PERFORMANCE EVALUATION OF AD-HOC ROUTING PROTOCOLS IN UNDERWATER ACOUSTIC SENSOR NETWORKS

By

Naveed Bakhsh Qadri

2006-NUST-MS PhD-ComE-12

Thesis Advisor

Dr. Ghalib A. Shah

A thesis submitted to the faculty of Computer Engineering Department College of Electrical & Mechanical Engineering, National University of Sciences and Technology, Pakistan in partial fulfillment of the requirements for the degree of MSc in Computer Engineering
2010

Abstract

Underwater acoustic sensor networks(UWASNs) is an emerging technology, comprising of sensor nodes and unattended automated vehicles (AUVs), all working in a collaboration to sense various phenomenon, process digital information, store processed data and communicate among each other and base stations. UWASNs have the capability and potential of supporting large set of applications ranging from oceanic geographical surveys to tactical surveillance. Underwater Acoustic propagation is characterized by high and variable delays, fading effect, Doppler spread and multi path which in turn lead to a limited bandwidth and high error rates. At the same time, battery life of the sensor nodes and their data storage capacity is limited. So there is a need to find a suitable routing protocol that takes all these limitations into consideration and makes communication in underwater networks viable. In this paper we focus on existing mobile ad-hoc routing protocols which are widely accepted and have been tested across the globe. This is the first attempt to analyze the performance of these protocols in underwater acoustic networks environment. The first challenge is to come up with a reliable simulation environment for underwater networks in ns-2. Currently, ns-2 does not support simulation for underwater networks. We extended ns-2 for underwater networks by adding underwater propagation, network interface (data link layer) and underwater physical models. After having a working underwater networks simulation model, we then proceeded with our study. We used performance metrics like packet delivery ratio, average end-to-end delay, throughput, routing overhead and energy consumption of the sensor nodes. AODV, DSDV, DSR and OLSR are compared for their performance at different traffic conditions, number of nodes and depths. By analyzing our simulation results,

we found that AODV is recommended for denser underwater networks but with less traffic. DSDV is suitable for higher traffic conditions with optimal number of nodes.

Acknowledgements

All praise to Allah-The Almighty for His blessings and help. Without His guidance and willingness it was not possible to complete the thesis.

I am greatly in debt to all my teachers at College of Electrical & Mechanical Engineering, NUST for their dedication towards teaching and making me not only a better computer professional but also a good human being.

I express my profound gratitude to my respectable supervisor Dr. Ghalib A. Shah for his encouragement, co-operation, invaluable guidance and expert opinions during the project. I would also like to thank the members of guidance and examination committee Dr. Farooque Azam, Dr. Rashid Ahmad and Dr. Asia Khanum for their support, encouragement and valuable input.

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

This chapter gives a highlight of the research conducted. Overview of the background, scope and objectives of the research and organization of the thesis is presented in this chapter.

1.1 Overview

Underwater acoustic sensor networks (UWASN) is an emerging technology and a lot of work all across the globe is going on in this field. In order to clearly understand the dynamics and nuts & bolts of UWASNs, it is very important to grasp a knowledge of terrestrial wireless sensor networks, acoustic propagation in water, channel characteristics and challenges associated in designing an UWASN. The routing in wireless networks is a challenging problem because wireless networks experience larger and variable delays, plus the mobility of nodes magnifies the challenge. But as we take a wireless sensor network underwater, the routing becomes an even greater challenge.

1.1.1 Wireless Sensor Networks

A wireless sensor network comprises of autonomous sensors spatially distributed over a certain area. These sensors work in a cooperative manner to monitor physical or environmental conditions like temperature, pressure, vibrations, humidity, motion or chemicals etc. A sensor network usually constitutes a wireless ad-hoc network, meaning that the network architecture is infrastructure less and each sensor supports multi-hop routing protocols because several nodes may forward data to the base station. Each sensor in the network is equipped with a transceiver for wireless communication, a micro-controller and a battery. Sensor nodes vary in sizes and can be as small as a grain. Prices of sensors also vary according to the size and functionality incorporated in them. By constraining the size and cost of a sensor node, constraints are put on resources like energy, processing power, memory and bandwidth. A typical wireless sensor network is show in Figure 1 where sensor nodes are sensing the environmental phenomenon and relaying the collected data wirelessly to a gateway sensor over a multi hop path. The gateway sensor is connected to a base station where the collected data is gathered and converted into information.

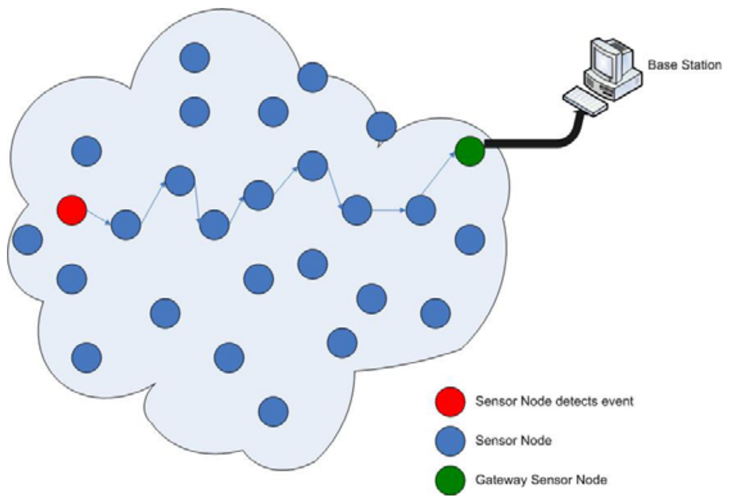


Figure 1: A typical wireless sensor network.

Wireless sensor networks find their way to a variety of applications in our lives. They are used for environmental monitoring, area monitoring and industrial monitoring. Monitoring climatic changes like change in temperature at glaciers, temperature change patterns around active volcanoes, seismic activities along the shores are few examples from a very long list. In area monitoring, sensor networks are widely used in early warning systems where a specific area is monitored for movement. Sensor networks can be deployed in battle fields to detect intrusions and thus give a tactical advantage over the enemy. In industrial and agricultural fields, wireless sensor networks are deployed to detect various phenomenons which help in fine tuning the processes.

Some of the key characteristics of wireless sensor networks are as follow [14] :

- Sensor nodes have limited power.

- Sensor nodes are rugged in nature and can with stand harsh environmental conditions.
- Sensor networks are prone to failures but they have the capability to cope with frequent node failures.
- Nodes with in a wireless sensor network can be stationary/fixed or can be mobile.
- The network topology of wireless sensor networks is always changing and very dynamic in nature.
- Usually WSNs are deployed over a large area with high spatial density.
- Wireless sensor networks are intended for 'un-manned' operations.
- Wireless sensor networks are highly scalable in terms of node capacity; however bandwidth of gateway nodes may be a limiting factor.

1.1.2 Underwater Acoustic Communication

Underwater acoustic communication is the technique of sending and receiving messages underwater using acoustical signal (sound waves). Underwater communication technology using sound waves was experimented upon by the American Navy during World War 2. This was later successfully employed for communications between ships and submarines. Sound waves exhibit maximum efficiency in traversing water i.e. sound waves can travel through water with minimum losses in amplitude of the wave front as compared to optical and electro-magnetic (EM) waves [2]

Sound waves travel as mechanical vibrations through water. Speed of propagation of sound waves and their attenuation with distance traversed depend on the density of the medium. Since water is denser than air, speed of sound is faster and its attenuation rate is lower in water as compared to air, so the acoustic signal can traverse faster and farther in water than air.

1.1.2.1 Underwater Acoustic Sensor Networks

Underwater acoustic sensor networks consist of small, intelligent; battery powered digital devices called sensor nodes. These devices carry different types of sensors and can communicate with each other wirelessly [1] . Efficiency of communication is increased by cluster formation algorithms. Nodes with-in good signal receiving distances of one another form a cluster with one node as the cluster head or underwater-sink. All nodes communicate with their respective cluster heads only, using TDMA techniques due to its good short range communication properties while cluster heads communicate with each other using CDMA [15] because of the good long distance communication properties of the CDMA MAC protocol.

1.1.2.2 Electromagnetic and Optical waves in water

Electromagnetic wave (EM), which is the most suited means of wave propagation in terrestrial networks show poor results in water. Although its speed of propagation in water is almost the same as the speed of light, but the sea water is saline and therefore conductive. This conductive nature makes the medium act like a capacitor. An electromagnetic wave passing through a capacitive medium can be modeled as passing through a low pass filter. Higher frequencies will get attenuated more than lower ones. This means EM signals with low frequencies exhibit deeper penetrations through water than higher frequencies. E.g. experimentation with MICA Motes shows that using a carrier frequency of 433 MHz, a MICA Mote can communicate up to 120 cm in water. Conversely an EM wave of 30 Hz will penetrate very long distances but would require high power and very big antennas. [15] Laser can be modulated and used in line-of-sight communication systems above surface. Water due to its transparent nature can be used for laser communications but in underwater environments, due to movement of underwater currents, sudden changes in temperatures can be expected. This forms layers of dense and rare mediums between communicating devices.

Laser signals entering to/ from rare and denser mediums will experience refraction and may not reach the intended receiver at all. This is one of the main reasons for using acoustic signals at MAC layer in underwater channels.

1.1.2.3 Underwater Channel Characteristics

Sea water behaves different from fresh water. In seas and oceans, unequal warming by sun causes temperature changes and produces “layers” of water at different temperatures. This also cause huge water movements called ocean currents. The biggest and continuous current flows in the Atlantic Ocean called the Gulf Stream which starts from the continent of Africa and ends on the coast of Europe. Similarly sea beds may contain salt rocks which get dissolved in water and increase salt concentrations in that area. This will cause salinity levels of water in that area more than the surrounding water and thereby causing a layer of water with different conductance. Sea water therefore cannot be treated as a homogenous medium; it has “*pockets*” or “*layers*” of regions with varying physical and chemical properties. Ocean currents move these different pockets or layers causing continuous changes in the homogeneity of the medium. A long distance signal traveling through this medium will experience changes in propagation speed, diffraction, spreading, fading, attenuation, distortion etc. Design of communication systems therefore have to include all these “peculiarities” of the medium in order for it to be functional.

