ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

ENERGY AWARE NEGOTIATION BASED DATA DISSEMINATION PROTOCOL FOR WIRELESS SENSOR NETWORKS

M.Sc. Thesis by

Özcan ÖZYURT

Department : Computer Engineering

Programme : Computer Engineering

Supervisor: Assist. Prof. Dr. Feza BUZLUCA

OCTOBER 2005

PREFACE

I would like to thank to Assist. Prof. Dr. Feza BUZLUCA for all support, supervision, and motivation during my graduate studies.

I would like to thank to Ertan ONUR for his contributions to this study. Thanks to my friends Burak GÜRDAG, Aydın ULAŞ, Gürkan GÜR, Burak GÖRKEMLİ, and İsmail Caner BİROL for their friendship and motivation during my study.

Thanks to my family for their patience and endless support throughout all of my education life.

Özcan ÖZYURT

OCTOBER 2005

TABLE OF CONTENTS

ABBREVIATIONS

WSN	: Wireless Sensor Network
MANET	: Mobile Ad Hoc Networks
EAR	: Energy Aware Routing
SPIN	: Sensor Protocols for Information via Negotiation
EASPIN	: Energy Aware SPIN

LIST OF TABLES

Page Number

LIST OF FIGURES

Page Number

TELSİZ DUYARGA AĞLAR İÇİN ENERJİNİN FARKINDA MÜZAKERE TABANLI VERİ YAYMA PROTOKOLÜ

ÖZET

Düşük maliyetli cihazların geliştirilmesi ile telsiz duyarga ağları için yapılan uygulamaların sayısı da artmıştır. Bu küçük cihazların duyma, işleme ve iletişim birimleri vardır. Sınırlı güç kaynakları ile iletişim mesafeleri kısadır. Bu sınırlar yüzünden telsiz duyarga ağlarının bütün katmanlarında enerji yönetimi ve hata bağışıklığı düşünülmelidir. Bu tezde ağ katmanı düşünülmüştür. Tasarsız ağlar ve telsiz duyarga ağları için bir çok yönlendirme protokolü tanımlanmıştır. Duyarga ağlar, tasarsız ağların bir çeşidi olmasına rağmen, tasarsız ağlar için düşünülmüş olan bazı protokoller telsiz duyarga ağları için uygun değildir, çünkü telsiz duyarga ağları veri merkezlidir. Bu tezde enerjinin farkında bir veri yayma protokolü tanımlanmıştır. SPIN temel olarak alınmış ve üzerine yönlendirme sırasında maliyetin düşünüldüğü, EAR protokolündekine benzer bir mekanizma getirilmiştir. Sonuçlar bu yöntemin toplam enerji kullanımını azalttığını ve ağın ömrünü uzattığını göstermiştir.

ENERGY AWARE NEGOTIATION BASED DATA DISSEMINATION PROTOCOL FOR WIRELESS SENSOR NETWORKS

SUMMARY

Wireless Sensor Network (WSN) applications are increased with development of low-cost devices. These tiny sensor nodes have sensing, data processing, and communication units. They have limited power and can communicate in short distances. Because of the limitations of these nodes, energy management and fault tolerance must be thought in all communication layers of sensor networks. In this thesis, network layer issues of a WSN are considered. There are lots of routing protocols for Ad-hoc networks and WSN. Although sensor networks are some kind of Ad-hoc networks, some protocols for Ad-hoc networks are not suitable for WSN, because WSNs are data-centric. In this thesis an energy aware fault tolerant data dissemination protocol called EA-SPIN (Energy Aware SPIN) is introduced for sensor networks. It is based on SPIN (Sensor Protocols for Information via Negotiation) protocol, but it has also a multi-hop cost effective routing mechanism, which is similar to EAR (Energy Aware Routing). The experimental results show that the proposed protocol can reduce total energy consumption and increase the network lifetime.

1. INTRODUCTION

Improvements in sensor devices and wireless network technology have increased the importance of the wireless sensor network (WSN) applications, such as habitat monitoring, enemy tracking in battlefield, surveillance, etc. [1].

Mobile Ad Hoc Network (MANET) is the closest network model to the WSN with non-fixed network topology, limited power, and wireless communication. Although there are some similarities, the protocols of MANET may not be applied directly to the sensor networks because of the below differences [1,2].

- Sensor networks are data centric, so they are involved gathering data, but MANETs are used for distributed computing.
- In sensor networks data flow from the sensor nodes to the sink. In MANET, data flow is irregular.
- The number of nodes in a WSN is more than of MANET.
- Sensor nodes are much more limited in terms of computation and communication capabilities. They have very limited power resource, while MANET can have rechargeable nodes.
- Sensor nodes are cheaper than the MANET nodes.

WSNs are usually deployed in densely in a sensor field. They contain lots of nodes. The nodes have sensing, computing and wireless communication capabilities [4]. Some additional capabilities can be add to sensor node with hardware units. Some of these units are mobilizer, global positioning system, or power generator [5].

These sensor nodes usually are assigned with gathering data from the sensor field. The collected data are sent to sink node, which is connected to the user terminal over Satellite or Internet link. The routes from sensor nodes to sink node is a multi-hop path in an infrastructureless architecture.

There are some requirements and constraints for wireless sensor networks [4,5]. First of all, sensor nodes shall be cheap, because they are deployed densely. Because of the hardware constraints, the sensor nodes have limited energy and storage capacity.

Routing is an issue for WSNs. There are three categories. Depending on how the protocol finds the route, namely proactive, reactive, and hybrid. In proactive approach, the route is determined before it is required, such as directed diffusion [6]. Whereas in reactive routing, the route is found on demand, such as PEGASIS [7] and EAR [8], hybrid routing protocols are the combination of these two approaches, such as Rumor Routing [9].

Direct communication, flat, and clustering protocols are the classes of routing protocols according to the participating style of a node. In direct communication, nodes send their data to sink directly. All nodes have same roles in flat routing, but in hierarchical routing the nodes can be assigned with different roles.

Negotiation-based, multipath-based, query-based, and location-based are the categories of the protocols according to the protocol operation. Before data is send to nodes, the receivers are asked, whether they require this data or not, so the redundant data transmission is reduced in negotiation based protocols [10]. In multipath routing, more than one path is used to send data. Sink node requires data via a query in query-based protocols [6,8]. Some protocols require the location information of the nodes. They are called location-aware protocols.

1.1 Problem Statement and Proposed Solution

In this thesis, a new data dissemination protocol based on Sensor Protocols for Information via Negotiation (SPIN) [10] is introduced for wireless sensor networks. It is called Energy Aware SPIN (EA-SPIN).

SPIN is a negotiation based data dissemination protocol. It is designed for resolving the problems in classic data dissemination protocols, such as flooding and gossiping. The classic data dissemination protocols have three main problems, implosion, overlap, and resource blindness. Implosion problem states that a node can receive same data from two nodes. Overlap means that two nodes can sense same data and send this data to same sensor node. Because the receiver node does not know whether it has data, redundant data transmissions occur. Also, flooding does not

check the resources. SPIN solves these problems with a negotiation and resource management mechanism. In SPIN, a sender asks to the receiver before sending data whether the receiver has data. If the receiver has this data, sender does not send this advertised data.

Although SPIN solves these problems, it has also a problem. All nodes can be sink in SPIN, so is not energy and sink aware. The aim of the SPIN is sending a data to all nodes in the network, so nodes still can send redundant data in SPIN. We thought that this not a cost effective approach, because the deployment cost increases, if we designed all nodes as sink.

We designed EASPIN for a network, which contains different types of nodes. Unlike SPIN, its goal is transmitting data to sink node with minimum cost. It gets the cost calculation mechanism of EAR [11] and modifies some steps of the mechanism. Also, the message structures are different. In EAR, destination node starts the messaging for establishment, but in EA-SPIN source node sends first message to start route-finding phase. Its data communication phase is similar to SPIN, but it maintains its routing table during communication.

The main goal of EASPIN is increasing the network lifetime, so it reduces the redundant data transmissions with using cost effective paths to sink. It does not use only the path with minimum cost. Paths are selected with a inverse probability according to their costs, so the energy consumption is distributed to all network. This mechanism increases the network lifetime.

1.2 Summary Of Chapters

The thesis is presented in 6 chapters. The summary of chapters are given below :

- Chapter 2 describes the WSN architecture and design challenges.
- Chapter 3 describes the routing protocols for WSN. The classification of routing protocols is explained and the routing protocols, which are used in our study, are analyzed.
- Chapter 4 describes our proposed protocol EASPIN. The messages, phases and communication scenarios are given in this chapter.

- Simulation environment is presented in chapter 5. Also, the simulations we have run and the results of these simulations.
- The aim and the result of this thesis is presented in conclusion chapter. The suggestions for the future work are given also.

2. WIRELESS SENSOR NETWORKS (WSNs)

The improvements in science and technology has affected development of WSNs. Due to the improvements in semiconductor technology, the communication and computation devices have become cheaper and smaller each passing year [3]. This increases the number of applications, which uses these tiny sensor nodes. The sensor nodes must have at least sensing, data processing, and communication components [2]. In Figure 2.1, the basic hardware structure of a sensor node is shown [1]. In this structure all units are assigned with different tasks. Sensor unit collects the requested data and send this analog data to the analog digital converter (ADC). ADC translates the analog data to digital data, so central processing unit (CPU) can understand the sensed data. Moreover, ADC informs the sensor unit what to do. Communication unit obtain the communication between CPU unit and sensor network. A node can communicate with other sensor nodes or users. CPU is the computation and the decision center of a sensor node. It processes the received data, send the query to ADC, find the neighbor to send the data, etc [1].

The components can be added according to the sensor network application requirements. From this point of view, the application designer should consider these hardware constraints. For example, a node can have a power generator, a location finding system, or a mobilizer if the node is mobile [2].

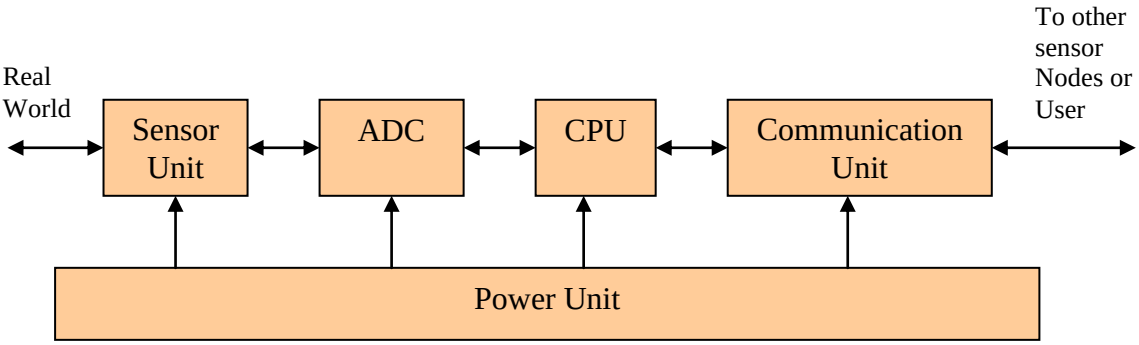


Figure 2.1: Components of a sensor node [1].

A sensor network consists of a lot of sensor nodes, which is densely deployed. Sensors must not be deployed in to predetermined locations, so WSN can be used in inaccessible terrains. In Figure 2.2 is shown WSN architecture. The sensor nodes, which are scattered in a sensor field, can collect data from the field and send data to the sink node. The route is a multihop path in an infrastructureless architecture [2]. Sink node is a special node, which can be communicated with *Task Manager Node* over internet or satellite connection. In respect of this, sink node can have additional hardware units, such as a bigger battery and a powerful communication component for long distance.

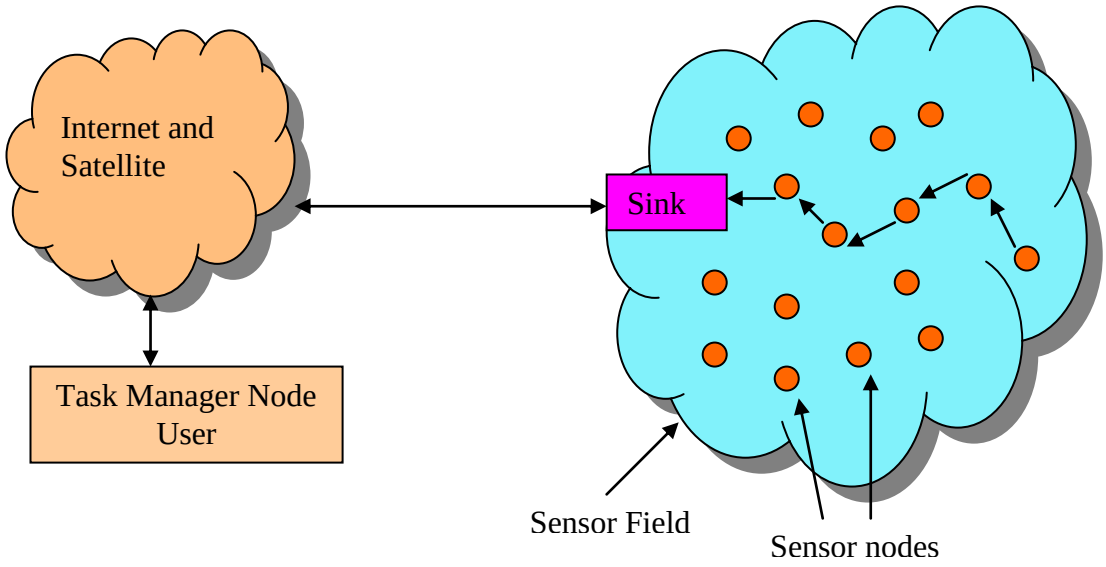


Figure 2.2: WSN Architecture [2].

Sensor nodes can sense different objects and events, such as heat, motion, voice, etc. Sensor networks can be used for monitoring an environment, objects in this environment and relations between these objects [3]. In respect of this, different applications can be designed, such as monitoring events in a hazardous environment, surveillance of enemy activities, border surveillance, etc. [2].

2.1 WSN Design Factors

There are some features and requirements for a sensor network, such as varying network size, low cost, long network life time, self-organization, application awareness, and data centric application [1]. Also sensor nodes have some limitations

because of the hardware requirements. They have limited power and storage capacity.

These requirements and limitations brings some design factors, which must be considered while designing a WSN, such as fault tolerance, scalability, production costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption [2].

Fault Tolerance – Lack of power, physical damage or environmental effects may cause the sensor nodes to fail. The sensor network shall continue to do its overall task, although some nodes are died.

Scalability – A sensor network consists of a large number of sensor nodes, which are deployed densely. For example, hundreds of sensor node in a 10 m^3 region. New protocols or algorithms should work with this number of nodes.

Production Costs – A sensor network has lots of sensor nodes, so the price of a single node is important for the overall cost of sensor network. The cost of a sensor network can be feasible if the cost of a sensor node is less than US$1.

Hardware Constraints – A sensor network consists of Sensor Unit, ADC, CPU, Communication unit, and Power unit [1]. According to application some other components, such *location finding system, power generator, and mobilizer,* can be added to a sensor node [2]. All of these subunits should be fit into a matchbox-sized module, so it size should be smaller than a cubic centimeter.

Sensor Network Topology – Deploying a large number of sensor nodes densely is analyzed in three phases. Predeployment and deployment phase, post-deployment phase, and Redeployment of additional nodes phase. Sensor nodes placed randomly, e.g. throwing from a plane, and after this deployment the topology can change with failures. In the last phase, new nodes can be redeployed to cover the failure nodes' areas.

Transmission Media – In WSN, sensor nodes communicate over wireless channel. Radio, Infrared or optical media can be used for this communication. In infrared and optical media, sensor nodes must see each other.

Power Consumption – The sensor nodes are small and cheap devices, so they have limited power resources. However they behave not only a consumer node, also they are routers in that ad-hoc structure.

2.2 WSN Protocol Stack

In Figure 2.3, the protocol stack of WSN is shown. The WSN protocol stack has planes, which covers all OSI layers. This is the difference between other communication architecture. Power management, Mobility management, and task management should be considered all in OSI layers.

