CLUSTER BASED SERVICE DISCOVERY FOR HETEROGENEOUS WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

ARTHI.V (0810108002)
VISHNU PRIYA.M (0810108060)
KARTHIKA.K (0810108303)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE-641049.

ANNA UNIVERSITY OF TECHNOLOGY,
COIMBATORE-641047.

APRIL 2012

ANNA UNIVERSITY OF TECHNOLOGY COIMBATORE 641047

BONAFIDE CERTIFICATE

Certified that this project report entitled **"Cluster based service discovery for heterogeneous wireless sensor networks"** is the bonafide work of **Arthi.V**, **Vishnu Priya.M and Karthika.K** who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported here does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mrs.V.VANITHA	Prof.N.JAYAPATHI
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Associate Professor,	Professor,
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Kumaraguru College of Technology,	Kumaraguru College of Technology,
Coimbatore-641 049.	Coimbatore-641 049.

The candidates with University Register Nos. 0810108002, 0810108060 and 0810108303 were examined by us in the project viva-voce examination held on _______________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the project entitled **"Cluster based service discovery for heterogeneous wireless sensor networks"** 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 of the requirements for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University.

Place **:** Coimbatore

Date **:**

(Arthi.V)

(Vishnu Priya.M)

(Karthika.K)

ACKNOWLEDGEMENTS

We express our hearty gratitude to our beloved Chairman of our Institution, **Arutselvar Dr. N. Mahalingam, B.Sc., F.I.E.,** and Co-Chairman **Dr. B.K. Krishnaraj Vanavarayar,** for giving us this great opportunity to pursue this course.

We thank, **Dr.Ramachandran,** Principal, Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facilities to work on this project.

We would like to make a special acknowledgement and thanks to **Dr. S. Thangasamy,** Dean and **Prof.N.Jayapathi,** Head of Department of Computer Science and Engineering, for his support and encouragement throughout the project.

We express deep gratitude and gratefulness to our guide **Mrs.V.Vanitha, M.E.,** Associate Professor, Department of Computer Science and Engineering, for her supervision, enduring patience, active involvement and guidance and all **members of staff** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited. We also express our profound gratitude to our parents and friends for their moral support.

TABLE OF CONTENTS

ABSTRACT

Wireless Sensor Networks (WSN) has many potential applications in various fields such as military services, collaborative and distributed computing, emergency operations, wireless sensor networks and hybrid wireless networks. Dynamic Service Discovery and Advertisement (SDA) has also brought significant issues to the networking technologies. Accordingly, all devices should be able to advertise their own services and discover their required services dynamically. Automatic discovery of services and resources is a crucial feature to achieve the expected user friendliness in Wireless Sensor Networks (WSN). Due to limited computing power, scarce bandwidth, high mobility and the lack of a central coordinating entity, service discovery in these networks is a challenging task. We aim at providing an energy efficient service discovery protocol which can be achieved through gathering the nodes into clusters. Each cluster is assigned a cluster head. The communication of information and messages takes place only between the cluster heads. The cost of communication of information and overhead due to data traffic can be considerably reduced through cluster formation. The selection of cluster head can be calculated using a number of parameters such as energy and distance from base station. Our objective is to provide a directory for each cluster where the services provided by the nodes of that particular cluster can be registered.

LIST OF ABBREVIATIONS

Abbreviation	**Definition**
WSN	Wireless Sensor Network
SDP	Service Discovery Protocol
C4SD	Clustering for Service Discovery
DSR	Dynamic Source Routing
DSDV	Destination Sequenced Distance Vector Routing

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. WIRELESS SENSOR NETWORKS

Wireless Sensor Networks (WSNs) is an emerging technology that opens a wide perspective for future applications in ubiquitous computing and ambient intelligence. Wireless sensor nodes form a dense, large scale network and are expected to function unattended. Their hardware limitations have led to the design of new protocols at the lower communication levels, such as physical, MAC and routing. As the technology evolves, the focus changes toward sensor networks providing services to clients in a wide range of applications. We make the distinction between homogeneous and heterogeneous WSNs.

A homogeneous WSN is composed of tiny, resource-constrained devices, using the same platform and having the same hardware capabilities. The functionality of a homogeneous WSN serves mainly the purpose of gathering the sensed data and sending it to a central location. The typical research questions focus on prolonging the lifetime of the network, by designing energy-efficient protocols which distribute the communication overhead evenly among the sensor nodes. Fig. 1 shows a wireless sensor network.

A heterogeneous WSN employs a range of different devices, which are able to cooperate in order to achieve a global goal by combining the individual capabilities of the nodes. Small and cheap sensor nodes are

deployed with high density and easily attached to people or objects moving in the environment, while the more powerful nodes are able to provide persistent data storage, intensive processing and actuation.
In such a network, the objective is to distribute the workload depending on the capabilities of the nodes. We argue that heterogeneous WSNs have the potential to provide higher quality networking and system services than the homogeneous counterparts.

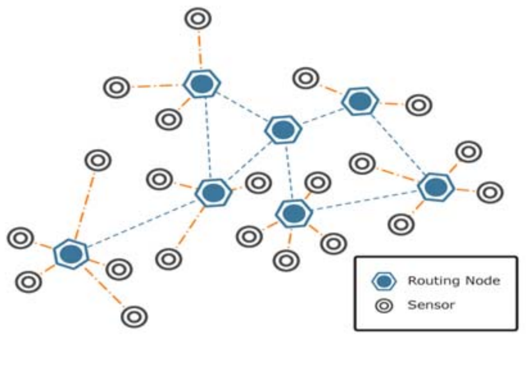

Figure 1. Wireless Sensor Network

1.2 Service Oriented WSN

Service-oriented architecture in WSN, helps in rapid development of new applications. In a service oriented WSN, a typical application may require several different services, e.g., temperature monitoring, intrusion

detection, environmental sensing, which are provided by service providers, that are the sensors. In a service oriented WSN, a typical application may require several different services, e.g., temperature monitoring, intrusion detection, environmental sensing, which are provided by service providers, that are the sensors.

1.2.1 Service Registry

A service registry allows us to organize information about services and provides facilities to publish and discover services.

1.2.2 Service Discovery

Service discovery generally refers to the detection of services offered by the nodes in a sensor network. It can also be defined as the process of finding services that match the
requirements of the service requestor.

1.3 EXISTING SYSTEM

The traditional method for service discovery in ad hoc networks is based on flooding, which has the advantage of zero maintenance overhead. However, flooding has obvious limitations with regard to energy-efficiency and scalability, and it is mostly suitable for homogeneous networks. The problem is how to design a service discovery protocol suitable for heterogeneous wireless sensor networks that reduces the workload of the resource-constraint devices and avoids the significant traffic induced by the traditional flood-based solutions in dense networks.

1.4 PROPOSED SYSTEM

We propose a solution based on clustering, where a set of nodes, selected based on their capabilities, acts as a distributed directory of service registrations for the nodes in their cluster. In this way,

(1) The communication costs are reduced, since the service discovery messages are exchanged only among the directory nodes, and

(2) The distribution of workload takes into account the capabilities of the nodes.

CHAPTER 2

LITERATURE SURVEY

2.1 Clustering Algorithms for Service Discovery and Provisioning

The evolution of computer technology follows a trajectory of miniaturization and diversification. The technology has developed from mainframes (large computers used by many people) to personal computers (one computer per person) and recently, embedded computers (many computers per person). One of the smallest embedded computers is a wireless sensor node, which is a battery powered miniaturized device equipped with processing capabilities, memory, wireless communication and sensors that can sense the physical parameters of the environment. A collection of sensor nodes that communicate through the wireless interface form a Wireless Sensor Network (WSN), which is an ad-hoc, self organizing network that can function unattended for long periods of time. Although traditionally WSNs have been regarded as static sensor arrays used mainly for environmental monitoring, recently, WSN applications have undergone a paradigm shift from static to more dynamic environments, where nodes are attached to moving objects, people or animals. Applications that use WSNs in motion are broad, ranging from transport and logistics to animal monitoring, health care and military, just to mention a few. These application domains have a number of characteristics that challenge the algorithmic design of WSNs. Firstly, mobility has a negative effect on the quality of the wireless communication and the performance of networking protocols. Nevertheless, it has been shown that mobility can enhance the functionality of the network by exploiting the movement patterns of mobile

objects. Secondly, the heterogeneity of devices in a WSN has to be taken into account for increasing the network performance and lifetime. Thirdly, the WSN services should ideally assist the user in an unobtrusive and transparent way. Fourthly, energy-efficiency and scalability are of primary importance to prevent the network performance degradation. This thesis focuses on the problems and enhancements brought in by network mobility, while also accounting for heterogeneity, transparency, energyefficiency and scalability. We propose a set of algorithms that enable WSNs to self-organize efficiently in the presence of mobility, adapt to and even exploit dynamics to increase the functionality of the network. Our contributions include an algorithm for motion detection, a set of clustering algorithms that can be used to handle mobility efficiently, and a service discovery protocol that enables dynamic user access to the WSN functionality. In short, the main contributions of this thesis are the classifications of service discovery protocols and clustering algorithms; a generalized clustering algorithm for wireless sensor networks; Cluster-based service discovery for wireless sensor networks.