1.1.3 Ad-hoc Routing Protocols

The protocols performing routing activities in mobile ad-hoc networks are referred as ad-hoc routing protocols. In the start, nodes in an ad-hoc network are not familiar with their network topologies. Ad-hoc routing protocols enable them to discover it. On joining an ad-hoc network, a new node may announce its presence by broadcasting and should listen to announcements made by neighboring nodes. Once a node learns about neighboring nodes and cost involved in reaching them, this information can be exchanged and thus routes to all the nodes in the network can be built. This may sound a very easy task but it is a very complicated task as topology of an ad-hoc network is changing all the time. So there is a need of routing protocol that can cope with such conditions and ad-hoc routing protocols does the job. Four ad-hoc routing protocols are studied in this research are discussed in detail in the coming sections. These routing protocols are Ad-hoc On Demand Distance Vector routing protocol (AODV), Destination Sequenced Distance Vector routing protocol (DSDV), Dynamic Source Routing protocol (DSR) and Optimized Link State Routing protocols (OLSR). These protocols are widely used in wireless sensor networks and are globally accepted.

1.2 Scope of the research

Network layer is responsible for determining a path between a source and destination. Some of the characteristics like extremely high and variable delays are better addressed at network layer. During the past recent years, huge advancements have been made in the routing protocols for ad-hoc wireless networks and sensor networks. Due to the different nature of underwater networks environment and specific requirements of the related applications, it is very much likely that these existing mobile ad-hoc protocols may not be suitable for underwater networks. In this research we tend to explore this. None of these protocols have yet been tested in underwater networks. This is one of the first attempts of its kind. Four widely accepted and globally tested ad-hoc routing protocols, Dynamic Source Routing Protocol (DSR), Destination Sequenced Distance Vector Protocol (DSDV), Ad-hoc On Demand Distance Vector Routing Protocol (AODV) and Optimized Link State Routing Protocol (OLSR) are selected for this study. These protocols will be studied in context with the underwater sensor networks. Their performance and behavior will be studied and compared among themselves.

1.3 Research Objectives

The major objective of this research study is to analyze the performance of mobile ad-hoc routing protocols in underwater acoustic sensor networks. The intention here is to find how these protocols react when introduced to underwater networks environment. We are also interested to find what network conditions affect the performance of routing protocols and how their performance is affected. Based on various metrics, routing protocols will be compared with each other for their performance. This comparison will be made with respect to every network condition and on overall basis. At the end of the study we'll be able to see a summarized comparison of performance of routing protocols. It is part of our objective to find the reason for depicted behavior by each routing protocol and possibly suggest improvements that can help to improve the overall performance of routing protocols in underwater acoustic sensor networks.

1.4 Thesis Organization

In Chapter 1 an overall introduction of the research work is presented. An overview of underwater acoustic sensor networks is given along with a brief discussion about wireless sensor networks and ad-hoc routing protocols. The problem statement is also made in this chapter followed by defining the overall objectives of this research work. Chapter 2 discusses the four mobile ad-hoc routing protocols selected for this study. Their working and operating details is discussed briefly in this chapter. A detail about underwater acoustic sensor network is provided. The basics of acoustic communications are discussed here. The architecture of the underwater sensor networks along with the challenges involved in designing and implementing underwater networks is presented in this chapter. Chapter 2 also discusses the applications of UWASNs in our daily life. Network Simulator is introduced. The major components and features of ns-2 are discussed here. In chapter 3, the underwater simulation model for ns-2 is discussed along with the metrics used to evaluate the performance of routing protocols. The simulation scenarios designed to evaluate the effect of different traffic conditions, different node depths and different number of nodes is discussed in detail in this chapter. In chapter 4 captured results are presented in graphical and analytical form. Discussion is made on the obtained results. Results are discussed with respect to the metrics discussed in chapter 4. Based on the gathered results comparisons are made among the routing protocols. In chapter 5, results are concluded and future work is proposed.

Chapter 2

Literature Review

This chapter includes the summary of mobile ad-hoc routing protocols, underwater acoustic sensor networks and network simulator 2. The chapter encompasses the background work related to our area of research.

2.1 Mobile Ad-hoc Routing Protocols Studied

Mobile ad-hoc networks constitute a number of nodes, which have the capability of communicating over wireless medium and thus forming an arbitrary and dynamic network with wireless links. The topology of mobile ad-hoc networks is always changing, as nodes are allowed to leave or join at any time. Due to the mobility of the nodes, routing in ad-hoc networks is a challenging problem. Nodes in ad-hoc networks do not start out familiar of their network's topology, instead they have to discover it. It is expected from a routing protocol to satisfy the specific requirements of mobile ad-hoc networks. The two conceptual approaches used in ad-hoc routing are proactive and reactive approach. In proactive approach, nodes periodically exchange messages that contain network information. These exchanges are made irrespective of the fact whether a route is required or not. The other approach is reactive where nodes exchange messages containing network information only when it's needed. Such an approach may cause the packet latency to increase because some time will be required to discover the routes. However, reactive routing protocols significantly overcome the wasted effort required in maintaining unused routes. In this work, four widely accepted and tested routing protocols are selected for studying their behavior in underwater networks. Dynamic source routing protocol, ad-hoc on-demand distance vector routing protocol, destination sequenced distance vector routing and optimized link state routing protocol are studied and briefly discussed in the coming section.

2.1.1 Dynamic Source Routing Protocol (DSR) [4]

Dynamic Source Routing Protocol (DSR) is specifically designed for multi-hop wireless ad-hoc networks. DSR is a reactive routing protocol and allows the network to self-organize and self-configure without any dependence on the existing network infrastructure or administration. The DSR protocol allows a node to dynamically discover a source route across multiple hops. Every data packet sent contains a complete ordered list of nodes through which a packet must pass in order to reach the desired destination. This mechanism avoids the need for having up-to-date routing information in all the intermediate nodes through which a node is passed. The other nodes can cache the source route present in the packet header making route discovery faster. Principally there are two major mechanisms involved in this routing protocol, *Route Discovery* and *Route Maintenance*. These two mechanisms work together in collaboration and allow the nodes to discover and maintain source routes to any destination in the network. All the working of DSR is purely on-demand and thus scales the routing overhead to only currently needed or in-use paths.

2.1.1.1 Route Discovery Mechanism

If two nodes need to communicate with each other, let A be the sender node and B be the destination node, then A needs a source route to B. A looks for a valid route to B in its cache. If A finds an entry in its cache then A places this route into the header of the sending packet and sends the packet. The packet follows the sequence of hops to the destination B. No route discovery is initiated in this scenario. If the cache did not return a valid route then route discovery is initiated.

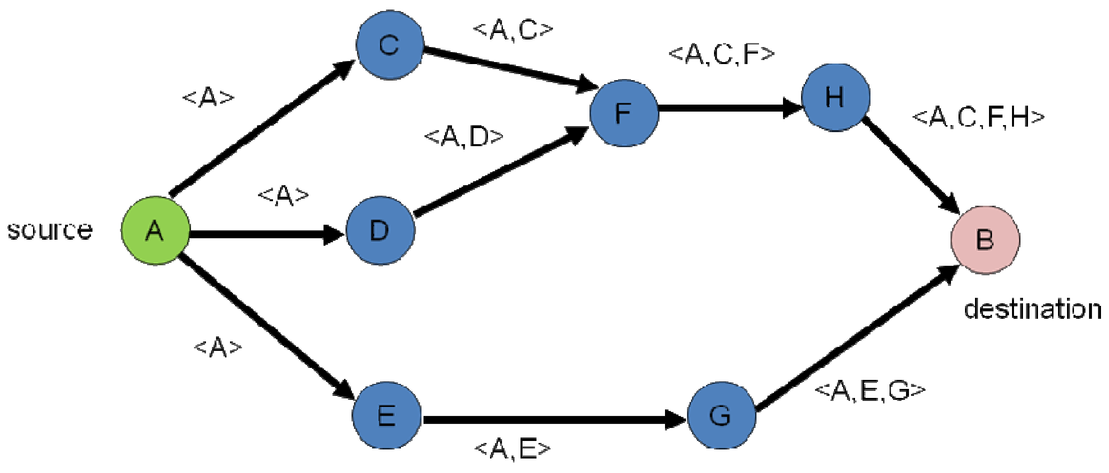


Figure 2: DSR - Node A sends ROUTE REQUEST for route to Node B

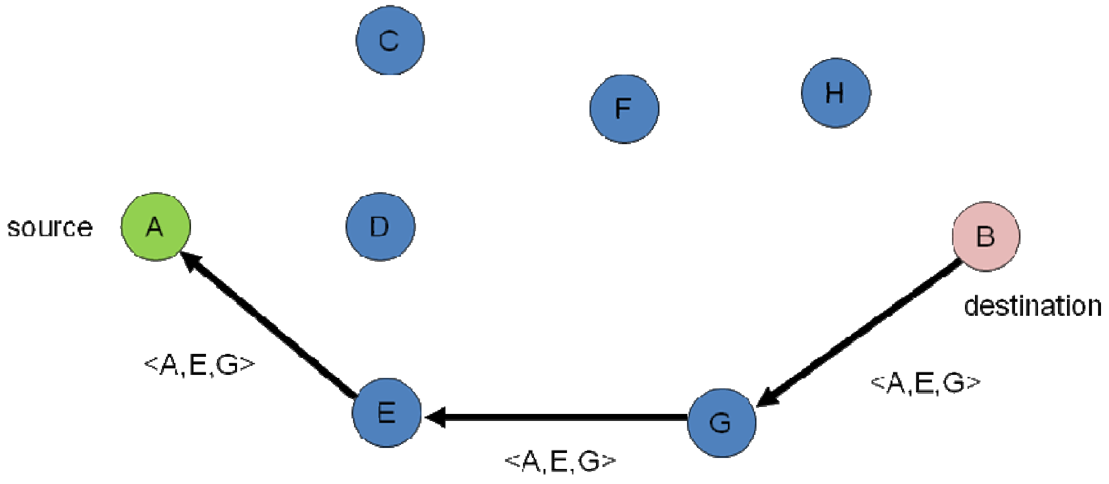


Figure 3: DSR - Propagation of ROUTE REPLY message from Node B to Node A

The major steps involved in a route discovery process are as follows and are shown in Figure 2 and Figure 3:

- A broadcasts a ROUTE REQUEST message to all the nodes in its transmission range.
- Non-target nodes will forward this message when they receive it for the first time.
- Before forwarding, non-target nodes will add their address to the route record in the packet.
- Forwarding nodes check the request id and source node id to avoid retransmissions.

