns_ at 10.0 "$ftp11 start"
$ns_ at 14.0 "$ftp11 stop"

 $ns_ at 10.0 "$ns_ trace-annotate \" Dynamic composition service
started from  special servicing-node-$sn to responsing node-$rsp
\""

Agent/TCP set packetSize_    800

set tcp12 [new Agent/TCP]
$tcp12 set class_ 2
set sink13 [new Agent/TCPSink]
$ns_ attach-agent $node_($rsp) $tcp12
$ns_ attach-agent $node_($sn) $sink13
$ns_ connect $tcp12 $sink13
set ftp12 [new Application/FTP]
$ftp12 attach-agent $tcp12
```

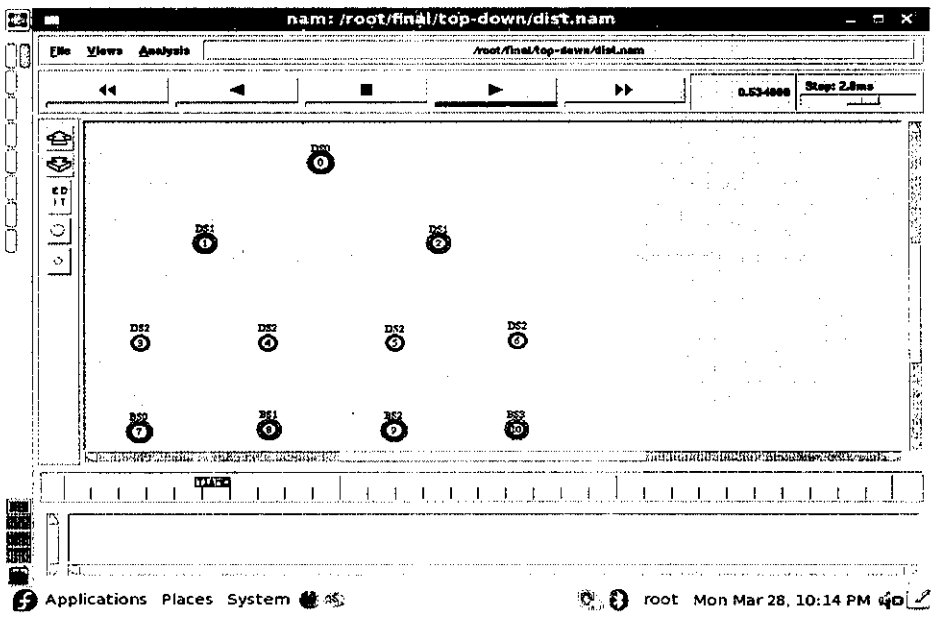
```
$ns_ at 15.0 "$ftp12 start"
$ns_ at 19.0 "$ftp12 stop"
$ns_ at 15.0 "$ns_ trace-annotate \" Resource is
discovered,responsing node transfer the resource from node-$rsp
to special servicing-node-$sn through Dynamic composition \""


#*******responsing*********

Agent/TCP set packetSize_    800

set tcp13 [new Agent/TCP]
$tcp13 set class_ 2
set sink14 [new Agent/TCPSink]
$ns_ attach-agent $node_($sn) $tcp13
$ns_ attach-agent $node_($r) $sink14
$ns_ connect $tcp13 $sink14
