

P-2830



# Energy Efficient Cluster Based Service Discovery in Wireless Sensor Networks

A PROJECT REPORT

*Submitted by*

C.SIVAKUMAR  
K.SRINIVASAN

71205104047  
71205104052

*In partial fulfillment for the award of the degree*

*Of*

BACHELOR OF ENGINEERING

*In*

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,  
COIMBATORE

ANNA UNIVERSITY: CHENNAI-600 025

APRIL 2009



**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “Energy Efficient Cluster Based Service Discovery in Wireless Sensor Networks” is the bonafide work of

**C.SIVAKUMAR  
K.SRINIVASAN**


**71205104047  
71205104052**

who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported herein 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 other candidate.



**Signature**  
Dr. S.Thangasamy  
**HEAD OF THE DEPARTMENT**

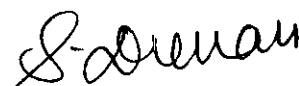
Department of Computer Science &  
Engineering,  
Kumaraguru College of Technology,  
Coimbatore – 641 006



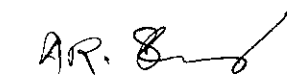
**Signature**  
Mrs. V.Vanitha  
**SUPERVISOR,**  
**Assistant Professor,**

Department of Computer Science &  
Engineering,  
Kumaraguru College of Technology,  
Coimbatore – 641 006

The candidates with University Register number 71205104047 and 71205104052 were examined by us in the project viva-voice examination held on 27.04.09.



**Internal Examiner**



**External Examiner**

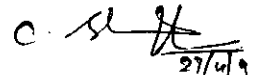
## DECLARATION

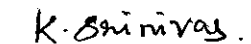
We hereby declare that the project “**Energy Efficient Cluster Based Service Discovery in Wireless Sensor Networks**” is a record of original work done by us and to the best of my knowledge, a similar work has not been submitted to Anna University or any institution, for fulfillment of the requirement 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, Chennai.

Place: Coimbatore

Date: 27/04/09

  
27/4/09  
(Sivakumar C)

  
(Srinivasan K)

## ACKNOWLEDGEMENT

The exhilaration achieved on successful completion of any task should be shared with the people behind the venture. At the onset, we thank the management of our college for having provided the excellent facilities to carry on the project.

We extend our sincere thanks to our vice principal **Mr.V.Annamalai** 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 our beloved Professor, **Dr. S. Thanagasamy**, Ph.D., Dean and Head of the Computer Science and Engineering Department, for his support and encouragement throughout the project.

We express deep gratitude and gratefulness to our guide **Mrs. V.Vanitha**, M.E., Assistant Professor, Department of Computer Science and Engineering, for her supervision, enduring patience, active involvement and guidance.

We would like to thank our project coordinator, **Mrs. P. Devaki**, M.S., Assistant Professor, Department of Computer Science and Engineering for her support during the course of our project.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We humbly acknowledge the inspiration received from the Almighty without whose help the project would not have taken place.

## ABSTRACT

**Wireless Sensor Networks** (WSNs) is an emerging technology that opens a wide perspective for future applications in ubiquitous computing and ambient intelligence. Wireless Sensor Network could contain hundreds of sensors that collect and some cases pre-process data before it is send to central node for final processing .Sensor networks are the key to gathering the information needed by smart environments, whether in buildings, utilities, industrial, home, shipboard, transportation systems automation, or elsewhere.

Service discovery is the action of finding a service provider for a requested service. When the location of the demanded service (typically the address of the service provider) is retrieved, the user may further access and use it.

Designing a service discovery protocol for WSN environments implies a number of challenges. Since sensor nodes are likely to be battery powered, the first objective is to minimize the energy consumption. As the energy is spent mostly during communication, minimizing the energy consumption translates into minimizing, the communication cost. The problem is challenging especially in large scale dense networks, where significant nature of the wireless communication. A second challenge is to repeat rapidly to the network topology changes, which directly affect the consistency of the distributed directory.

We propose a solution for discovery in WSNs, based on clustering. The set of clusterhead nodes act as a distributed directory of service registrations for the nodes in their cluster. In this way we can achieve low discovery cost, since the service discovery messages are exchanged only among the nodes from the distributed directory. The main goal is to minimize the energy consumed both during maintenance and service discovery phases. The contributions of this paper are therefore:

- A lightweight clustering algorithm that builds a distributed directory of service registrations.
- An energy-efficient service discovery protocol that exploits the clustering structure.

## LIST OF SYMBOLS AND ABBREVIATIONS

|      |   |                                   |
|------|---|-----------------------------------|
| WSNs | – | Wireless Sensor Networks          |
| SD   | – | Service Discovery                 |
| DMAC | – | Directional Medium Access Control |
| C4SD | – | Cluster For Service Discovery     |
| DHT  | – | Distributed Hash Table            |
| CH   | – | ClusterHead                       |

## Table of Contents

| <b>CHAPTER NO</b> | <b>TITLE</b>                                  | <b>PAGE NO</b> |
|-------------------|---|----------------|
| <b>1</b>          | <b>INTRODUCTION</b>                           | <b>1</b>       |
|                   | 1.1 Wireless Sensor Network                   | 1              |
|                   | 1.2 WSN Applications                          | 2              |
|                   | 1.3 WSN Characteristics                       | 3              |
|                   | 1.4 Network Architecture                      | 3              |
|                   | 1.4.1 Hierarchical network architecture       | 4              |
|                   | 1.4.2 Flat Network Architecture               | 4              |
|                   | 1.5 Service discovery in WSNs                 | 5              |
|                   | 1.5.1 Service discovery architectures         | 7              |
|                   | 1.5.1.1 Entities and interaction patterns     | 7              |
|                   | 1.5.1.2 Classical Service Discovery Protocols | 8              |
|                   | 1.5.2 Service Discovery Techniques            | 9              |
|                   | 1.5.2.1 Service registration                  | 9              |
|                   | 1.5.2.2 Service discovery                     | 10             |
| <b>2</b>          | <b>LITERATURE REVIEW</b>                      | <b>11</b>      |
|                   | 2.1 Directional Medium Access Control         | 11             |
|                   | 2.2 Distributed Hash Table                    | 13             |
| <b>3</b>          | <b>CLUSTERING</b>                             | <b>15</b>      |
|                   | 3.1 What is Clustering                        | 15             |
|                   | 3.2 Clustering in WSN                         | 15             |
| <b>4</b>          | <b>PROPOSED CLUSTERING SCHEME</b>             | <b>16</b>      |
|                   | 4.1 Designing Challenges                      | 16             |
|                   | 4.2 C4SD (Cluster For Service Discovery)      | 18             |

|     |   |    |
|-----|---|----|
| 4.3 | Adjacent cluster Formation              | 18 |
| 4.4 | Maintenance in face of topology changes | 19 |
| 5   | IMPLEMENTATION                          | 21 |
| 5.1 | Service Registration                    | 21 |
| 5.2 | Service Discovery                       | 21 |
| 5.3 | Simulation Environment                  | 22 |
| 6   | RESULTS OBTAINED                        | 23 |
| 7   | CONCLUSION                              | 24 |
| 8   | FUTURE WORK                             | 25 |
| 9   | REFERENCES                              | 26 |
| 10  | APPENDIX                                | 28 |
|     | - Sample Code                           | 28 |
|     | - Sample Output                         | 31 |



# 1 INTRODUCTION

## 1.1 Wireless Sensor Network

Wireless Sensor Networks represent a new data collection paradigm in which adaptability plays important role. Wireless sensor networks are comprised of a vast number of ultra-small fully autonomous computing, communication and sensing devices, with the very restricted energy and computing capabilities that co-operate to accomplish a large sensing task. Such networks can be very useful in practice i.e. in the local detection of remote crucial events and the propagation of data reporting their realization to a control center. Typical sensor network scenarios involve scattering a large number of wireless nodes from an aircraft across an area of interest. The nodes then form the network through which collected data is routed to a base station. Adaptation is necessary to deal with the unpredictable network topologies that result from sensor node scatters and to manage resources (energy in particular) efficiently in response to changing conditions and requirements.