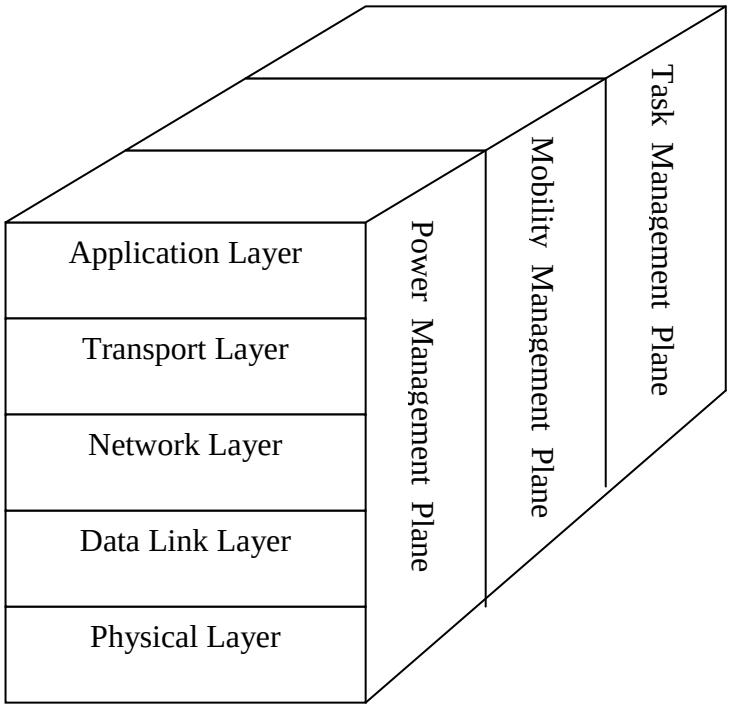


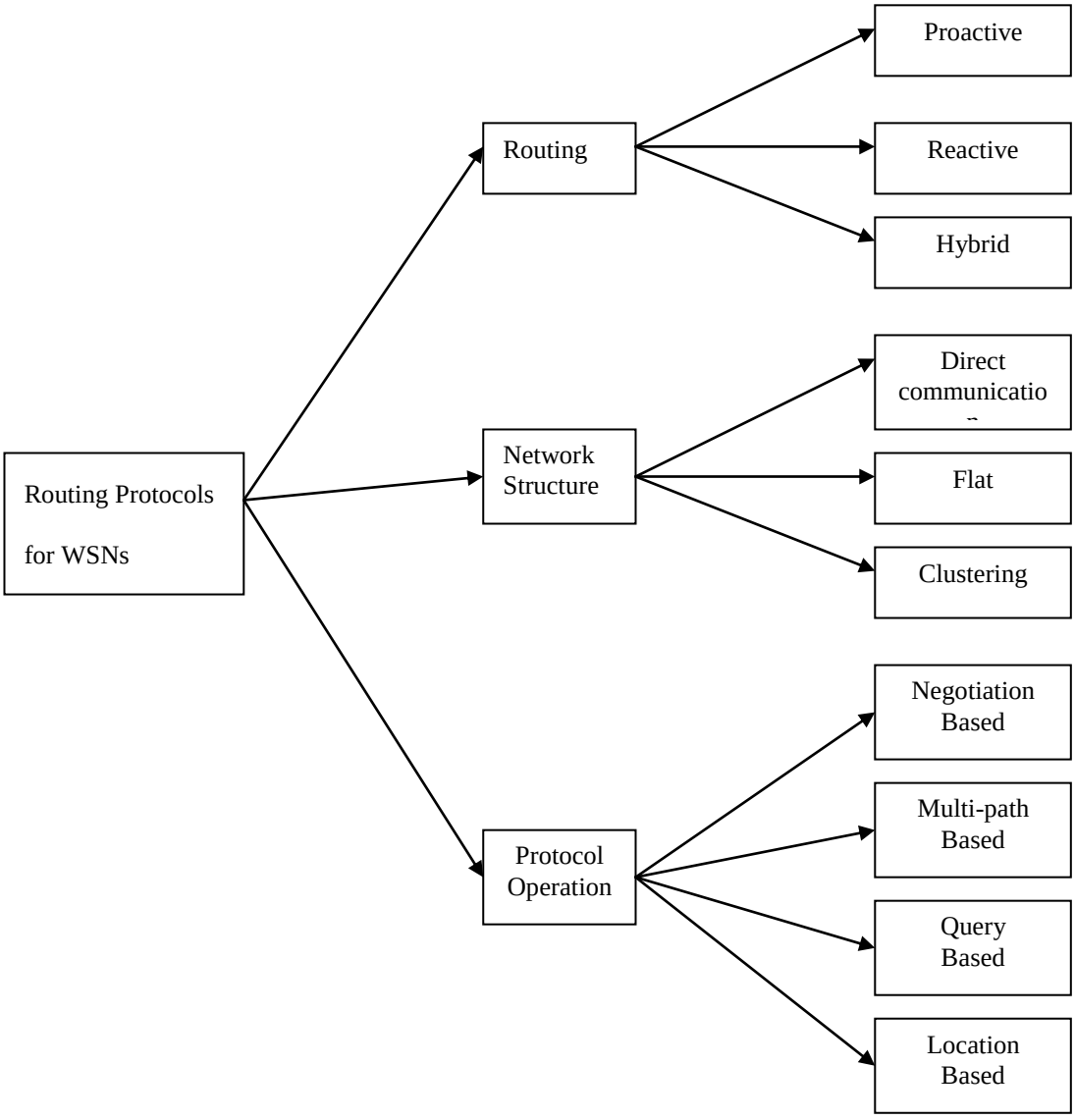
Figure 2.3: The WSN protocol stack [2]

In Physical layer, the modulation, transmission, and receiving techniques are addressed. In data link layer, MAC protocols, which are power-aware and are able to minimize collusion [2], are designed. The network layer is responsible for routing packets. The difference between Ethernet and WSN network layer is that, power management is considered in WSN network layer. The transport layer controls the flow of data. Different type of applications can be built in application layer. Moreover, in power management, mobility management, and task management planes, the power consumption and the sensing tasks are controlled.

3. ROUTING PROTOCOLS FOR WSN

3.1 Classification of Routing Protocols

Different parameters are used in different studies for the classification of routing protocols. We classified the routing protocols as shown in Figure 3.1 [1, 4].



The parameters of first level classification are route finding, network structure, and protocol operation. According to the route finding, routing protocols can be divided into three categories. In proactive protocols, route is found before it is required. These protocols must maintain routing tables. In reactive protocols, the route to the

destination is found, when it is demanded. The hybrid protocols are a combination of proactive and reactive protocols [1].

There are three categories in network structure class. In direct communication protocols, a sensor node communicates with sink node directly. This type of protocols is suitable for small networks. In flat routing, all nodes are same type nodes. However, in hierarchical routing, nodes can have different roles. Cluster structures are established around the sink nodes. Hierarchical routing is more scalable than other types of protocols [1].

According to the protocol operation, protocols can be divided into four categories [4]. In negotiation based protocols, source nodes ask to the destination nodes whether the destination nodes need the data. This mechanism reduces the redundant data transmission. In multipath based routing, more than one path is used for sending data to sink. Query based routing can be used if the application is based on queries. The queries send to the sensor nodes from sink node to gather specific information. Some protocols needs to know location information. They are called location-aware routing protocols [4,1].

3.2 Energy Aware Routing (EAR)

There are some criterions for routing protocols, such as low power, fault tolerant, and scalable. *Network survivability* is more useful metric for routing protocol performance [8]. If we think this criterion, most of the protocols are not useful, although they are designed for energy efficiency, because they find an optimum path. Traffic flows on these paths, so the energy consumption of the nodes along those paths increases.

EAR has designed to increase network lifetime. EAR is a reactive routing protocol, such as AODV [11] and Directed Diffusion [6]. The difference between EAR and other reactive protocols is that EAR uses a single path, which is selected according to the calculated probability. This means EAR selected a different path for each data, so not only one optimum path is used. This obtain that all nodes contributes to the traffic, so the life of the nodes, which are along the optimum path, increases. This means the network lifetime increases.

Protocol Phases:

EAR has three phases [8] .

- **Setup Phase:** Sink node floods an interest message to find all the routes from source to sink node. In this phase all nodes build their routing tables and calculates its average cost and the probabilities for each neighbor. These probabilities are inversely proportional to the cost.
- **Data Communication Phase:** The node, which has data, select a neighbor from its routing table. This selection is made with the probability of the neighbor in the routing table. The relay node, which gets the data, selects a neighbor a neighbor and send data to it. This steps continues until data packet reaches the destination node.
- **Route Maintenance Phase:** Setup phase is rerun periodically to keep the routing tables up-to-date.

3.3 Flooding

In classic flooding, if a node has a data, it sends the data its all neighbors. The relay nodes, which get the data, make copies of the data and send the data its neighbors, except the node, from which the data is received. The node records the data, which it forwarded. If it receives same data again, it does nothing. The algorithm finishes when all nodes receive a copy of data [10].

3.4 Gossiping

Gossiping is an alternative approach to the classic flooding [12]. In gossiping the node, which has data, selects a node randomly and send data to this node. The relay node selects a node randomly again and sends data to it. The selected can be any neighbor of node, so data can be send again to the node, which the data is received from. This is a guard for the deadlock situation.

The gossiping goal is consuming less energy compared with flooding, but the hop count and latency increase.

3.5 Sensor Protocols for Information via Negotiation (SPIN)

SPIN is a data dissemination protocol family, which is based on negotiation for wireless sensor networks. The SPIN philosophy grew out of the limitations of sensor networks, such as limited power and storage capacity, and the problems of the classing flooding [12].

3.5.1 Problems in Flooding

Actually flooding is simple and powerful protocol for data dissemination, but energy consumption is not considered. Also it has three more main problems [10].

- **Implosion:**

  In flooding, a node sends data to its neighbors and does not check whether the neighbor has data. Because of this, the implosion problem occurs. A node can receive same data from a node, although it has already received the data from another node. In Figure 2.1, the problem can be seen. The node A has a data. It floods this data to B and C. After that B and C make copies the data and send their neighbors. Because D is a neighbor both B and C, it receives the data two times. This causes wasting energy and bandwidth resources [10].

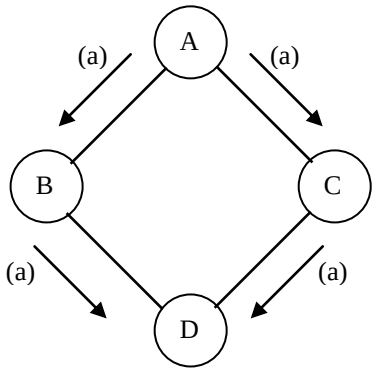


Figure 3.1: The Implosion Problem

- **Overlap:**

  Sensor nodes are deployed densely, so their coverage areas are intersect. Two nodes can gather same data from same geographic area. Figure 3.2 illustrates what happens in flooding when two nodes cover same area. Node A and node B covers the area r, and C is a neighbor both A and B. C receives same data two times, so the energy and bandwidth resources are wasted [10].

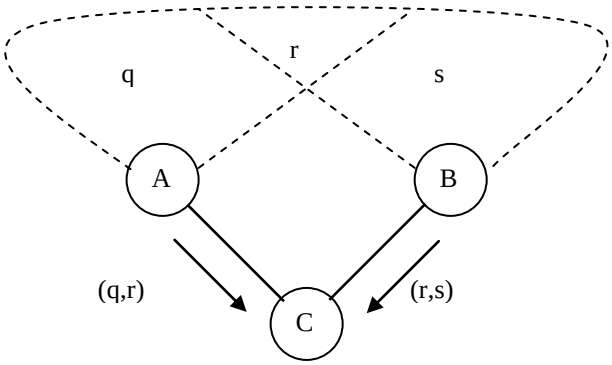


Figure 3.2: The overlap problem

- **Resource blindness:**
  In classic flooding, a node does not behave in respect of its available energy. However, the energy awareness is important for sensor networks. The sensors should be "resource aware" and change their behavior according to the their available resources [10].

There are two keys in design of SPIN family, *negotiation* and *resource-adaptation* [10]. SPIN uses negotiation to solve the overlap and implosion problems. In flooding, if a node has a data, it sends this data to its neighbors, although some of its neighbors have data, too. In SPIN, a node asks to other nodes, whether they have data. This means, the node sends a descriptor of data before data. The descriptor of data is called *meta-data.* It must be shorter than actual data. If the actual data can be separated into pieces, then meta-data should be able to separate. To solve the *resource blindness* problem, all nodes have one resource manager that keeps the available resources. The node modifies its behaviors in respect of the available resources.

3.5.2 SPIN Messages

There are three type of message in SPIN [10].

- **ADV – Advertisement:** A node sends this message, if it has a data to advertise. ADV message contains a meta-data to describe the data.
- **REQ – Request:** The node, which received ADV message, send this message, if it does not have data. The meta-data in ADV message is added to this message before sending to the node, which has advertised the data.
- **DATA:** This message contains the data, which is described with meta-data.

The strenght of SPIN is its simplicity. It has three message described above. If a node has a data, it broadcast an advertise(ADV) message. Data is described with a meta-data, which is send in ADV message. A node, which receive the ADV message checks whether it has the data. If it does not have data, it sends REQ message to the source node. Source node sends DATA to requester nodes.

3.5.3 SPIN Protocols

There are four protocols in SPIN family.

- **SPIN-PP:** This protocol is designed for point-to-point communication and it is used if two nodes can communicate without interference from the other nodes [4]. It is 3 stages protocol, which is illustrated in Figure 2.3. In this Figure 2.3 [10]. In this Figure A advertises its data to B. B sends a REQ message to A to indicate that it does not have data and it requests to get data. After A receives the REQ message from B, it sends DATA to B. After B receives DATA, it advertises the data to its neighbors. B sends DATA to its neighbors, which responds to the ADV message with a REQ message.

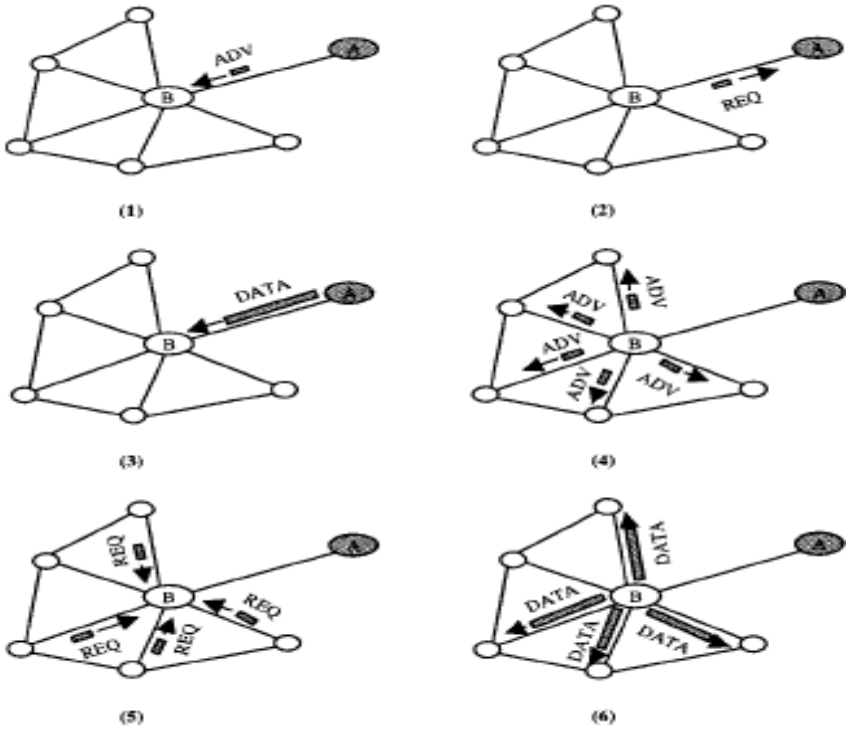


Figure 3.3: The SPIN-PP protocol sequence [10].