set ftp13 [new Application/FTP]
$ftp13 attach-agent $tcp13
$ns_ at 20.0 "$ftp13 start"
$ns_ at 24.0 "$ftp13 stop"

$ns_ at 20.0 "$ns_ trace-annotate \" Discovered resource is
serviced to requested node-$r from responsing node-$rsp through
special servicing-node-$sn as per Dynamic composition\""

proc awk1 { } {
exec awk -f dbserver.awk ds-server-db
}
proc awk2 { } {
exec awk -f reqst.awk rst ds-server-db
}
proc awk3 { } {
exec awk -f rspon.awk rsp1 ds-server-db
}
proc awk4 { } {
exec awk -f service.awk sn1 ds-server-db
}

$ns_ at 5.0 "awk1"
$ns_ at 5.0 "awk2"
$ns_ at 5.0 "awk3"
$ns_ at 5.0 "awk4"

proc update2 { } {
global rsp f4
set fsize [file size "request-node-detail"]
set  f3 [open request-node-detail r]
set contents [read $f3 $fsize]
puts $contents
close $f3
set f4 [open ds($rsp) a]
puts $f4 "$contents"
```

7.2 SCREEN SHOTS

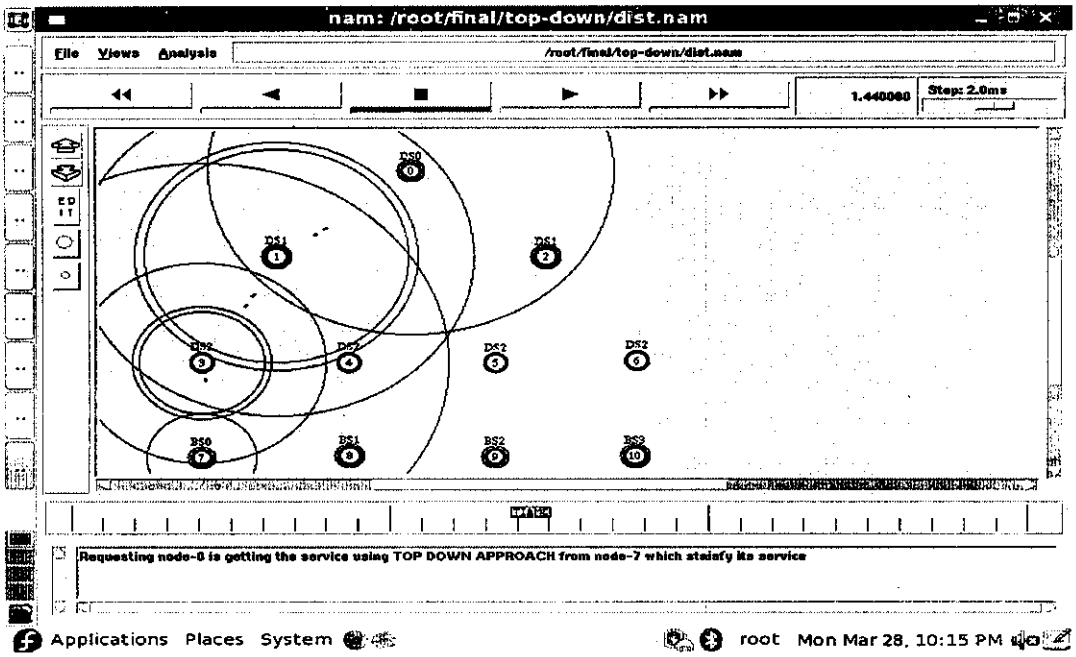
- **Top down**



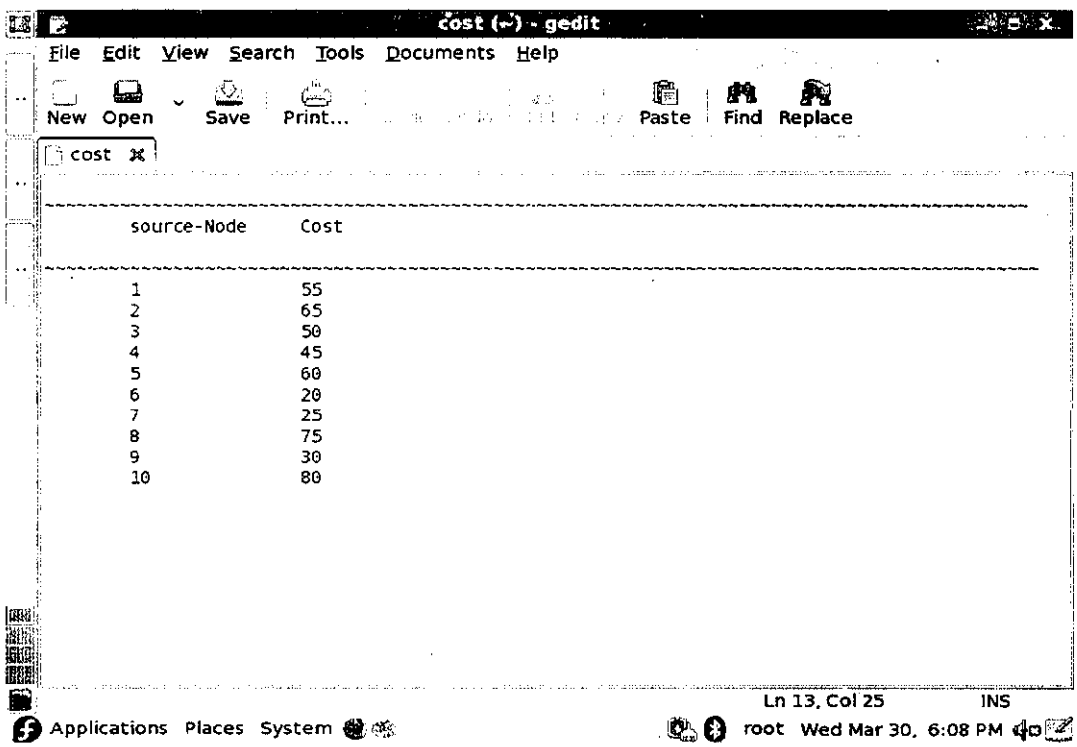
- **Initialize Position of Nodes**