2.2. A classification of clustering algorithms for wireless sensor networks

Clustering represents a method to handle heterogeneity in a resource-constraint environment, we are interested in a clustering solution that can be used to support service discovery in dynamic WSNs. Therefore, we present a survey and classification of clustering algorithms designed for wireless ad-hoc and sensor networks. Firstly, we introduce the clustering concept and the main reason for using it within this network environment. Secondly, we

define the main classification criteria and describe each of the categories in turn. Thirdly, we provide an overview of the different clustering solutions, pointing out their characteristics and explaining the main construction steps. Fourthly, we give a comparative table that summarizes the characteristics of the various clustering algorithms according to the classification.

2.3. Service Discovery Based on Past User Experience

Web service technology provides a way for simplifying inter-operability among different organizations. A piece of functionality avail-able as a web service can be involved in a new business process. Given the steadily growing number of available web services, it is hard for developers and services appropriate for their needs. The main research e®orts in this area are oriented on developing a mechanism for semantic web service description and matching. In this paper, we present an alternative approach for supporting users in web service discovery. Our system implements the implicit culture approach for recommending web services to developers based on the history of decisions made by other developers with similar needs. We explain the main ideas underlying our approach and report on experimental results.

2.4. A Survey of Service Discovery Protocols in Multi hop Mobile Ad Hoc Networks

Mobile ad hoc networks (manets) include a variety of devices, such as cell phones, PDAs, laptops, and other relatively larger devices. These devices can move at high or low speeds or even remain stationary, entering and leaving the system when switched on or off. Such a variety of devices

also offers a variety of services. A *service* is any tangible or intangible facility a device provides that can be useful for any other device. Services comprise those for software and hardware. A software
service, for example, can be a simple file, such as an MP3 file, or a software implementation of an algorithm, such as converting one audio file format to another. A hardware service, for example, can be a printer that a mobile device can use wirelessly. To benefit from these services, a device must be able to both locate them in the network and invoke them. *Service discovery protocols* (SDPs) enable these capabilities. SDPs play a role in wired networks, singlehop wireless networks, and multihop manets. In wired networks, the many industry standard SDPs include Sun's Jini, Microsoft's Universal Plug and Play, IBM's Salutation, and the Internet Engineering Task Force's Service Location Protocol. In single-hop wireless networks, SDPs include DEAPSpace and the industry standard Bluetooth SDP. Wired networks have stationary devices, whereas single-hop wireless ad hoc networks have restricted mobility with a low rate of devices joining and leaving. This survey focuses on multihop manet SDPs, one of the many approaches to realizing pervasive computing environments. In these networks, devices can have unrestricted mobility— that is, no constraints on joining or leaving the network—making service discovery difficult. (SDPs for wireless sensor networks, such as the ZigBee standard SDP, are out of our work's scope, because such SDPs focus on energy optimization rather than mobility.) Most SDPs in wired networks work at the application level. In theory, any such application level SDPs can work for manets as well. In practice, however, mobility and resource scarcity introduce two dimensions that SDPs for wired or single-hop wireless networks don't take into account:

location awareness and physical proximity between the service provider and the user. Thus, physical proximity remains undetectable.
In multihop manets, the service provider and user's physical proximity is a factor. Using bandwidth efficiently and reducing link breakages requires services that are physically close and at fewer hops. The Open Systems Interconnection architecture layering approach works well in wired networks without resource constraints, but it introduces inefficiencies in resource-constrained manets. Crosslayering can improve efficiency, for example, in a specially designed SDP that integrates service discovery with routing. Many proposals have focused on multihop manet SDPs, but none are as developed as those for wired or singlehop ad hoc networks. Several comparative studies cover SDPs in all three types of networks, such as those by Chunglae Cho and Duckki Lee;1 Feng Zhu, Matt Mutka, and Lionel Ni;2 and Raluca Marin- Perianu, Pieter Hartel, and Hans Scholten. 3 However, none closely analyzes only SDPs for multihop manets. We selected 12 multihop manet SDPs,4–15 identified the basic building blocks of SDPs in general, and analyzed the selected SDPs in the context of these building blocks.

CHAPTER 3

METHODOLOGY

3.1. BLOCK DIAGRAM FOR C4SD

The basic flow of the diagram in Figure 2 indicates the initial phase of creation of nodes followed by the formation of clusters. The Service Registry has three major processes involved which includes addition of new services followed by registering the specific service and then the Service Discovery has searching of service required with identification of the same.

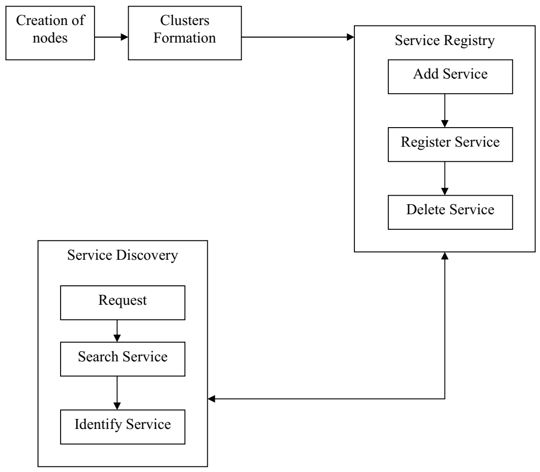

Figure 2. Block Diagram for C4SD

CHAPTER 4

IMPLEMENTATION

4.1 MODULE DESCRIPTION

4.1.1. CLUSTERING

Several clustering algorithms have been proposed to support scalable MAC and routing protocols in large ad-hoc and sensor networks. The election of the clusterheads usually involves (1) the dissemination of a set of parameters of each node either to the whole network or to group of nodes and (2) the comparison of these parameters in order to choose the best nodes as clusterheads.

- Firstly, electing the cluster heads based on information from nodes which are multiple hops away leads to high overhead and slow reaction to topology changes.
- Secondly, maintaining complete intra-cluster information regardless of the capabilities of the nodes is an expensive task for resource-lean devices.
- Thirdly, the complexity of the multi-layer clustering algorithms leads to a lot of effort in building and maintaining the desired structure.

To sum up, we are interested in a simple clustering solution, which can react quickly to topology changes and that requires a low construction and maintenance effort.

An algorithm which meets our conditions is DMAC , which constructs and maintains an independent dominating set. Nodes decide based on their one-hop neighbourhood information, which assures rapid reaction to topology changes. DMAC has also the advantage that the topology can change during the cluster formation. However, DMAC suffers from the chain reaction phenomenon, where a single topology change in the network may trigger significant changes in dominating set. For a distributed directory composed of nodes from the dominating set, the chain reaction causes additional overhead for maintaining consistent service registries. We compare the performance of our clustering algorithm with DMAC, when using them as structural basis for our service discovery protocol.

4.1.1.1. DESIGN CONSIDERATION

The clustering algorithm constructs an overlay network which facilitates the discovery of services in an energy-efficient fashion. We discuss from the design perspective several techniques for reducing the communication cost during

(1) Discovery of services and
(2) Maintenance of the distributed directory.

During the discovery process, messages are exchanged among the cluster head nodes. To minimize the discovery cost, the root nodes have to be sparsely distributed on the deployment area, as a high density of root nodes (or clusters)would lead to a high communication cost during discovery.

• Make decisions based on 1-hop neighbourhood information.

• Avoid chain reactions.