Sensor networks are the key to gathering the information needed by smart environments, whether in buildings, utilities, industrial, home, shipboard, transportation systems automation, or elsewhere. Recent terrorist and guerilla warfare countermeasures require distributed networks of sensors that can be deployed using, e.g. aircraft, and have self-organizing capabilities. In such applications, running wires or cabling is usually impractical. A sensor network is required that is fast and easy to install and maintain.

Wireless Sensor Network could contain hundreds of sensors that collect and some cases pre-process data before it is send to central node for final processing. In most cases sensors are deployed in remote location without capability to replace battery. This means that one of the key elements for distributed sensors is long lifetime covering both reliability and energy efficiency because the battery limits lifetime of the sensors. Energy efficiency should be taken account in all designs phases starting from system design e.g. topology down to physical implementation constraints. In the literature has shown that

energy efficiency can be improved in various areas when designing wireless sensor network e.g. VLSI design, protocols and network topology.

Many wireless sensor network applications require information about the geographic locations, approximate geographical localization is also needed for many sensor network applications. Manual recording and entering the positions of each sensor node is impractical for very large sensor networks. To address the problem of assigning an approximate geographic coordinate to each sensor node, many automated localization algorithms have been developed.

As the technology evolves, the sensor nodes are expected to self-organize and adapt in face of mobility, failures, changes of network tasks and requirements. Nodes are aware of their own capabilities and are able to cooperate with other nodes in the network, for the purpose of providing networking and system services. The focus changes toward sensor networks providing services to clients in a wide range of applications.

In this project, a prerequisite for providing a service-oriented functionality is the ability to search for services in a WSN. A service discovery protocol has three participating entities: the service provider, the service consumer and the service directory. The latter represents a group of devices responsible for maintaining a distributed repository of service descriptions, which is accessed by the consumer in the search process. The result of a service lookup is the address of one or more service providers.

## **1.2 WSN Applications**

The applications for WSNs are many and varied, but typically involve some kind of monitoring, tracking, and controlling. Specific applications for WSNs include

- habitat monitoring,
- object tracking,
- nuclear reactor control,
- fire detection,
- traffic monitoring.

In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes.

### 1.3 WSN Characteristics

Unique characteristics of a WSN include:

- Limited power they can harvest or store
- Ability to withstand harsh environmental conditions
- Ability to cope with node failures
- Mobility of nodes
- Dynamic network topology
- Communication failures
- Heterogeneity of nodes
- Large scale of deployment
- Unattended operation

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

The base stations are one or more distinguished components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user.

### 1.4 Network Architecture

The two main architecture alternatives used for data communication in a sensor network are the hierarchical and the flat network architectures. The hierarchical network architecture is energy efficient for collecting and aggregating data within a large target region, where each node in the region is a source node. Hence hierarchical network protocols are used when data is to be collected from the entire sensor network. Flat

network architecture is more suitable for transferring data between a source destination pair separated by a large number of hops.

#### **1.4.1. Hierarchical network architecture**

One way of minimizing the data transmissions over long distances is to cluster the network so that signaling/control overheads can be reduced, while critical functions such as media access, routing, and connection setup could be improved. While all nodes typically function as switches/routers, one node in each cluster is designated as the cluster head(CH), and traffic between nodes of different clusters must always be routed through their respective CHs or gateway nodes that are responsible for maintaining connectivity among neighboring CHs. The number of tiers within the network can vary according to the number of nodes, resulting in hierarchical network architecture.

Two tiers of clusterhead where the double lines represent that CHs of tier-1 are cluster-members of the cluster at the next higher level that is tier-2. A proactive clustering algorithm for sensor networks, called LEACH is one of the initial data gathering protocols introduced by MIT's researchers Heinzelman et. Al. Each cluster has a CH that periodically collects data from its cluster members, aggregates it and send it to an upper level CH. Only the CH needs to perform additional data computations such as aggregation, etc. and the rest of the nodes sleep unless they have to communicate with the CH. In order to evenly distribute this energy consumption, all the nodes in a neighborhood takes turns to become the CH for a time interval called the cluster period.

#### **1.4.2. Flat Network Architecture**

In flat network architecture, all nodes are equal and connections are setup between nodes that are in close proximity to establish radio communications, constrained only by connectivity conditions and security limitations. Route discovery can be carried out in sensor networks using flooding that do not require topology maintenance as it is a reactive way of disseminating information. In flooding, each node receiving data packets

broadcasts till all nodes or the node at which the packet was originated gets back the packet. But in sensor networks, flooding is minimized or avoided as nodes could receive multiple or duplicate copies of the same data packet due to nodes having common neighbors or sensing similar data. Intanagonwiwat et. al. have introduced a data dissemination paradigm called directed diffusion for sensor networks, based on a flat topology. The query is disseminated (flooded) throughout the network with the querying node acting as a source and gradients are setup towards the requesting nodes to find the data satisfying the query. The query is propagated towards the requesting node along multiple paths shown by dashed lines. The arcs show how the query is directed towards the event of interest similar to a ripple effect. Events (data) start flowing towards the requesting node along multiple paths. To prevent further flooding, a small number of paths can be reinforced among a large number of paths initially explored to form the multihop routing infrastructure so as to prevent further flooding. One advantage of the flat networks is the ease of creating multiple paths between communicating nodes, thereby alleviating congestion and providing robustness in the presence of failures.

## **1.5 Service discovery in WSNs**

- A service in the network can be any software or hardware entity that a user might utilize.
- Service discovery is the action of finding a service provider for a requested service. When the location of the demanded service (typically the address of the service provider) is retrieved, the user may further access and use it.
- The main goal in the service discovery is that it must be done in an energy efficient manner.

## Service Discovery in the network

The main task for the SD component inside the network is to provide support for the instantiation and reconfiguration of the nodes. When a new node is introduced in the network it has to learn about the capabilities and services of the other nodes. This knowledge enables optimal division of work and maximizes the efficiency of the cooperative problem solving. Because of the dynamic nature of WSNs (from introduction of new nodes, depletion of the existing ones, mobility, communication errors, etc.) this resource discovery has to be periodically repeated during the whole lifetime of the network. These activities entail both address-centric and data-centric parts.

In the first group are the discovery mechanisms that tend to establish a link between the requested service/resource and the unique identity of the node that offers this service. One typical example for this type is the control of the acting elements in the network, i.e. actuator control. This communication is mostly done in a *request/reply* fashion, because the return information about the successfulness of the request can have significant impact on the future actions. This creates tight identity and temporal coupling between the entities and is usually addressed with an address-centric discovery scheme. The same conditions occur in the network management task, when the operator needs to target his commands to specific nodes: turning particular on or off, selective reprogramming of the nodes, gateway selection, etc.

The second group of discovery tasks are the ones that do not require that a specific binding is created between the unique identity of the providers and the services they offer. If the network supports data-centric routing of messages, this binding is not necessary as the entities can communicate between each-other even without knowing the unique ID of the other side.

For example, on instantiation, the temperature sensing node might issue a resource discovery request for other temperature nodes in his vicinity so that he can determine its sleeping schedule. After overhearing the replies in the neighborhood the node computes the optimal schedule and then rebroadcasts its decision back. We can see that the underlying data-centric routing implicitly performs the same function as the

classical service discovery task. This means that the focus of the SD capability is shifted towards providing some additional services on top of this basic functionality.

One such service might be to optimize the discovery task in the case of frequent requests for fixed or slowly changing services by caching the information on the data-centric routing level. Another one is the creation and maintenance of a metadata repository of the attributes used in the network that will guarantee a consistent use of the data-centric naming both inside and outside the network.

### **1.5.1 Service Discovery architectures**

Systems providing discovery for wide-area networks including the discovery of Internet services are presented and it deals with the recently very active research topic of distributed hash tables which are implemented in peer to peer systems.

#### **1.5.1.1 Entities and interaction patterns**

This section contains a short visual introduction to the basic interaction pattern between requesters and providers of services during a typical SD task. The discovery process implicitly defines the necessary interfaces that have to be exposed to the application layer on both sides.

#### **Requesting a service**

The basic interface provided to a SD client is shown in Figure 1.1. The discovery process is simple: The client application issues a request to the node's lower layers and expects a service reply (within a given timeout). In the case that a provider is found a reply is issued is shown in the Scenario 1.1(a). The reply includes the network address of the provider, to be used for subsequent communication between client and provider. After a successful discovery phase, additional communication between client and provider will be necessary in most cases. This information exchange is modeled by a *Service Invocation* and *Invocation Acknowledgment* handshake. Naturally, more or other data messages can be exchanged. End to end routing of data messages between client and provider will be needed. In the case that no provider is available, no reply will be issued by the network. In this case the client will have to wait for a timeout.