- Forwarding nodes also check if their address is already present in the route record. This is done to avoid loops.
- The destination node B sends a ROUTE REPLY message when it receives the ROUTE REQUEST message.
- In case of bi-directional links, the ROUTE REPLY uses a reverse path of the ROUTE REQUEST.
- In case of uni-directional links, the destination node B repeats a similar process for finding a route to initiator node A. Route cache will be checked for existing routes, if no route is present then a discovery process is initiated.
- In order to avoid infinite route discoveries, the destination node B will store the original ROUTE REQUEST message.
- The source node when receives the ROUTE REPLY message adds the source route to its route cache for future quick access.
- Once the source node A has a route to destination node B, it starts the transmission through the sequenced hops mentioned in the header of packets.

2.1.1.2 Route Maintenance

The maintenance mechanism is based on a very simple approach that every node that originates or forwards a packet using the source route is responsible for confirming the receipt of the packet by the next hop. If a source node A sends a packet to destination node B through nodes C and D, then A is responsible for receipt of C, C for D and D for B. Packet is retransmitted until a receipt is received or maximum number of retransmissions is achieved. If no receipt is received, the node transmits a ROUTE ERROR message to the original sender. This ROUTE ERROR message is used to indicate a broken link. The sender node will react to this broken link by removing the route from cache and looking for another route in the cache. If no other route is present in cache then again a route discovery process is initiated.

2.1.2 Ad-hoc On Demand Distance Vector Routing Protocol [5]

Ad-hoc On Demand Distance Vector Routing protocol (AODV) is reactive routing protocol for mobile ad-hoc networks capable of both unicast and multicast routing. Due to its on-demand nature, AODV builds routes only when needed and keep them as long as they are needed by the sources. AODV joins the mechanism of DSR and Destination Sequenced Distance Vector routing protocol (DSDV). The specific characteristics of DSDV like periodic beaconing, hop-by-hop routing and sequencing and on-demand mechanism of DSR is combined to form the core working of AODV. AODV uses an algorithm with three simple basic objectives:

1. Discovery packets should be broadcasted only when necessary.
2. Local and general topology maintenance should be distinguished.
3. Changes in local connectivity should be passed to only those neighboring nodes that might need the information.

2.1.2.1 Route Discovery Process

AODV builds routes using a route request / route reply query cycle. Route discovery process is only initiated when a source node does not have a route to the desired destination node. The process is initiated by broadcasting a ROUTE REQUEST packet across the network. If receiving nodes are not the destination or does not have a valid route to the destination node, this ROUTE REQUEST is forwarded further. The forwarding nodes store a reverse path to the source node for themselves in their routing tables. Along with the source node’s IP address, current sequence number and broadcast ID, the ROUTE REQUEST also contains the most recent sequence number for the destination node. If ROUTE REQUEST reaches the destination node or a node that has a valid route to the destination, the node sends back a ROUTE REPLY message back towards the source containing the hops to the destination node and the recent sequence number.

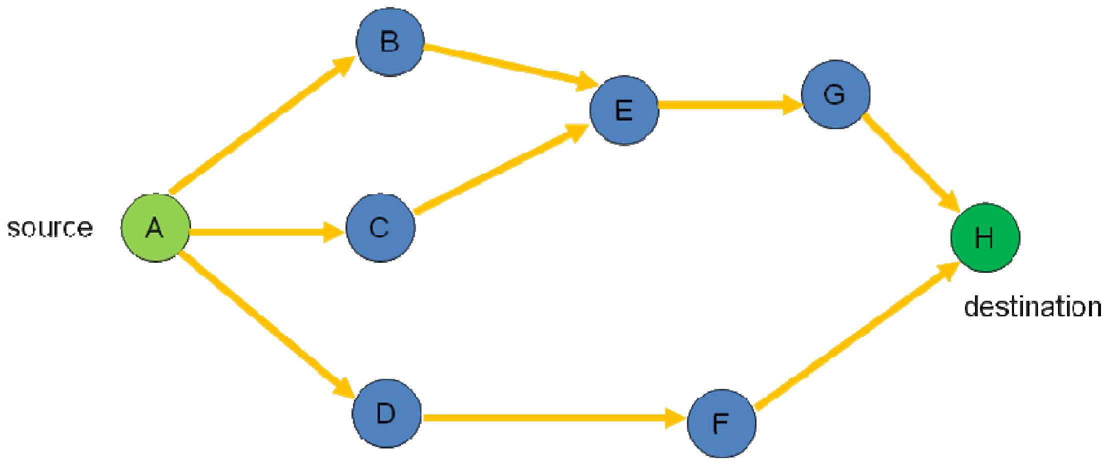


Figure 4: AODV - Propagation of Route Request (RREQ) message

All intermediate nodes that forward this reply message back to the source of the ROUTE REQUEST message, build a forward route to the destination node. When the source node receives this reply message, it can send packets to the destination over the already build forward route.

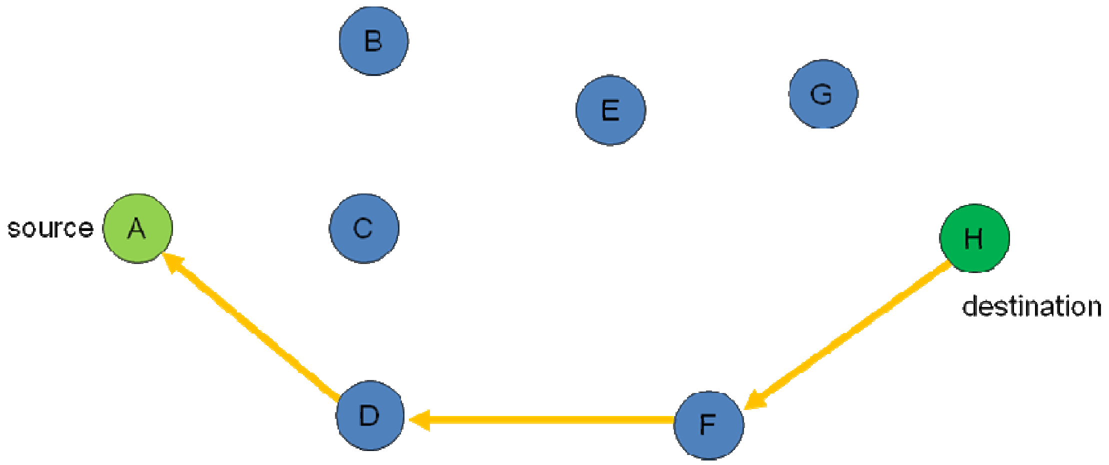


Figure 5: AODV - Path followed by Route Reply (RREP) packet

Due to AODV's hop-by-hop nature, intermediate nodes only store next hop routing information rather than complete routing table for the complete network topology.

2.1.2.2 Route Maintenance

A route is maintained as long as that route is active. A route is considered active when data packets periodically passes through the path from source to destination nodes. When source node stops sending the data, the links will be maintained only until the time out occurs. Once the time out occurs, links will be deleted from the routing tables. In case a node detects a link break in an active route, it sends a ROUTE ERROR message to its upstream neighbors. This messages is passed on and eventually reaches the source node. Nodes in AODV detect broken links by sending periodic HELLO messages and if receipts of three consecutive HELLO messages are not received, it is considered as a broken link. The HELLO messages enable the nodes to detect broken links before sending the packets. But this has a disadvantage of using bandwidth for periodic HELLO messages. A source nodes when receives the ROUTE ERROR can start a fresh route discovery process for establishing a link to destination.

2.1.2.3 Route Table Management

AODV maintains a *soft-state* associated with every entry in the route table. A *soft-state* is the additional useful information other than the source and destination sequence numbers. A timer is associated with the reverse path and is called *request expiration timer*. The expiry time depends on the topology, greater the size of the ad-hoc network greater will be the expiry time. This timer serves to delete reverse path entries from those nodes that do not lie in the path form source to destination. *Route caching timeout* is another parameter associated with every entry. This is time after which a route entry is considered invalid. In each routing table entry, the address of active neighbor node is also recorded. A neighbor node is considered active if it originates or forwards at least one packet for that particular destination during the most recent *active timeout* duration. This information is required for updating the nodes whenever there is a link break along the path. In AODV, a node only maintains the route table entries for destinations for which it is interested unlike traditional table driven routing protocols. Each routing table entry contains following information:

1. Destination address
2. Next hop address
3. Metric (Number of hops)
4. Sequence number for the destination
5. Active neighbors involved in this route
6. Expiry time for route table entry

Whenever a route is used, the time out time is reset to current time plus the active route timeout. If a new a route is available to a mobile node, destination sequence numbers are compared. The route having the greater destination sequence number is opted. If sequence numbers turn out to be exactly same then route is selected on the basis of metric. Route with the smaller metric will be chosen in such a case.

2.1.3 Destination Sequenced Distance Vector Protocol [7]

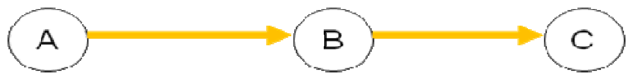
The Destination Sequenced Distance Vector routing protocol (DSDV) is a proactive routing protocol using Bellmann-Ford Algorithm. Two nodes communicate with each using routing tables. Routing tables are stored at each node and contains all available destinations and number of hops to each destination.

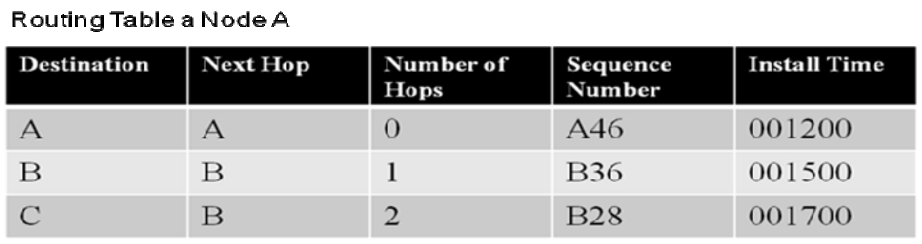
2.1.3.1 Routing Table Management

In DSDV every node has a routing table. Every routing table carries the available destinations and cost involved in accessing that destination. Entries in routing table are marked with a sequence number which is generated by the destination node. Ad-hoc networks have an every changing topology and routing protocol needs to fully aware of this. In order to cope up with this challenge, nodes periodically transmit updates or whenever there is a change in the network topology. These update packets advertise which nodes are accessible from each node in the network and the number of hops required to reach these nodes. The update packet has a metric of one for one-hop neighbors and it is increased by one by each forwarding node. Additionally a sequence number is also tagged by original node to the update packet. The data broadcast by each node contains

1. The destination address.
2. Number of hops required to reach the destination.
3. Sequence number as originally tagged by the destination.