• Distribute the knowledge depending on the capabilities of the nodes.

4.1.1.2. CONSTRUCTION OF CLUSTERS

• Nodes that have the highest capability grades among their neighbours declare themselves clusterheads and broadcast a SetRoot message announcing their roles.

• The remaining nodes choose as parent the neighbour with the highest capability grade.

• When a node receives a SetRoot message from its parent, it learns the cluster membership and rebroadcasts the SetRoot message.

4.1.1.3. KNOWLEDGE ON ADJACENT CLUSTERS

The root nodes learn about the adjacent clusters from the nodes placed at the cluster borders. During the propagation of the broadcast message SetRoot down to the leaf nodes, the message is also received by nodes from adjacent clusters. These nodes store the adjacent root identity in their Ru(v) sets and report it to their parents. The information is propagated up in the tree with a message which we term UpdateInfo. Through this message, nodes learn the next hops for the paths leading to the clusters adjacent to their sub-trees. In particular, the root nodes learn the adjacent clusters and the next hops on the paths to reach their clusterheads.

The events of receiving messages SetRoot and UpdateInfo from Algorithm describe how the knowledge and the paths to adjacent clusters are updated for a given node v.

This means that if v is informed about a root node from one neighbour, but it knows already about this root through another neighbour, v does not propagate the information to the parent again. Fig. 3 shows the cluster head formation.

4.1.1.4. CLUSTER HEAD FORMATION

i. Let v be the cluster
ii. Let r be the cluster head
iii. Initially r(v) is selected randomly
iv. After every round, the energy and the distance of each node is calculated.
v. The node with the high energy value and the distance within the threshold value is chosen as the cluster head.

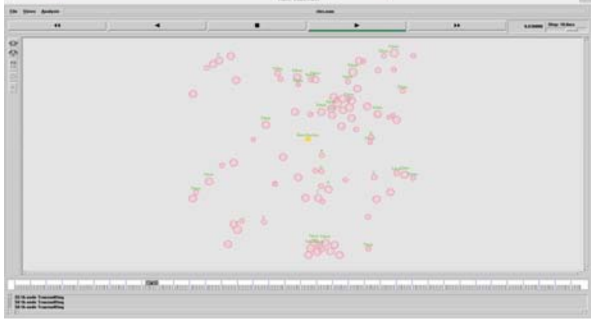

Figure 3. Cluster Head Formation

4.1.2. SERVICE REGISTRATION

During service registration, each node sends the local service descriptions and the descriptions received from its children to the parent node. In this way, a node learns the service descriptions of the nodes placed lower in hierarchy. In particular, a root node is informed of all the service descriptions offered by the nodes in its cluster. Since the registration process requires unicast messages to be transmitted from children to parents, it can be integrated with the transfer of knowledge on adjacent clusters. Thus, the message UpdateInfo is used for both service registrations and transferring the knowledge on adjacent clusters.

Algorithm shows the integrated version of the UpdateInfo message, where a node updates the information on both the adjacent clusters and the known services.

In the following we describe how the distributed service registry is kept consistent when the topology changes. In the case of a parent reselection, a child node v registers the services from its sub-tree with the new parent p1, and notifies the old parent p0 (if it is still reachable) to purge the outdated service information. The process is transparent to the other nodes in the sub tree rooted at v. If the overall service information at p0 and p1 changes due to the parent reselection, the modifications are propagated up in the hierarchy.

i. Each cluster maintains its own registry.
ii. The attributes of the nodes are stored in a structure.
iii. An array variable for the structure is created.

iv. The services of the nodes are registered in the registry thus created.

4.1.3 SERVICE DISCOVERY

The service discovery process uses a distributed directory of service registrations. Suppose a node in the network generates a service discovery request ServDisc. The request is first checked against the local registrations. In the case where no match is found, the message is forwarded to the parent. This process is repeated until the ServDisc message reaches the root of the cluster.

When a root node receives a ServDisc message and it does not find a match in the local registry, the message is forwarded to the roots of the adjacent clusters. Each node v along the path checks its Ru(v) sets and picks a neighbour that has a path to the root of the adjacent cluster. In the case where a link is deleted and v cannot forward the ServDisc message, it chooses another neighbour that provides a path to destination. If such a neighbour does not exist, v informs its parent that it no longer has a route to the next cluster.

The result of a service search is typically the address of one or more service providers. This response can be returned by the first node that finds a match in its registry for the requested service. However, in certain situations it may be preferable that the service provider itself issues a reply for the service request.

In these situations, the ServDisc message is forwarded down the cluster until it reaches the service provider. In the case where the link to the service provider is deleted or the service description is no longer valid, the

service request is sent back to the root node which forwards it to the adjacent clusters.

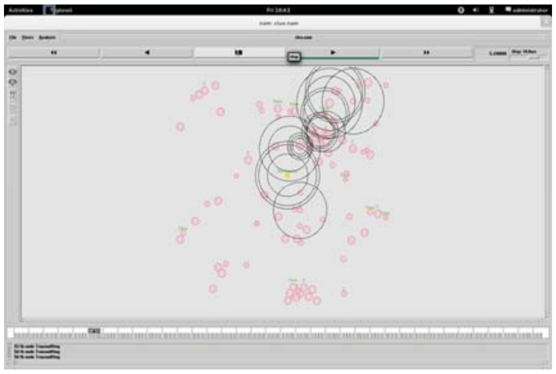

Figure 4. Data Transmission between clusters

The service discovery reply may follow the reverse cluster-path to the client, or any other path if a routing protocol is available. For the first case, if there is a cluster partition, the path can be reconstructed using the same search strategy as for the ServDisc message, where this time the service is the address of the client. Fig. 4 shows the transmission of data between several clusters.

Caching the service discovery messages is a technique that allows us to cope with mobility. Root nodes cache the ServDisc messages for a limited period of time. If a newly arrived node registers a service for which there is a match in the cache, the root node can respond to the old service request.

Moreover, when a root node is notified on a new adjacent cluster, it sends the valid service request entries from its cache to the new clusterhead. As a result, the overall hit ratio is improved.

Algorithm describes the protocol, where replies are generated by nodes in the distributed directory and no caching is implemented. The message ServDisc has four parameters: the neighbour u that sends the request, the service description s, and the final destination d of the message (typically a root node) and a flag f. The flag indicates whether the message is a fresh service discovery request, or it is a failure notification of a previous attempt to reach an adjacent cluster. In the latter case, the failed route is erased from the knowledge on adjacent clusters and another message is sent using an alternate path.

i. Service discovery is a function which allows automatic detection of services offered by the nodes on a network.
ii. A node initially requests for the service and it is searched in its local registry.
iii. If the service is not found then the request is forwarded

CHAPTER 5

RESULTS

5.1 GRAPHS

The graph in Figure 5 indicates the probability of retrieving the required services. It is calculated by the number of services retrieved divided by the total number of services provided by the fixed nodes.

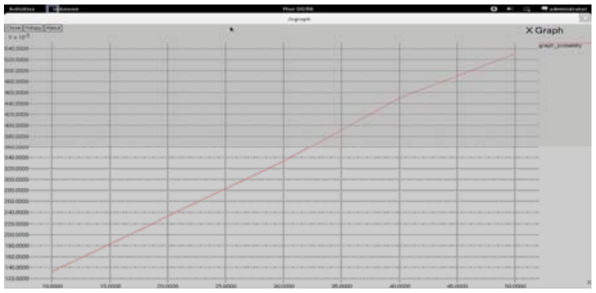

Figure 5. Probability of servicing the requests

The graph in Figure 6 demonstrates the time taken to service fixed number of services by increasing the number of nodes. It is found that the time consumption increases with the increase in the number of nodes.

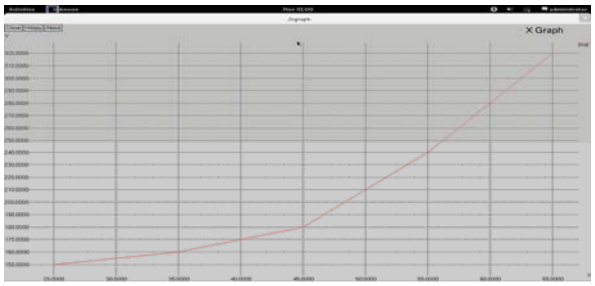

Figure 6. Time taken for servicing the fixed number of services by increasing the no. of nodes

The graph in Figure 7 denotes the precision of service discovery. It is calculated by the number of total services divided by the number of relevant services.