## Providing a service

The Provider node interface is also simple: Applications shall be able to register and deregister their services with the service discovery layers

### 1.5.1.2 Classical Service Discovery Protocols

Discovery protocols provide features to spontaneously lookup services with a low communication overhead. Note that the connotation of *service* in the traditional context refers to a network service like a printer or *ssh* server; The traditional view of *service discovery* within a local area network is different from the usage scenarios in multi-hop wireless networks we discuss here.

The protocols originally address one local administrative domain although extensions are sometimes discussed. *Service Lookup Protocol (SLP)* is a proposed standard for service discovery in TCP/IP networks. It provides a dynamic configuration mechanism for applications on local area networks. It automatically selects matching services specified by a client query. It may be implemented with or without the use of *directory agents*. If implemented without *directory agents*, local multicast to all service providers is used to discover services. *Sun's Project Jini* is a *Java*-based framework for heterogeneous networks and software components.

Service discovery is provided by the *Lookup Service* which resides on a designated node to which other nodes may send advertisements or lookup messages. Cascading of multiple lookup services is possible to enable discovery outside the local domain. In addition to employing the central role of *Lookup Services* Jini imposes several constraints on the participants as it relies on *Java* and *Remote Method Invocation (RMI)*. Although *Jini* has often been termed an ad-hoc networking protocol because of its ability to spontaneously integrate new services and applications, it does not provide a suitable support for wireless sensor networks, because of its heavyweight RMI based protocols and its centralized structure.



Distinctive features are its transport independence and the support of code mobility: Devices store drivers, which enable their use, for a variety of operating systems in a repository, so clients may obtain the code to utilize the device. Clients and servers access their local Salutation Manager (SM) to discover or advertise services. Communication among SMs is independent of underlying transport protocols, although to communicate two SMs must use the same protocol. In a network without SMs, local broadcast is used.

The protocol interaction is very similar to using SLP without directory agents. Discovery requests are sent to a designated local multicast address upon which replies are unicast from matching service providers. None of the protocols described above is really suited for the mobile ad hoc networks environment. They are either based on aggregating information into central directories which would impose single points of failure or on maintaining a network wide multicast tree used for periodic service advertisements.

## **1.5.2 Service Discovery Techniques**

### **1.5.2.1 Service registration**

Each node keeps a registry of service descriptions of the nodes placed below in hierarchy. The root node knows 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 easily integrated with the transfer of knowledge on adjacent clusters.

The message UpdateInfo is used for both service registrations and transferring the knowledge on adjacent clusters. 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 topology changes. In the case of a parent reselection, a child node  $v$  registers the services from its sub-tree with the new parent  $p_1$ , and notifies the old parent  $p_0$  (if it is still reachable) to purge the outdated service information. The process is transparent for the other nodes in the sub-tree rooted at  $v$ . If the overall service

information at  $p_0$  and  $p_1$  changes due to the parent reselection, the modifications are propagated up in the hierarchy.

#### 1.5.2.2 Service discovery

The service discovery process uses the 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.

In case 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. The next hop on the path leading to the adjacent cluster is decided by every node that acts as forwarder of the ServDisc message. In the case where a link is deleted and  $v$  cannot forward the ServDisc message, it chooses another neighbor that provides a path to destination. If such a neighbor does not exist,  $v$  informs its parent that it no longer has a route to the next cluster. 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. 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 learns of 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.

The protocol without caching implemented. The message ServDisc has four parameters: the neighbor  $u$  that sends the request, the service descriptions, 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.

## 2 LITERATURE REVIEW

### 2.1 Directional Medium Access Control

DMAC is a viable existing clustering technique for our service discovery protocol. Its simplicity and good performance results make it suitable for sensor environments. DMAC achieves fast convergence, as nodes decide their roles based only on 1-hop neighborhood information. DMAC constructs the clusters based on unique weights assigned to nodes. The higher the weight, the more suitable is the node for the clusterhead role. The difference with our clustering algorithm is that DMAC imposes a Maximum cluster height of one, whereas our protocol in principle may lead to arbitrary cluster height.

For the construction of clusters, DMAC uses two types of broadcast messages, Clusterhead and Join, announcing the roles of the nodes to their neighbors. The role decision of a node is dependent on the decisions of the neighbors with higher weights. Therefore, a single topology change may trigger reclustering of a whole chain of dependent nodes. This phenomenon is called chain reaction. For a distributed directory composed of clusterhead nodes, the chain reaction leads to high overhead for maintaining consistent service registries. The impact of the cluster height and the chain reaction on the performance of the service discovery protocol, in comparison with our proposed clustering solution.

The DMAC protocol with an omni directional antenna and directional antennas is proposed to improve spatial reuse. In the DMAC protocol, the blocking algorithm for directional antennas is used to prevent collision of packets. The directional antenna overheard RTS or CTS is blocked and does not use for data transmission. Node having unblocked directional antennas uses ORTS to solve the problem of deafness.

The node having one or more blocked directional antennas uses DRTS to improve spatial reuse. The operation of the DMAC protocol. Node A transmits ORTS to node B as directional antennas of node A are unblocked. Node B sends OCTS to node A. Then, the DDATA and DACK packets are transmitted. Node C in the coverage of node A overhears ORTS of node A and the directional antenna is blocked. To improve spatial



reuse, node C transmits DRTS to node D as node C has blocked directional antenna. Node E sends DRTS to node C in communication because Node E does not overhear DRTS of node C. Node E retransmits DRTS to node C in deafness after a backoff period. The operation of the DMAC protocol: The circle centered at each node shows its transmission range. An arrow represents the transmission direction. Exponentially increased at every retransmission. Therefore, the throughput performance of ad hoc networks is severely degraded. Directional transmission has trade-off between spatial reuse and deafness.

### **Drawbacks**

- The role decision of a node is dependent on the decisions of the neighbors with higher weights. Therefore, a single topology change may trigger reclustering of a whole chain of dependent nodes.
- For a distributed directory composed of clusterhead nodes, the chain reaction leads to high overhead for maintaining consistent service registries.
- DMAC imposes a max. cluster height of one which results in many cluster formation which causes Cause additional overhead .

## **2.2 Distributed Hash Table**

**Distributed hash tables** are a class of decentralized distributed systems that provide a lookup service similar to a hash table: (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows DHTs to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

DHTs form an infrastructure that can be used to build more complex services, such as distributed file systems, peer-to-peer file sharing and content distribution systems, cooperative web caching, multicast, anycast, domain name services, and instant

messaging. Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the eDonkey network, the Storm botnet, and the Coral Content Distribution Network. DHT implementations CAN (Content Addressable Network), Chord, Pastry, P-Grid, Tapestry.

Each node maintains a set of links to other nodes (its *neighbors* or routing table). Together these links form the overlay network. A node picks its neighbors according to a certain structure, called the network's topology.

All DHT topologies share some variant of the most essential property: for any key  $k$ , the node either owns  $k$  or has a link to a node that is *closer* to  $k$  in terms of the keyspace distance defined above. It is then easy to route a message to the owner of any key  $k$  using the following greedy algorithm: at each step, forward the message to the neighbor whose ID is closest to  $k$ . When there is no such neighbor, then we must have arrived at the closest node, which is the owner of  $k$  as defined above. This style of routing is sometimes called key based routing.

Beyond basic routing correctness, two important constraints on the topology are to guarantee that the maximum number of hops in any route (route length) is low, so that requests complete quickly; and that the maximum number of neighbors of any node (maximum node degree) is low, so that maintenance overhead is not excessive.

The structure of a DHT can be decomposed into several main components. The foundation is an abstract **keyspace**, such as the set of 160-bit strings. A **keyspace partitioning** scheme splits ownership of this keyspace among the participating nodes. An **overlay network** then connects the nodes, allowing them to find the owner of any given key in the keyspace.