- **SPIN-EC:** This protocol is a version of SPIN-PP. It checks its energy level and participates in the protocol if it can handle all stages of the protocol. This

cannot prevent the node from receiving ADV and REQ message, but it is a guard for receiving DATA message, which is longer than other message.

- **SPIN-BC:** The nodes use broadcast channels in this protocol. All nodes in the range receive the messages. However, node must wait, if another node uses the channel. 3 stages protocol sequence is same, but nodes do not send REQ message immediately. They wait a random time to send REQ. If another node completes the protocol sequence in this time, the node receives data also. After that it does not send REQ message. This prevents sending redundant REQ messages. This protocol is illustrated in Figure 3.4.

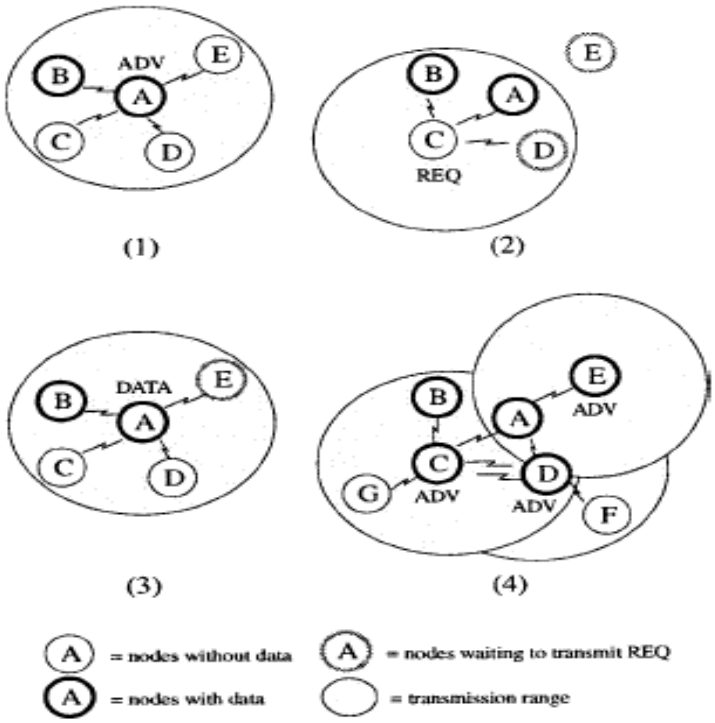


Figure 3.4: SPIN-BC protocol sequence [10].

- **SPIN-RL:** This protocol is designed for a channel with packet loss. SPIN-BC is improved. Each node keeps records of ADV messages. If it has not get DATA message from source node, although it has sent the REQ message, it resends the REQ message to another node, which have same data. Furthermore, there is a limit for the frequency of sending data. The source node does not respond requests an amount of time, after it send data once. This prevents sending data unnecessarily.

3.6 SPMS - Shortest Path Minded SPIN

Shortest Path Minded SPIN (SPMS) is motivated by the SPIN protocol. It is a new approach for SPIN protocols. It is designed for reducing the energy consumption and latency of SPIN [13].

In SPMS, all nodes maintain a routing table for the nodes, which can be reached. A node knows shortest paths to other nodes in its zone. To find the shortest paths, Distributed Bellman Ford (DBF) algorithm is used. The 3-stage SPIN protocol sequence is used, but data flows along the shortest path. The source node sends ADV message to all nodes in its range. The nodes, which are in the next hop of the source node, send REQ message and gets data from source. Other nodes wait the ADV of this data from closest node along the shortest path between source and them.

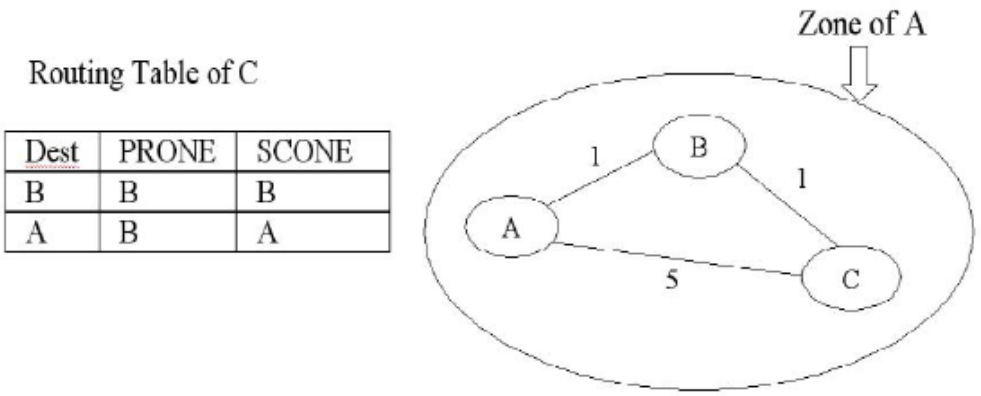


Figure 3.5: SPMS sample network [13].

4. PROPOSED PROTOCOL EA-SPIN (ENERGY AWARE SPIN)

This protocol is based on EAR and SPIN, which are explained before [8, 10]. The main difference between the SPIN and EAR is that the destination node starts route discovery process in EAR, whereas the source node starts the path establishment in SPIN. Each protocol has strong and poor parts according to the assumptions.

In this thesis, a new approach is designed with combining and improving the strong features of these two protocols and their versions. EA-SPIN goal is extending SPIN to a cost effective approach. It consider transmitting data to the sink node, whereas a node considers only sending data to its neighbors in SPIN. EA-SPIN inherits a cost calculation mechanism from EAR and modify in respect of its needs.

4.1 Messages

SPIN has three messages, ADV, REQ, and DATA as mentioned before [10]. EA-SPIN adds one more message, Not Request (NREQ).

- **ADV – Advertisement:** A node sends this message, if it has a new data. ADV message contains meta-data to describe the data. In addition, cost and the remaining energy of the sender.
- **REQ – Request:** The receiver send this message, if it does not have data and its cost is smaller than the sender of ADV message. The meta-data in ADV message is added to this message, so sender node can find the data.
- **NREQ:** This message is the response of ADV message, if the node has data, which is defined with meta-data, and the receiver's average cost is smaller than the sender's average cost.
- **DATA:** This message contains the data, which is described with meta-data.

SPIN messages are used in EASPIN, but some fields are added to use in route finding phase. They are shown in Figure 4.1.

Message-Type	Meta-Data	Average Cost	Remaining Energy

Figure 4.1: The additional EASPIN message parts

4.2 Data Communication

In data communication phase, the negotiation of SPIN and routing selection in EAR is used. In addition the route maintenance is made at the same time. Protocol sequence is explained below. The algorithm is shown in Figure 4.2. A node waits messages from target or other nodes. This algorithm should be considered as an event based system. The constants, variables and the methods are defined below.

DataCommunication method: This is the main method of the node, which is given in Figure 4.2. In this method, the node waits a signal from target, a message from a neighbor, or waits a timeout of ADV message.

ProcessREQMessage method: This method is called when a node receives a REQ message. In this method, node decides its next behavior and updates its routing table.

RouteMessage method: This method is used for selecting a neighbor to send data according to the probability values of the nodes in routing table.

AdvTimeOut method: This method is called when an ADV timeout is occurred. This means, a node send an ADV message for a data and waits REQ and NREQ messages until this timeout occurred.

INITIAL_AVERAGE_COST: This is a constant, which indicates that the cost of node is not calculated, yet. This node behaves like a SPIN node, because the it is not clear whether it can reach to the sink or not.

NO_PATH_TO_SINK: This is a constant, which indicates that the node can not reach to the sink node.

Metadata2dataTable: This table contains the data according to the metadata. This table is used to record and find data.

Metadata2AdvMessagingTable: A node saves a record for each data, which is in its responsibility.This means a node has received data from target or its neighbor. It has send and ADV message for this data and waits to send it to a neighbor.

AverageCostOfSender: This is the average cost of the node, which sends the message.

AverageCostOfNode: This is cost of the node, which runs this method. It is the receiver node.

Nodes have routing tables with the parameters given below.

- **NodeName:** Each node have a unique name in the network.
- **AverageCost:** This is the average cost of the node for sending one bit to sink node. Average cost calculation is explained in Updating Routing Table section.
- **Probability:** The neighbor to send data is selected according to this probability. The probability calculation is explained in updating routing table section.
- **Distance:** Distance is a parameter for energy consumption. Nodes use greater energy for sending data to away nodes than closer nodes. Distance is calculated with ADV and REQ messages. Maximum energy is used for sending REQ messages, so the receiver node can calculate cost of the sender node via lost energy on the way.

At the beginning of a node's life, its average cost value is set to INITIAL_AVERAGE_COST. This means that its average cost is not calculated, yet. After that it begins to wait a message from target or a neighbor. The behavior of the node is explained below for different types of messages.

If node receives a SIGNAL message from a target, it adds this new data to its metadata2DataTable. After that it has this data, so it should not get another copy of this data. If it is not sink node, it has the responsibility of this data. Its mission is transmitting data to sink node. As a single node it can only send this data to its neighbors.

For selecting a neighbor node, which does not have this data, it sends ADV messages to all its neighbors. It sets a timer with a $TOut_{ADV}$ value and starts to wait the messages from its neighbors.

If message is an ADV message, it checks the average cost value of sender. If the average costs of sender and receiver are not INITIAL_AVERAGE_COST, the

receiver node checks whether the sender's cost is smaller than its cost. If this is true, node does nothing, because sender is closer to the sink node. The aim of the protocol is reducing the messages, so receiver node does not send any message.

If average cost of sender or receiver is INITIAL_AVERAGE_COST or the receiver has a smaller cost value than it checks whether it has the advertised data. If it has this data and the receiver cost is not INITIAL_AVERAGE_COST or NO_PATH_TO_SINK, it sends a NREQ message to say that it has data and it is closer than sender node to the sink node. Otherwise it should get data, so it sends a REQ message.

If received message is a NREQ message, node checks whether the average cost of sender is INITIAL_AVERAGE_COST or NO_PATH_TO_SINK. If this is true, than node ignores the message because these values shows that it is not determined whether the sender can reach to the sink node. Although one of the neighbors has a copy of data, it waits other REQ messages. If the sender of NREQ message has a calculated average cost, receiver removes data from its metadata2AdvMessagingTable. It means it will not send data, because another node, which is closer to the sink, has data. Node can know that because nodes send NREQ message if it has data and its cost is smaller than the ADV message sender.

If node receives a REQ message, it runs the processREQMessage algorithm, which is shown in Figure 4.3. It will explain in that section.

If node receives a DATA message, it behaves like it receives a SIGNAL from target. It adds data to its table and sends ADV messages to its neighbors.

If a node receives a REQ message, it runs the processREQMessage algorithm, which is shown in Figure 4.3. In this algorithm node checks whether its metadata2advMessagingTable contains metadata, which is received with REQ message. If metadata cannot be found in the table, it means data is not in this node's responsibility. This can be happened, if all nodes in the routing table have send REQ messages, and node has selected one of them, or a valid NREQ message has been received. In this situation, node ignores REQ message.

If average cost of the sender is NO_PATH_TO_SINK, node does nothing because the requester cannot reach to the sink.

```
DataCommunication()

{Initialize}
averageCost ← INITIAL_AVERAGE_COST
{Waiting A Message}
For every received message
        If Message is SIGNAL
                Add data to metadata2dataTable.
                Broadcast an ADV Message.
                Set a Timer with TOut_ADV
        End If
        If Message is ADV
                If averageCostOfSender < averageCostOfNode AND
                averageCostOfSender != INITIAL_AVERAGE_COST
                AND
                averageCostOfSender != NO_PATH_TO_SINK
                AND
                averageCostOfNode != INITIAL_AVERAGE_COST
                AND
                averageCostOfNode != NO_PATH_TO_SINK
                return
        Else-If metaData2dataTable contains metadata
                Send NREQ message.
        Else
                Send REQ message.
        End If
        If Message is NREQ
                If averageCostOfSender != INITIAL_AVERAGE_COST OR
                averageCostOfSender != NO_PATH_TO_SINK
                   remove advertise message from metadata2advMessagingTable
                End If
        End If
        If Message is REQ
                result ← processREQMessage()
        End If
        If Message is DATA
                Add data to metadata2dataTable.
                Broadcast an ADV Message.
                Set a Timer with TOut_ADV
        End If
End-For
```

Figure 4.2: DataCommunication algorithm

If average cost of the sender is zero, it shows that the sender node is a sink node. Receiver clears its routing table and adds only this sink node to its routing table.

If average cost of sender is INITIAL_AVERAGE_COST, it adds this requester to the requesterWithInitCostTable. This table is used in updating routing table section to show that some neighbors have not calculated their average cost values, but they may be able to reach sink node. Receiver sends DATA to requester because the situation of the node is not clear, so nodes behave like SPIN nodes.

If average cost of sender is not INITIAL_AVERAGE_COST, then the requester is added to requesterTable. This table contains the records for sender nodes, which have sent valid REQ messages.

If requester table contains all nodes in the routing table that means all nodes in the routing table have send REQ messages. In this situation, node must not wait anymore, because the nodes in routing table should have the best-cost values. Node runs the updateRoutingTable algorithm for each node in the requester table. The updateRoutingTable algorithm will be explained in the next section. After running this algorithm, the routing table of a node is updated according to the changes in the network. For example, another node is added to network, or a node dies.

If requester table contains all nodes in the routing table, node selects a neighbor according to the routeMessage algorithm. It removes the advMessaging from metadata2advMessagingTable and sends DATA to selected node. After that, it does not accept new REQ messages, because it has sent DATA and it is not responsible for this DATA.

RouteMessage algorithm is used for selecting a node to send DATA. In the updateRoutingTable algorithm the probabilities are calculated for each node. In routeMessage method a random value is generated and compare with the probabilities of nodes in routing tables. Method returns the selected node.

A node set a timer after it sends ADV messages. After this timer expires, AdvTimeout algorithm is run. Node checks the advMessaging objects, which are expired, in metadata2AdvMessaging table. After that node does these checks given below.

- If No REQ message is received or all requester nodes' average costs are NO_PATH_TO_SINK, this node cannot reach to sink node, so it set its average cost to NO_PATH_TO_SINK

- If all requester nodes' average costs are INITIAL_AVERAGE_COST, then it set its average cost to INITIAL_AVERAGE_COST, because all of its neighbors have not initialized its average cost value.
- Otherwise, it updates its routing table for each node in the requester table.

At the end of the algorithm node selects a node for each data with expired timer and

```
ProcessREQMessage()

If advMessaging is not found in metadata2advMessagingTable
        Do nothing
End If
If  averageCostOfSender = = NO_PATH_TO_SINK
        Do nothing
End If
If  averageCostOfSender = 0 {Requester is Sink Node}
        RoutingTableCapacity ←1
        Add requester to routing table
        For each requester
                UpdateRoutingTable(requester)
        End For
End If

If averageCostOfSender = INITIAL_AVERAGE_COST
        Add requester to requesterWithInitCostTable
        Send DATA to requester
Else
        Add requester to requesterTable
End If

If  requesterTable contains all nodes in routingTable
   (all nodes in routingTable send REQ messages)
        For each requester
                UpdateRoutingTable(requester)
        End For
End If

If  requesterTable contains all nodes in routingTable
   (all nodes in routingTable send REQ messages)
        selectedNode ←routeMessage()
        remove advMessaging from metadata2advMessagingTable
        send DATA to selecteNode
End If
```

send DATA to selected node.