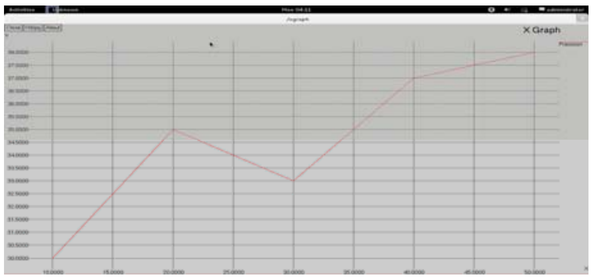

Figure 7. Precision for service discovery.

CHAPTER 6

CONCLUSION

Thus service discovery protocol for heterogeneous WSNs relies on a clustering structure that offers distributed storage of service descriptions. The clusterheads act as directories for the services in their clusters. The structure ensures low construction and maintenance overhead, reacts quickly to topology changes and avoids the chain-reaction problems. A service lookup results in visiting only the directory nodes, which ensures a low discovery cost. The evaluation of the proposed discovery solution addresses a number of questions regarding the performance and the tradeoffs implied by the clustering approaches.

Firstly, we focus on the performances of the service discovery protocol depending on the underlying clustering structure, by comparing our clustering algorithm with DMAC. We show that the chain reaction of DMAC determines reclustering and re-registration of services with new clusterheads, implying higher maintenance overhead. The smaller-height clusters of DMAC leads to faster convergence and higher hit ratio. The hit ratio is improved to more than 98% for both protocols if a mechanism of limited-time caching is implemented for service discovery messages. Due to the lower cluster density, our protocol has a lower discovery cost in both implementation alternatives.

Secondly, we compare the performance of our solution with the traditional flooding approach. The cost of service discovery when using a distributed directory is much lower than the flooding alternative, as the

service discovery messages visit only the nodes which are part of the distributed directory.

However, flooding has the advantage of zero-maintenance cost. Therefore, in dynamic and sparse networks, when service requests are infrequent, flooding is more energy efficient than cluster-based protocols, which spend a lot of effort in maintaining the consistency of service registries. However, flooding does not distinguish among nodes with different capabilities. We show that for the proposed cluster based solutions, the resource-lean devices experience a low overhead, while the more powerful nodes consume more energy both during maintenance and discovery of services.

Thirdly, we investigate the limit in the network heterogeneity where clustering is still feasible for implementation in a WSN. We model the resources of the nodes according to a Pareto distribution, where the number of resource constraint nodes is high in comparison to the more powerful devices. We notice that for a high Pareto index, the availability of resources is the main factor that influences the clustering structure. The performance of the service discovery protocol decreases while increasing the Pareto index, such that for rather homogeneous networks, flooding is more efficient.

For future work we plan to further test the feasibility of our solution by implementing the clustering and service discovery protocols on sensor nodes.