A typical use of the DHT for storage and retrieval might proceed as follows. Suppose the keyspace is the set of 160-bit strings. To store a file with given *filename* and *data* in the DHT, the SHA1 hash of *filename* is found, producing a 160-bit key  $k$ , and a message  $put(k, data)$  is sent to any node participating in the DHT. The message is forwarded from node to node through the overlay network until it reaches the single node

responsible for key  $k$  as specified by the keyspace partitioning, where the pair  $(k, data)$  is stored. Any other client can then retrieve the contents of the file by again hashing *filename* to produce  $k$  and asking any DHT node to find the data associated with  $k$  with a message  $get(k)$ . The message will again be routed through the overlay to the node responsible for  $k$ , which will reply with the stored *data*.

### **Drawbacks**

- A disadvantage of most DHT approaches is that they have high maintenance costs, due to the during changes in the overlay network as a result of peers joining and leaving.
- A peer which is responsible for a certain key (for example the hash-key of the string 'Sensor') can become a bottleneck due to dynamic changes.
- Collision occurs when there exists a same hash-key for the different string or filename.

## 3 CLUSTERING

### 3.1 What is Clustering

- Clustering is the process of grouping the nodes i.e cluster is a subset of nodes.
- A network may consist of any number of clusters.
- Each cluster has a head named Clusterhead on unique, and several cluster members, where head is selected on certain criteria assigned to each methodology.
- Nodes divided in virtual group according to some rules
- Nodes belonging in a group can execute different functions from other nodes.

The elements in clustering are,

- Cluster member
- Clusterhead
- Gateway node
- Intra-Cluster link
- Cross-cluster link

### 3.2 Clustering in WSN

- Involves grouping nodes into clusters and electing a CH
- Members of a cluster can communicate with their CH directly
- CH can forward the aggregated data to the central base station through other CHs
- Clustering Objectives
- Allows aggregation
- Limits data transmission
- Facilitate the reusability of the resources
- CHs and gateway nodes can form a virtual backbone for intercluster routing
- Cluster structure gives the impression of a smaller and more stable network
- Improve network lifetime
  - Reduce network traffic and the contention for the channel
  - Data aggregation and updates take place in CHs

## 4 PROPOSED CLUSTERING SCHEME

### 4.1 Designing Challenges

Designing a service discovery protocol for WSN environments implies a number of challenges. Since sensor nodes are likely to be battery powered, the first objective is to **minimize the energy consumption**. As the energy is spent mostly during communication, minimizing the energy consumption translates into minimizing the communication cost. The problem is challenging especially in large scale, dense networks, where significant traffic is generated due to the intrinsic broadcast nature of the wireless communication. A second challenge is to react rapidly to the network topology changes, which directly affect the consistency of the distributed directory.

The most straightforward method for searching in a WSN is based on flooding, which has the advantage of zero maintenance overhead. However, flooding has obvious limitations with regard to energy-efficiency and scalability. A better approach for WSNs is based on **clustering**, where a set of designated nodes acts as a distributed directory of service registrations for the nodes in their cluster. In this way, the communication costs are reduced, since the service discovery messages are exchanged only among the nodes from the distributed directory.

The several design perspective techniques for reducing the communication cost during (1) **discovery of services** and (2) **maintenance of the distributed directory**. Our service discovery protocol uses an underlying clustering structure, where the clusterheads (or root nodes) form a distributed directory of service descriptions. During the discovery process, messages are exchanged among the clusterhead nodes. Therefore, the design issue for minimizing the discovery cost is that the root nodes have to be sparsely distributed on the deployment area. The clustering algorithm should construct an independent set of clusterheads, i.e. two root nodes are not allowed to be neighbors. In the following, we give the design considerations for minimizing the communication cost during the maintenance of the distributed directory:



- **Make decisions based on 1-hop neighborhood information.** Clustering algorithms that require each node to have complete topology knowledge over a number of hops are expensive with regard to the maintenance cost. We aim to build a lightweight clustering structure that requires only the 1-hop neighborhood topology information.
- **Avoid chain reactions.** Several clustering algorithms suffer from the chain reaction problem, where a single topology change in the network may trigger significant changes in clustering structure. For a distributed directory composed of clusterhead nodes, a chain reaction leads to high overhead for maintaining consistent service registries. Therefore, an energy-efficient solution should avoid chain reactions, such that local topology changes determine only local modifications of the directory structure.
- **Distribute the knowledge on adjacent clusters among cluster members.** The knowledge on adjacent clusters should be distributed among the ordinary nodes. Only the root needs to know all the nearby clusters.
- The construction of clusters follows the idea of a greedy algorithm, where nodes choose a neighbor with higher capability grade as parent, while other nodes that do not have such a neighbor are roots. The message SetRoot is used for propagating the address of the root node to all the members of the clusters. The Initialization phase and the event SetRoot from Algorithm 1 give a formal description for the construction of clusters. The protocol works as follows:
  - Nodes that have the highest capability grades among their neighbors declare themselves clusterheads and broadcast a SetRoot message announcing their roles.
  - The remaining nodes choose as parent the neighbor 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.

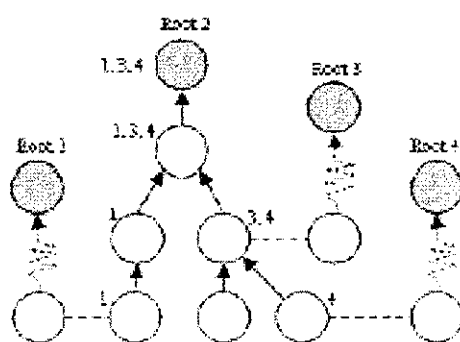
The number of clusters is an important measure for the performance of a clustering algorithm that is intended to be used as a basis for a search mechanism. A high

density of clusterheads leads to a large number of loops that occur during the discovery process.

## 4.2 C4SD (Cluster For Service Discovery)

- In a network, Nodes that have the highest capability grade among their neighbors is declared as Clusterhead.
- The node's capability grade is nothing but the number of services it offers.
- Set of clusterhead nodes act as a distributed directory.
- Adjacent cluster is formed, which plays.
- Each clusterhead maintains the registry for the services it offers and their children
- In time of service discovery, the clusterhead is referred for the service provider.

## 4.3 Adjacent cluster Formation



Show of adjacent cluster formation.

Once the clustering structure is set up, the root nodes need to establish links to the 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. Figure above gives an intuitive example of learning the adjacent clusters. The events of receiving messages **SetRoot** and **Update-Info** from Algorithm describe how the knowledge and the paths to adjacent clusters is updated for a given node. Duplicate **UpdateInfo** messages are discarded: a node sends the message **UpdateInfo** to its parent if and only if the set of known root nodes changes.

#### 4.4 Maintenance in face of topology changes

We analyze how the clustering structure adapts to dynamic environments. We term the events regarding topology changes **LinkAdd** and **LinkDelete**. Algorithm 1 gives a detailed description of the behavior of node  $v$  when these events occur. In short, there are two situations where nodes adjust their cluster membership:

- A node discovers a new neighbor with a higher capability grade than its current parent. The node then selects that neighbor as its new parent.
- A node detects the failure of the link to its parent. The node then chooses as new parent the node with the highest capability grade in its neighborhood. Besides reclustered, topology changes may also require modifications in the knowledge on adjacent clusters. The **SetRoot** message informs nodes about the cluster membership of their neighbors, while the **UpdateInfo** message is used for transmitting the updates from children to their parents. We distinguish the following situations:

- A node  $v$  detects a new neighbor from a different cluster. Consequently,  $v$  adds the root of that cluster to its knowledge.
- A node  $v$  switches from parent  $p_0$  to  $p_1$ . Then  $v$  (1) notifies  $p_0$  to remove the information associated with  $v$  and (2) sends the list of adjacent clusters to  $p_1$ .
- A node  $v$  detects the failure of the link to one of its neighbors  $u$ . As a result,  $v$  erases the knowledge associated with  $u$ .
- Any change of global knowledge at node  $v$  results in transmitting the message `UpdateInfo` from  $v$  to its parent.

### Clustering algorithm :

- Nodes with one-hop distance forms the cluster.
- Each node has its own capability grade on unique.
- Capability grade is the no. of services that the node offers.
- The node with highest capability grade is declared as a Cluster head.
- Sending messages to the other nodes in the cluster is done through `SetRoot()`
- `LinkAdd()` is used when it finds the neighbor with the highest capability grade than its current parent .
- `LinkDelete()` is used to delete the link of the node.