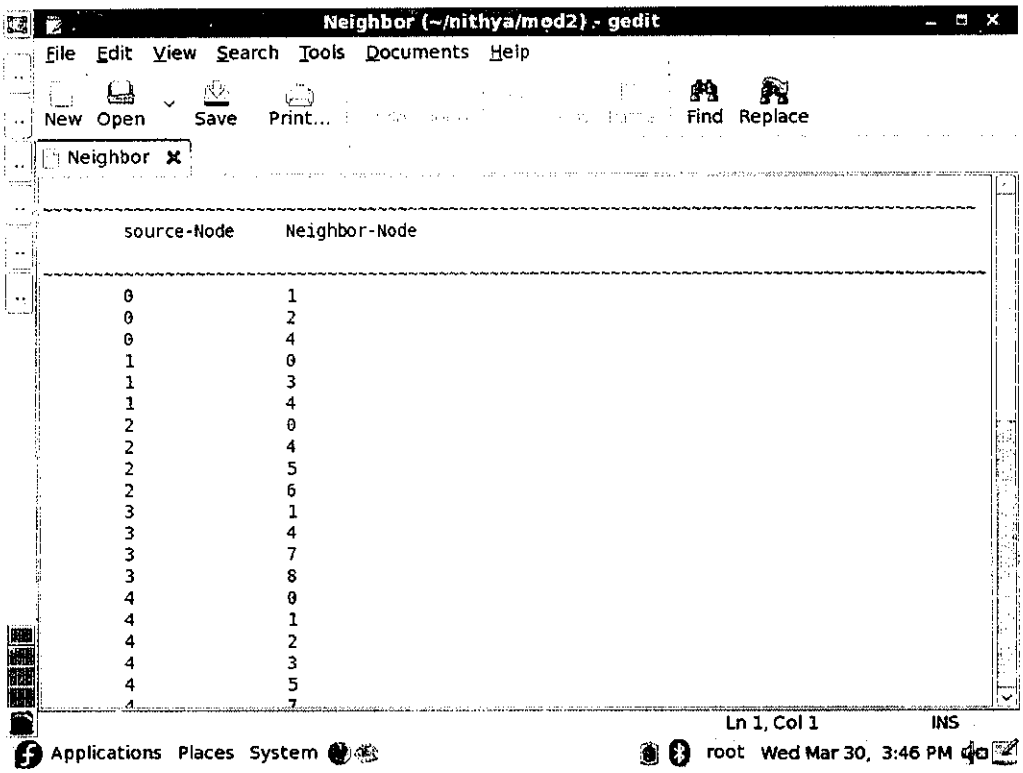
- **Requesting the service**



- **Cost of Each Node**

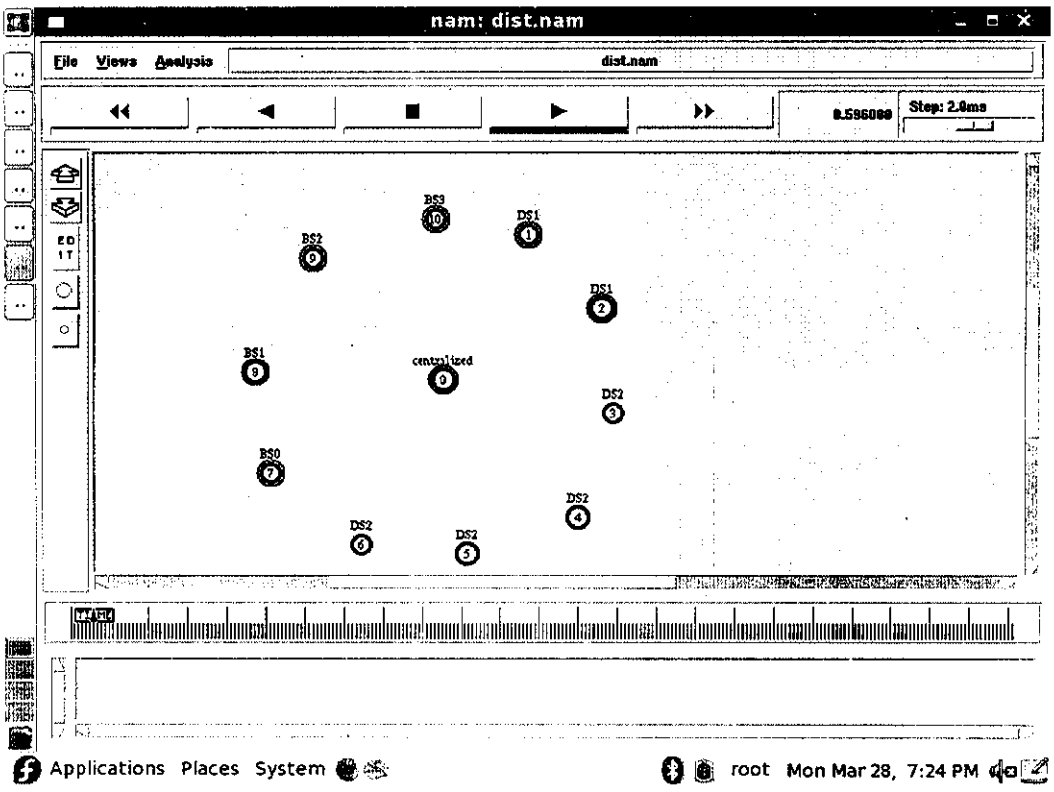


- **Neighbor Details**

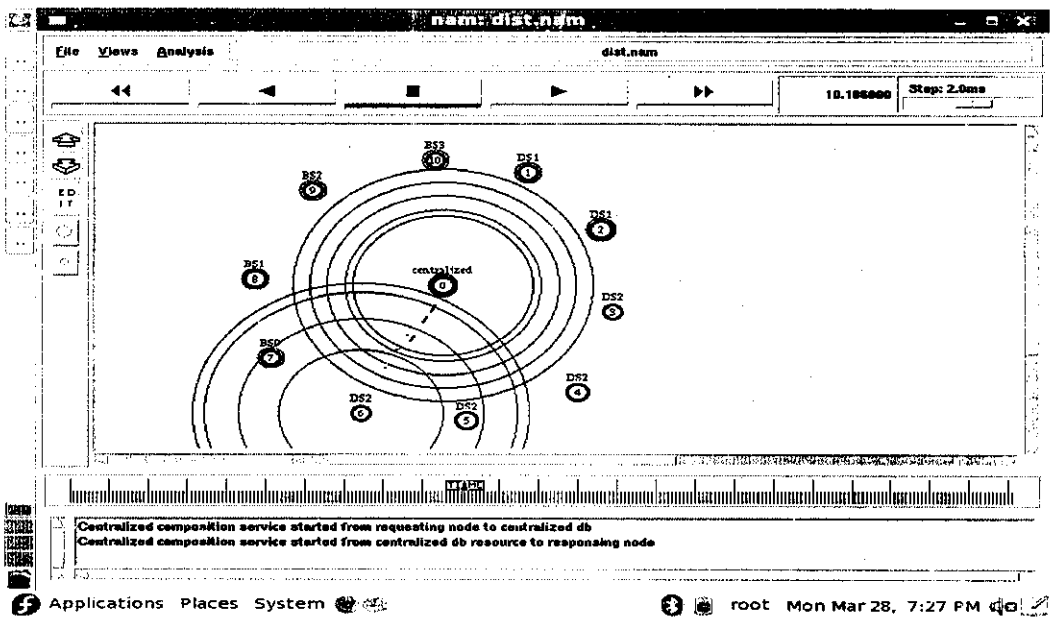


- **Centralized approach**

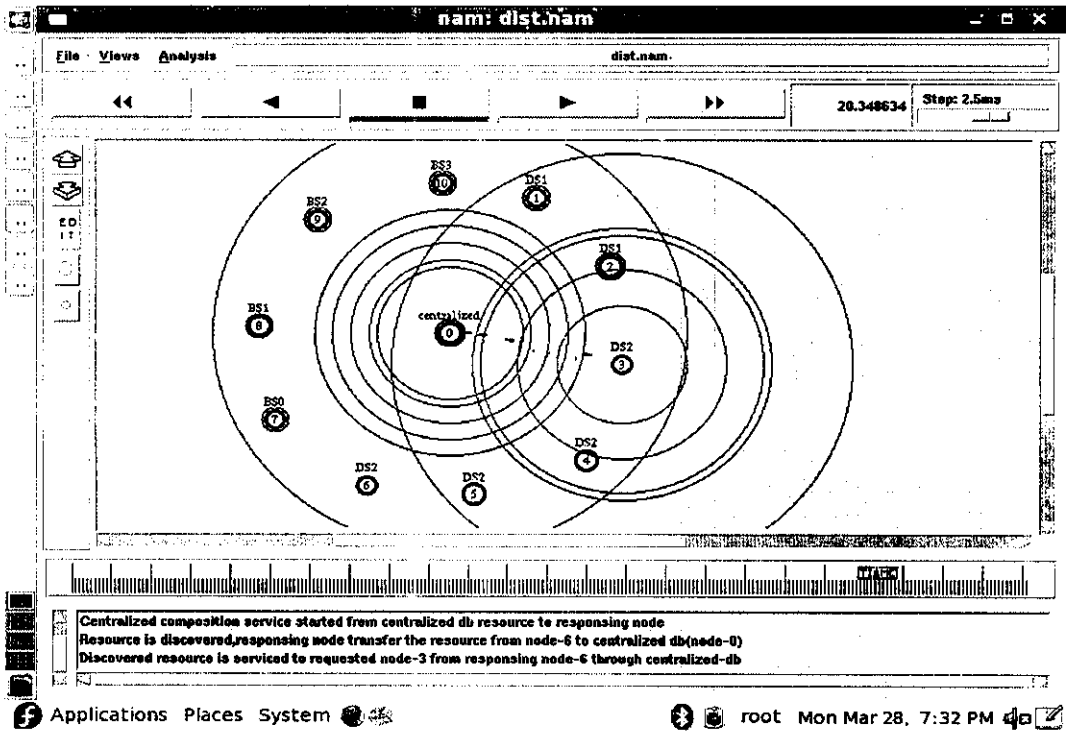
- **Initialize Position of Nodes**



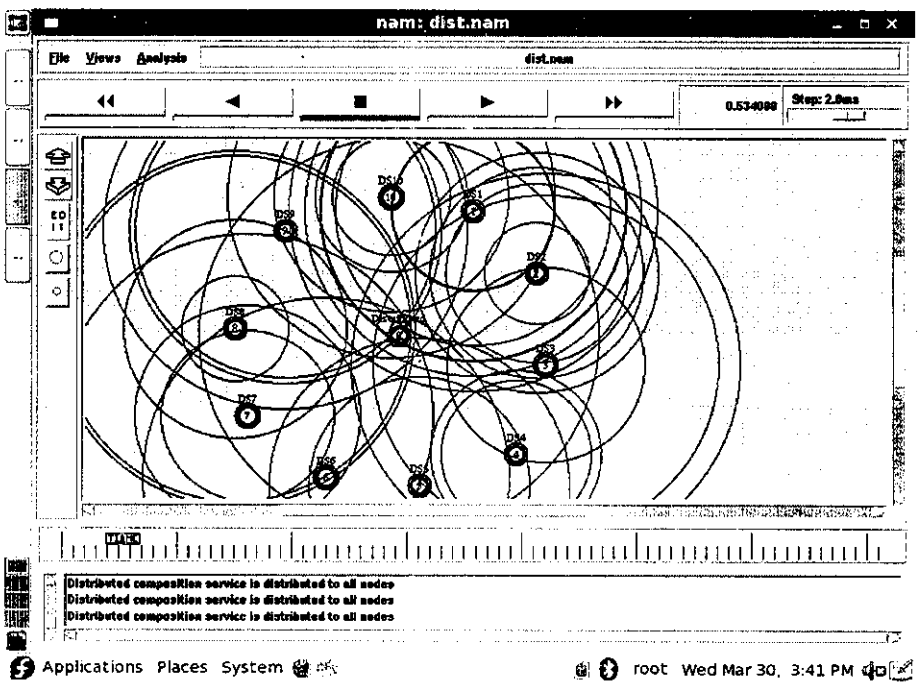
- **Requesting using centralized node**



- **Responding using centralized node**

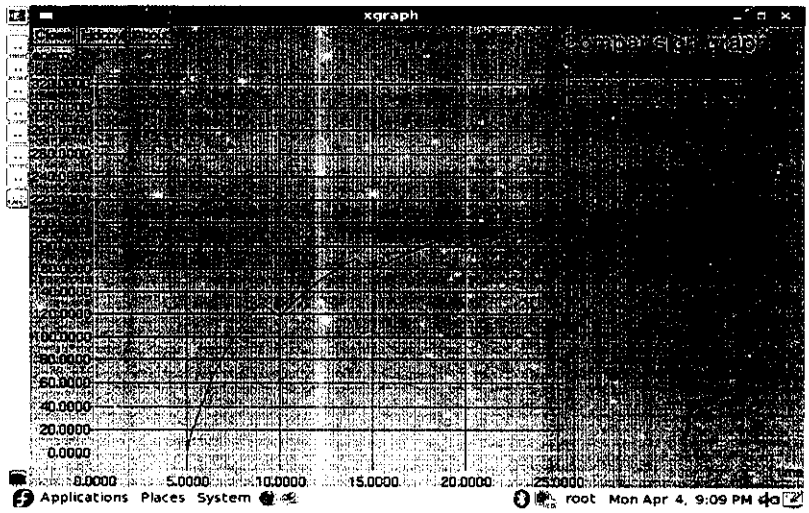