APPENDIX

TCL FILE:

```
Mac/802_11 set bandwidth_ 2Mb
set MESSAGE_PORT 42
set BROADCAST_ADDR -1


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

# 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)            CMUPriQueue                ;# 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)             90                        ;# number of
mobilenodes
set val(rp)           TEEN                      ;# routing
protocol
set val(x)       1500                ;# X axis distance
set val(y)       1500                ;# Y axis distance
set opt(energymodel)    EnergyModel                ;# Initial
Energy
set opt(radiomodel)     RadioModel                 ;#
Transmission Model
set opt(initialenergy)  100                        ;# Initial
energy in Joules

# Creating Simulator Object
set ns [new Simulator]

# Creating NAM File
set namTracefile [open clus.nam w]
$ns namtrace-all-wireless $namTracefile $val(x) $val(y)

# Creating Multiple Trace
```

```
set traceFile [open clus.tr w]
$ns trace-all $traceFile


# Creating Topology
set topo [new Topography]
set val(rp) DSR
$topo load_flatgrid $val(x) $val(y)

# Creating GOD(General Operation Director) Object
set god_ [create-god $val(nn)]

# Parameters
Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.0818
Phy/WirelessPhy set freq_ 914e+6

Mac/802_11      set dataRate_  2.0e6



# Configuring the Nodes in the Topology.
$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 \
           -energyModel $opt(energymodel) \
           -initialEnergy $opt(initialenergy) \
           -macTrace ON \
           -movementTrace ON \
           -idlePower 0.0 \
           -rxPower 0.5 \
           -txPower 1.0 \
           -sleepPower 0.1 \
           -transitionTime 0.003 \
                -channelType $val(chan) \
           -maxSenRange 20 \
           -maxTransRange 40 \



 #Creating Group-1 Nodes
for { set i 0 } { $i<=89 } { incr i } {
set node($i) [$ns node]
}
```

```
# Parameters to the Node
for { set i 0 } { $i<=19} { incr i } {
#$node($i) random-motion 1
#$node($i) set X_ 0.0
#$node($i) set Y_ 0.0
#$node($i) set Z_ 0.0
$node($i) color black
$ns initial_node_pos $node($i) 30
}

for { set i 20 } { $i<=39} { incr i } {
#$node($i) random-motion 1
#$node($i) set X_ 0.0
#$node($i) set Y_ 0.0
#$node($i) set Z_ 0.0
$node($i) color black
$ns initial_node_pos $node($i) 35
}
for { set i 40 } { $i<=89} { incr i } {
#$node($i) random-motion 1
#$node($i) set X_ 0.0
#$node($i) set Y_ 0.0
#$node($i) set Z_ 0.0
$node($i) color black
$ns initial_node_pos $node($i) 50
}
#source "./nodes90"

set c1 pink2



set rate 1000
set ll T

$ns at 0.0 "$node(0) setdest 782.467 861.396 1500"
$ns at 0.0 "$node(0) color gold"
$ns at 0.0 "$node(0) label Base-Station"
$ns at 0.0 "$node(1) setdest 26.849427280175 469.216197286027
1500"
$ns at 0.0 "$node(1) color $c1"
$ns at 0.0 "$node(2) setdest 1194.700496270718 287.237329582070
1500"
$ns at 0.0 "$node(2) color $c1"
$ns at 0.0 "$node(3) setdest 880.408858094607 742.668039858097
1500"
$ns at 0.0 "$node(3) color $c1"
$ns at 0.0 "$node(4) setdest 98.735652290521 1383.147766626146
1500"
$ns at 0.0 "$node(4) color $c1"
```

```
$ns at 0.0 "$node(5) setdest 1106.298373222831 931.357825575962
1500"
$ns at 0.0 "$node(5) color $c1"
$ns at 0.0 "$node(6) setdest 1061.678392190744 1159.462065242257
1500"
$ns at 0.0 "$node(6) color $c1"
$ns at 0.0 "$node(7) setdest 1328.661111966127 1019.871451340910
1500"
$ns at 0.0 "$node(7) color $c1"
$ns at 0.0 "$node(8) setdest 1358.815127781952 1036.973710681901
1500"
$ns at 0.0 "$node(8) color $c1"
$ns at 0.0 "$node(9) setdest 1419.116781134226 1209.639476403830
1500"
$ns at 0.0 "$node(9) color $c1"
$ns at 0.0 "$node(10) setdest 872.181439635749 617.629595209601
1500"
$ns at 0.0 "$node(10) color $c1"
$ns at 0.0 "$node(11) setdest 558.406507048885 524.011784813938
1500"
$ns at 0.0 "$node(11) color $c1"
$ns at 0.0 "$node(12) setdest 595.387613014463 1297.992909672327
1500"
$ns at 0.0 "$node(12) color $c1"
$ns at 0.0 "$node(13) setdest 1205.186194432603 869.961984910093
1500"
$ns at 0.0 "$node(13) color $c1"
$ns at 0.0 "$node(14) setdest 410.667111612568 855.388189298274
1500"
$ns at 0.0 "$node(14) color $c1"
$ns at 0.0 "$node(15) setdest 1185.895398763531 50.784929786638
1500"
$ns at 0.0 "$node(15) color $c1"
$ns at 0.0 "$node(16) setdest 718.534734555340 1265.082938977809
1500"
$ns at 0.0 "$node(16) color $c1"
$ns at 0.0 "$node(17) setdest 970.646294973425 1144.024896541885
1500"
$ns at 0.0 "$node(17) color $c1"
$ns at 0.0 "$node(18) setdest 1201.303062814064 835.723326734838
1500"
$ns at 0.0 "$node(18) color $c1"
$ns at 0.0 "$node(19) setdest 1493.297579140844 569.083545559407
1500"
$ns at 0.0 "$node(19) color $c1"

$ns at 0.0 "$node(20) setdest 344.253814221545 263.455564807811
1500"
$ns at 0.0 "$node(20) color $c1"
$ns at 0.0 "$node(21) setdest 1038.896331118987 559.282638093175
1500"
$ns at 0.0 "$node(21) color $c1"
```

```
$ns at 0.0 "$node(22) setdest 1151.637192941396 1431.170925889992
1500"
$ns at 0.0 "$node(22) color $c1"
$ns at 0.0 "$node(23) setdest 1293.980460648410 1470.671374327421
1500"
$ns at 0.0 "$node(23) color $c1"
$ns at 0.0 "$node(24) setdest 832.565698114612 627.406133688166
1500"
$ns at 0.0 "$node(24) color $c1"
$ns at 0.0 "$node(25) setdest 795.312906015917 1036.180298199296
1500"
$ns at 0.0 "$node(25) color $c1"
$ns at 0.0 "$node(26) setdest 580.560331176163 1346.929273139986
1500"
$ns at 0.0 "$node(26) color $c1"
$ns at 0.0 "$node(27) setdest 886.370390622188 403.914046593505
1500"
$ns at 0.0 "$node(27) color $c1"
$ns at 0.0 "$node(28) setdest 1230.679057766973 575.759407309340
1500"
$ns at 0.0 "$node(28) color $c1"
$ns at 0.0 "$node(29) setdest 1048.389420510503 1277.816264491012
1500"
$ns at 0.0 "$node(29) color $c1"

$ns at 0.0 "$node(30) setdest 556.622387789529 527.772436564944
1500"
$ns at 0.0 "$node(30) color $c1"
$ns at 0.0 "$node(31) setdest 489.153899192813 249.995116144182
1500"
$ns at 0.0 "$node(31) color $c1"
$ns at 0.0 "$node(32) setdest 852.356874218730 67.590268924547
1500"
$ns at 0.0 "$node(32) color $c1"
$ns at 0.0 "$node(33) setdest 1258.111857154236 1365.542485851990
1500"
$ns at 0.0 "$node(33) color $c1"
$ns at 0.0 "$node(34) setdest 1356.913681779533 1365.073194595805
1500"
$ns at 0.0 "$node(34) color $c1"
$ns at 0.0 "$node(35) setdest 202.620222034167 675.021411712387
1500"
$ns at 0.0 "$node(35) color $c1"
$ns at 0.0 "$node(36) setdest 1485.343246593000 1465.204157927647
1500"
$ns at 0.0 "$node(36) color $c1"
$ns at 0.0 "$node(37) setdest 798.967989786841 1291.350201494257
1500"
$ns at 0.0 "$node(37) color $c1"
$ns at 0.0 "$node(38) setdest 877.499908679161 516.992923695251
1500"
$ns at 0.0 "$node(38) color $c1"
```

```
$ns at 0.0 "$node(39) setdest 1378.365806466921 603.748562531476
1500"
$ns at 0.0 "$node(39) color $c1"

$ns at 0.0 "$node(40) setdest 852.392424665770 424.036005092666
1500"
$ns at 0.0 "$node(40) color $c1"
$ns at 0.0 "$node(41) setdest 1128.195032864941 1435.194244833012
1500"
$ns at 0.0 "$node(41) color $c1"
$ns at 0.0 "$node(42) setdest 4.887129649642 1192.592094084051
1500"
$ns at 0.0 "$node(42) color $c1"
$ns at 0.0 "$node(43) setdest 284.671215389765 234.278964165138
1500"
$ns at 0.0 "$node(43) color $c1"
$ns at 0.0 "$node(44) setdest 1254.562343458870 1042.733765297715
1500"
$ns at 0.0 "$node(44) color $c1"
$ns at 0.0 "$node(45) setdest 114.210389730465 545.455723175497
1500"
$ns at 0.0 "$node(45) color $c1"
$ns at 0.0 "$node(46) setdest 253.764686939087 87.777004171539
1500"
$ns at 0.0 "$node(46) color $c1"
$ns at 0.0 "$node(47) setdest 1183.316428128122 1099.647861492530
1500"
$ns at 0.0 "$node(47) color $c1"
$ns at 0.0 "$node(48) setdest 179.705491494095 1433.776783925847
1500"
$ns at 0.0 "$node(48) color $c1"
$ns at 0.0 "$node(49) setdest 707.507103020669 1308.900409320203
1500"
$ns at 0.0 "$node(49) color $c1"

$ns at 0.0 "$node(50) setdest 615.184353305573 726.153999110691
1500"
$ns at 0.0 "$node(50) color $c1"
$ns at 0.0 "$node(51) setdest 1366.892724254561 1491.980859272187
1500"
$ns at 0.0 "$node(51) color $c1"
$ns at 0.0 "$node(52) setdest 1427.679607401918 591.621711048604
1500"
$ns at 0.0 "$node(52) color $c1"
$ns at 0.0 "$node(53) setdest 309.174407112477 194.693435913938
1500"
$ns at 0.0 "$node(53) color $c1"
$ns at 0.0 "$node(54) setdest 693.185313530859 1057.686952757390
1500"
$ns at 0.0 "$node(54) color $c1"
$ns at 0.0 "$node(55) setdest 279.757976110480 683.356973840448
1500"
$ns at 0.0 "$node(55) color $c1"