### Advantages

- Reduces communication messages to be transmitted in search of service discovery since it refers in the Clusterhead only.
- Less cluster density.
- Reclustering is avoided.
- Adjacent clustering is done which helps to discover the service Provider in the neighbor cluster.
- Reduces energy consumption

## 5 IMPLEMENTATION

As per the proposed Clustering scheme, the clustering is done on a specific network. The following explains the implementation of the Service Discovery and Service Registration.

### 5.1 Service Registration

- Each node keeps a registry of services it offers.
- When a clusterhead is elected, head node maintains a service registry of its cluster members.

#### Algorithm

The *UpdateInfo()* message is used to do the service registration process by updating the cluster members services to its clusterhead.

### 5.2 Service Discovery

- Service discovery is the action of finding a service provider for a requested service. When the location of the demanded service (typically the address of the service provider) is retrieved, the user may further access and use it.
- The main goal in the service discovery is that it must be done in an energy efficient manner .

## Algorithm

- The service discovery process uses the 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. The service discovery reply may follow the reverse cluster-path to the client, or any other path if a routing protocol is available.

## 5.3 Simulation Environment

A Web service (also Web Service) is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network". Web services are frequently just Web application programming interfaces (API) that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

The implementation is done in c#.net platform. In that platform, the default web service application is used for searching and using of the services provided. The programming language is c#. In web service application the sample services are included and has been used.

## **6 RESULTS OBTAINED**

Set of clusterhead nodes act as a distributed directory of service registrations for the nodes in their cluster. In this way we achieved low discovery cost, since the service discovery messages are exchanged only among the nodes from the distributed directory. The main goal of minimizing the energy consumed both during maintenance and service discovery phases is achieved.

## 7 CONCLUSION

An energy-efficient solution to service discovery in wireless sensor networks. The discovery protocol 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, avoids the chain-reaction problems and keeps a sparse network of nodes in the distributed directory.

Our comparison with DMAC shows different performances of the service discovery protocol depending on the underlying clustering structure. We show that the chain reaction of DMAC determines reclustering and re-registration of services with new clusterheads, implying higher maintenance overhead. Our clustering algorithm achieves fewer clusters and consequently, lower discovery 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. Our protocol has a lower discovery cost in both implementation alternatives.



## **8 FUTURE WORK**

We consider introducing dynamic capability grades, in order to avoid overloading the root and parent nodes with service registrations. The idea is that nodes that reach their memory limit decrease the capability grade and thus, a part of their children will register to other nodes.

## 9 REFERENCES

- [1] M. Balazinska, H. Balakrishnan, and D. Karger. INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery. In *Pervasive '02*, pages 195–210, August 2002.
- [2] S. Basagni. Distributed clustering for ad hoc networks. In *ISPAN '99*, pages 310–315, Washington, DC, USA, 1999. IEEE Computer Society.
- [3] J.Wu and M. Zitterbart. Service awareness in mobile ad hoc networks. Boulder, Colorado, USA, March 2001. Paper Digest of the 11th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN).
- [4] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *WCMC: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.
- [5] S. Das, C. E. Perkins, and E. M. Royer. Ad hoc on demand distance vector (AODV) routing. Internet-Draft Version 4, IETF, October 1999.
- [6] C. Frank and H. Karl. Consistency challenges of service discovery in mobile ad hoc networks. In *MSWiM '04*, pages 105–114, New York, NY, USA, 2004. ACM Press.
- [7] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.
- [8] U. C. Kozat and L. Tassiulas. Service discovery in mobile ad hoc networks: An overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, 2(1):23–44, June 2003.

[9] R. Marin-Perianu, H. Scholten, and P. Havinga. CODE: A description language for wireless collaborating objects. In ISSNIP '05, pages 169–174. IEEE Computer Society Press, December 2005.

[10] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In INFOCOM' 97, volume 3, pages 1405–1413. IEEE, April 1997.

## 10 APPENDIX

### Sample Code

#### Cluster.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SN
{
    class Cluster
    {
        public string ClusterName;
        public int clusterSize;
        public int RemainclusterSize;
    }
}
```

#### Node.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace SN
{
    public class Node
    {
        public String NodeName;
        public String ClusterName;
        public String ServiceName;
        public int ServiceCount;
        public String NodeType;
    }
}

using System.Windows.Forms;

namespace SN
{
    static class Program
    {
        /// The main entry point for the application.

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

```

```

        Application.Run(new mdiForm());
    }
}

```

### **CommonFunction .cs**

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SN
{
    class CommonFunction
    {
        public static bool isNumber(String num)
        {
            try
            {
                Double.Parse(num);
                return true;
            }
            catch
            {
            }
            return false;
        }
    }
}

```

### **mdiForm .cs**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SN.SN;
using System.Collections;

namespace SN
{
    public partial class mdiForm : Form
    {
        public static DataSet nodeDS=new DataSet();
        public static DataSet ClusterDS = new DataSet();
        public static ArrayList ClusterList = new ArrayList();
    }
}