DSDV protocol requires all the nodes to broadcast its routing table to all its current neighbors.




Routing Table a Node A

Destination	Next Hop	Number of Hops	Sequence Number	Install Time
A	A	0	A46	001200
B	B	1	B36	001500
C	B	2	B28	001700

Figure 6: Routing table at Node A using DSDV

The receiving nodes update their routing tables on the basis of sequence numbers. If the sequence number is greater or equal than the current one, routing tables are updated. DSDV uses two different types of update packets. One is full dump which is exchanged periodically and contains the complete snapshot of the network. The other update type is incremental which contains the information changed since last full dump.

2.1.3.2 Route Maintenance

Broken links are detected by link or physical layer components or if a node does not receive any broadcast message from neighbor for a longer period of time. When this happens, the detecting node immediately informs the rest of the network by broadcasting an update message. DSDV invalidates all broken links by immediately assigning an infinite metric and increments sequence number. The incremented sequence number forces all the nodes to update their routing tables and this link is invalidated by all the nodes. Once the link is established again, the detecting node again transmits an update message and the network immediately reacts to it and route is updated by all the nodes.

2.1.4 Optimized Link State Routing Protocol [6]

The Optimized Link State Routing Protocol (OLSR) is a table driven proactive routing protocol designed for mobile ad-hoc networks. It periodically updates the network topology information by sending HELLO messages. Every node selects a set of neighbor nodes and designate them as *Multipoint Relays* (MPRs). In OLSR, only nodes chosen as MPRs forward control traffic and provide an efficient mechanism for flooding by reducing the number of transmissions. OLSR can be broadly divided into following three mechanism

1. Neighbor sensing mechanism
2. Flooding using Multipoint Relays
3. Route calculations

2.1.4.1 Neighbor Sensing Mechanism

Changes in the neighborhood of a node are detected through this mechanism. Two nodes are considered neighbors when they are directly connected to each other and transmission can occur in both directions. If nodes A and B are neighbors, then node X is considered to be a two-hop neighbor of node A, if X is not the neighbor of node A and there is a symmetric link between node A and B and a symmetric link between node B and node X.

Neighbor sensing is done by periodically sending HELLO messages. HELLO messages contain the sender node’s address along with the list of neighbors of the sending node and link status. A node which receives the HELLO message can thus generate information for its two-hop neighbor and link status in the neighborhood. In this way every nodes become aware of not only neighbors but also two-hop neighbors.

2.1.4.2 Flooding using Multipoint Relays

In normal flooding techniques, also referred as full or pure flooding, a node retransmits broadcast packet when it receives it for the first time and duplicate copies are dropped and not forwarded. Such a technique significantly increases the network overhead and networks where nodes are mobile and have limited power; this significantly affects the overall performance of the network. In neighbor sensing mechanism, HELLO messages are exchanged only among neighbors. As mobile ad-hoc can be considerably large and dynamic, the more efficient way of distributing the topology information is by the use of multipoint relays. The intention here is to allow spreading of information to each node without making any duplicate and unnecessary retransmissions. The multi relay concept significantly decreases the flooding overhead as compared to pure flooding.

Every node selects a set of nodes as multipoint relays (MPRs). The two-hop neighbor information is exploited to get a minimal MPR set. The MPR is chosen in such a way that a node can reach all its two-hop neighbors through the selected multipoint relays. Multipoint relay selector set is maintained by every node which keeps track of nodes that selected current node as the MPR. The protocol only allows MPR node to retransmit a broadcast packet if it is received by a node for which that node is in the multipoint selector set. If same packets are received again, they are simply dropped thus significantly reducing the flooding overhead.

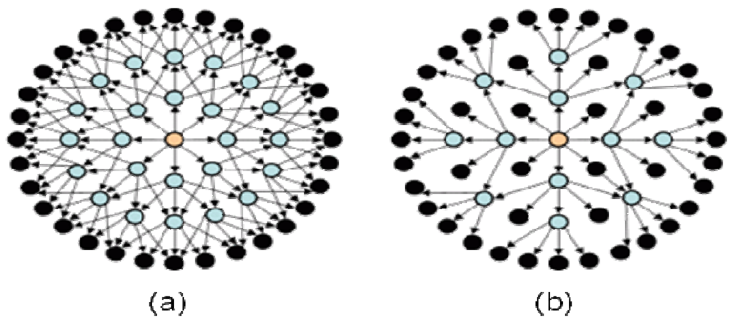


Figure 7: Flooding in OLSR (a) Normal Flooding (b) Flooding through MPRs (Blue nodes are forwarding)

2.1.4.3 Route Calculations

All nodes that are not selected as MPRs by any other node periodically sends a topology control message. The topology control message is spread in the network using the mechanism described in the earlier section. A topology control message contains the address of the originator node and MPR set for that node. The MPR nodes announce this information periodically in their control messages. So, a node announces to the network, that it has approach to the nodes which have selected it as an MPR. As a result all nodes receive a partial topology graph. Shortest path algorithm is applied on the partial topology graph to compute the optimal path. The nodes maintain the topology information only for specific period of time. Once that time expires, topology graph is removed.

2.2 Underwater Acoustic Sensor Networks

Underwater acoustic sensor networks (UWASNs) consist of variable number of sensors and vehicles deployed over a certain area for performing various monitoring tasks. These sensors are small, intelligent; battery powered digital devices, capable of communicating wirelessly with each other and to the base station.

2.2.1 UWASNs vs. Terrestrial Sensor Networks

The dynamics of underwater networks is totally different from the traditional terrestrial sensor networks. The biggest difference is the propagation medium. In terrestrial sensor networks the propagation medium is air and in UWASNs it is water. These two mediums have entirely different characteristics. Electromagnetic (EM) waves are used in terrestrial sensor networks but EM waves cannot be used in UWASNs. Electromagnetic waves do not propagate over large distances at high frequencies. As reported by Robotic Embedded Systems Laboratory (RESL), EM waves at 433MHz have a transmission range of only 120cm. Only extra low frequencies can penetrate through water, but transmission via low frequencies requires very large antennas. Similarly, optical waves cannot be used for transmission in underwater networks because they suffer badly due to scattering. Moreover, optical transmission requires precision equipment for pointing the laser beam. Due to these reasons, acoustic propagation is the most suited option for underwater networks. Some other differences between UWASNs and terrestrial sensor networks are as follows.

1. Sensors for terrestrial sensor networks are inexpensive as compared to underwater sensors. Sensors used in underwater networks are expensive because of complex transceivers and extra protection required against harsh environmental conditions.
2. In terrestrial sensor networks deployment is easy and usually sensor nodes are densely deployed. However, in underwater networks deployment is a costly and challenging process and that is why sensors are sparsely deployed in underwater networks.
3. Power requirements of underwater sensors are higher because of complex transmission techniques and greater transmission distances. In case of terrestrial sensor networks, recharging sensor nodes is easier. Sensors can be easily collected and batteries can be replaced or solar energy can be exploited to recharge the sensor nodes. Solar energy cannot be exploited in underwater networks and reclaiming deployed nodes is a costly process.

4. Underwater sensors have grater memory requirements than terrestrial sensors because they might have to cache the transmission data due to impaired or unavailable channel.

2.2.2 UWASN Architecture

The network topology becomes crucial when we consider the energy consumption of the nodes, capacity of network and its reliability. Therefore, an optimized topology is required. The underwater sensor network topology is an open issue and a lot of research is currently going on. The basic architectures proposed and widely accepted are as follows [3] .

2.2.2.1 Static two-dimensional underwater sensor networks

Such networks have sensors anchored to the bottom of the ocean. All sensors are kept at the same depth and all nodes are static. No free movement in any direction is allowed. Sensors are interconnected to each other via wireless acoustic links. Sensors are equipped with two acoustic transceivers, vertical and horizontal. Horizontal transceiver is used to communicate with other sensors and sinks at the same depth. Vertical transceiver is for communication with surface station. Sensor nodes in such networks are also referred as ocean bottom nodes.

2.2.2.2 Static three-dimensional underwater sensor networks

In three dimensional underwater sensor networks, sensor nodes float at different depths. Usually nodes are deployed with the help of an inflatable buoy and anchors on the bottom of the sea. The amount of compress air within the buoy determines the depth or height of the nodes with respect to ocean floor. Such a deployment is very challenging. Extra precautions are required to make sure complete 3D coverage of the network, because underwater channel is unpredictable and has ever changing dynamics. So the 3D deployment has to be done in such a way that network topology is always connected and there always exists a path from a sensor to sink.

2.2.2.3 Mobile three-dimensional underwater sensor networks

Autonomous underwater vehicles (AUVs) carrying single or multiple sensors makeup the mobile 3D underwater sensor network. These AUVs are truly mobile in all the directions. This mobility makes communications extra challenging because the topology is constantly changing and so as the routes.

2.2.3 Acoustic propagation: The Basics

Sound waves travel in the form alternating compressions and refractions. These compressions and refractions are detected by the receiver as changes in pressure.

2.2.3.1 Speed of Sound

The speed of sound in water is dependent on pressure (depth), temperature and the salinity of the water. This is why, propagation speeds differ for fresh water and seawater. The average approximate speed of sound in fresh water and sea water at atmospheric pressure are 1450 m/s and 1500 m/s respectively. Speed of sound is directly proportional to the change in temperature, pressure and salinity. A change of 1 °C in the temperature causes a change of ~ 4 m/s. A 1% change in salinity causes a change of ~1 m/s [13] . Various empirical equations have been derived to accurately calculate speed of sound on the basis of temperature, depth and salinity. The speed of sound ***c*** as a function of temperature ***T*** in degrees Celsius, Salinity ***S*** in parts per thousand and depth ***z*** in meters as given by Mackenzie [13] and is shown in Equation 1.

$$c(T,S,z) = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5(S-35) + a_6z + a_7z^2 + a_8T(S-35) + a_9Tz^3$$

Equation 1: Nine-term equation for speed of sound in oceans by K.V. Mackenzie.