- **Distribution Approach**

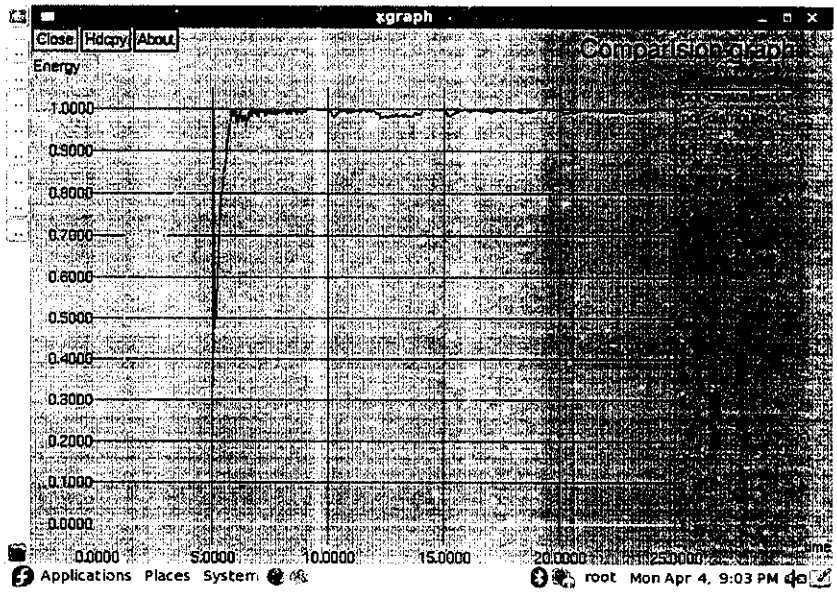


- **Comparisons Graph for Centralized and Distributed Approach**

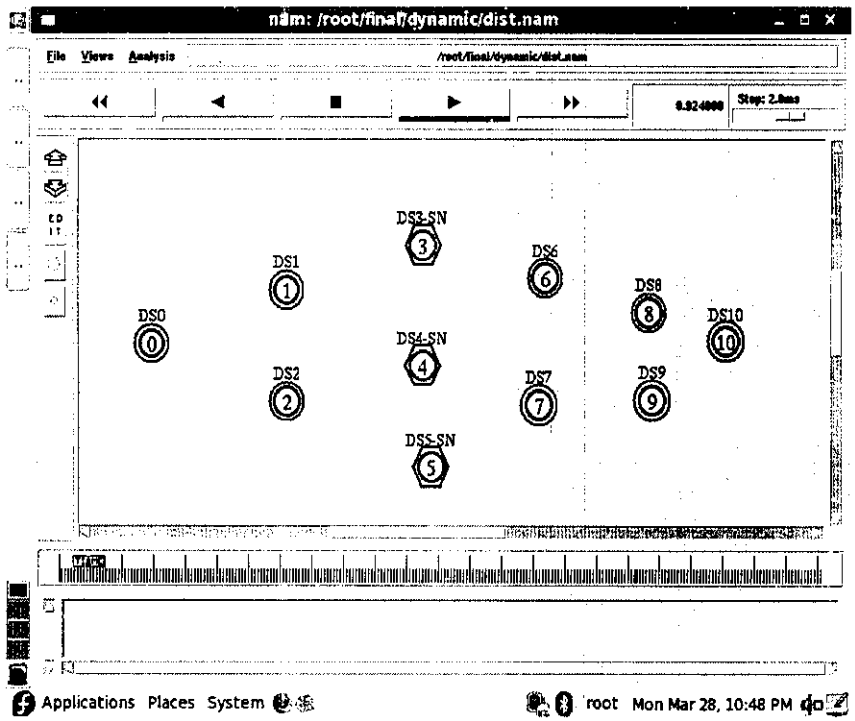
Throughput



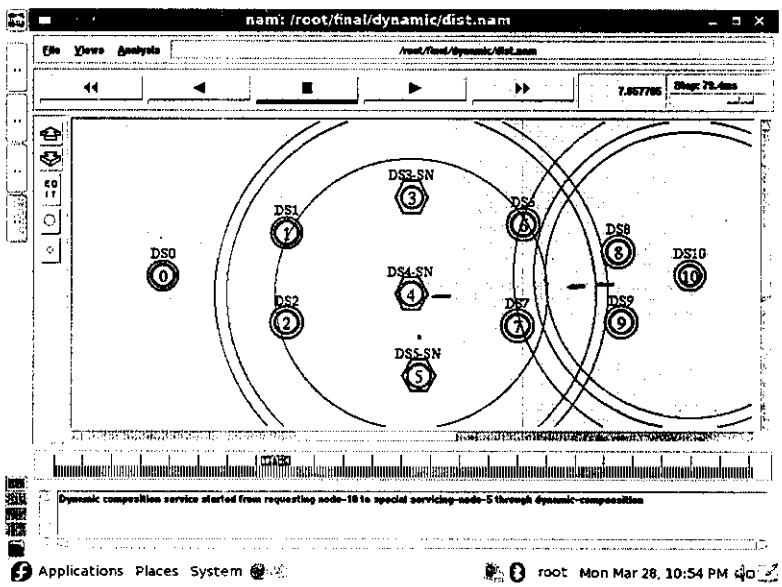
Packet Drop Ratio



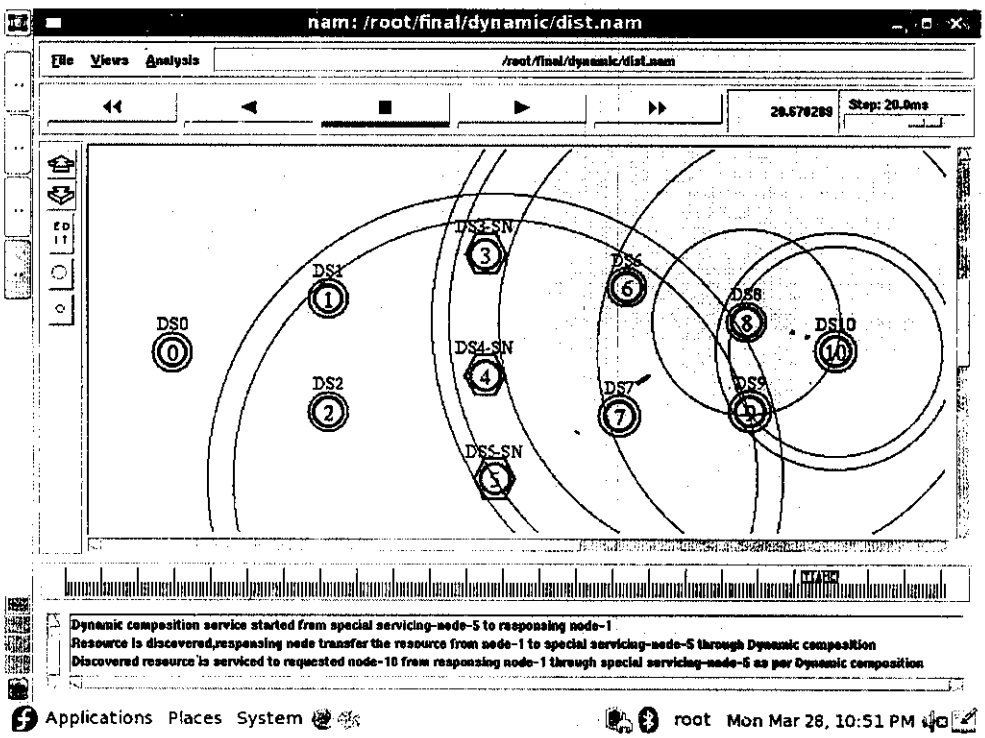
- **Dynamic Service Composition - Initialize Position of Nodes**



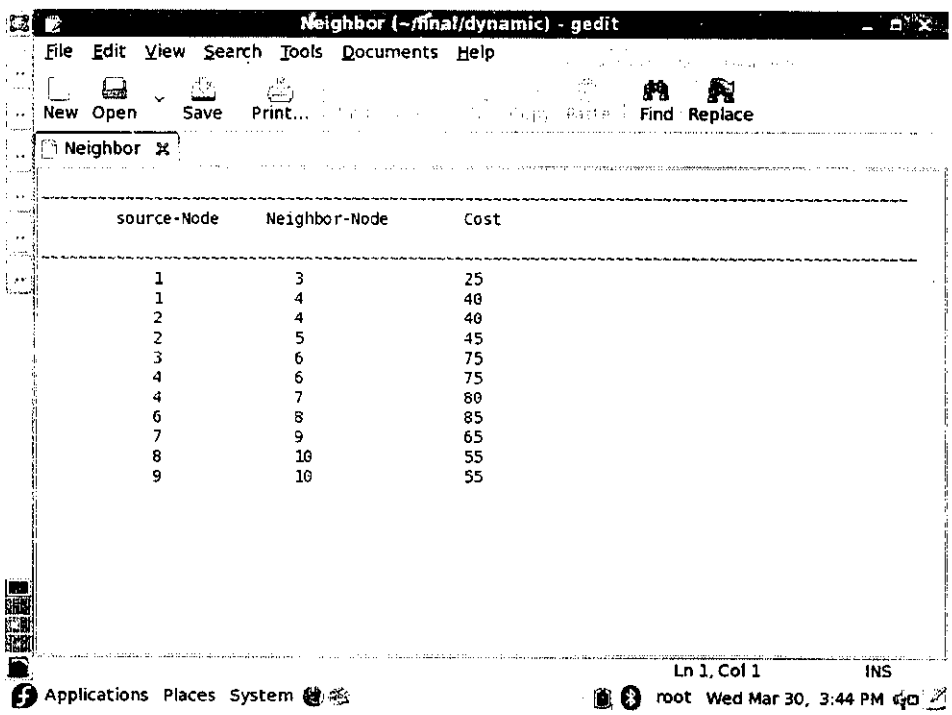
- **Requesting Node to Special Node**



- **Reply from Special Node to Responding Node**