```

```

public static ArrayList NodeList = new ArrayList();
public static Hashtable ClisterHashMap = new Hashtable();

public mdiForm()
{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
}

private void clusterNameToolStripMenuItem_Click(object sender,
    EventArgs e)
{
    frmCluster obj = new frmCluster();
    obj.MdiParent = this;
    obj.Show();
}

private void createNodeToolStripMenuItem_Click(object sender,
    EventArgs e)
{
}

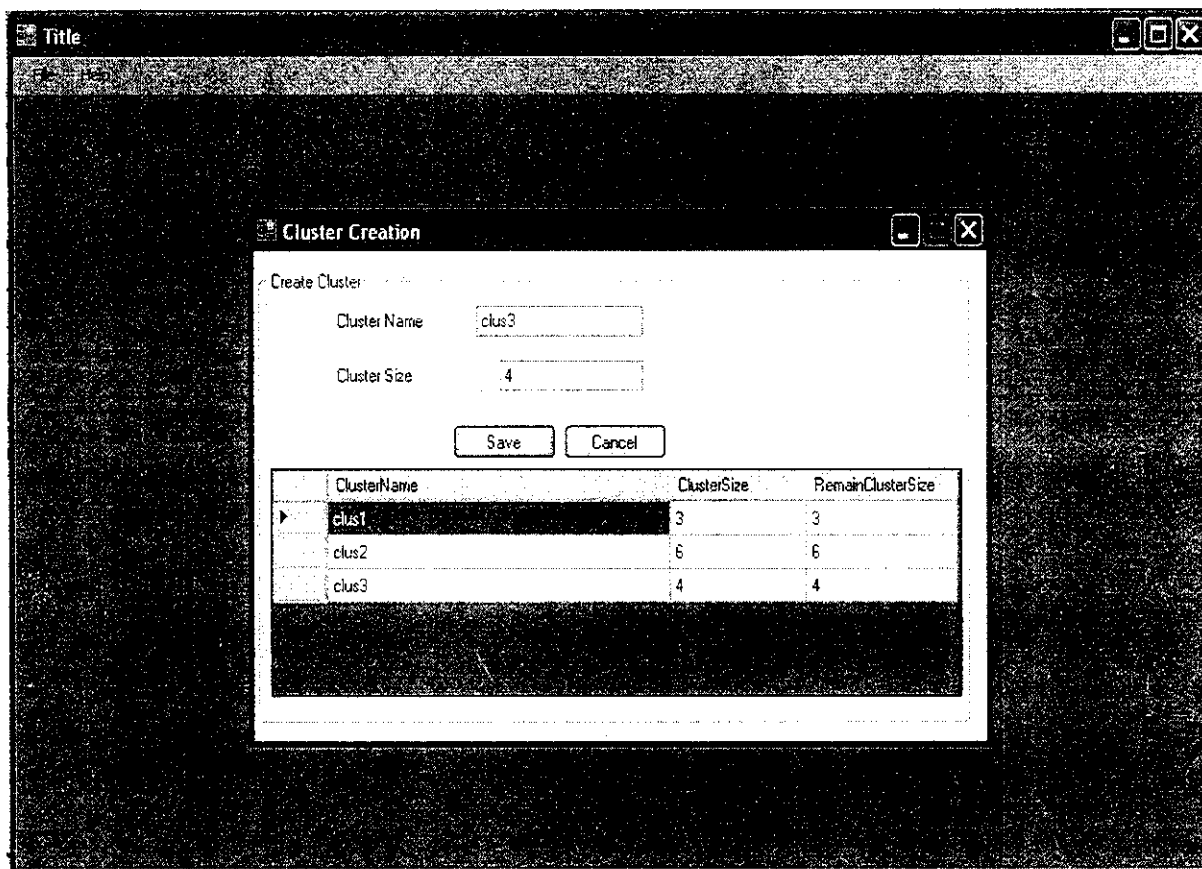
private void nodeToolStripMenuItem_Click(object sender,
    EventArgs e)
{
    frmNodeCre obj = new frmNodeCre();
    obj.MdiParent = this;
    obj.Show();
}

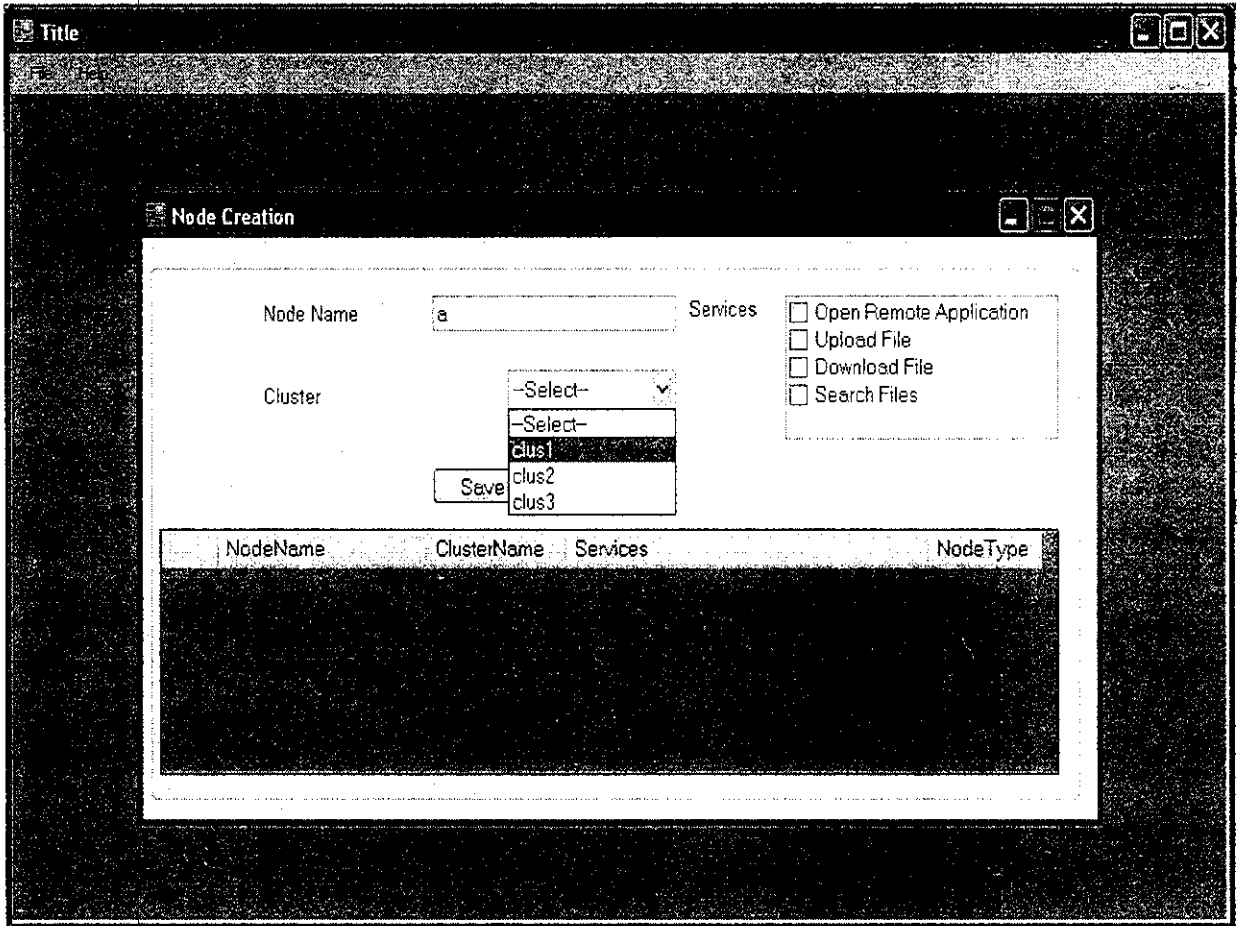
private void serviceDiscoveryToolStripMenuItem_Click(object
    sender, EventArgs e)
{
    ClusterMainForm obj = new ClusterMainForm();
    obj.MdiParent = this;
    obj.Show();
}

private void deleteOrMoveNodeToolStripMenuItem_Click(object
    sender, EventArgs e)
{
    frmNodeMove obj = new frmNodeMove();
    obj.MdiParent = this;
    obj.Show();
}
}
}

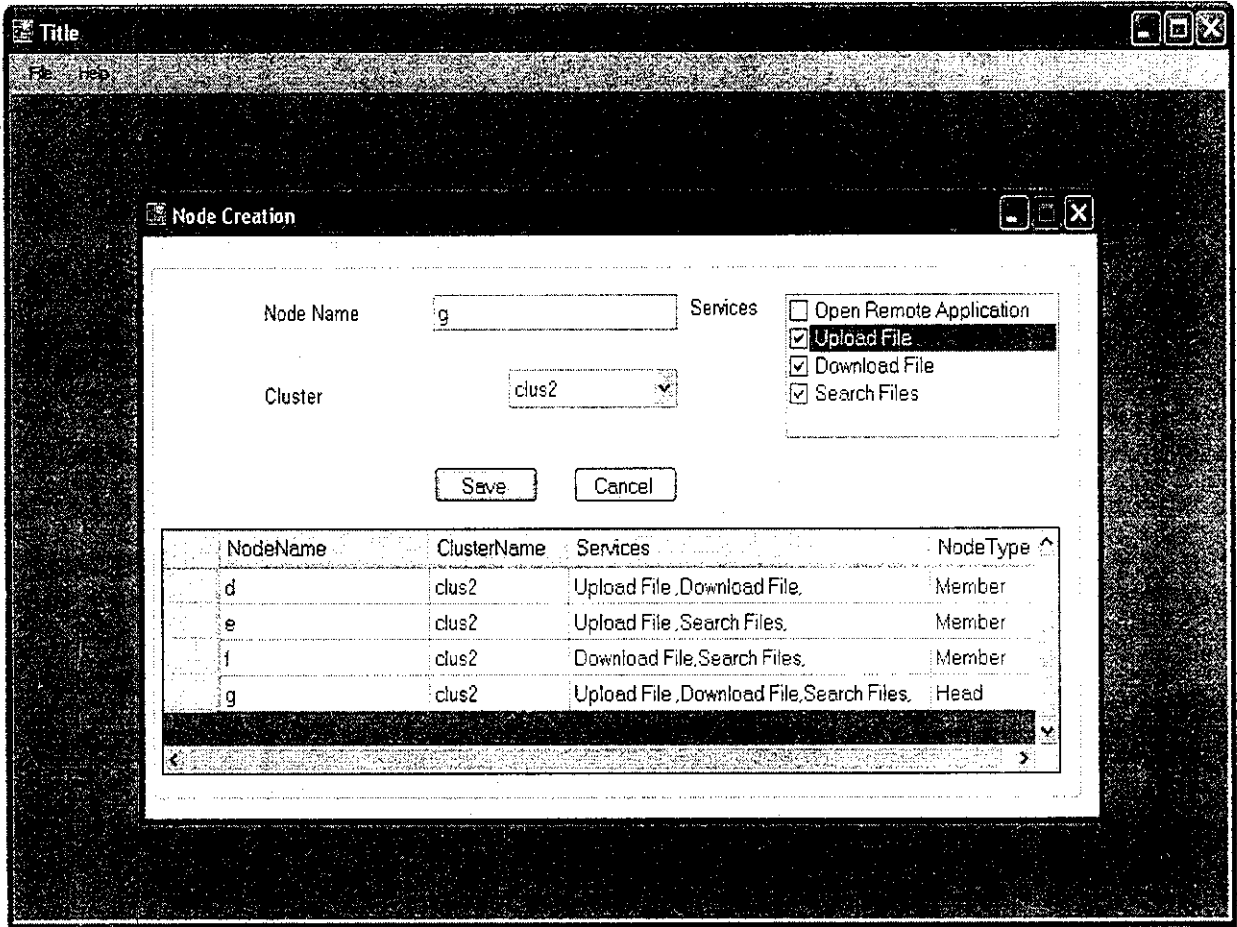
```