Where constants a_1, a_2…. a_8 are as follows:

$a_1 = 1448.96$	$a_2 = 4.591$	$a_3 = -5.304\times10^{-2}$	$a_4 = 2.374\times10^{-4}$
$a_5 = 1.340$	$a_6 = 1.630\times10^{-2}$	$a_7 = 1.675\times10^{-7}$	$a_8 = -1.025\times10^{-2}$

$$a_9 = -7.139\times10^{-13}$$

This equation has a standard error of 0.070 m/s for salinities between 25 and 400 ppt.

2.2.3.2 Bandwidth and Range Limitation

The typical frequencies associated with acoustic communication are between 10 Hz and 1 MHz, higher frequencies are rarely used because they are quickly absorbed [2] . Bandwidth of the underwater acoustic channel is limited and dramatically dependant on both frequency and transmission range. Long range systems may have a bandwidth of only few kHz, while short range systems will high orders of bandwidth. Usually underwater communication links are classified as very short, short, medium, long and very long. In Table 1 available bandwidth for the above mentioned classes of acoustic links is given [1] .

	Range (km)	Bandwidth (kHz)
Very Long	1000	<1
Long	10-1000	2-5
Medium	1-10	≈10
Short	0.1-1	20-50
Very Short	<0.1	>100

Table 1: Available Bandwidth for different range of underwater acoustic links.

Acoustic links are also classified as vertical or horizontal, depending upon the direction of the sound ray with respect to the ocean floor.

2.2.3.3 Factors influencing acoustic propagation

Factors that influence the acoustic communication in underwater networks are mentioned below [1] .

- **Path loss due to Attenuation**: It is mainly caused due to absorption due to conversion of acoustic energy into heat. The attenuation is also caused by scattering, refraction and dispersion of sound waves in water.
- **Path loss due to Geometric spreading**: As the sound waves travel in water, wave fronts expand and sound energy spreads. The most common kinds are spherical and

cylindrical spreading which characterizes deep and shallow water communication respectively.

- **Reverberation**: Reverberation causes a large number of echoes to build up and gradually decays as the sound is absorbed in the environment. The rough boundaries, fish and other organisms underwater causes the sound to scatter and are a major cause of reverberation. In underwater communication these reverberations or background noise can be of much longer duration than the original transient signal. For correctly detecting an acoustic signal it should have a higher level than reverberation threshold.
- **Man made noise**: This kind of noise is caused by small boats & ships, machines, drilling sites and other fishing or exploration activities.
- **Ambient Noise**: The natural movements of water like tides, currents, winds, rain and other seismic activities contribute towards ambient noise.
- **Multipath effect**: Acoustic propagation is degraded by multipath propagation since it generates inter symbol interference.
- **High and variable delay**: The propagation speed in underwater acoustic channel is five orders of magnitude lower than in radio channel. This large propagation delay results in reduced throughput and prevents from accurately calculating the round trip time (RTT) which is an important parameter for many protocols.
- **Doppler spread**: When the paths between to two communicating ends change due to either one's movement, a shift in the frequency of transmitted signal is introduced. This shifting phenomenon is referred as Doppler shift. Signals travelling in different paths may have different Doppler shifts corresponding to different rates of change in phase. The difference in Doppler shifts between different signal components contributing to a single fading channel tap is known as the Doppler spread [ref].

2.2.4 Related Work

In the last couple of years, huge advancements have been made towards underwater networks. In this section, we discuss the existing protocols/ideas developed specifically for underwater networks. In [16] , Vector based forwarding approach is proposed which uses packets to carry position information about the sender, destination and currently forwarding node. Using these positions, any node will be able to determine the forwarding direction. A receiving node determines to forward the packet based on its position, position of the forwarding node and the angle of arrival (AOA) of the signal. If the node finds itself close to the forwarding path, it records its own position and forwards the packet, otherwise packet is discarded. The protocol, using this technique, forms a "*routing pipe*" from source to destination where all nodes should forward packets and nodes outside the pipe should discard the packets. Multi Meshed-Tree Protocol (MMT) is presented in [17] which use the concept of constructing number of meshed trees from different origins also called "roots". The roots act as gateways in UWASs. The size and the growth of meshed trees are dependent on the number of hops from the root and QoS required. As the hops increase, QoS is degraded. In [18] , authors presents energy-aware spanning tree protocol (E-Span), specifically tailored for underwater acoustic sensor networks. The objective of E-Span is to span the tree without having any cycles in such a way that all nodes of the network are covered and are connected to the root via shortest path while considering the residual energy in the nodes while selecting the root. The root node is responsible for coordinating the links and the routes connected through it, hence a root node is selected to be the node with the highest level of energy among all other nodes. Configuration messages are used to exchange the information about the residual energy. As the selection of root node is entirely based on the fact that which node has the highest energy level, so it is very much possible that a selected root may not provide the minimum number of hops to the destination.

Most of the routing protocols for underwater networks are either limited to just being an approach or under development. Some protocols have been developed, but at the moment they cannot be used commercially as they have not been tested thoroughly in live scenarios and accepted globally. This is why we choose to used DSR, DSDV, AODV and OLSR in our study. These routing protocols have been widely used in terrestrial sensor networks and accepted globally. Because of their mature nature, we have a high degree of confidence level in using these routing protocols for our study.

2.2.5 Applications of underwater networks [1]

UWASNs find their way into a number of practical applications. Some of those applications are listed below.

1. UWASNs can be used for environmental monitoring ranging from monitoring pollution levels to monitoring changes in ecology. Monitoring ocean currents can better help in predicting weather forecast and ensuring safe journeys in the sea. Marine life, in general, and especially endangered species can also be tracked and monitored using underwater networks.
2. Undersea explorations are another major application area. UWASNs can aid in detecting oil & gas fields and minerals.
3. Great disasters like Tsunami can be avoided with the help of UWASNs monitoring seismic activities of remote areas on the ocean bed.
4. UWASNs can play a vital role in sea navigation. Hazards on seabed like rocks, icebergs, wrecks and shoals can be detected and warnings can be passed on to the navigation system for safe routing.
5. UWASNs also find their way into military and naval applications like tactical surveillance, reconnaissance, and targeting and intrusion detection.

2.2.6 Underwater Network Design Challenges

As a lot of factors influence the acoustic propagation, the underwater network designing faces numerous challenges. Some of the major challenges are listed below.

- Bandwidth is very limited.
- Multipath and fading inversely affects the underwater channel.
- Propagation delay in underwater networks is very high as compared to radio waves in terrestrial channel. Also this delay is highly variable and highly effected by the physical characteristics of water like temperature, salinity and speed of currents.
- Underwater communication experiences high bit error rates and temporary losses of connectivity due to shadow zones.
- Sensor nodes have limited battery powers. Replacing these batteries frequently is an expensive, time consuming and tedious process. Also solar energy cannot be used to recharge the sensor nodes.
- Fouling and corrosion can cause the underwater sensor nodes to fail.

2.3 Network Simulator 2 (ns-2)

The Network Simulator is an event drive discrete simulator developed by UC Berkeley. Network Simulator is widely used across the globe for research and academic purposes. It has the capability of supporting simulations of a variety of protocols and network topologies. It is suitable for designing new protocols, comparing various protocols and traffic evaluations. Ns-2 is now developed and maintained by a number of collaborating institutes and researchers. Network Simulator 2 is open source and is distributed freely. Various versions of ns-2 for different operating systems like Linux, Solaris, FreeBSD, Mac OS X and Windows are available.

2.3.1 Structure of ns-2

Object oriented programming approach is used to build ns-2. Methods are written in C++ and OTcl (an object oriented variant of programming language Tcl). Tcl is pronounced as "tickle" and a very simple scripting language similar to Python and Perl. Tcl script is the primary method for invoking ns-2 simulations. A user writes the simulation script using Tcl. The simulation environment has to be configured to set various components required for simulation purposes. These components include event scheduler objects, setup module libraries and network component libraries. Once the environment is set, the simulation script is written by putting all the required components together. The ns-2 simulator interprets the script and triggers the various procedures as specified and required by the simulation script. The event scheduler is one of the major components of ns-2 other than network components. Events like sending and receiving packets, start and stop tracing are triggered by event schedules.

Some parts of ns-2 that require greater efficiency and fast processing is written in C++, while others are written in OTcl. OTcl linkage is used to map C++ methods and variables to their corresponding methods and variables in OTcl. The flow of the simulation in network simulator is given in Figure 8.

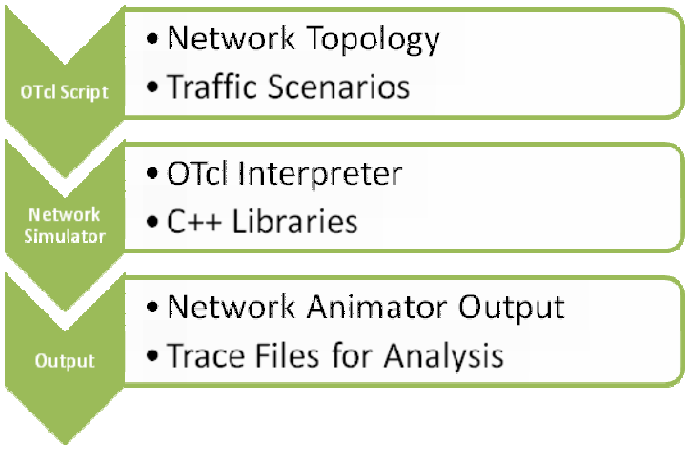


Figure 8: Simulation flow in ns-2

Because of its open source nature, we can modify existing components or add new network components in ns-2 and set them in the simulation. C++ objects are controlled by OTcl objects and every linked class hierarchy in C++ has a corresponding class hierarchy in OTcl.