Figure 4.3: ProcessREQMessage algorithm

```
node RouteMessage()

randomValue ←randomize a value between 0and 1

For each node in routingTable
        If  node.probabilityBeginValue <= randomValue AND
            node.probabilityEndValue > randomValue

            return node
        End If
End For
```

Figure 4.4: RouteMessage algorithm

```
AdvTimeOut()

currentTime ←getCurrentSimulationTime()
initialize a metaDataArray
For each advMessage in metadata2AdvMessagingTable
        If advMessage.timeOutValue < currentTime
                Add metadata to metaDataArray.
                If  No REQ message is received OR
                    all requester's averageCosts are  NO_PATH_TO_SINK
                        averageCost = NO_PATH_TO_SINK
                Else If all requester's averageCosts are
                INITIAL_AVERAGE_COST
                        averageCost = INITIAL_AVERAGE_COST
                Else
                        For each requester of advMessage
                                UpdateRoutingTable(requester)
                        End For
                End If
        End If
End For

For each metadata in metadataArray
        selectedNode ←routeMessage()
        remove advMessaging from metadata2advMessagingTable
        send DATA to selecteNode
End For
```

Figure 4.5: AdvTimeout algorithm

4.3 Updating Routing Table

In EASPIN, a routing table is used for selecting a route. Actually, a node does not select a route. A source node selects one of its neighbors, and this neighbor selects

one of its neighbors again. However, selecting a neighbor is similar to finding a route.

ADV and REQ messages are used to update its routing table in EASPIN. This calculation has some steps, which are similar to EAR setup phase [8]. The source nodes do calculations given below, so they update their routing tables and calculates their average costs. The algorithm is shown in Figure 4.6 and explained below.

```
UpdateRoutingTable(requester)

e_ij ←consumedPower for one unit from requester
cost ← CostOfRequester + e_ij^α * energyOfRequester^β

update cost of requester in routingTable

If size of routingTable > 3   {tableCapactiy is set 3}
    Find nodes with minimum cost
    Remove other nodes from routingTable
End If

totalCost ← Calculate total cost (C1 + C2 + C3)
totalInverseCost ← Calculate inverse costs (1/C1 + 1/C2 + 1/C3)

probabilityCurrentValue ← 0
for each node
        propability of node ← ((1 / costOfNode) / totalInverseCost);
        node.probabilityBeginValue = probabilityCurrentValue;
        node.probabilityEndValue = node.probabilityBeginValue
                                        + node.probability;
        probabilityCurrentValue = node.probabilityEndValue;
End For

If size of routingTable = 0
        averageCost = INITIAL_AVERAGE_COST
Else
        averageCost ←0
        For each node in routingTable
            averageCost += (node.probability * node.cost);
        End For
End If
```

Figure 4.6: UpdateRoutingTable algorithm

1. The source nodes send ADV message to their neighbors. The average cost and the remaining energy of the source node are added to message. After that it collects the REQ messages until timeout occurs. In this state, it has a table

with requester nodes and their average costs. The cost of the sink node is zero, so if source node is near the sink node, it receives a REQ message from sink with average cost zero.

$$Cost(N_{\sin k}) = 0 \tag{3.1}$$

2. The source node calculates the energy metric of the requester nodes, which send the REQ messages. After that the node calculates the total cost of the path as given below.

$$C_{N_j,N_i} = Cost(N_i) + Metric(N_j, N_i) \tag{3.2}$$

For calculating the cost metric different formulas can be used, which consider the available energy or used energy. We used the same formula, which is used in EAR [8].

$$C_{i,j} = e_{ij}^{\alpha} R_i^{\beta} \tag{3.3}$$

e_{ij} is the required energy for transmitting and receiving packet and R_i is the available energy of node i. α and β parameters can change in respect of different needs.

3. A routing table is filled with low cost routes. Unlike EAR, a parametric value k determines the record count. For example, if k equals 3, then 3 records with lowest cost are added to the routing table. This is important for setup phase. It is assumed that the node knows the best paths, if the routing table of a node is full.

4. Node N_j assigns a probability for all N_j according to the cost of paths inversely.

$$P_{N_j,N_i} = \frac{\frac{1}{C_{N_j,N_i}}}{\sum_{k \in RT_j} \frac{1}{C_{N_j,N_k}}} \tag{3.4}$$

5. Node N_j calculates the average cost of the paths in routing table.

$$Cost(N_j) = \sum_{k \in RT_j} P_{N_j,N_k} C_{N_j,N_k} \tag{3.5}$$

At the end of this phase, the source node updated its routing table and its average cost.

4.4 Data Communication Scenarios

The protocol sequence and node behaviors are explained with a small network shown in Figure 4.7. The important parts of the protocol are illustrated with sample networks. In this sample network, there are 3 sensor nodes and a sink node. Each node has a routing table. At the beginning, the routing tables are empty, except the sink node, sensor nodes have INIT cost. The average cost of sink node is zero.

The link costs are given with numbers for the simplicity. This cost is dependant to the distance and the remaining energy of the nodes. As an example, we select the costs given in the Figure 4.7.

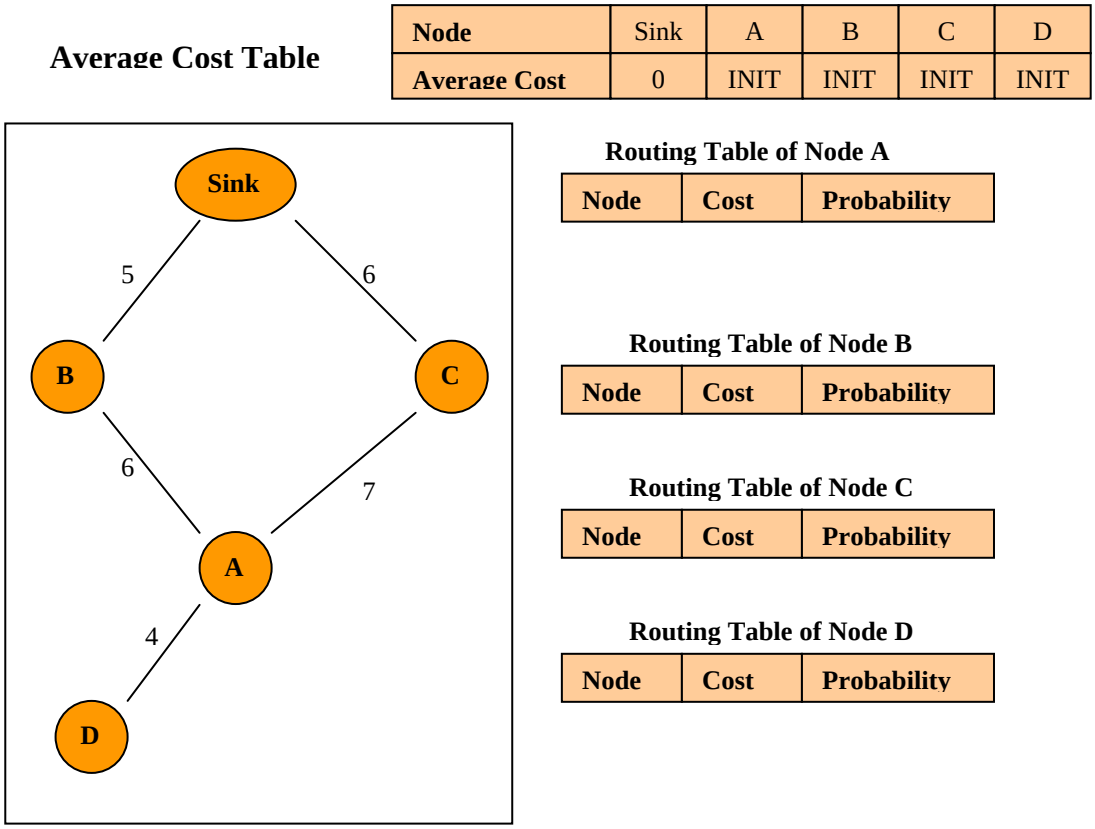


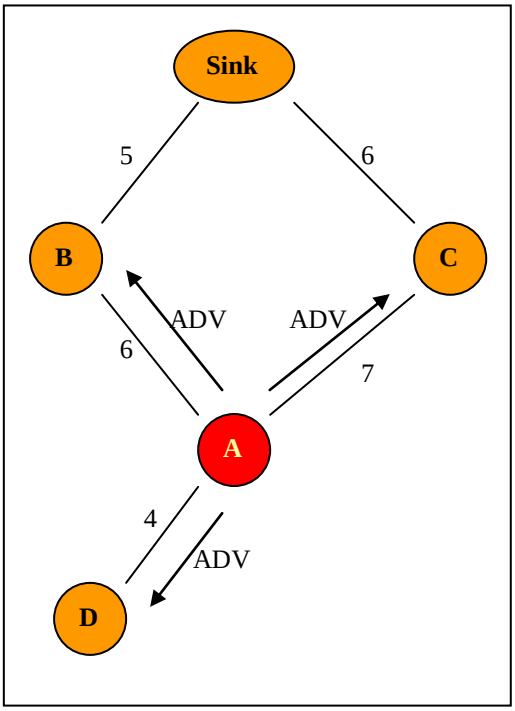
Figure 4.7: Data Communication Scenarios Sample Network

4.4.1 Initial Scenario

The initial scenario shows the protocol sequence in an initial network. At the beginning the routing tables are empty and the average costs are initial.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	INIT	INIT	INIT



Routing Table of Node A

Node	Cost	Probability

Routing Table of Node B

Node	Cost	Probability

Routing Table of Node C

Node	Cost	Probability

Routing Table of Node D

Node	Cost	Probability

Figure 4.8: Initial Scenario (1) – Node A receives first SIGNAL

In Figure 4.8, Node A receives a SIGNAL message. This means Node A has a new data and it must send this data to sink node. Node A sends ADV messages to its neighbors like SPIN protocol, but it adds its average cost in ADV messages also. Its average cost is INIT, which indicates that A does not know, whether there is a path from A to sink.

In the second step shown in Figure 4.9, Node B, D, and C checks the average cost values. The cost values of the all nodes are INIT, so they send REQ messages because they do not know, which node is close to the sink node (1).

Node A checks the average costs of requester nodes. All of them are INIT. Node A understands that B, C, and D have not calculated their average costs, yet, so Node A sends DATA to all requester nodes (2). At this time the protocol sequences of SPIN and EA-SPIN are same because the initialization of the costs are not finished. B, C and D receive DATA. The red nodes in the figures have data.

If the node is not sink and receives a data from target or its neighbors, it is responsible to send this data to another node. In this situation, B, C, and D have received data and they are not sink nodes, so they must advertise this data.

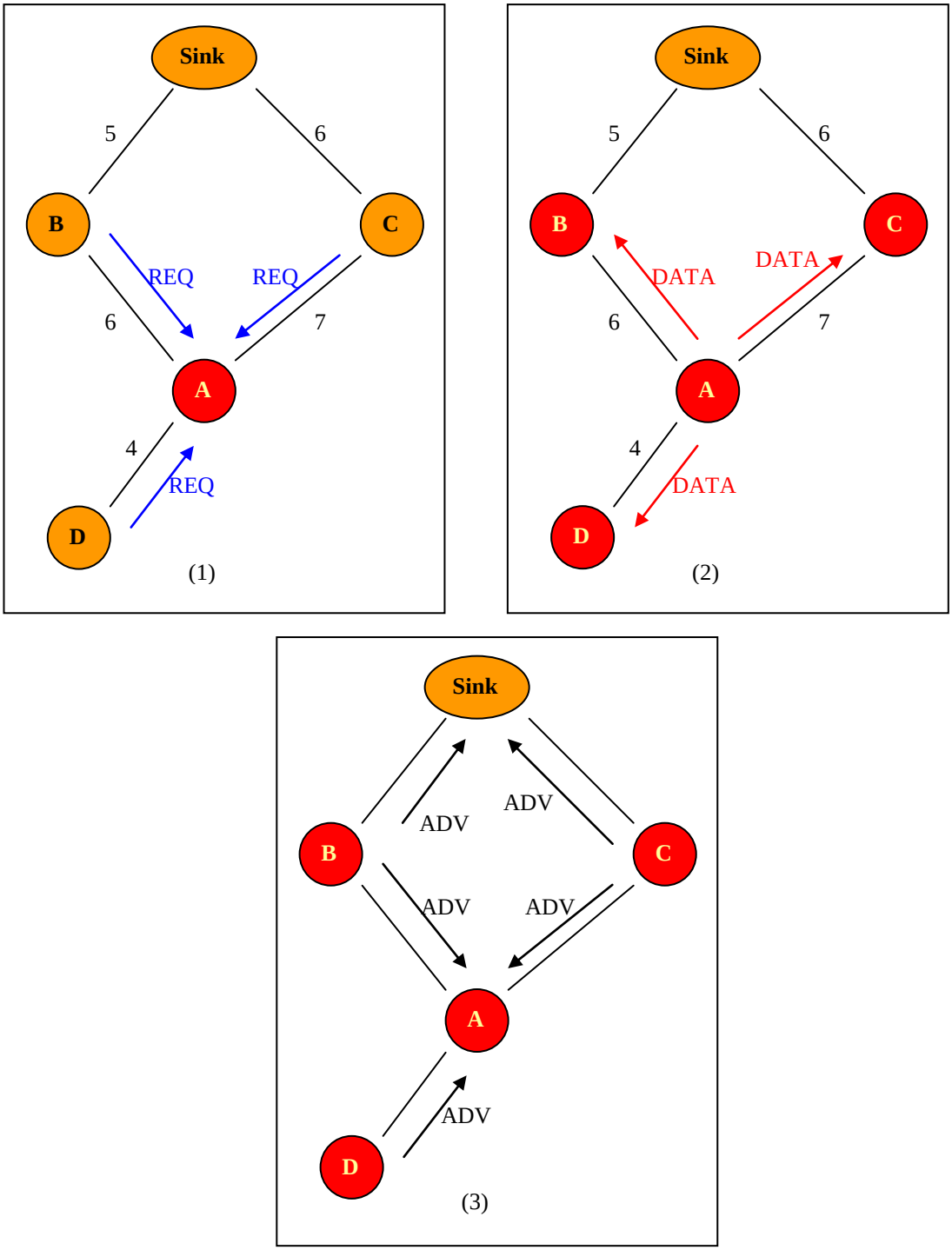


Figure 4.9: Initial Scenario (2) – Behaviors of the nodes after first SIGNAL

B, C, and D send ADV messages, which contain their cost values, to their neighbors, which shown in Figure 4.9. B and C send ADV message to Node A and sink node. D has only one neighbor, Node A, so it sends to A.

The responses of these ADV messages and cost calculation of the B, C, and D nodes are explained separately with the following figures. The nodes, which receive ADV

message controls the checkpoints given below and behaves according to these checkpoints, which are explained before.

- If the cost of sender and receiver are calculated before, it means that if the cost values are not INITIAL_AVERAGE_COST, and the cost of the receiver is greater than the cost of sender, the receiver node does nothing, because the sender node is closer than it to the sink node.
- If the first check is failed, receiver node checks whether it has data.
    - If the receiver node does not have data and its average cost is not NO_PATH_TO_SINK, it sends REQ message.
    - If the receiver node has data, it checks whether its cost is INIT or NO_PATH_TO_SINK and if it is true, it does nothing, because it does not need this data, but it does not know, whether there is a path to sink from it. Otherwise, receiver sends NREQ message to indicate that it has data and it is closer than the ADV sender to the sink node.

In Figure 4.10, the responses of ADV message from B and the result of the cost calculation of B are shown. Node A and sink node have received the ADV message from B. Sink nodes controls the checkpoints given before and sends REQ messages for getting data.

- The cost of B is INIT as in Figure 4.8 and the cost of sink node is zero. According to the first check, which is given before, one of the nodes have INIT cost, so sink node controls the second check.
- Sink node does not have data and its average cost is not INIT or NO_PATH_TO_SINK, so it requests this data, with sending a REQ message to B.

Node A controls the checkpoints and does nothing according to the second check, because it has data and its average cost INIT.

As a result, B receives only one REQ message, which is received from sink node. The average cost of the sink is zero, so B knows that the requester is a sink node and it clears its routing table and adds the sink node to the table. It calculates the cost of the sending one unit data to sink node.

$$C_{N_B,N_{\sin k}} = Cost(N_{\sin k}) + Metric(N_B, N_{\sin k}) = 0 + 5 = 5$$

After that node B calculates the probability of the sink node.