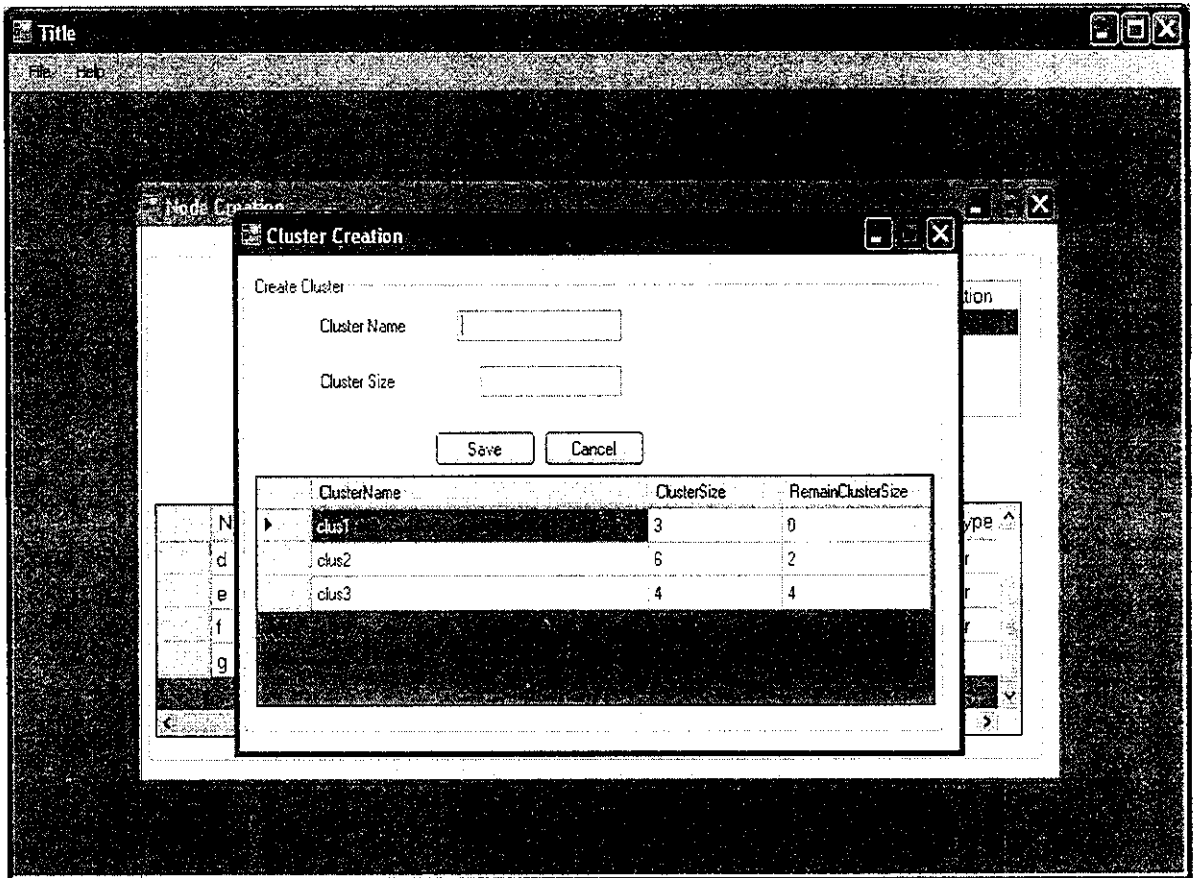
## Sample Output

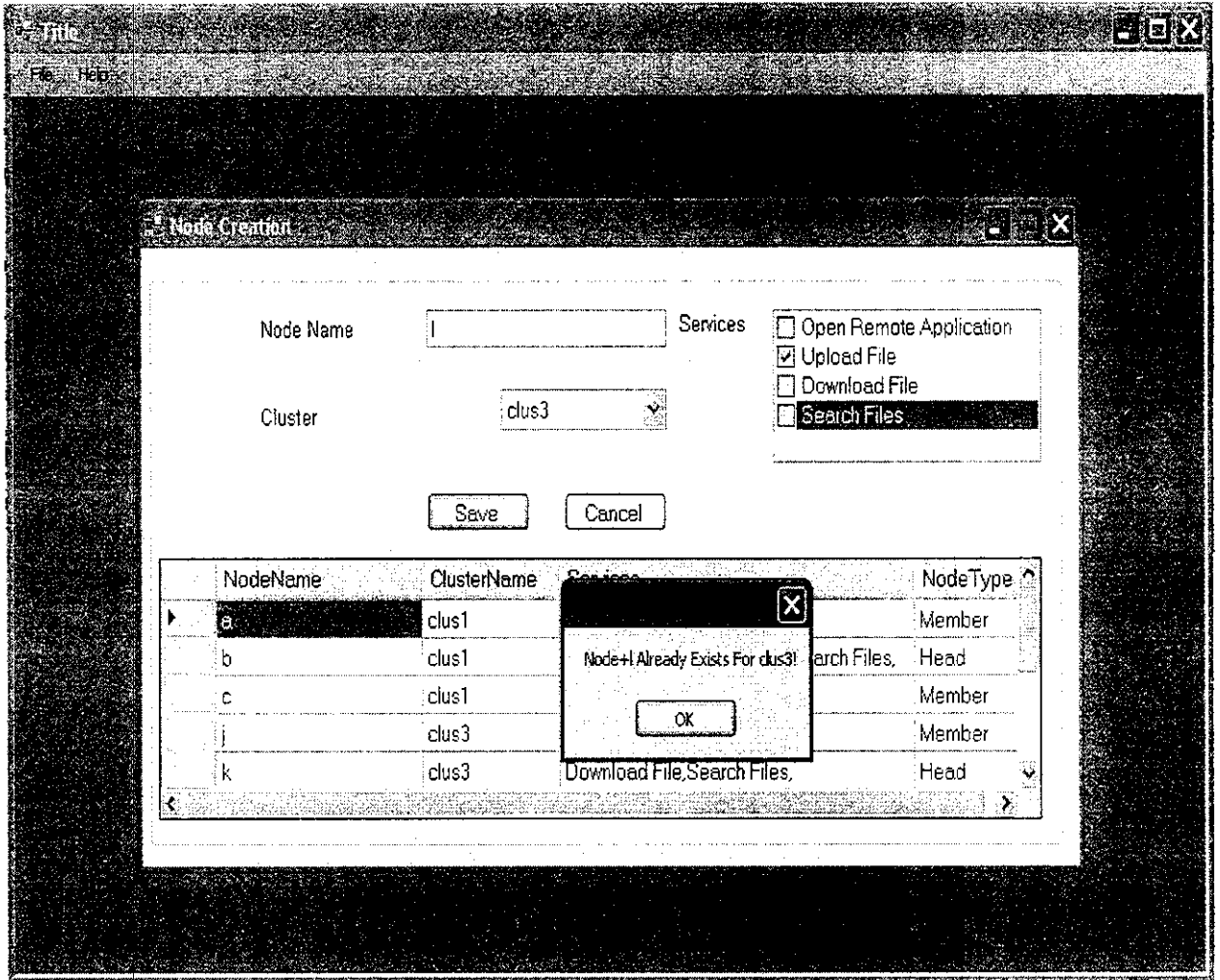


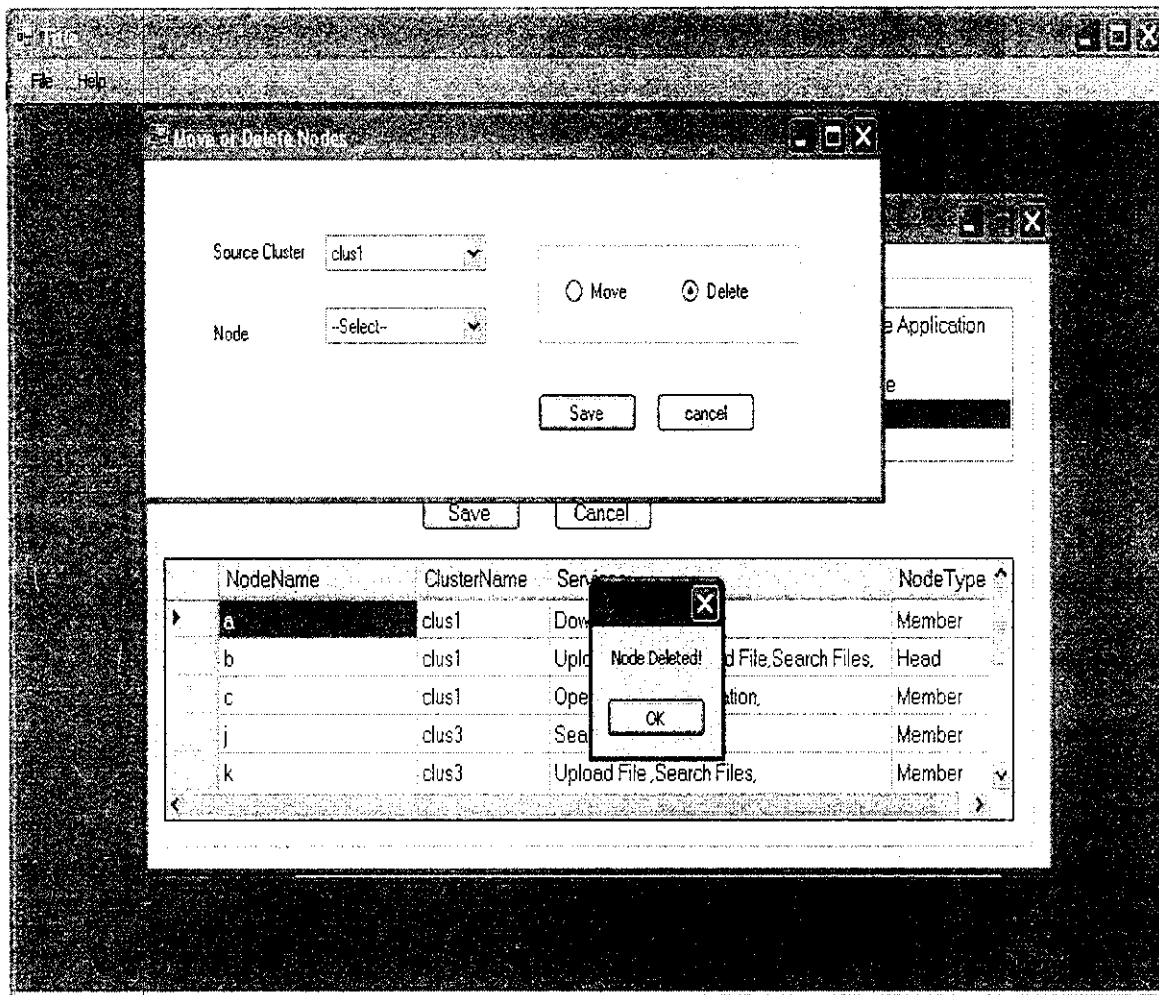


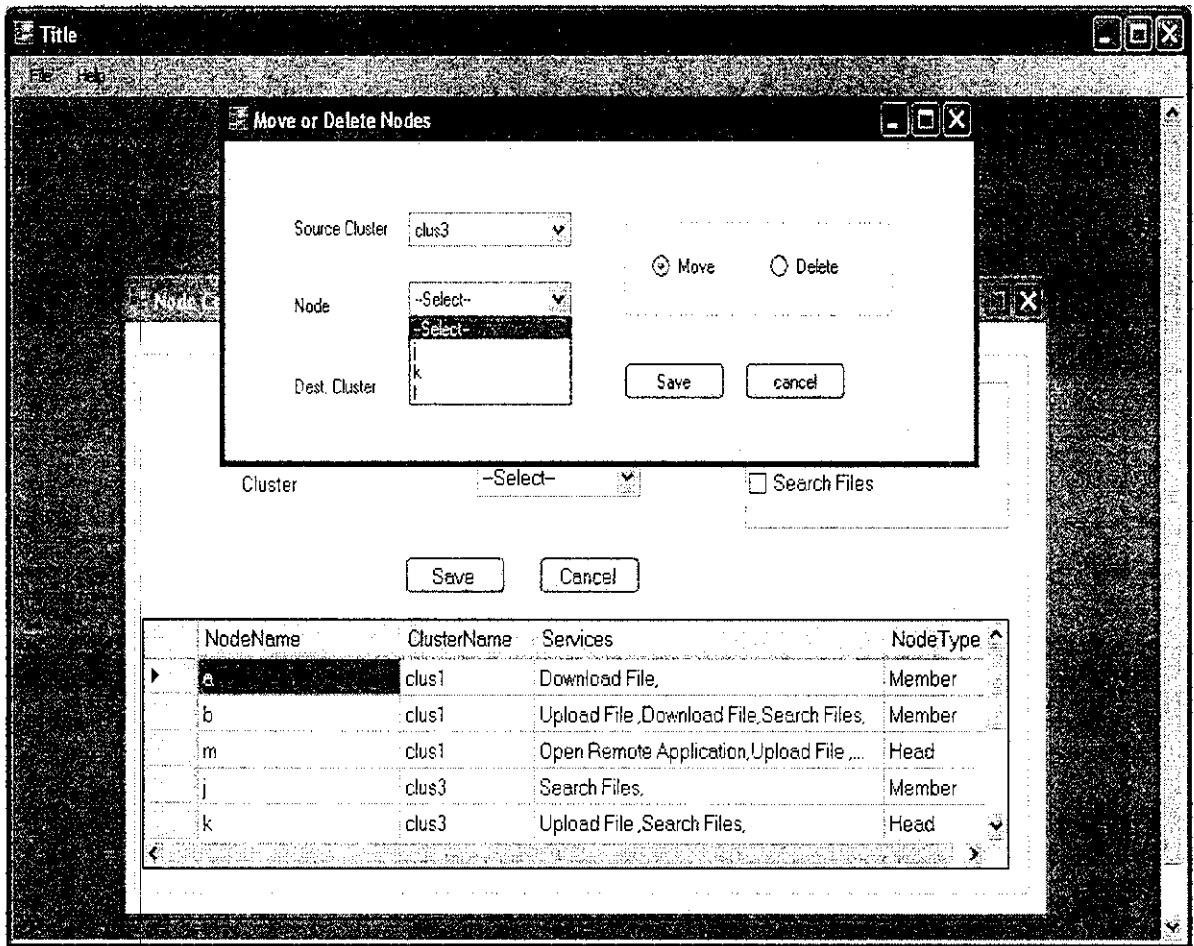












clus1  
clus2  
clus3

| NodeName | ClusterName | Services                                  |
|----------|-------------|---|
| a        | clus1       | Download File,                            |
| b        | clus1       | Upload File ,Download File,Search Files,  |
| m        | clus1       | Open Remote Application,Upload File ,Down |

Process Open Remote Appl  
Open cmd  
Ok

C:\WINDOWS\system32  
Microsoft Windows XP [U] User  
(C) Copyright 1985-2381  
C:\Documents and Settings

Searching Service...  
Service Open Remote Application Not Found in clus2 Cluster  
Service Open Remote Application Found clus1 Cluster  
Searching Service...  
Service Open Remote Application Not Found in clus2 Cluster  
Service Open Remote Application Found clus1 Cluster

ClusterMainForm

clus1

**clus2**

clus3

| NodeName | ClusterName | Services                                  |
|----------|-------------|---|
| d        | clus2       | Upload File, Download File,               |
| e        | clus2       | Upload File, Search Files,                |
| f        | clus2       | Download File, Search Files,              |
| g        | clus2       | Upload File, Download File, Search Files, |

Process Search Files

File Name

AUTOEXEC.BAT

Searching Service...

Service Open Remote Application Not Found in clus2 Cluster

Service Open Remote Application Found clus1 Cluster

Searching Service...

Service Open Remote Application Not Found in clus2 Cluster

Service Open Remote Application Found clus1 Cluster

Searching Service...

Service Search Files Found clus2 Cluster