2.3.2 Features of ns-2

Wired and wireless simulations are supported in ns-2 along with their tracing and visualization. Some the major features are as follows:

1. Network Topology: ns-2 supports simulations for wired, wireless and wired-cum-wireless networks. It also supports simulations for satellite networks.
2. Propagation Models: Multiple propagation models are provided by ns-2 including free space, two-ray ground and shadowing model.
3. Routing Protocols: For wired networks ns-2 supports Distance Vector routing (DV), Link state routing (LS) and Protocol Independent Multicast-Sparse Mode (PIM-SM) for routing to multicast groups.
4. Transport Protocols: Transmission Control Protocol (TCP), User Datagram Protocol (UDP) for unicast and Scalable Reliable Multicast (SRM) for multicast are supported by ns-2.
5. Traffic source applications: FTP, Web, telnet, CBR, real audio, etc can be used in simulations.
6. Queues: Different types of queues are supported by ns-2 including drop-tail, Random Early Detection (RED), Fair Queuing (FQ), Stochastic Fair Queuing (SFQ) and Deficit Round Robin (DRR).
7. QoS: Integrated Services and Differentiated Services models are present in ns2
8. Energy Model: The energy model in ns-2 represents the energy in a mobile host. The model has arguments for initial energy, energy usage for transmitting each packet and receiving every packet.

9. Visualization Tools: ns-2 includes a tool for viewing the simulation, called Network Animator (NAM). NAM is a Tcl/Tk based animator used for viewing network simulation traces.
10. Helpful Utilities: ns-2 also includes some very useful utilities like mobile movement generator used for generating movement patterns and traffic generators used for creating desired traffic scenarios.

2.3.3 Tracing Capabilities of ns-2

The ns-2 simulator records all events of a simulation in a single file called trace file. This trace file is a simple text file listing each and every event that has occurred during the simulation time. There are two different formats available for wireless simulations, namely old and new wireless trace formats.

2.3.3.1 Old Wireless Trace File Format

The old wireless trace format starts with a $ sign followed by the action type related to the processing of packet. The four possible action types for the packet can be send, receive, drop or forward. The action type is then followed by a number of other values for the items mentioned in **Error! Reference source not found.**. The exact actual trace format is like this.

$<action type> %.9f %d (%6.2f %6.2f) %3s %4s %d %s %d [%x %x %x %x]

For all wireless simulations the values specified in **Error! Reference source not found.** are recorded.

Additional trace information is also recorded according to the specific used protocol. The complete list of trace items specific to protocols can be found at [8] .

2.3.3.2 New Wireless Trace File Format

The structure of the new wireless trace is completely changed from the older format. The lines of the trace begin with the action flags, similar to older format, but are followed by flag/value pairs. The flags are a combination of a dash and alphabetical letter, specifying the type of the flag. The general format of new wireless trace is as follows.

<***Event Type***> <***General Tag***> <***Node Property Tags***> <***IP Level Packet Information***> <***Next Hop Information***> <***MAC Level Packet Information***> <***Application Specific Information***>

Chapter 3

Methodology

This section describes how this study was conducted. The first objective was to come up with a reliable simulation environment for underwater networks in Network Simulator. After the simulation model was set, a set of metrics were defined in order to evaluate the performance of various routing protocols. Our metrics include packet delivery ratio, average end-to-end delay, throughput, routing overhead and energy consumption of nodes. These metrics enable us to judge the performance of routing protocols on the basis of single criteria. Once the metrics are established, different simulation scenarios are constructed. These scenarios are designed in ns-2 using Tcl programming language. The traffic scenarios implement different traffic conditions, node depth and number of nodes. The trace files generated by running the all the different simulation scenarios are evaluated for studying the effect of variable traffic conditions, node depth and number of nodes in the network on the performance of routing protocol with respect to our performance metrics. Trace files are evaluated via scripts written in AWK and code written in Matlab. The further detail of each step is in the following sections.

3.1 Underwater Simulation Model for ns-2

The underwater simulation model is based on the implementation of [9] . The ns-2 simulator divides the layers below the MAC layer into four components: Propagation, Channel, Physical, and Modulation. The model provides the implementation of underwater acoustic channel, underwater physical layer and underwater acoustic propagation model. The simulation model correctly calculates and adjusts the network conditions. The attenuation is calculated on the basis of spreading loss and Thorp's approximation. Underwater channel noise is incorporated into the model. All the factors like turbulence, wind, thermal and shipping noise are accounted for. The calculations for propagation delay are done by accounting factors like depth of the nodes, salinity of water and temperature. The physical layers let us adjust some key parameters like transmission power, carrier sense and receive thresholds. The model is capable of correctly predicting the transmission power required to successfully meet the receive threshold of the receiving node and the available bandwidth.

The detail of files provided by this model is as follows:

1. Underwater-phy.h and underwater-phy.cc provides the implementation of underwater physical layer. These files go into the "*mac*" directory of ns-2 installation. From the TCL script it can be used as "**set val(netif) Phy/UnderwaterPhy ;**"
2. Underwaterchannel.h and underwaterchannel.cc implements the underwater channel and is placed in directory "*mac*" of ns-2 installation and can used from Tcl as "**set val(chan) Channel/UnderwaterChannel ;"**
3. Underwater.h and underwater.cc is the implementation of underwater acoustic propagation model. These files are added in the "*mobile*" directory of ns-2 installation. This propagation model can be assigned form the TCL script as "**set val(prop) Propagation/Underwater ;**"

Propagation models are responsible for calculating the signal-to-noise ratio at the receiver after attenuation and ambient noise are taken into account, as well as the interference range of a signal. Thorp's approximation for absorption loss is shown here in the form of pseudo code.

```
Thorp (frequency)
    1.f ← POW (frequency, 2);
    2.if f > 0.4
    3.     then
    4.      atten ← 0.11 * f /(1+f)+
    5.                 44* (f /(4100 + frequency))+
    6.               2.75 * POW (10, -4) * f +
    7.              0.003;
    8.  else
    9.         atten ← 0.002 +
    10.                0.11 * (f / (1+f)) +
    11.               0.011 * f ;
    12. return atten;
```

Calculation of the SNR at the receiver is done in a function that overloads the Pr function in ns2 in combination with the ambient noise calculation. In order to find the center frequency, the distance between nodes is calculated. Lines 3–8 calculate the AN factor for each of the possible frequencies for the transmission. In addition to this attenuation, signal fading in the underwater environment is affected by the orientation of the link. As each AN value is calculated, the frequency with the lowest AN factor (largest value of the AN variable) is tracked. Finally, the AN factor that corresponds to that frequency is combined with the transmitted power to calculate the SNR at the receiver.

```
Pr (transmitter, receiver)
    1. Pt ← transmitter → GetTxPr();
    2. Distance ← CalcDist (transmitter, receiver);
    3. for i ← 0 to Num_Freq
    4. do
    5.      AN[i] ← -(k * 10 * log10(distance)+
    6.                distance * Thorp(freq[i])+
    7.                orientation (transmitter, receiver)+
    8.                log10(Noise(freq[i])));
    9. if AN[i] > AN[max_index]
    10.     then
    11.         max_index ← i;
    12. Pr ← Pt + AN[max_index];
    13. return Pr;
```

Noise calculation is done considereing all the factors that contribute to the noise. Turbulance, Wind, Shipping and Thermal factors influence the over all noise of the channel.

$$N(f) = Nt\ (f) + \ Nw\ (f) + \ Ns\ (f) + \ Nth\ (f)$$

Where N(f) is the total noise in the channel for a particular frequency f and $N_t(f)$, $N_w(f)$, $N_s(f)$ and $N_{th}(f)$ are turbulence, wind, shipping and thermal noise components for that particular noise.

Physical Layer calculates the final statistics used in the simulation with respect to

- Packet Reception
- Packet Error
- Transmission Time
- Propagation Delay.

For most of the calculations, calls are made to propagation and channel classes.

Primary C++ function used at this layer is bandwidth calculation given the distance between the transmitter and receiver. The physical layer lets us adjust following important physical parameters through bounded variables. Those parameters include: Capture threshold, Carrier sense threshold, Receive power threshold, Transmitted signal power, Power for transmission, Power for reception, Idle power consumption, Sleep power consumption and Transition power consumption.

3.2 Performance Evaluation Metrics

The objective of this thesis is to evaluate the performance of the ad-hoc routing protocols in underwater acoustic sensor networks environment. In order for protocols to be compared with each, there is a need to define some metrics so that we can judge them and make our comparisons in a fair manner. Following metrics are used to compare the performance of routing protocols.

3.2.1 Packet Delivery Ratio

The fraction of the packets received by receivers out of the packets that were sent by the application is called packet delivery ratio (PDF) [10] . This ratio is important to find the packet drop and loss ratios. In our case all the traffic sources are constant bit rate sources (CBR) so we can express PDF in the following manner.

$$Packet\ Delivery\ Ratio = \frac{\sum CBR\ packets\ recieved\ by\ sinks}{\sum CBR\ packets\ sent\ by\ sources}$$

Equation 2: Calculating packet delivery ratio (PDF)

3.2.2 Average End-to-End delay

The time required by a packet to traverse from source to destination is called the end-to-end delay [11] . Average end-to-end delay is the sum of end-to-end delays for all the packets received divided by the total number of packets received by the receiver.

$$Avg.\, End\ to\ End\ Delay = \frac{\sum_{i=1}^{N}(Packet\ Receive\ Time_i - \ Packet\ Send\ Time_i)}{N}$$
$$where\ N\ is\ the\ total\ number\ of\ packets\ received$$

Equation 3: Calculation for average end-to-end delay for the received packets

In the calculation of end-to-end delays all delays are included like transmission delay, propagation delay, processing delay and queuing delay.

3.2.3 Throughput

The throughput is defined as the total data received by the receivers divided by the time from the start of the transmission to the time last packet was received [12] .

$$Throughput = \frac{(\sum CBR\ packets\ received\ by\ sinks)\ x\ Packet\ size}{(Time\ of\ recieving\ last\ packet - \ Simulation\ start\ time)}$$

Equation 4: Overall throughput of the network

Throughput is the rate of successful delivery messages over a communication link and usually expressed in the terms of bits per seconds (bps).

3.2.4 Routing Overhead

The number of packets sent across the network for establishing routes, exchanging routing information and performing route maintenance are considered routing overhead, as these packets do not contribute towards the actual through put of the networks. The routing control packets are also considered an overhead because they come with a cost of management processing, delays and bandwidth utilization.

$$Routing\ Ovrhead = \sum Routing\ packets\ transmitted$$

Equation 5: Calculating routing overhead

Another important terminology used regarding the routing overhead is Normalized Routing Load (NRL). NRL is the number of routing packets sent across the network divided by the total number of packets received.

$$NRL = \frac{\sum Routing\ packets\ transmitted}{\sum CBR\ packets\ received\ by\ sinks}$$

Equation 6: Calculating normalized routing load

The NRL tells us how many routing control packets are generated for successfully sending one data packet.