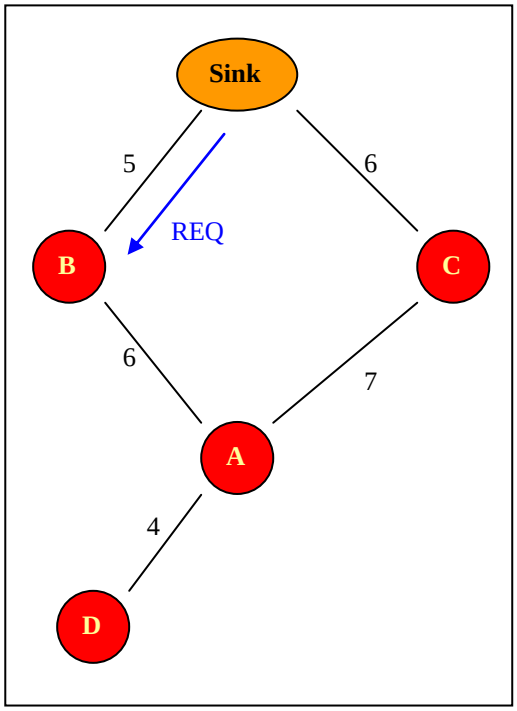
$$P_{N_B,N_{\sin ki}} = \frac{\frac{1}{C_{N_B,N_{\sin k}}}}{\sum_{k \in RT_j} \frac{1}{C_{N_B,N_{\sin k}}}} = \frac{\frac{1}{5}}{\frac{1}{5}} = 1$$

The probability of sending data to sink node is 1. Actually, if a node is a neighbor of the sink node, it sends data to this sink node. If a node is neighbor of two sink nodes, it sends data one of the sink nodes, whose REQ message is received first. It is thought that if it is received earlier, this sink node is closer than other node.

As a result B sends DATA to the sink node, which is shown in Figure 4.11.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	INIT	INIT	INIT



Routing Table of Node A

Node	**Cost**	**Probability**

Routing Table of Node B

Node	**Cost**	**Probability**

Routing Table of Node C

Node	**Cost**	**Probability**

Routing Table of Node D

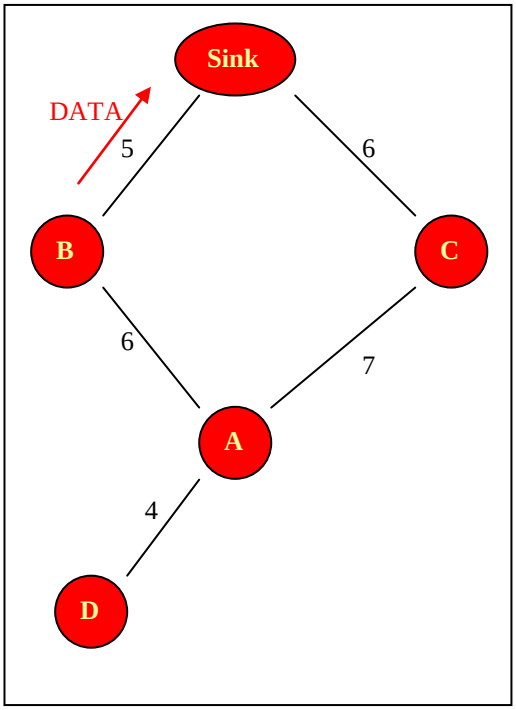
Node	**Cost**	**Probability**

Figure 4.10: Initial Scenario (3) – Responses for ADV from B

In Figure 4.12, the responses of ADV message from C are shown. The result of the cost calculation of C is shown in Figure 4.13. Node A and sink node have received the ADV message from C and only sink node sends REQ message, because the situation is similar to the ADV message from B.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	5	INIT	INIT



Routing Table of Node A

Node	Cost	Probability

Routing Table of Node B

Node	Cost	Probability
Sink	5	1

Routing Table of Node C

Node	Cost	Probability

Routing Table of Node D

Node	Cost	Probability

Figure 4.11: Initial Scenario (4) - Cost Calculation of B

The average cost of the sink is zero, so C knows that the requester is a sink node and it clears its routing table and adds the sink node to the table. It calculates the cost of the sending one unit data to sink node.

$$C_{N_C,N_{\sin k}} = Cost(N_{\sin k}) + Metric(N_C, N_{\sin k}) = 0 + 6 = 6$$

After that node B calculates the probability of the sink node.

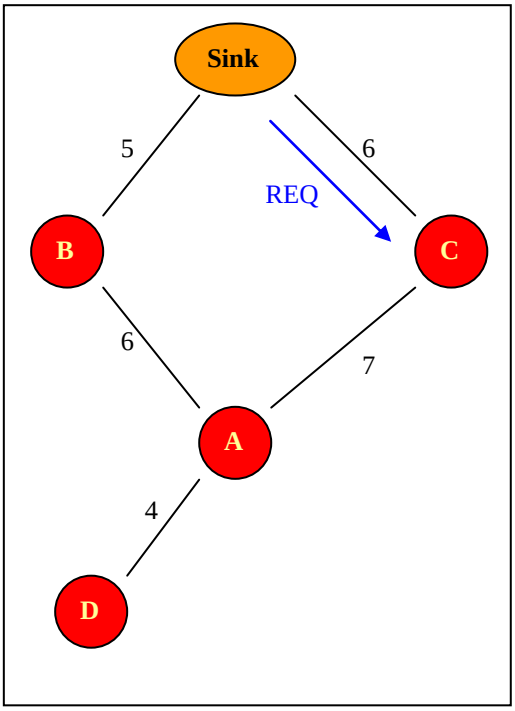
$$P_{N_C,N_{\sin ki}} = \frac{\frac{1}{C_{N_C,N_{\sin k}}}}{\frac{1}{C_{N_C,N_{\sin k}}}} = \frac{\frac{1}{6}}{\frac{1}{6}} = 1$$

The probability of sending data to sink node is 1. As a result, C sends data to sink node.

After this protocol sequence, sink node has data and does not need anymore, so Node D does not have to transmit data.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	5	INIT	INIT



Routing Table of Node A

Node	Cost	Probability

Routing Table of Node B

Node	Cost	Probability

Routing Table of Node C

Node	Cost	Probability

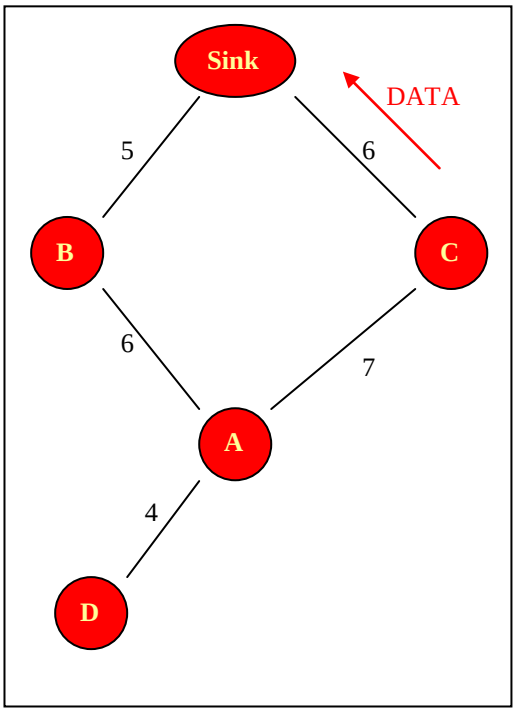
Routing Table of Node D

Node	Cost	Probability

Figure 4.12: Initial Scenario (5) – Responses for ADV from C

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	INIT	**6**	INIT



Routing Table of Node A

Node	Cost	Probability

Routing Table of Node B

Node	Cost	Probability

Routing Table of Node C

Node	Cost	Probability
Sink	6	1

Routing Table of Node D

Node	Cost	Probability

Figure 4.13: Initial Scenario (6) - Cost Calculation of C

In Figure 4.14, the responses of ADV message from D and the result of the cost calculation of D are shown. Only node A has received the ADV message from D. Node A controls the checkpoints given before and does nothing.

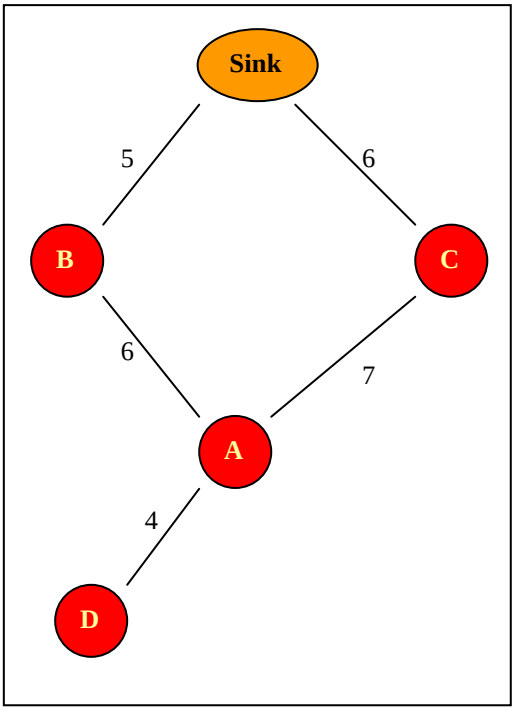
- The cost of D and A are INIT as in Figure 4.8. According to the first check, which is given before, one of the nodes have INIT cost, so sink node controls the second check.
- Node A has data and its cost is INIT, so it does not know, whether it can reach to the sink node. As a result, node A does nothing.

The behavior of Node D is shown in Figure 4.14. Node D waits until a timer with $Tout_{ADV}$ value is expired. After that, it checks the REQ messages. Because it has not received any REQ messages, it clears its routing table and set its average cost as Not Available (NA). As a result, node D does nothing.

At the end of this protocol sequence of the first data, all nodes have received this data. The routing tables and average costs of the B, C, D filled.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	5	6	**NA**



Routing Table of Node A

Node	**Cost**	**Probability**

Routing Table of Node B

Node	**Cost**	**Probability**
Sink	5	1

Routing Table of Node C

Node	**Cost**	**Probability**
Sink	6	1

Routing Table of Node D

Node	**Cost**	**Probability**

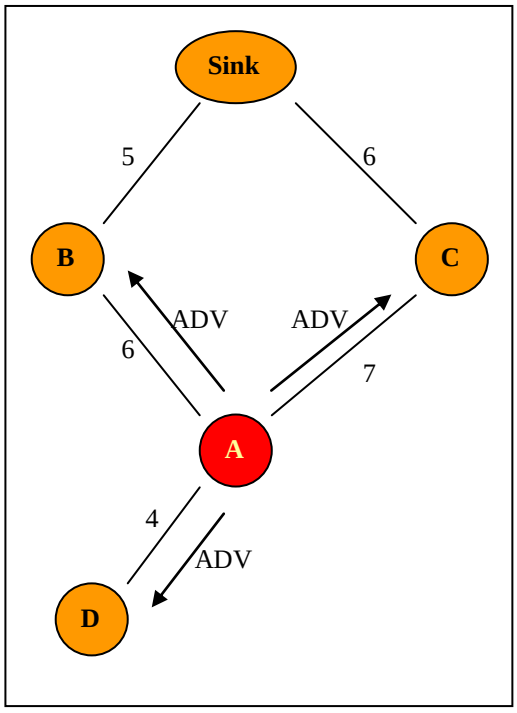
Figure 4.14: Initial Scenario (7) – Cost Calculation of D

In the previous section, first SIGNAL is sent to the sink node. After a while, node A is received another SIGNAL. Protocol sequence starts again. Node A sends ADV messages to its neighbors, which is shown in Figure 4.15.

The neighbors of Node A send REQ messages after some checks given below, which is shown in Figure 4.16. They check whether average cost of Node A or B is INIT or NO_PATH_TO_SINK. Node A has INIT cost, so they check whether it has this DATA. They do not have data, so it sends REQ message to A.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	5	6	**NA**



Routing Table of Node A

Node	**Cost**	**Probability**

Routing Table of Node B

Node	**Cost**	**Probability**
Sink	5	1

Routing Table of Node C

Node	**Cost**	**Probability**
Sink	6	1

Routing Table of Node D

Node	**Cost**	**Probability**

Figure 4.15: Initial Scenario (8) – ADV messages for second SIGNAL

The result of cost calculation and node selection of node A is shown in Figure 4.16. A receives request messages from B, C, and D and calculates the cost of sending data for each node.

- The cost of sending data to B.

  $C_{N_A,N_B} = Cost(N_B) + Metric(N_A, N_B) = 5 + 6 = 11$

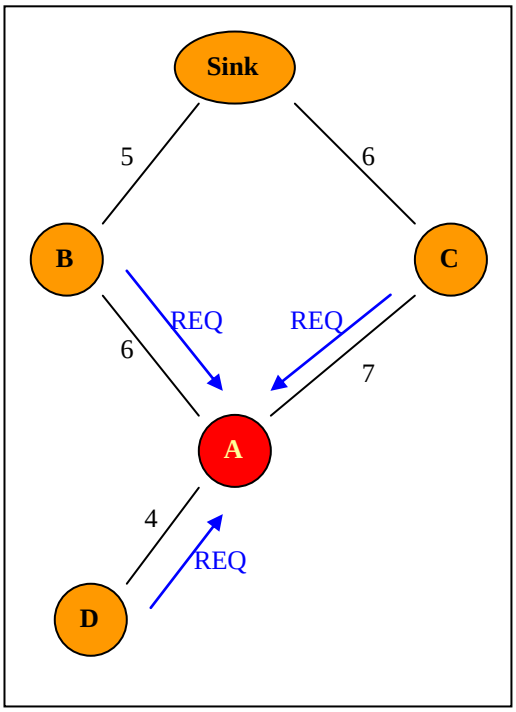
- The cost of sending data to C.

  $C_{N_A,N_C} = Cost(N_C) + Metric(N_C, N_A) = 6 + 7 = 13$

- A does not calculate the cost of sending data to D, because the average cost of node is NA (NO_PATH_TO_SINK). This shows that, D cannot reach to the SINK from another way.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	INIT	5	6	NA



Routing Table of Node A

Node	Cost	Probability

Routing Table of Node B

Node	Cost	Probability
Sink	5	1

Routing Table of Node C

Node	Cost	Probability
Sink	6	1

Routing Table of Node D

Node	Cost	Probability

Figure 4.16: Initial Scenario (9) – Responses for second SIGNAL

After that node A calculates the probability values for B and C, which is shown in Figure 4.17. This probability values are used while selecting a node to send data. Node a does not calculate this probability for Node D. Node A does not add Node D to its routing table, because the average cost of Node D shows that Node D cannot reach to sink from another way.

- The probability of B for route selection.

$$P_{N_A,N_B} = \frac{\frac{1}{C_{N_A,N_B}}}{\frac{1}{C_{N_A,N_B}} + \frac{1}{C_{N_A,N_C}}} = \frac{\frac{1}{11}}{\frac{1}{11} + \frac{1}{13}} = 0.54$$

- The probability of C for route selection.

$$P_{N_A,N_C} = \frac{\frac{1}{C_{N_A,N_C}}}{\frac{1}{C_{N_A,N_B}} + \frac{1}{C_{N_A,N_C}}} = \frac{\frac{1}{13}}{\frac{1}{11} + \frac{1}{13}} = 0.46$$

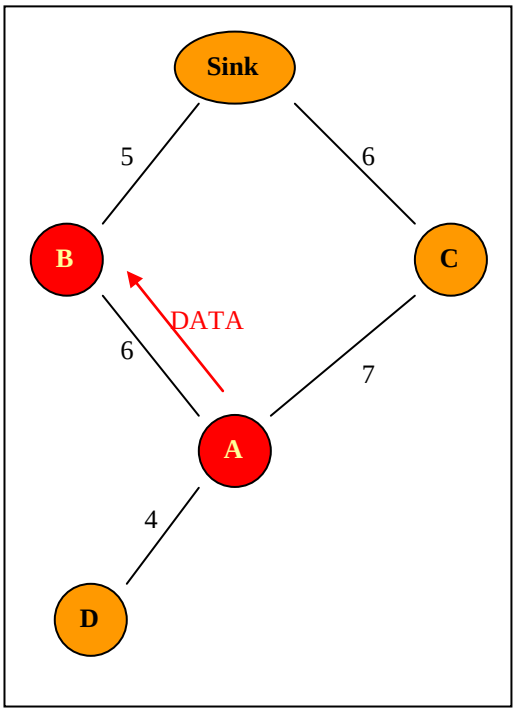
At the end Node A calculates its average cost according to the values in its routing table.

$$Cost(N_A) = \sum_{k \in RT_A} P_{N_A,N_k} C_{N_A,N_k} = P_{N_A,N_B} C_{N_A,N_B} + P_{N_A,N_C} C_{N_A,N_C} = 0.54 * 11 + 0.46 * 13 = 11.92$$

Node A has filled its routing table and has calculated its average cost. After that it selects a node to send data according to the probabilities in routing table. In this example, we choose Node B.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	11.92	5	6	**NA**



Routing Table of Node A

Node	**Cost**	**Probability**
B	11	0.54
C	13	0.46

Routing Table of Node B

Node	**Cost**	**Probability**
Sink	5	1

Routing Table of Node C

Node	**Cost**	**Probability**
Sink	6	1

Routing Table of Node D

Node	**Cost**	**Probability**

Figure 4.17: Initial Scenario (10) – Cost calculation of Node A.