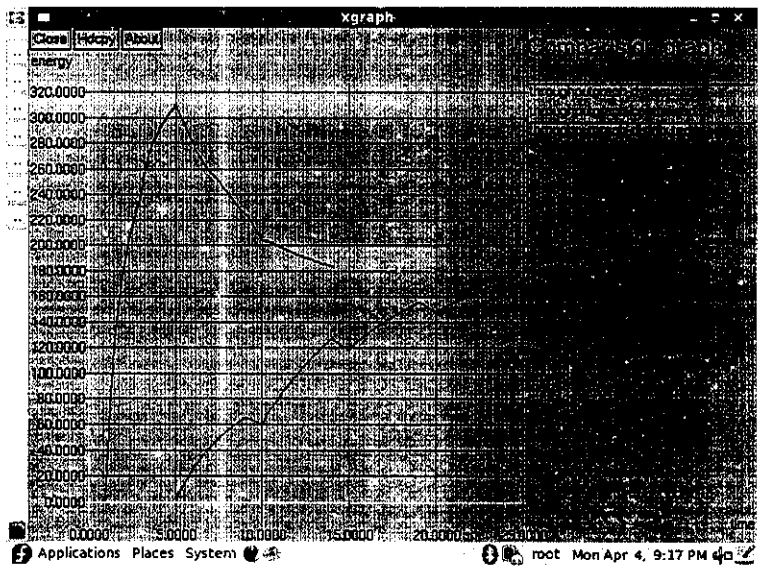


- **Neighbor**

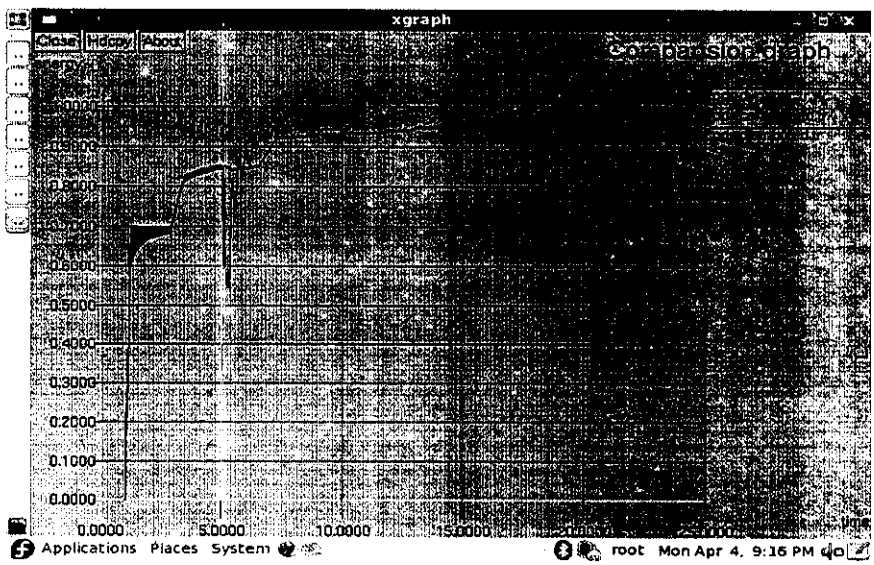


- **Comparison Graph for Dynamic Service, Centralized and Distributed approaches**

Throughput



Packet Drop Ratio Graph



REFERENCES

[1] Akyildiz I.F., Sankarasubramaniam W. Su, Y. and Cayirci E, "A Survey on Sensor Networks," vol. 38, No. 4, Pages 393–422, 2002.

[2]Mokhtar Aboelaze, Fadi Aloul, "Current and Future Trends in Sensor Networks: A Survey", IEEE, 2005.