3.2.5 Energy consumption

In underwater networks, battery life of the sensor has always been a major issue. The replacement or recharging solutions are not very viable. So it is always expected from the underwater network components to consume lesser and lesser energy of the nodes and yet not compromising the overall functionality and stability of the network. All nodes are monitored for their energy levels. Node failures due to energy depletion are recorded. Routing protocols are compared on the basis of their impact on the energy consumption rate of the sensor nodes.

3.3 Simulation Scenarios

The simulation scenarios were created to evaluate the performance of ad-hoc routing protocols with respect to varying traffic conditions, varying number of nodes and varying depth of the nodes. For all the scenarios, trace files are evaluated on the basis of the metrics discussed in the earlier section.

3.3.1 General Simulation Settings

Some parameters of simulation environment are kept constant for the all the simulation runs. Those parameters are discussed below.

- **Node Deployment**: Nodes are deployed randomly using a C++ code. For all scenarios, nodes are deployed over an area of 500m x 500m.
- **Physical Layer Parameters**: Phy/Underwater-Phy is set as the network interface type for all the mobile nodes deployed. The additional physical layer parameters are given in Table 2.

	Parameter	Variable	Value
1	Capture Threshold (db)	Phy/UnderwaterPhy set CPThresh_	10.0
2	Carrier Sense Threshold (W)	Phy/UnderwaterPhy set CSThresh_	0.284
3	Receive Threshold (W)	Phy/UnderwaterPhy set RXThresh_	4.0
4	Transmission Power (W)	Phy/UnderwaterPhy set Pt_	65
5	Frequency (Hz)	Phy/UnderwaterPhy set freq_	300

Table 2: Important Parameters for Underwater Physical Layer

- **MAC Layer Settings**: MAC 802.11 is chosen as the MAC layer protocol. As we are aware of the fact that MAC 802.11 is not specifically designed for use in underwater acoustic networks, so some parameters are adjusted accordingly. The changed parameters are given in Table 3**Error! Reference source not found.**.

	Parameter	Variable	Value
1	Minimum Contention Window	Mac/802_11 set CWMin_	4
2	Maximum Contention Window	Mac/802_11 set CWMax_	32
3	Slot Time (sec)	Mac/802_11 set SlotTime_	0.020
4	Short Inter Frame Space (sec)	Mac/802_11 set SIFS_	0.010

Table 3: Important parameters for MAC 802.11 adjusted for underwater networks

- **Channel**: Channel/UnderwaterChannel is set as the channel type for all simulations.
- **Propagation Model**: Propagation/Underwater is assigned as the propagation model for all simulations.
- **Transport Protocol**: User Datagram Protocol (UDP) is chosen as the transport protocol. UDP is preferred over TCP because of the noisy & lossy and unpredictable nature of underwater acoustic channel.

- **Application Layer Traffic Generators**: Constant Bit Rate (CBR) is used to generate traffic over UDP in all simulations.
- **Energy Model**: In simulation patterns, all nodes are assigned a same energy level of 1000 joules. The power required for receiving is set to 350mW and the energy consumption for transmitting is set to 2.5W. These settings remain constant for all simulation runs.

3.3.2 Simulation Settings For Scenarios With Different Traffic Conditions

In order to study the effect of varying traffic conditions on the performance of the routing protocols. 25 nodes are deployed randomly at the depth of 50 meters. Simulations are run for different number of constant bit rate (CBR) connections for AODV, DSDV, DSR and OLSR. Routing protocols are compared for their performance with 1 CBR connection and incrementing the number of connections to 5 CBR connections. Sources and sinks are also selected randomly.

3.3.3 Simulation Settings For Scenarios With Different Node Depths

For studying the effect of depth on the performance of routing protocols, 25 sensor nodes are deployed with 3 CBR connections. The deployment of the nodes and the selection of source and sinks are purely random. The depth of the node is varied in these simulations, starting from the depth of 10 meters and going up to the depth of 100 meters. Routing protocols are compared on the basis of the metric already discussed in the earlier section with respect to the change in the depth of the sensor nodes.

3.3.4 Simulation Settings For Scenarios With Different Number of Nodes

In this case number of CBR connections is set to 3 and depth of the nodes is taken as 50 meters. However, the number of nodes is kept variable. Simulation runs are made starting with just 5 sensor nodes and then increasing it to 25 nodes.

3.4 Trace Analysis

Simulations run by ns-2 result in large text based trace files. These trace files contain the record of all the events that are triggered while the simulation was running. AWK programming language is used to write the scripts for analyzing the trace files. AWK is a special programming language designed for processing text-based, either in files or in data streams. AWK treats a file as a collection of records where each line is one complete record. Each line is broken into a sequence of fields. Each field in trace file corresponds to a flag or a value as per the tracing format of ns-2. The trace formats are discussed in chapter 2. Using AWK scripts, trace files are read line by line and required values are read field by field and stored and manipulated for calculating packet delivery ratio, throughput, average end-to-end delay, routing overhead and energy consumption of the nodes. The raw calculations produced by AWK scripts are used by Matlab code for performing complex calculations and generating graphs. The AWK scripts and Matlab code are provided in Annexure B.

C h a p t e r   4

Analysis of Results

In this chapter, the results of the simulation scenarios, given in Chapter 3, have been presented. A comparison among the studied routing protocols is performed using the performance metrics presented in earlier section. In the end the results are concluded and propositions are made.

4.1 Packet Delivery Ratio (PDF)

The fraction of packets received by a receiver out of the total packets sent by the send is called packet delivery ratio (PDF). The following sections exhibit the comparison of PDF among different routing protocols for different simulation scenarios.

4.1.1 PDF for different traffic conditions

Packet delivery ratio as a function of number of CBR connections is shown in Figure 9. It is observed that delivery ratio decreases as the number of connections increase. OLSR has the least delivery ratio than other protocols in every traffic condition. The maximum delivery ratio for OLSR is just above 6 percent for 1 connection scenario and less than 2 percent for heavy traffic conditions. DSR shows a sharp decline in delivery ratio as the number connections increase. With 1 CBR connection the PDF for DSR is higher than 95 percent.

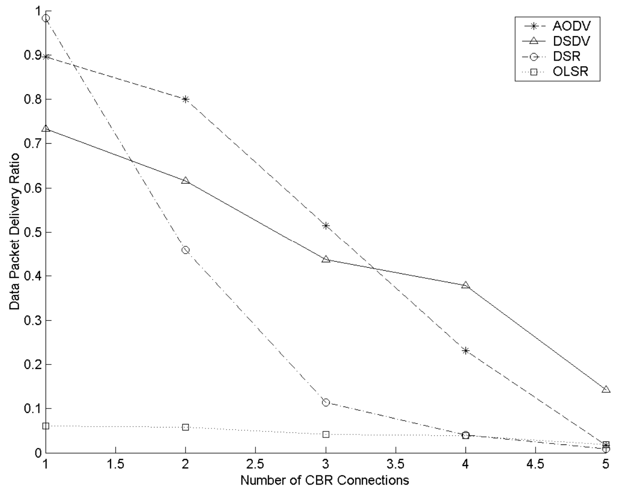


Figure 9: Packet delivery ratio for different number of CBR connections

By increasing the number of CBR connections from 1 to just 2 causes the delivery ratio to decrease by half and subsequently it falls to less than 1 percent for maximum traffic conditions. DSDV has the best overall delivery ratio. Even at high traffic conditions DSDV shows 8 times more delivery ratio than AODV and OLSR and 17 times more ratio than DSR. Also, DSDV does not exhibit any sudden sharp decline in the delivery ratio. Performance of AODV with respect to PDF turns out to be inversely proportional with the number of connections. This is expected from a reactive protocol. As more packets are created by the application layer, more route requests are created, thus causing more routing control packets to float in the network which in turn reduces the actual throughput. AODV performs better than DSDV in case of lesser number of connections with a packet delivery ratio of up to 90 percent but shows a degraded performance in case of higher number of connections with delivery ratio of less than 2 percent for the scenario with highest CBR traffic.

4.1.2 PDF for different node depths

Packet delivery ratio vs. node depth is shown in Figure 10. It is clearly evident that AODV performs better than DSDV, OLSR and DSR at all depth as it has the highest data delivery ratio of more than 50 percent.

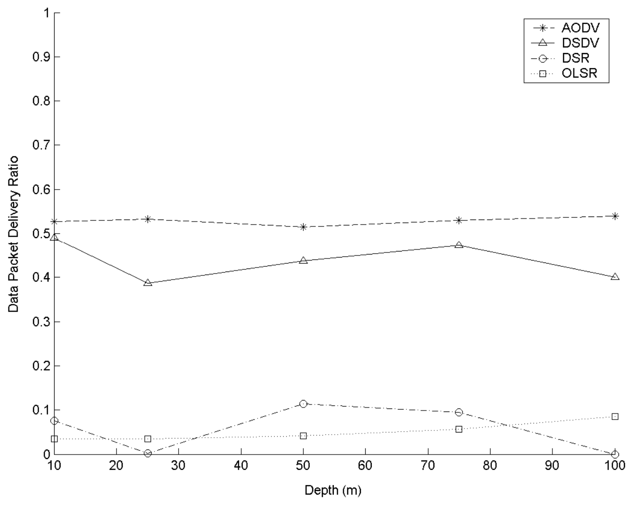


Figure 10: Packet delivery ratio for different node depths

Delivery ratio for AODV remains almost unaffected by the change in depth. The slight variations observed are because of channel characteristics. DSDV has significantly high data delivery ratios then OLSR and DSR but lesser than AODV for all depths. The maximum PDF observed is 49 percent and minimum recorded PDF is 38 percent. DSR and OLSR both very low PDFs. DSR has almost a PDF of zero at the depths of 25 meters and 100 meters. However the results are slightly higher than OLSR at other depths. Although, OLSR has very low PDFs for all depths, but the PDF remains unaffected by the variations in depth.