```

```
$ns at 0.0 "$node(56) setdest 1381.470289976760 1002.665805759488
1500"
$ns at 0.0 "$node(56) color $c1"
$ns at 0.0 "$node(57) setdest 143.258846085016 1411.812079292003
1500"
$ns at 0.0 "$node(57) color $c1"
$ns at 0.0 "$node(58) setdest 772.979975976012 426.683016179611
1500"
$ns at 0.0 "$node(58) color $c1"
$ns at 0.0 "$node(59) setdest 260.695926082496 1471.261068136036
1500"
$ns at 0.0 "$node(59) color $c1"

$ns at 0.0 "$node(60) setdest 1334.476832200583 466.509587022940
1500"
$ns at 0.0 "$node(60) color $c1"
$ns at 0.0 "$node(61) setdest 829.420437342359 1291.064114901548
1500"
$ns at 0.0 "$node(61) color $c1"
$ns at 0.0 "$node(62) setdest 499.804534008574 966.100982145640
1500"
$ns at 0.0 "$node(62) color $c1"
$ns at 0.0 "$node(63) setdest 1081.642536209349 1345.606737277983
1500"
$ns at 0.0 "$node(63) color $c1"
$ns at 0.0 "$node(64) setdest 1006.537580247926 953.082355144697
1500"
$ns at 0.0 "$node(64) color $c1"
$ns at 0.0 "$node(65) setdest 4.240990006860 423.595228723206
1500"
$ns at 0.0 "$node(65) color $c1"
$ns at 0.0 "$node(66) setdest 1242.610293431892 414.648889155818
1500"
$ns at 0.0 "$node(66) color $c1"
$ns at 0.0 "$node(67) setdest 917.831821345961 485.912084828530
1500"
$ns at 0.0 "$node(67) color $c1"
$ns at 0.0 "$node(68) setdest 1111.917648180125 1225.085119208718
1500"
$ns at 0.0 "$node(68) color $c1"
$ns at 0.0 "$node(69) setdest 295.815 1295.05 1500"
$ns at 0.0 "$node(69) color $c1"


$ns at 0.0 "$node(70) setdest 942.761 1086.63 1500"
$ns at 0.0 "$node(70) color $c1"
$ns at 0.0 "$node(71) setdest 978.720 1121.16 1500"
$ns at 0.0 "$node(71) color $c1"
$ns at 0.0 "$node(72) setdest 1014.7 1067.93 1500"
$ns at 0.0 "$node(72) color $c1"
$ns at 0.0 "$node(73) setdest 1062.17 1092.38 1500"
$ns at 0.0 "$node(73) color $c1"
$ns at 0.0 "$node(74) setdest 1013.26 1148.49 1500"
```

```
$ns at 0.0 "$node(74) color $c1"
$ns at 0.0 "$node(75) setdest 954.27 1167.19 1500"
$ns at 0.0 "$node(75) color $c1"
$ns at 0.0 "$node(76) setdest 947.078 1037.71 1500"
$ns at 0.0 "$node(76) color $c1"
$ns at 0.0 "$node(77) setdest 1073.68 1138.42 1500"
$ns at 0.0 "$node(77) color $c1"
$ns at 0.0 "$node(78) setdest 872.267 1059.3 1500"
$ns at 0.0 "$node(78) color $c1"
$ns at 0.0 "$node(79) setdest 1076.56 1013.26 1500"
$ns at 0.0 "$node(79) color $c1"

$ns at 0.0 "$node(80) setdest 909.56 93.4078 1500"
$ns at 0.0 "$node(80) color $c1"
$ns at 0.0 "$node(81) setdest 944.126 32.2906 1500"
$ns at 0.0 "$node(81) color $c1"
$ns at 0.0 "$node(82) setdest 783.075 6.1802 1500"
$ns at 0.0 "$node(82) color $c1"
$ns at 0.0 "$node(83) setdest 823.355 95.3758 1500"
$ns at 0.0 "$node(83) color $c1"
$ns at 0.0 "$node(84) setdest 843.497 27.7571 1500"
$ns at 0.0 "$node(84) color $c1"
$ns at 0.0 "$node(85) setdest 964.349 78.111 1500"
$ns at 0.0 "$node(85) color $c1"
$ns at 0.0 "$node(86) setdest 799.155 54.0489 1500"
$ns at 0.0 "$node(86) color $c1"
$ns at 0.0 "$node(87) setdest 1000.1 6.0593 1500"
$ns at 0.0 "$node(87) color $c1"
$ns at 0.0 "$node(88) setdest 880.904 55.0916 1500"
$ns at 0.0 "$node(88) color $c1"
$ns at 0.0 "$node(89) setdest 717.615 577.2158 1500"
$ns at 0.0 "$node(89) color $c1"



set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]
set sink6 [new Agent/LossMonitor]
set sink7 [new Agent/LossMonitor]
set sink8 [new Agent/LossMonitor]
set sink9 [new Agent/LossMonitor]
set sink10 [new Agent/LossMonitor]
set sink11 [new Agent/LossMonitor]
set sink12 [new Agent/LossMonitor]
set sink13 [new Agent/LossMonitor]
set sink14 [new Agent/LossMonitor]
set sink15 [new Agent/LossMonitor]
set sink16 [new Agent/LossMonitor]
set sink17 [new Agent/LossMonitor]
```

```
set sink18 [new Agent/LossMonitor]
set sink19 [new Agent/LossMonitor]
set sink20 [new Agent/LossMonitor]
set sink21 [new Agent/LossMonitor]
set sink22 [new Agent/LossMonitor]
set sink23 [new Agent/LossMonitor]
set sink24 [new Agent/LossMonitor]
set sink25 [new Agent/LossMonitor]
set sink26 [new Agent/LossMonitor]
set sink27 [new Agent/LossMonitor]
set sink28 [new Agent/LossMonitor]
set sink29 [new Agent/LossMonitor]
set sink30 [new Agent/LossMonitor]
set sink31 [new Agent/LossMonitor]
set sink32 [new Agent/LossMonitor]
set sink33 [new Agent/LossMonitor]
set sink34 [new Agent/LossMonitor]
set sink35 [new Agent/LossMonitor]
set sink36 [new Agent/LossMonitor]
set sink37 [new Agent/LossMonitor]
set sink38 [new Agent/LossMonitor]
set sink39 [new Agent/LossMonitor]
set sink40 [new Agent/LossMonitor]
set sink41 [new Agent/LossMonitor]
set sink42 [new Agent/LossMonitor]
set sink43 [new Agent/LossMonitor]
set sink44 [new Agent/LossMonitor]
set sink45 [new Agent/LossMonitor]
set sink46 [new Agent/LossMonitor]
set sink47 [new Agent/LossMonitor]
set sink48 [new Agent/LossMonitor]
set sink49 [new Agent/LossMonitor]
set sink50 [new Agent/LossMonitor]
set sink51 [new Agent/LossMonitor]
set sink52 [new Agent/LossMonitor]
set sink53 [new Agent/LossMonitor]
set sink54 [new Agent/LossMonitor]
set sink55 [new Agent/LossMonitor]
set sink56 [new Agent/LossMonitor]
set sink57 [new Agent/LossMonitor]
set sink58 [new Agent/LossMonitor]
set sink59 [new Agent/LossMonitor]
set sink60 [new Agent/LossMonitor]
set sink61 [new Agent/LossMonitor]
set sink62 [new Agent/LossMonitor]
set sink63 [new Agent/LossMonitor]
set sink64 [new Agent/LossMonitor]
set sink65 [new Agent/LossMonitor]
set sink66 [new Agent/LossMonitor]
set sink67 [new Agent/LossMonitor]
set sink68 [new Agent/LossMonitor]
set sink69 [new Agent/LossMonitor]
set sink70 [new Agent/LossMonitor]
set sink71 [new Agent/LossMonitor]
set sink72 [new Agent/LossMonitor]
set sink73 [new Agent/LossMonitor]
set sink74 [new Agent/LossMonitor]
set sink75 [new Agent/LossMonitor]
set sink76 [new Agent/LossMonitor]
set sink77 [new Agent/LossMonitor]
set sink78 [new Agent/LossMonitor]
set sink79 [new Agent/LossMonitor]
set sink80 [new Agent/LossMonitor]
set sink81 [new Agent/LossMonitor]
set sink82 [new Agent/LossMonitor]
set sink83 [new Agent/LossMonitor]
set sink84 [new Agent/LossMonitor]
set sink85 [new Agent/LossMonitor]
set sink86 [new Agent/LossMonitor]
set sink87 [new Agent/LossMonitor]
set sink88 [new Agent/LossMonitor]
set sink89 [new Agent/LossMonitor]


# Attatching the Sink with Node

$ns attach-agent $node(0) $sink0
$ns attach-agent $node(1) $sink1
$ns attach-agent $node(2) $sink2
$ns attach-agent $node(3) $sink3
$ns attach-agent $node(4) $sink4
$ns attach-agent $node(5) $sink5
$ns attach-agent $node(6) $sink6
$ns attach-agent $node(7) $sink7
$ns attach-agent $node(8) $sink8
$ns attach-agent $node(9) $sink9
$ns attach-agent $node(10) $sink10
$ns attach-agent $node(11) $sink11
$ns attach-agent $node(12) $sink12
$ns attach-agent $node(13) $sink13
$ns attach-agent $node(14) $sink14
$ns attach-agent $node(15) $sink15
$ns attach-agent $node(16) $sink16
$ns attach-agent $node(17) $sink17
$ns attach-agent $node(18) $sink18
$ns attach-agent $node(19) $sink19
$ns attach-agent $node(20) $sink20
$ns attach-agent $node(21) $sink21
$ns attach-agent $node(22) $sink22
$ns attach-agent $node(23) $sink23
$ns attach-agent $node(24) $sink24
$ns attach-agent $node(25) $sink25
$ns attach-agent $node(26) $sink26
$ns attach-agent $node(27) $sink27
$ns attach-agent $node(28) $sink28
$ns attach-agent $node(29) $sink29
```

```
$ns attach-agent $node(30) $sink30
$ns attach-agent $node(31) $sink31
$ns attach-agent $node(32) $sink32
$ns attach-agent $node(33) $sink33
$ns attach-agent $node(34) $sink34
$ns attach-agent $node(35) $sink35
$ns attach-agent $node(36) $sink36
$ns attach-agent $node(37) $sink37
$ns attach-agent $node(38) $sink38
$ns attach-agent $node(39) $sink39
$ns attach-agent $node(40) $sink40
$ns attach-agent $node(41) $sink41
$ns attach-agent $node(42) $sink42
$ns attach-agent $node(43) $sink43
$ns attach-agent $node(44) $sink44
$ns attach-agent $node(45) $sink45
$ns attach-agent $node(46) $sink46
$ns attach-agent $node(47) $sink47
$ns attach-agent $node(48) $sink48
$ns attach-agent $node(49) $sink49
$ns attach-agent $node(50) $sink50
$ns attach-agent $node(51) $sink51
$ns attach-agent $node(52) $sink52
$ns attach-agent $node(53) $sink53
$ns attach-agent $node(54) $sink54
$ns attach-agent $node(55) $sink55
$ns attach-agent $node(56) $sink56
$ns attach-agent $node(57) $sink57
$ns attach-agent $node(58) $sink58
$ns attach-agent $node(59) $sink59
$ns attach-agent $node(60) $sink60
$ns attach-agent $node(61) $sink61
$ns attach-agent $node(62) $sink62
$ns attach-agent $node(63) $sink63
$ns attach-agent $node(64) $sink64