4.4.2. Data Communication Scenario II

In this scenario, the protocol sequence is illustrated, when more than one node have data. Node A and C have data. The protocol sequence is shown for Node A and C separately.

In Figure 4.18, Node A sends ADV messages to its neighbors. This is part of the protocol is always same. If a node has data it broadcast an ADV message without checking anything.

After that each node controls its ADV messages according to the checkpoints. The responses of nodes for ADV message from A are shown in Figure 4.19.

- Node B checks whether the average costs of it and sender has been calculated. The cost of node A is 11.92 and cost of node B is 5, so it checks whether its cost is greater than node A's cost. Its cost is smaller than node A's cost, so it checks the meta-data. Node B does not have this data, so it sends REQ message.
- Node C sends NREQ message to node A, because its cost is smaller than node A's cost and it has data. It sends NREQ message to indicate that there is a node, which is closer than the sender.
- Although node D does not have this data, it does not send any message, because its average cost is NO_PATH_TO_SINK, which shows that it cannot reach to sink from another path.

At the end, node A does not send DATA message, because it has received a NREQ message from C.

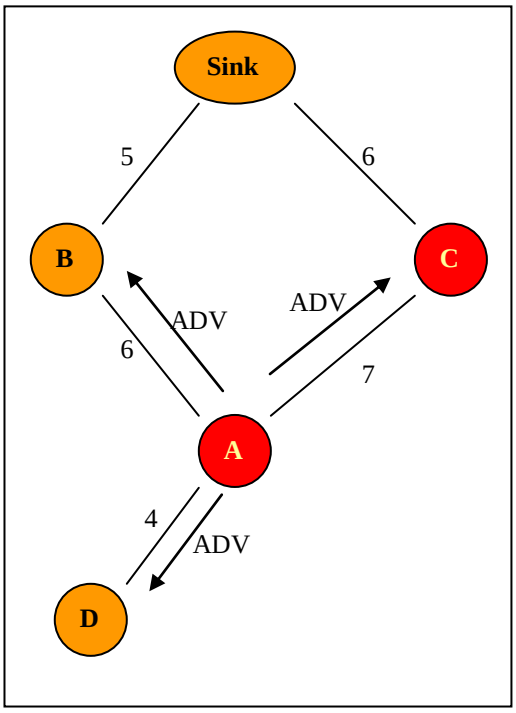
In Figure 4.20, C starts the protocol sequence. It sends ADV messages to its neighbors. After that Sink and A control the ADV message.

- Sink checks whether the average costs of it and sender has been calculated. The cost of Sink is 0 and cost of node C is 6, so Sink's cost is smaller than node C's cost, so it checks the meta-data. Sink does not have this data, so it sends REQ message.
- Node A checks whether the average costs of it and sender has been calculated. The cost of node A is 11.92 and cost of node C is 6, so it checks whether its cost is greater than node C's cost. Its cost is greater than node C's cost, so Node A does not send any message, because Node A understands, that Node C is closer to sink than it, from cost values.

At the end Node C has received a REQ message from Sink, so it sends data to it.

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	11.92	5	6	NA



Routing Table of Node A

Node	Cost	Probability
B	11	0.54
C	13	0.46

Routing Table of Node B

Node	Cost	Probability
Sink	5	1

Routing Table of Node C

Node	Cost	Probability
Sink	6	1

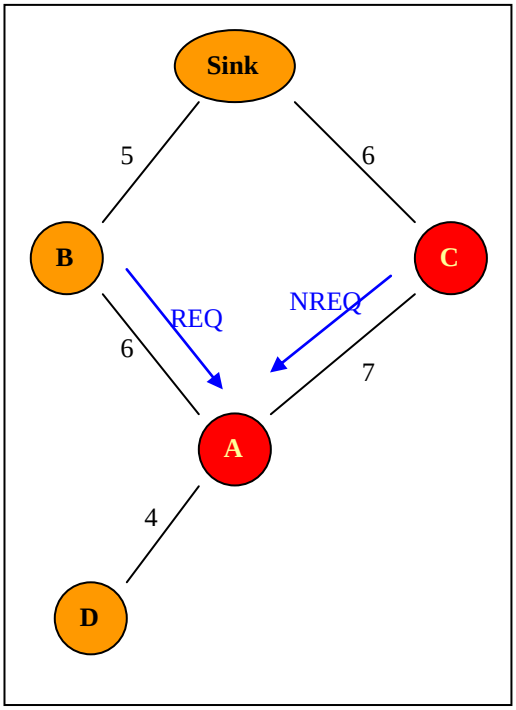
Routing Table of Node D

Node	Cost	Probability

Figure 4.18: Scenario II - ADV messages for first SIGNAL

Average Cost Table

Node	Sink	A	B	C	D
Average Cost	0	11.92	5	6	NA



Routing Table of Node A

Node	Cost	Probability
B	11	0.54
C	13	0.46

Routing Table of Node B

Node	Cost	Probability
Sink	5	1

Routing Table of Node C

Node	Cost	Probability
Sink	6	1

Routing Table of Node D

Node	Cost	Probability

Figure 4.19: Scenario II – Responses for ADV message from A.

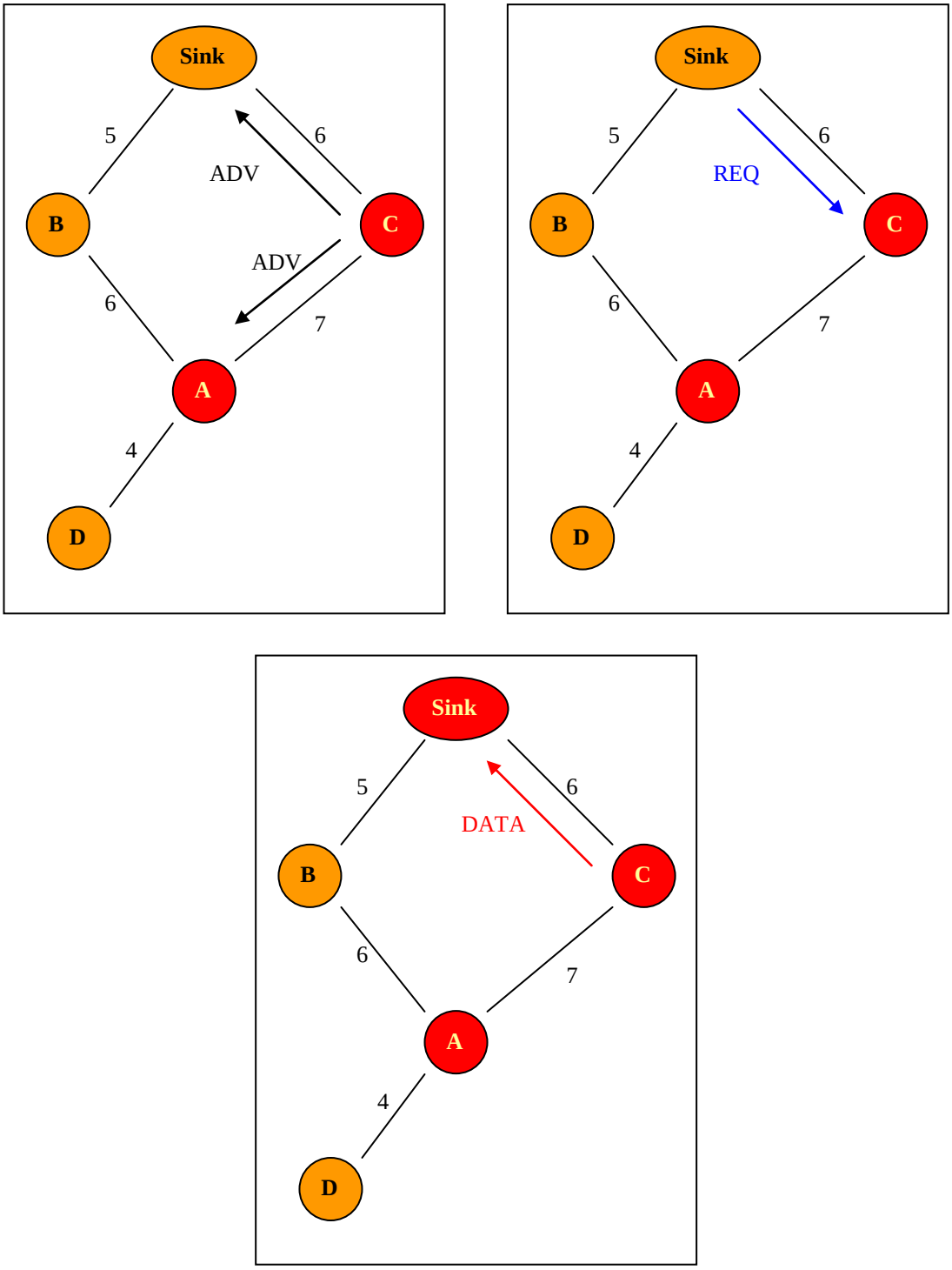


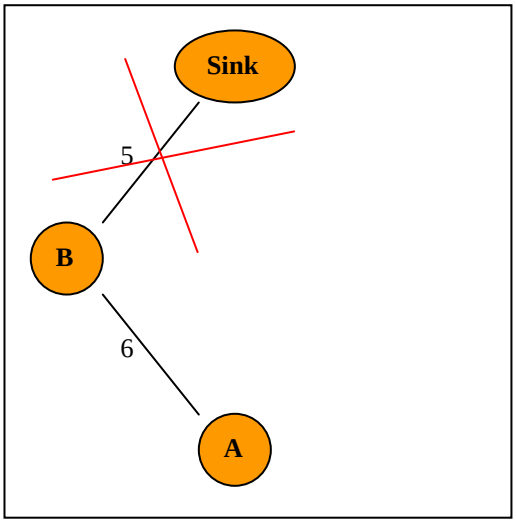
Figure 4.20: Scenario II – Route selection of C.

4.4.3 Data Communication Scenario III

In this scenario, the protocol sequence is illustrated, when the link between sink and nodes disconnect. In this situation, the nodes should understand that they couldn't reach to sink and set their average cost to NO_PATH_TO_SINK. This is important for redeployment phase of the network. If the nodes, which cannot reach to sink, do

not send and receive messages, they do not run out of energy and they can be used when new sensor nodes are deployed after while.

A sample network, which is shown in Figure 4.21, is used for illustration. There are three nodes in the network, sink, A, and B. They have initialized their routing tables and average cost values. As always in the protocol, the average cost of sink is zero. In the routing table of Node A, there is only B with cost 11 and probability 1. In the routing table of Node B, there is only sink node with cost 5 and probability 1. In this situation, the link between sink and B becomes disconnected. Sink node can be down or an environmental difficulty can be occurred.



Average Cost Table

Node	Sink	A	B
Average Cost	0	11	5

Routing Table of Node A

Node	Cost	Probability
B	11	1

Routing Table of Node B

Node	Cost	Probability
Sink	5	1

Figure 4.21: Scenario III – sample network

After that Node A received a SIGNAL message and sends ADV message to its neighbor B, which is shown in Figure 4.22.

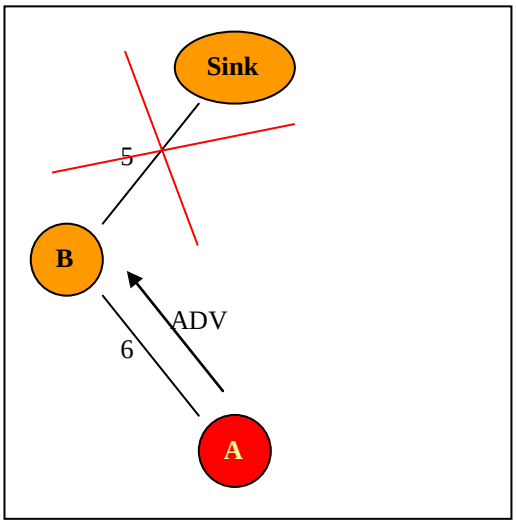
After B receives the ADV message, it checks the average cost values. Its cost is smaller than A's cost, so it checks the meta-data table. It does not have data and its cost has been calculated, so it send REQ message, which is shown in Figure 4.22. At this time, node A and B does not know that they cannot reach sink.

After A receives REQ message from B, it sends DATA message to B.

After B receives the data from A, it sends ADV messages to its neighbors, shown in Figure 4.22. It thinks that, one of its neighbors is sink node. Only Node A receives the ADV message from B. Node A checks their cost values. Node A's cost is greater

than Node B's cost, so A understands that B is closer to sink than it. Node does not send any message. Sink node does not receive ADV message anyway.

Node B waits until its ADV timer expires, and checks the REQ messages, which is shown in Figure 4.23. It does not receive any REQ messages; this means it has no neighbor, with valid cost values, so B clears its routing table and set its average cost values to NO_PATH_TO_SINK. At this time Node B understands that it cannot reach to sink node.



Average Cost Table

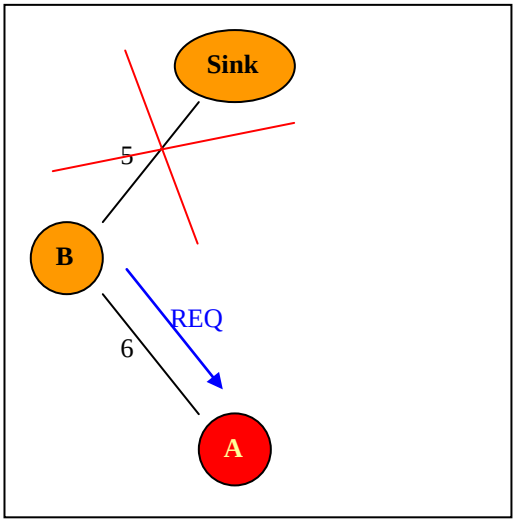
Node	Sink	A	B
Average Cost	0	11	5

Routing Table of Node A

Node	Cost	Probability
B	11	1

Routing Table of Node B

Node	Cost	Probability
Sink	5	1



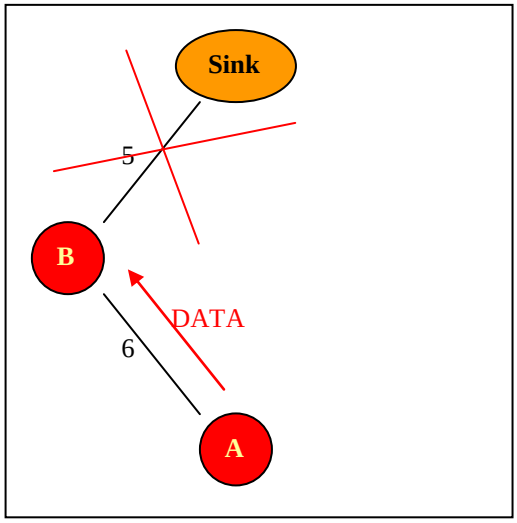
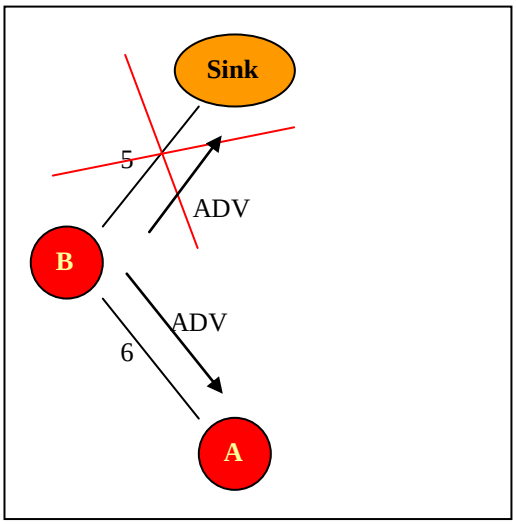


Figure 4.22: Scenario III – a

After a while, Node A receives another SIGNAL. It sends ADV message to B, which is shown in Figure 4.24. B checks its cost value, when it receives the ADV message. Its average cost value is NO_PATH_TO_SINK, so it does not send any message, because it knows that it cannot reach to sink node. Node A waits until ADV timer

expires and checks the REQ messages. It has not received any REQ message, so it clears its routing table and set its average cost value to NO_PATH_TO_SINK.