[3]Mauri Kuorilehto, "A Survey of Application Distribution in Wireless Sensor Networks", 2005

[4].Zoran Bojkovic, Bojan Bakmaz , "A Survey on Wireless Sensor Networks Deployment", ISSN: 1109-2742,Issue 12, Volume 7, December 2008

[5] Golatowski F, Blumenthal J, Handy M, Haase M, "Service oriented software architecture for sensor networks". Intl. Workshop on Mobile Computing , Rostock, Germany, pp 93–98,2003.

[6] Stephan Sommer, Christian Buckl, Alois Knoll, "Applying the Service Oriented Paradigm to Develop Sensor/Actuator Networks",2008.

[7] Hachem Moussa , Tong Gao , I-Ling Yen Farokh Bastani , Jun-Jang Jeng , "Toward effective service composition for real-time SOA-based systems" ,SOCA 4:17–31,2010.

[8] Xiumin Wang, Jianping Wang, Zeyu Zheng, Yinlong Xu, Mei Yang," Service Composition in Service Oriented Wireless sensor Networks with Persistent Queries", Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference, 2009.

[9] Farshad A. Samimi and Philip K. McKinley, " Dynamis: Dynamic Overlay Service Composition for Distributed Stream Processing", The 20th International Conference on Software Engineering & Knowledge Engineering, 2008.
[10] Manish Jain, Puneet Sharma, Sujata Banrjee , "QoS-Guranteed Path Selection Algorithm for Service Composition", in IEEE communications, FEB 2006.

[11] Han song-qiao, Zhang shen-sheng, Zhan yong, CAO Jian, "A QoS Aware Service Composition Protocol in Mobile Ad Hoc Networks", 2008.

[12] Oussama Kassem Zein and Yvon Kermarrec, "An Approach For Dynamic Composition of Services in Distributed Systems", Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Service, 2006.

[13] Ning Xu, "A Survey of Sensor Network Applications", IEEE Communications Magazine, 2002.

[14] Edgardo Avilés-López ,J. Antonio García-Macías, "TinySOA: a service-oriented architecture for wireless sensor Networks" ,SOCA 3:99–108, 2009.

[15] Kostas Kontogiannis Grace A. Lewis Dennis B. Smith, "The Landscape of Service-Oriented Systems: A Research Perspective for Maintenance and Reengineering", IEEE International Conference on Software Maintenance (ICSM 2008).

[16] Jongwoo Sung, Taehong Kim, Seonghoon Kim, Kyubaek Kim, "Service Oriented Wireless Sensor Network Toolbox for Consumer Applications", 2008

[17] Rudolf Sollacher, Christoph Niedermeier, Norbert Vicari, "Towards a Service Oriented Architecture for Wireless Sensor Networks in Industrial Applications", 2010.

[18] Jer emie Leguay, Mario LopezRamos, "Service Oriented Architecture for Heterogeneous and Dynamic Sensor Networks", 2008.

[19] Raluca , LOMBRISER, Mihai , Paul and Gerhard TRÖSTER, "Towards Activity Recognition in Service-Oriented Body Area Networks",2008.

[20] Benjamin Le Corre, J'er'emie Leguay, Mario Lopez-Ramos, Vincent Gay, Vania Conan, "Service Oriented Tasking System for WSN", 2008.

[21] K. Sahina, , M. U. Gumusay, "Service Oriented Architecure (SOA) based Web services", The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. 2008.

[22]Kirill Mechitov , Reza Razavi , "Architecture Design Principles to Support Adaptive Service Orchestration in WSN Applications",2007.

[23] Tuanloc NGUYEN, Abbas jamalipour, guy pujolle, "Middleware architecture for Service Creation in Wireless Sensor Networks", Wireless And Mobile Computing, Networking And Communications, IEEE International Conference , 2005.

[24] Flavia C. Delicato, Paulo Pires , José Ferreira de Rezende, "Service-oriented Middleware for Wireless Sensor Networks",2005.

[25] Kavi Kumar Khedo and R.K. Subramanian, "A Service-Oriented Component-Based Middleware Architecture for Wireless Sensor Networks", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.3, March 2009.

[26] Kamlendu Pandey, S.V. Patel, "A Novel Design of Service Oriented and Message Driven Middleware for Ambient Aware Wireless Sensor Network", International Journal of Recent Trends in Engineering, Vol 1, No. 1, 2009

[27] Jin Xiao and Raouf Boutaba, " QoS-Aware service composition and adaptation in Autonomic Communication", IEEE JSAC – autonomic communication systems, July 22, 2005.

[28] Manish Jain, Puneet Sharma, Sujata Banrjee , "QoS-Guranteed Path Selection Algorithm for Service Composition", in IEEE communications, FEB 2006.

[29] Farhan Hassan Khan, M.Younus Javed, Saba Bashir , Aihab Khan, Malik Sikandar Hayat Khiyal , "QoS Based Dynamic Web Services Composition &Execution", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 2, February 2010.

[30] Rodrigo Manto vaneli Pessoa, Eduardo Silva, Marten van Sinderen , "Enterprise Interoperability with SOA: a Survey of Service Composition Approaches",2008.

[31] David Mennie, Bernard Pagurek, "An Architecture to Support Dynamic Composition of Service Components", Workshop on Component-Oriented Programming, 2001