$ns attach-agent $node(65) $sink65
$ns attach-agent $node(66) $sink66
$ns attach-agent $node(67) $sink67
$ns attach-agent $node(68) $sink68
$ns attach-agent $node(69) $sink69
$ns attach-agent $node(70) $sink70
$ns attach-agent $node(71) $sink71
$ns attach-agent $node(72) $sink72
$ns attach-agent $node(73) $sink73
$ns attach-agent $node(74) $sink74
$ns attach-agent $node(75) $sink75
$ns attach-agent $node(76) $sink76
$ns attach-agent $node(77) $sink77
$ns attach-agent $node(78) $sink78
$ns attach-agent $node(79) $sink79
$ns attach-agent $node(80) $sink80
$ns attach-agent $node(81) $sink81
$ns attach-agent $node(82) $sink82
```

```
$ns attach-agent $node(83) $sink83
$ns attach-agent $node(84) $sink84
$ns attach-agent $node(85) $sink85
$ns attach-agent $node(86) $sink86
$ns attach-agent $node(87) $sink87
$ns attach-agent $node(88) $sink88
$ns attach-agent $node(89) $sink89

proc distance { n1 n2 nd1 nd2} {
global r
set dis [open E-Distance.txt a]
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 {$d<=300} {
puts $dis "\t$nd1\t\t$nd2\t\t$d\t\tclose"

}
if {$d>=500} {
puts $dis "\t$nd1\t\t$nd2\t\t$d\t\tfar"
}
if {$d>300 && $d<500} {
puts $dis "\t$nd1\t\t$nd2\t\t$d\t\tmedium"
}
}
if {$d<250} {
if {$nd2!=$nd1} {
puts $nbr "\t$nd1\t\t$nd2\t\t$d"
}
}
close $dis
close $nbr
}



proc energy {stnode etnode stime etime} {
set etp [open etmp w]
puts $etp "$stnode $etnode $stime $etime"
close $etp

}


for {set j 0} {$j<=89} {incr j} {
$ns at 8.5 "distance $node(0) $node($j) 0 $j"
}
$ns at 12.0 "energy 0 89 0 12.0"
```

```
$ns at 5.1 "$node(1) color $c1"
$ns at 5.1 "$node(1) label C "

$ns at 5.1 "$node(48) label C "

$ns at 5.1 "$node(10) label C "
$ns at 5.1 "$node(19) color $c1"
$ns at 5.1 "$node(19) label C "
$ns at 5.1 "$node(29) color $c1"
$ns at 5.1 "$node(29) label C "
$ns at 5.1 "$node(39) color $c1"
$ns at 5.1 "$node(39) label C "
$ns at 5.1 "$node(61) color $c1"
$ns at 5.1 "$node(61) label C "
$ns at 5.1 "$node(76) color $c1"
$ns at 5.1 "$node(76) label C "
$ns at 5.1 "$node(83) color $c1"
$ns at 5.1 "$node(83) label C "
$ns at 5.1 "$node(18) color $c1"
$ns at 5.1 "$node(18) label C "
$ns at 5.1 "$node(49) color $c1"
$ns at 5.1 "$node(49) label C "


$ns at 5.38 "$node(61) label Taken"
$ns at 5.38 "$node(76) label Taken"
$ns at 5.38 "$node(18) label Taken"
$ns at 5.38 "$node(49) label Taken"
$ns at 5.38 "$node(1) label Taken"
$ns at 5.38 "$node(19) label Taken"
$ns at 5.38 "$node(29) label Taken"
$ns at 5.38 "$node(39) label Taken"
$ns at 5.38 "$node(83) label Taken"

$ns at 5.38 "$node(62) label C "
$ns at 5.38 "$node(62) color $c1"
$ns at 5.38 "$node(78) label C "
$ns at 5.38 "$node(78) color $c1"
$ns at 5.38 "$node(15) label C "
$ns at 5.38 "$node(15) color $c1"
$ns at 5.38 "$node(52) label C "
$ns at 5.38 "$node(52) color $c1"
$ns at 5.38 "$node(6) label C "
$ns at 5.38 "$node(6) color $c1"
$ns at 5.38 "$node(23) label C "
$ns at 5.38 "$node(23) color $c1"
$ns at 5.38 "$node(32) label C "
$ns at 5.38 "$node(32) color $c1"
$ns at 5.38 "$node(44) label C "
$ns at 5.38 "$node(44) color $c1"
$ns at 5.38 "$node(80) label C "
$ns at 5.38 "$node(80) color $c1"
```

```
$ns at 5.53 "$node(62) label Taken"
$ns at 5.53 "$node(78) label Taken"
$ns at 5.53 "$node(15) label Taken"
$ns at 5.53 "$node(52) label Taken"
$ns at 5.53 "$node(6) label Taken"
$ns at 5.53 "$node(23) label Taken"
$ns at 5.53 "$node(32) label Taken"
$ns at 5.53 "$node(44) label Taken"
$ns at 5.53 "$node(80) label Taken"

$ns at 5.53 "$node(63) label C "
$ns at 5.53 "$node(63) color $c1"
$ns at 5.53 "$node(70) label C "
$ns at 5.53 "$node(70) color $c1"
$ns at 5.53 "$node(16) label C "
$ns at 5.53 "$node(16) color $c1"
$ns at 5.53 "$node(51) label C "
$ns at 5.53 "$node(51) color $c1"
$ns at 5.53 "$node(9) label C "
$ns at 5.53 "$node(9) color $c1"
$ns at 5.53 "$node(26) label C "
$ns at 5.53 "$node(26) color $c1"
$ns at 5.53 "$node(37) label C "
$ns at 5.53 "$node(37) color $c1"
$ns at 5.53 "$node(45) label C "
$ns at 5.53 "$node(45) color $c1"
$ns at 5.53 "$node(86) label C "
$ns at 5.53 "$node(86) color $c1"

$ns at 5.67 "$node(63) label Taken"
$ns at 5.67 "$node(70) label Taken"
$ns at 5.67 "$node(16) label Taken"
$ns at 5.67 "$node(51) label Taken"
$ns at 5.67 "$node(9) label Taken"
$ns at 5.67 "$node(26) label Taken"
$ns at 5.67 "$node(37) label Taken"
$ns at 5.67 "$node(45) label Taken"
$ns at 5.67 "$node(86) label Taken"

$ns at 5.67 "$node(67) label C "
$ns at 5.67 "$node(67) color $c1"
$ns at 5.67 "$node(73) label C "
$ns at 5.67 "$node(73) color $c1"
$ns at 5.67 "$node(13) label C "
$ns at 5.67 "$node(13) color $c1"
$ns at 5.67 "$node(53) label C "
$ns at 5.67 "$node(53) color $c1"
$ns at 5.67 "$node(3) label C "
$ns at 5.67 "$node(3) color $c1"
$ns at 5.67 "$node(28) label C "
$ns at 5.67 "$node(28) color $c1"
$ns at 5.67 "$node(31) label C "
$ns at 5.67 "$node(31) color $c1"
```

```
$ns at 5.67 "$node(40) label C "
$ns at 5.67 "$node(40) color $c1"
$ns at 15.67 "$node(82) label C "
$ns at 15.67 "$node(82) color $c1"


proc attach-CBR-traffic { node sink size interval } {
   #Get an instance of the simulator
   set ns [Simulator instance]
   #Create a CBR sink14 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 cb 0
set cb1 0
set k 2.0
set l 2.0
set b 1

for { set i 1 } { $i<= 40} { incr i } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink0 34 .042]
$ns at $k "$cbr($cb) start"
#incr i
#$ns at $k "$ns trace-annotate \"$i th node Broadcasting Residual
Energy\""
$ns at $l "$cbr($cb) stop"
incr cb
}
for { set i 1 } { $i<= 89} { incr i } {
$ns at $k "$ns trace-annotate \"$i th node Broadcasting Residual
Energy\""
}


set cb 20
set k 3.8
set l 3.8
set b 1
for { set j 1 } { $j<= 89} { incr j } {
set cbr($cb) [attach-CBR-traffic $node($j) $sink0 32 .042]
$ns at $k "$cbr($cb) start"
incr j
incr j
incr j
$ns at $l "$cbr($cb) stop"
incr cb
```

```
}

set cb 80
set k 5.2
set l 5.2
set b 1
for { set j 1 } { $j<= 89} { incr j } {
set cbr($cb) [attach-CBR-traffic $node($j) $sink0 32 .042]
$ns at $k "$cbr($cb) start"
incr j
incr j
incr j
incr j
$ns at $l "$cbr($cb) stop"
incr cb

}

set cb 80
set k 5.4
set l 5.4
set b 1
for { set j 1 } { $j<= 89} { incr j } {
set cbr($cb) [attach-CBR-traffic $node($j) $sink0 32 .042]
$ns at $k "$cbr($cb) start"
incr j
incr j
incr j
incr j
$ns at $l "$cbr($cb) stop"
incr cb

}

set cb 80
set k 5.55
set l 5.55
set b 1
for { set j 1 } { $j<= 89} { incr j } {
set cbr($cb) [attach-CBR-traffic $node($j) $sink0 32 .042]
$ns at $k "$cbr($cb) start"
incr j
incr j
incr j
incr j
$ns at $l "$cbr($cb) stop"
incr cb



}
```

```
set cb 60
set cb1 0
set k 4.0
set l 4.0
set b 1
for { set i 1 } { $i<= 56} { incr i } {
 if { $i >=2 && $i<=9 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink1 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"

}

if { $i >=11 && $i<=18 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink10 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}

if { $i >=20 && $i<=28 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink19 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}


if { $i >=49 && $i<=56 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink48 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}

if { $i >=30 && $i<=38 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink29 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}
if { $i >=40 && $i<=47 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink39 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}
if { $i >=49 && $i<=58 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink39 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}
if { $i >=59 && $i<=69 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink39 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}
if { $i >=70 && $i<=79 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink39 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}
if { $i >=80 && $i<=88 } {
set cbr($cb) [attach-CBR-traffic $node($i) $sink39 1024 .042]
$ns at $k "$cbr($cb) start"
$ns at $k "$ns trace-annotate \"$i th node Transmitting\""
$ns at $l "$cbr($cb) stop"
incr cb

}
}

proc finish {} {
      global ns traceFile

      #exec nam -r 2m clus.nam &

          close $traceFile
      $ns flush-trace
      exit 0
#exec xgraph energy.xg &
      }
```

```
$ns at 60.0 "finish"

puts "Start of simulation.."

#set ns [new Simulator]
set myobj [new Agent/MyVar]

$myobj set x_tcl 5
$myobj set y_tcl 10
$myobj add

#$ns run

$ns run
```