Average Cost Table

Node	Sink	A	B
Average Cost	0	11	NA

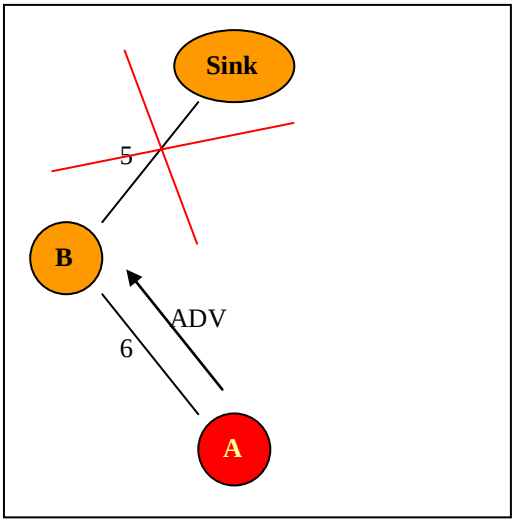
Routing Table of Node A

Node	Cost	Probability
B	11	1

Routing Table of Node B

Node	Cost	Probability
~~Sink~~	~~5~~	~~1~~

Figure 4.23: Scenario III – b



Average Cost Table

Node	Sink	A	B
Average Cost	0	NA	NA

Routing Table of Node A

Node	Cost	Probability
~~B~~	~~11~~	~~1~~

Routing Table of Node B

Node	Cost	Probability
~~Sink~~	~~5~~	~~1~~

Figure 4.24: Scenario III – c

4.5 Fault Tolerance in EA-SPIN

Failure can be in different times.

1. Failure of source node after it sends the data to the selected node and the selected node has received all data.

   Protocol tolerates this situation, because the responsibility of data is not on source node.

2. Failure of intermediates node after getting ADV message and before sending REQ message.

In this situation, the source node recognizes this failure, because the source node does not receive the REQ message. The source node updates its routing table and selects another node according to the probability.

3. Failure of intermediate nodes after sending REQ message or failure of intermediate nodes after receiving data from source node.

   If the source node selects a dead node, which is died after source sends the REQ message, then the source node sends the data to this node. After that source node begins to wait an ADV message from the node, which it send the data. If the destination node does not send an ADV message, source node decides that there is a failure and send a new message.

5. SIMULATION AND RESULTS

We use Ptolemy2 simulation framework to simulate and analysis the protocols [15]. It is a simulation framework and contains different types of simulation directories, such as discrete event and continues time. Visual-sense is an extension of Ptolemy2 for wireless sensor networks. In Figure 5.1 shows the GUI of Ptolemy2 simulation with our modules.

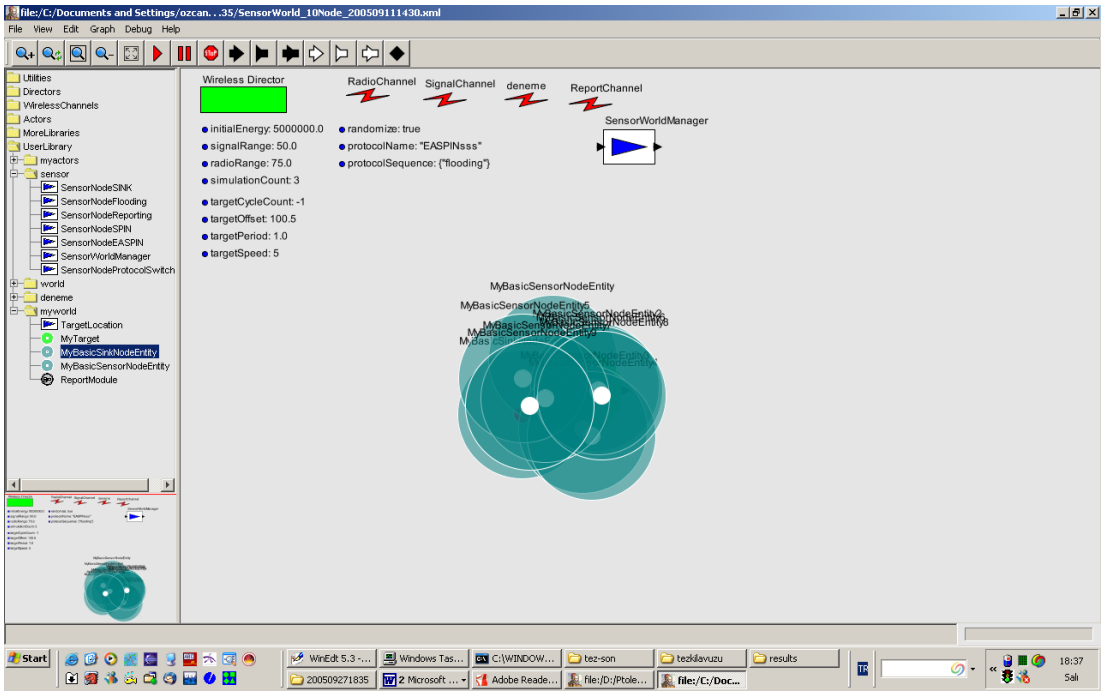


Figure 5.1: Ptolemy framework

5.1 Simulation Modules

We implemented some modules to simulate the protocols in a sensor field. Two channel are defined:

- **Radio Channel:** Sensor nodes use this channel to communicate with other sensor nodes and sink nodes.
- **Signal Channel:** Target uses this channel to send signals. Sensor nodes listen this channel to sense the target.

SensorWorldManager: This module is the manager of our simulation. It reads the parameters and run simulation according to these parameters. It scatters the sensor nodes randomly in the sensor field. Sensor nodes and targets about all events inform it. It is responsible for reporting the results. It has some parameters given below.

- **Range:** This parameter shows the sensor field. For example, {{150.0, 250.0}, {150.0, 250.0}} shows a 10000 m^2 field.
- **Seed:** This is the seed of randomize function.

SensorNode: A sensor node structure is shown in Figure 5.2. It consists of protocol modules and protocol switching modules.

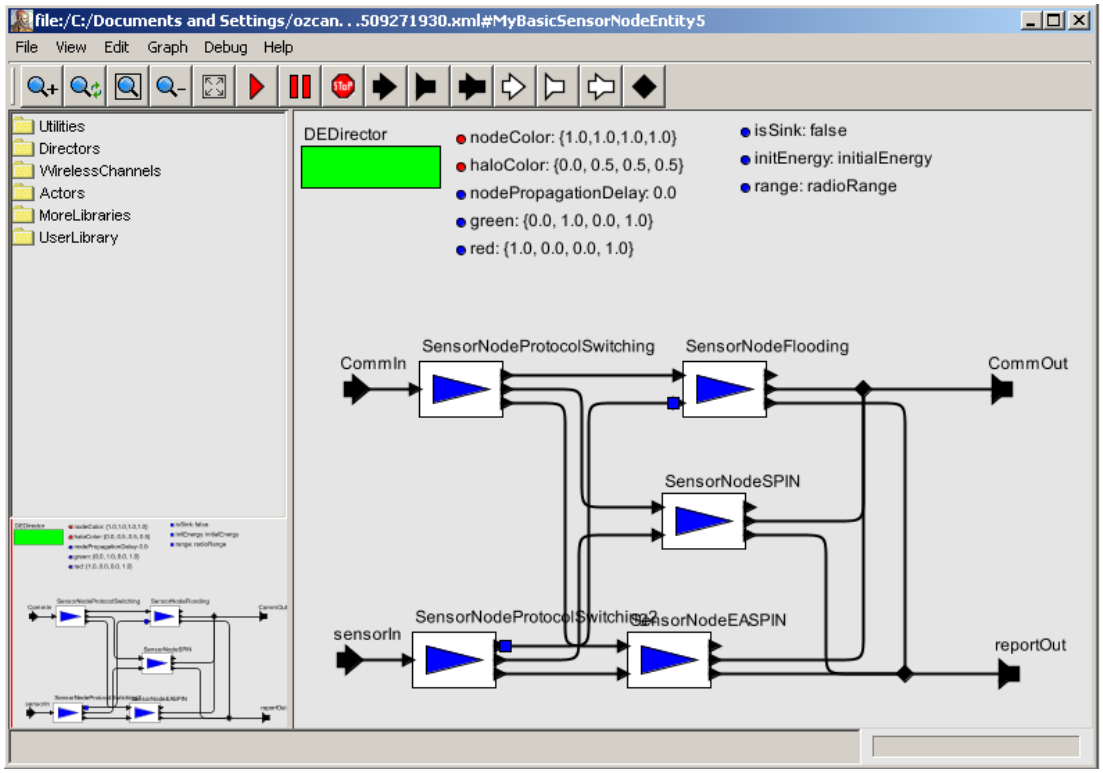


Figure 5.2: A sensor node structure

SensorNodeProtocolSwitching: This module gets the protocol name in message and sends the message to the related module.

SensorNodeFlooding: This module is the implementation of flooding.

SensorNodeSPIN: This module is the implementation of SPIN-PP.

SensorNodeEASPIN: This module is the implementation of EA-SPIN.

Target: This is the complex module of target, which is shown in Figure 5.3. It consists a clock and TargetLocation module. Every simulation time clock sends a signal to TargetLocation module. TargetLocation module sends the signal over SignalChannel.

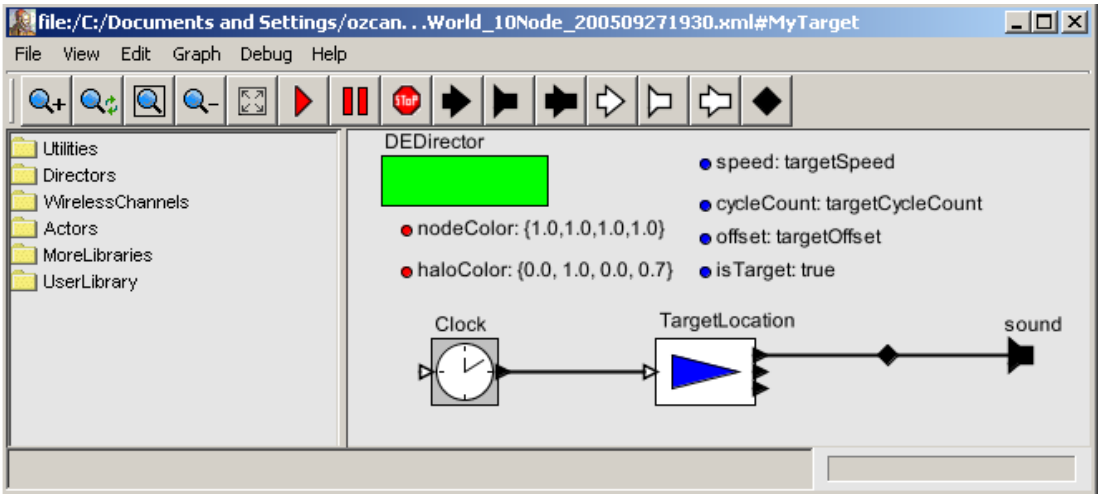


Figure 5.3: Target structure

TargetLocation: This module decides the target location. It gets the probabilities as a parameter. It selects a random value. According to this value and the probabilities of routes, it decides new location. As a result, target walks in a sensor field.

Simulation Parameters: Some of the simulation parameters are explained below.

- **initialEnergy:** This is the initial energy of sensor nodes.
- **signalRange:** This the target range. Sensor nodes in this range of target can hear the signals.
- **radioRange:** This is the range of communication channel of sensor nodes.
- **targetCycleCount:** The count of the target signal in a simulation.
- **targetOffset:** Target starts sending signals from this simulation time.
- **targetPeriod:** This is the period of signal, which is send by target.
- **targetSpeed:** This is the length of one target step.

5.2 Simulation Results

In our experiments, we constitute in a 10000 m^2 sensor field, which contains sensor nodes, a sink node and a target.

Sensor Field: In the initializing phase of simulations SensorWorldManager randomizes the location of sensor nodes, sink and target, if their randomize parameter is true. In our simulations, the same seed of the randomize function is 13L and range of field is {{150.0, 250.0}, {150.0, 250.0}}. According to this range values, X value of the node locations are uniformly distributed between 150 and 250, and Y value of the node locations are uniformly distributed between 150 and 250.

Initial Energy: The initial energy of nodes is set to 100 j. A sensor node is died if its energy is under a threshold. This threshold is set to 50 nj in our examples.

Radio Range: The radio range is set to 50 units. Sensors can send messages in this range.

Signal Range: Signal range is set to 50 units. Target send messages to the sensor nodes in this range.

Target Movement: Target moves randomly in sensor field according to a random seed. Each simulation is run 3 times with different target movements. The results are the average of these simulation results.

Energy Consumption: The energy level of a node is reduced, when it receives or sends a message. It is dependant on the message length and the distance to send. In our experiments, we are used the message length for data message is 500 bytes, which is 4000 bits, and for meta-data message is 16 bytes, which is 128 bits [10].

- **Transmitting:** The energy consumption for transmitting per bit is calculated with this formula given below (5.1) [16].

$$e_{t_x} = e_{t_e} + e_{t_a} d^{\alpha} \qquad (5.1)$$

  The parameters of this formula are explained below.

  - **e_{tx}:** This is the energy consumption while transmitting one bit.
  - **e_{te}:** This is the energy consumption of the transmitter electronics. This parameter is set 20 nj/bit [8].

- **e_{ta}:** This is the energy consumption of the transmit amplifier. This parameter is set 1 pj/bit [8].
- **d:** transmission distance. The receiver node according to the reduced energy calculates distance. In Flooding and SPIN, the distance is predefined. All messages are send with same energy, but in EASPIN the ADV and REQ messages are sending with maximum energy level. At this time, the receiver node calculates the distance and sends DATA according to this distance.
- **α :** The path-loss exponent. This parameter can change between 2 and 4. We set it 2 in our experiments.

The examples for energy consumption with distance 50 meters are given below.

- **Transmitting Data:**

  MessageLength*($e_{tx}=e_{te}+e_{ta}*d^{\alpha}$)=4000*(20nj+1pj*$50^2$)=80010 nj

- **Transmitting Meta-data:**

  MessageLength*($e_{tx}=e_{te}+e_{ta}*d^{\alpha}$)=128*(20nj+1pj*$50^2$)=2560.3 nj

- **Receiving:** A constant value, 30 nj/bit, is used for energy consumption of receiving per bit [8]. The energy consumption calculations for receiving data and meta-data are given below.
  - **Receiving Data:**

    MessageLength*(e_{rx})=4000*(30nj)=120000nj

  - **Receiving Meta-Data:**

    MessageLength*(e_{rx})=128*(30nj)=3840nj

5.2.1 Effects of sensor node count

Sensor node count is a parameter for simulations. We run the simulation with 10, 20, and 30 sensor nodes. The sensor field is not enlarged, so the sensor node density is increased. This means a node can hear more messages. The network lifetime (NLT) is our measurement parameter. The network lifetime is defined as the time of first

node dies. The simulations are run 3 times for all node count value with changing target movements and the average value is used as a result.