CC File:

```
#include <stdio.h>
#include <string.h>
#include "agent.h"


class MyAgent : public Agent
{
 private:

          void add();
        int x,y,z,i,j,n;
        int a,b;
        int p[10];

      char serv[50][25];

char s[50];

 protected:
          int command(int argc, const char* const* argv);
 public:
          MyAgent();
      struct nodes
         {
            int node_id;
            char service[10];
         }c[5];


};


MyAgent::MyAgent() : Agent(PT_UDP)
{
 bind("x_tcl", &x);
 bind("y_tcl", &y);
 bind("z_tcl", &z);
```

```
}


static class MyAgentClass: public TclClass
{
 public:
  MyAgentClass():TclClass("Agent/MyVar") {}

  TclObject* create (int, const char*const*)
 {
     return (new MyAgent());
 }
}my_obj;

int MyAgent::command(int argc, const char*const* argv)
{   if(argc==2)
  {
   if(strcmp(argv[1] ,"add") ==0)
   {
     add();
     return (TCL_OK);
    }
  }
 return(Agent::command(argc, argv));
}

void MyAgent::add()
{

FILE *out_file1,*out_file2,*out_file3,*out_file4;
char nowfile1[50]="reg1.txt";
char nowfile2[50]="reg2.txt";
char nowfile3[50]="reg3.txt";
char nowfile4[50]="output.txt";

out_file1=fopen(nowfile1,"w");
out_file2=fopen(nowfile2,"w");
out_file3=fopen(nowfile3,"w");
out_file4=fopen(nowfile4,"w");

strcpy(serv[0],"ATemp");
strcpy(serv[1],"APres");
strcpy(serv[2],"Humidity");
strcpy(serv[3],"BTemp");
strcpy(serv[4],"BPres");
strcpy(serv[5],"HRate ");
strcpy(serv[6],"PRate");
strcpy(serv[7],"Smoke");
strcpy(serv[8],"Aroma");
strcpy(serv[9],"Accel");
strcpy(serv[10],"Light");
strcpy(serv[11],"Gas");
strcpy(serv[12],"Bomb");
strcpy(serv[13],"Crack Detector");
```

```
strcpy(serv[14],"Dust Detector");
strcpy(serv[15],"Water Level");
strcpy(serv[16],"Vision Sensing");
strcpy(serv[17],"Movement Sensor");
strcpy(serv[18],"Obstacle Sensor");
strcpy(serv[19],"Damage Detector");
strcpy(serv[20],"Sound Sensor");
strcpy(serv[21],"Device Sensor");
strcpy(serv[22],"Depth Sensor");
strcpy(serv[23],"Altitude Sensor");
strcpy(serv[24],"UV Ray Sensor");


for(i=0;i<10;i++)
{
//p[i]=i;
j=i;
p[i]=rand()%20;

while(j--)
{
if(p[i]==p[j])
{
p[i]=rand()%20;
}
}
fprintf(out_file4,"\n Nodes: %d",p[i]);
}


fprintf(out_file4,"\nNode is requesting for service %s",serv[z]);

fprintf(out_file1,"\n============= SERVICE REGISTRY
=============\n");

for(i=0;i<10;i++)
{
fprintf(out_file1,"\n +-------+---------+");
fprintf(out_file1,"\n | Node%d | %s |",i,serv[i]);
}
fprintf(out_file1,"\n +-------+---------+");
fclose(out_file1);

fprintf(out_file2,"\n============= SERVICE REGISTRY
=============\n");
for(i=10;i<20;i++)
{
fprintf(out_file2,"\n +-------+---------+");
fprintf(out_file2,"\n | Node%d | %s |",i,serv[i]);
}
fprintf(out_file2,"\n +-------+---------+");

fclose(out_file2);
```

```
fprintf(out_file3,"\n============ SERVICE REGISTRY
=============\n");
for(i=20;i<26;i++)
{
fprintf(out_file3,"\n +-------+---------+");
fprintf(out_file3,"\n | Node%d | %s |",i,serv[i]);
}
fprintf(out_file3,"\n +-------+---------+");


fclose(out_file3);


for(i=0;i<26;i++)
{
strcpy(c[i].service,serv[i]);
}


for (i=0;i<26;i++)
{
if (strcmp(serv[z],c[i].service)==0)
{
fprintf(out_file4,"\nThe request is serviced by node %d", i);
break;
}
}
}
```

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer Networks, 38(4):393–422, March 2002.

[2] Ian Akyildiz and Ismail Kasimoglu. Wireless sensor networks: research challenges. 2(4):351–367, October 2004.

[3] Alan D. Amis, Ravi Prakash, Dung Huynh, and Thai Vuong. Max-min d-cluster formation in wireless networks. In INFOCOM (1), pages 32–41. IEEE Computer Society Press, 2000.

[4] Seema Bandyopadhyay and Edward J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In INFOCOM, pages 1713–1723. IEEE Computer Society Press, 2003.

[5] Suman Banerjee and Samir Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In INFOCOM, pages 1028–1037. IEEE Computer Society Press, 2001.

[6]. R.S. Marin-Perianu. *Wireless SensorNetworks in Motion: Clustering Algorithms for Service Discovery and Provisioning*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29.