The results are given in table 5.1 and Figure 5.4. It can be seen that network lifetime in flooding is the worst results. SPIN has better results and EASPIN has the best results. In Appendix A, other results are shown and they are similar to given below.

Table 5.1: The Network Life Time versus node count

Protocol \ Node Count	10	20	30
Flooding	142,5	53,8	34,5
SPIN	362,8	138,2	94,8
EASPIN	645,5	437,5	362,8

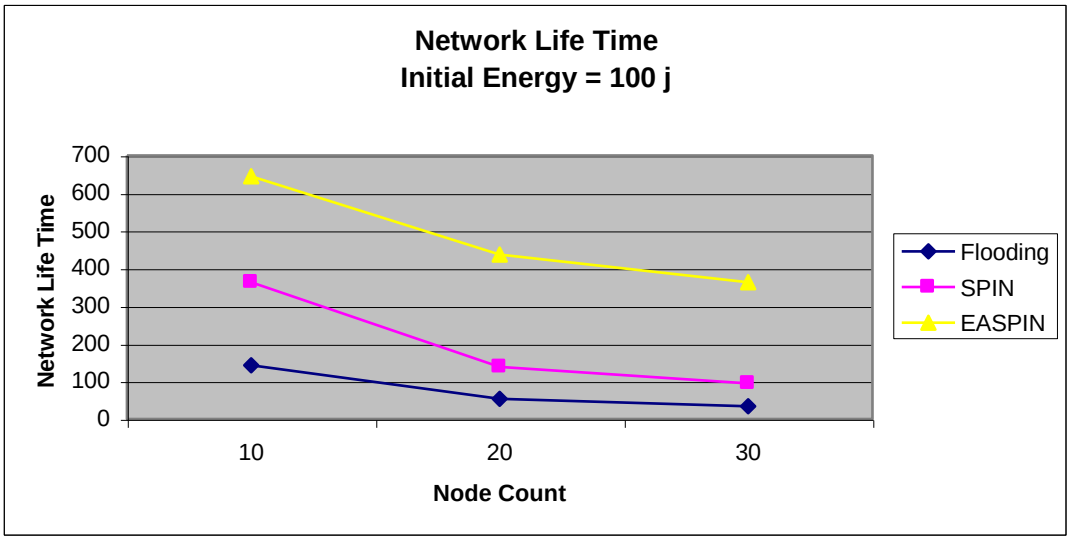


Figure 5.4: Network Life Time versus node count

5.2.2 Effects of network lifetime definition

In previous section, the network lifetime is defined as the time when one node is dead, but a network can continue do its tasks, although one node dies. Basically, after some nodes die, the links can be disconnect. We define NTL, the time when a

proportion of nodes are dead. This is a measurement parameter for our protocols, because the behavior of the protocols can be change.

We run the simulations in a network with 20 sensor nodes. The NTL is defined as time when 10%, 15%, and %20 of sensor nodes are dead. The results can be seen with a table in table 5.2 and with a graph in Figure 5.5. We can see that in the difference in flooding is too small. There is an improvement in SPIN. In EASPIN, the NTL is increased almost twice. It can be seen in graphic that EASPIN is better for greater proportion.

Table 5.2: The Network Life Time versus proportion of dead node

Protocol \ Proportion	First Node	10%	15%	20%
Flooding	53,8	55,8	57,83	58,2
SPIN	138,2	148,8	159,5	159,5
EASPIN	437,5	461,5	495,4	556,8

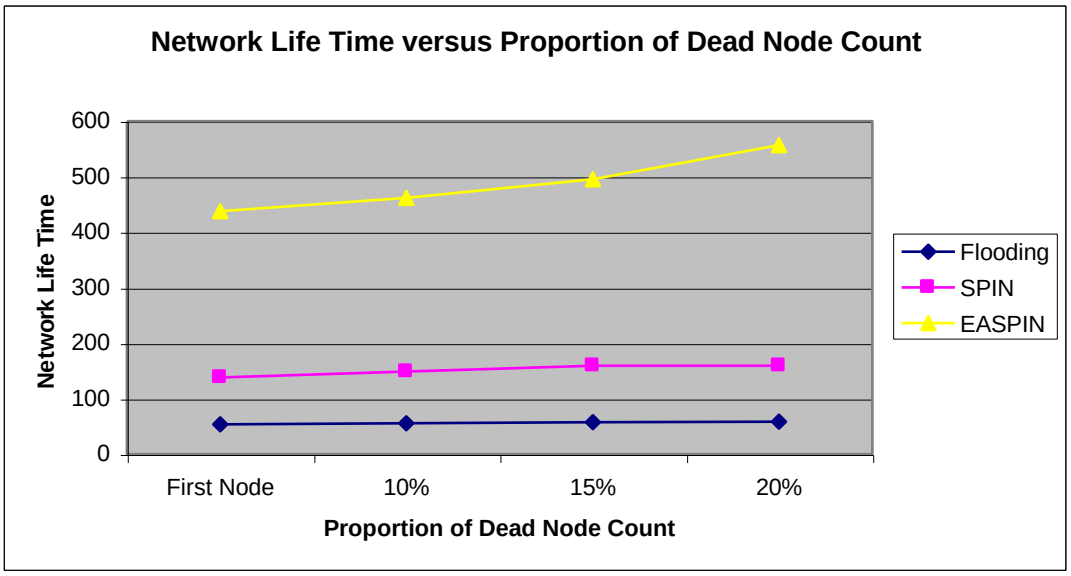


Figure 5.5: Network lifetime versus proportion of dead node

In flooding, only DATA messages to the dead nodes are reduced, so we see a little difference. In SPIN, there is a small improvement thanks to negotiation, but it is not close to EASPIN because it is not sink and energy aware. However, EASPIN is sink aware, so the DATA messages do not send to the nodes, which are from sink node.

6. CONCLUSION

Power efficiency is one of the critical design factors for WSN because of its architecture and constraints. Energy efficiency usually considered in Physical and MAC layer in networks. Unlike other network types, such as Ethernet, energy consumption must be considered in all layers in WSN.

In this thesis, we considered network layer and analyzed the routing protocols for WSN. There is not a protocol, which is suitable for all WSN applications because of the WSN characteristics. Protocol designer must consider the application requirements. It can be a time critical application, which is used in short term or the lifetime is important for it.

We presented a new data dissemination protocol called Energy Aware SPIN, for wireless sensor networks. The goal of the protocol is increase the network lifetime. This protocol combines the strong features of EAR and SPIN. Unlike SPIN, it is energy and sink node aware. SPIN already solves the implosion and overlap problems via negotiation, so EASPIN do not have these problems, because it gets negotiation mechanism from SPIN.

However, EASPIN has a different approach. The goal of SPIN is dissemination data to all nodes in network, whereas EASPIN try to transmit data to sink node. It calculates the cost of paths to the sink node and use one single path to send data. This path is selected inversely proportional of the costs. The cost calculation phase is inherited from EAR and modified some parts.

6.1 Simulations

In experimental results, it can be seen that total energy consumption of EA-SPIN is less than SPIN and flooding, because it reduces the redundant data transmissions, so the network lifetime is increased. The amount of changes are different for different parameters, but generally the network life time became twice or greater according to the SPIN.

We run the simulation with different node count to increase the density. The characteristics of result did not change. EASPIN has the best network lifetime value. The simulation has run with 20 nodes and different network lifetime definitions. It is assumed that the network is dead when 10%, 15%, and 20% of nodes are dead. We can see that EASPIN gives better results, while the proportion is increased because EASPIN is sink aware. If a node does not have a path to sink, it does not require data. If network is divided into subnets, which cannot be reach to sink, after some nodes are dead, data are not transmitted. Therefore, in a post deployment phase, these nodes can be used also.

6.2 Future Work

For future work, some open issues should be considered and analyzed.

- We propose to add multi-path approach to EA-SPIN. This can give better results for fault tolerance.
- Scheduling algorithms should be considered.
- Fault tolerance issues shall be analyzed. Some nodes die because of the environmental problems and energy consumption. Also, there can be a channel with loss or collusion. The behaviors of EA-SPIN in these situations shall be analyzed.

REFERENCES

[1] Jiang, Q. And Manivannan, D., 2004. Routing Protocols for Sensor Networks. *Consumer Communications and Networking Conference.*

[2] Akyildiz, I., Su, W., Sankarasubramaniam, Y. And Çayırcı, E., 2002. A Survey on Sensor Networks, *IEEE Communicaitons Magazine,* pp. 102-114

[3] Culler, D., Estring, D., and Srivastava, M., 2004. Overview of Sensor Networks, IEEE Computer, vol. 37, no 8, pp.41-49

[4] Al-Karaki, J. And Kamal, A., 2004. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communications*.

[5] Çöplü, T., 2004. Distributed spatial data aggregation and dilution based on hashing and relational algebra in wireless sensor networks. *Master's thesis,* ITU, Institute of Informatics, İstanbul.

[6] Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J. and Silva, F., 2003. Directed Diffusion for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking,* vol. 11, pp.2-16

[7] Lindsey, S. and Raghavendra, C., 2001. PEGASIS:Power-Efficient Gathering in Sensor Information Systems. *International Conference on Communications.*

[8] Shah, R. C. and Rabaey, M., 2002. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *IEEE Wireless Communications and Networking Conference (WCNC),* pp. 17-21, Orlando, FL.

[9] Braginsky, D. and D.Estrin, 2002. Rumor Routing Algorithm for Sensor Networks. Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia.

[10] Heinzelman, W., Kulik, J. and Balakrishnan, H., 2002. Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks. *Kluwer Academic Publisher Wireless Networks 8, 169-185.*

[11] Das, S., Perkins, C. and Belding-Royer, E., 2003. Ad hoc on demand distance vector (AODV) routing. *Network Working Group Request for Comments 3561.*

[12] Kaya, O. S., 2004. Query dissemination and processing in wireless sensor networks. *Master's thesis,* ITU, Institute of Science and Technology, İstanbul.

[13] Khanna, G., Bagchi, S. and Wu, Y., 2004. Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Networks. *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*

[14] Perrig, A., Szewczyk, R., Wen, V., Culler, D. and Tygar, J., 2001. SPINS: Security Protocols for Sensor Networks. *Proceedings of the7th Annual International Conf. On Mobile Computing and Networking.*

[15] Baldwin, P., **Kohli, S. and Lee, E. A.,** 2004. Modeling of Sensor Nets in Ptolemy II. *Proceedings of Information Processing in Sensor Networks (IPSN), Berkeley, CA, USA.*

[16] Haapola, J., Shelby, Z., Pomalaza-Raez, C., Mahonen, P., 2005. Cross-layer Energy Analysis of Multi-hop Wireless Sensor Networks. *Proceedings of the Second European Workshop.*

APPENDIX A: SIMULATION RESULTS

Table A.1: The Network Life Time versus node count. Target Count = 2

Node Count / Protocol	10	20	30
Flooding	96,5	26,5	19,5
SPIN	269,5	64,5	51,8
EASPIN	310,36	222,2	147,0

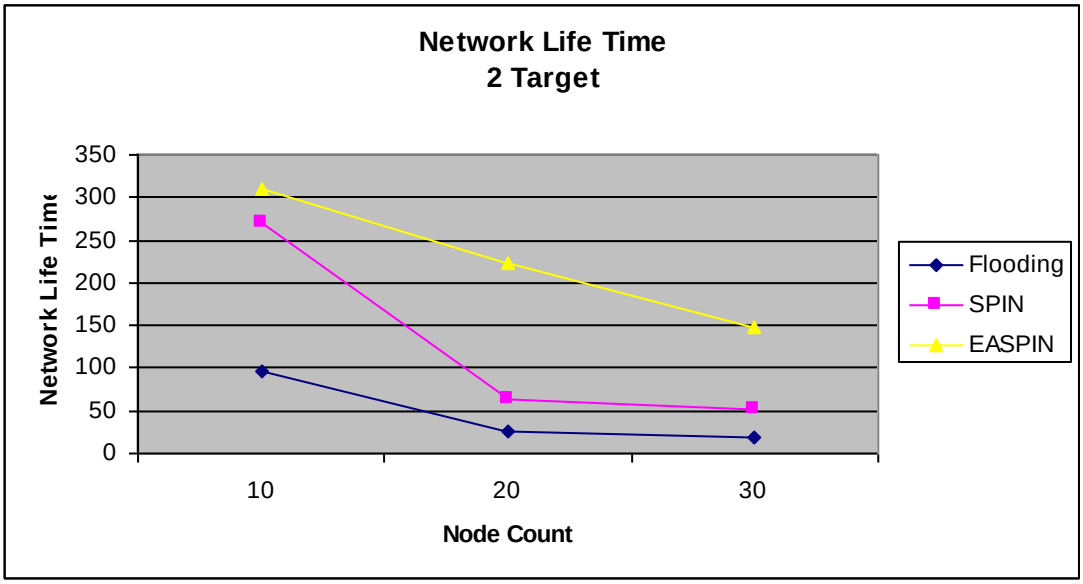


Figure A.1: Network Life Time versus node count. Target Count = 2

Table A.2: The Network Life Time versus node count. Sink Count = 2

Node Count / Protocol	10	20	30
Flooding	272,2	107,2	33,5
SPIN	756,6	282,6	91,5
EASPIN	1288,5	792,2	298,8

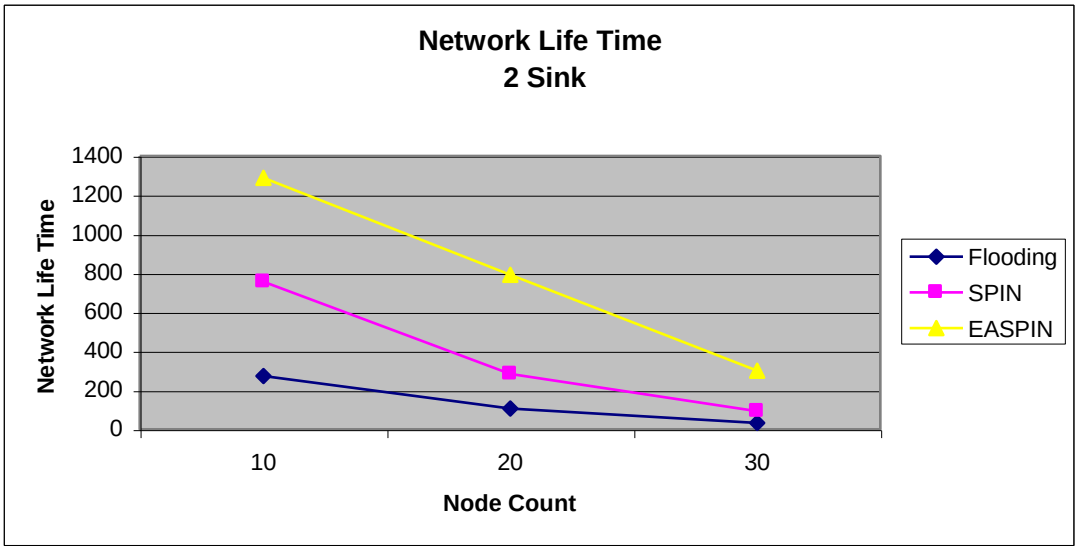


Figure A.2: Network Life Time versus node count. Sink Node Count = 2

AUTOBIOGRAPHY

I was born in Istanbul at 11 May 1979. I graduated from Istanbul Bahcelievler Anatolian High School at 1997 and had my BSc degree from Control and Computer Engineering Department of _Istanbul Technical University at June 2002. In September 2002, I was accepted to Computer Engineering M. Sc. Program at Istanbul Technical University. During that period, I have worked in AKT, Oksijen Technologies, and Argela Technologies as a software engineer. During that period, I worked on wireless networks under the supervision of Assoc. Prof. Dr. Feza Buzluca.