4.1.3 PDF for different number of nodes

Packet delivery ratio vs. different number of nodes is shown in Figure 11. AODV has the most stable packet delivery ratio and does not fluctuate much with the increase in number of nodes. AODV has a standard deviation of 2.27 for packet delivery ratio and DSR, DSDV and OLSR have standard deviations of 5.95, 6.43 and 13.08 respectively.

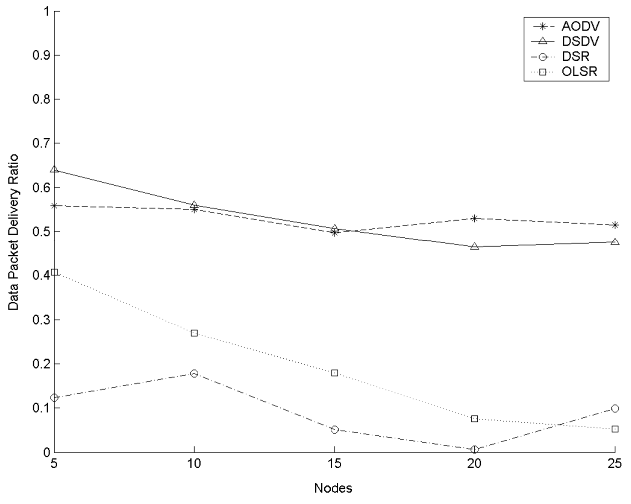


Figure 11: Packet delivery ratio for different number of nodes

DSDV also has a rather good PDF when compared to AODV. With 5, 10 and 15 number of nodes, DSDV has slightly higher delivery ratio than AODV and with 20 and 25 nodes, AODV performs better than DSDV. This is expected from both protocols. DSDV is proactive in nature and is table driven. Adding more nodes, require routing tables to be exchanged among more nodes thus creating a relatively greater overhead. The increased overhead causes the PDF to decline. This effect is obvious when network has lesser number of nodes and it norms out as network size grows. A similar graph trend is observed by OLSR because it is also a proactive routing protocol. The PDF values for OLSR are lesser than DSDV and ranges between 40 percent to slightly above 5 percent but the shape of the graph is almost identical to DSDV. Average delivery ratio is lowest for DSR for all different number of nodes and tends to fluctuate with the change in number of nodes.

4.2 Average End-to-End Delay

The sum of time taken by all the packets to move from source to destination divided by the total number of successful packets is called average end-to-end delay.

4.2.1 Average end-to-end delay for different traffic conditions

End-to-end delay vs. different number of CBR connections is shown in Figure 12. DSR clearly has the highest delay for all different traffic conditions.

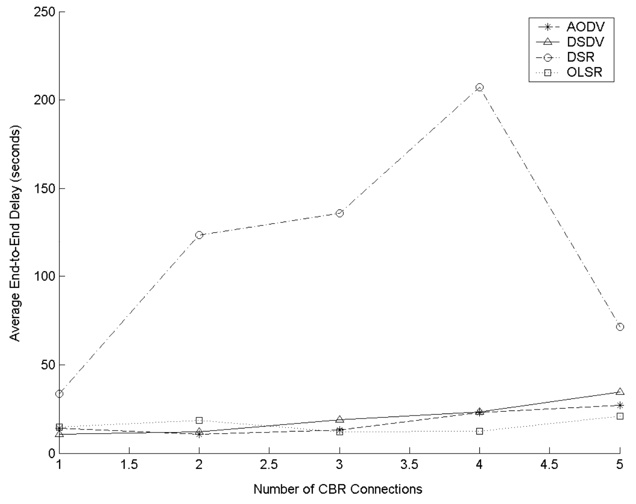


Figure 12: Average end-to-end delay for different traffic conditions

On average DSR has end-to-end delay 7 times higher than OLSR, 6 times higher than DSDV and 5 times higher than AODV. It is observed for DSR that increasing the traffic causes the end-to-end delay, however, when number of CBR connections is increased to 5, a sudden drop by a factor of 3 in end-to-end delay is observed. This because DSR is a source based routing protocol and routes are kept in cache for only a certain period of time. When data traffic is low, route validity timeout occurs more frequently causing the protocol to initiate discovery process. This discovery process causes the end-to-end delay to increase. But for higher traffic conditions, more routes are available through the node's route cache. Lesser discovery processes are initiated, thus reducing end-to-end delay significantly than the previous value. In case of AODV, DSDV and OLSR have comparable end-to-end delays and increase in traffic causes the delay to increase.

4.2.2 Average end-to-end delay for different node depths

The effect of depth on end-to-end delay is shown in Figure 13. DSR has the highest average end-to-end delay and is most affected by the change in depth. On average DSR has delay 6 times higher than AODV, 5 times higher than DSDV and 7 times higher than OLSR. AODV tends to maintain a steady end-to-end delay and does not fluctuate much with the increase or decrease in depth.

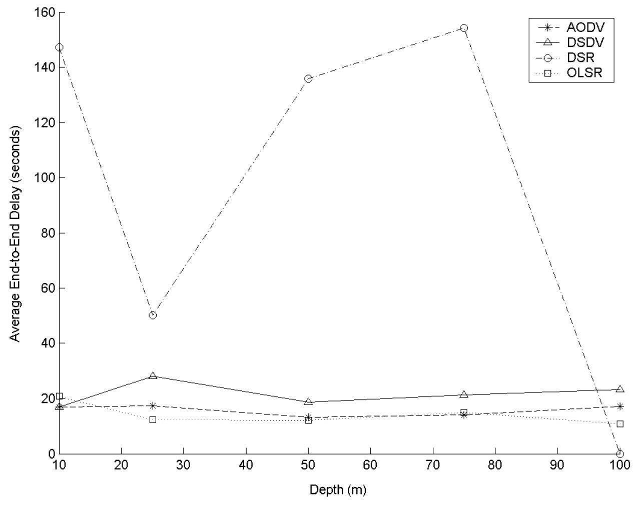


Figure 13: Average end-to-end delay for different node depths

DSDV has slightly higher delays for all depths than AODV. OLSR's graph for end-to-end delays is very much same to AODV and it actually has lower delays than AODV at the depths of 25 and 100 meters.

4.2.3 Average end-to-end delay for different number of nodes

DSR again has the highest delay when analyzed with respect to different number of nodes as shown in Figure 14. AODV, DSDV and OLSR have comparable end-to-end delays.

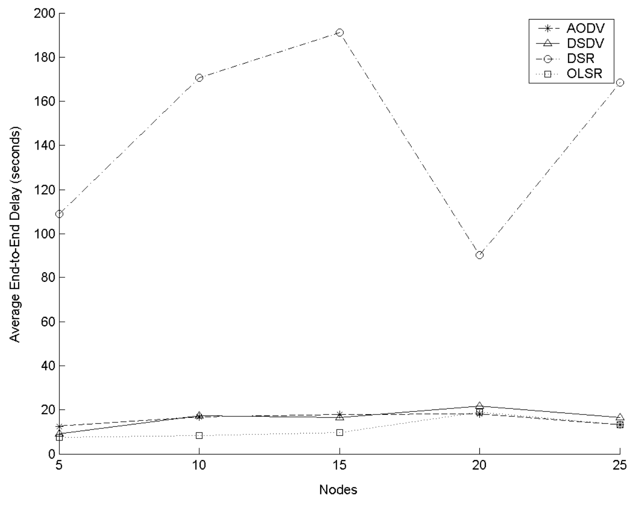


Figure 14: Average end-to-end delay for different number of nodes

Overall OLSR has lowest average end-to-end delay. AODV has the second lowest and DSDV has slightly higher average end-to-end delay than the other two protocols.

4.3 Throughput

Throughput is the rate of successful delivery messages over a communication link and usually expressed in the terms of bits per seconds. It is defined in Chapter 3. The following sections present the comparative findings of routing protocols with respect to throughput for different simulation scenarios.

4.3.1 Throughput for different traffic conditions

Throughput as a function of number of CBR connections is shown in Figure 15. Up to a certain point, throughput increases for AODV and DSDV as the traffic is increased. But with maximum traffic, throughput for AODV and DSDV drops to half of its previous value. Although, OLSR has the lowest data rate but at the same time it has the most steady data rate and remains unaffected by the change in traffic conditions. Throughput in case of DSR is highly affected by the traffic conditions and drops to very low data rate for maximum traffic conditions. DSDV performs exceptionally well at higher traffic conditions by having a data rate almost 6 times higher than AODV, which had highest data rate for lower traffic conditions.

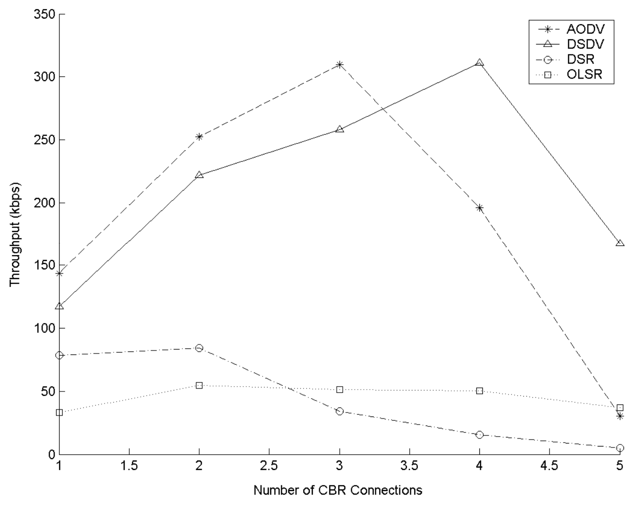


Figure 15: Throughput for different traffic conditions

At higher traffic conditions, routing overhead plays a key role in determining the overall throughput of the network. For reactive protocols like AODV and DSR, the routing overhead overwhelms the throughput and this why the graph takes a nose dive. However, with proactive protocols increasing the traffic does increase the route lookups but this does not adversely affect the throughput. As long as the topology is kept constant, amount of routing information exchanged will remain the same only the frequency of exchange of information will change due to broken or unavailable links caused by channel or energy depletion.

4.3.2 Throughput for different node depths

With the increase in node depth, throughput tends to improve slightly for OLSR and AODV as shown in Figure 16. AODV has the highest throughput for all depths and DSR has the least. Throughput for DSDV and DSR slightly fluctuates with the change in depth, showing a relative decrease in throughput at the depths of 25m and 100m.

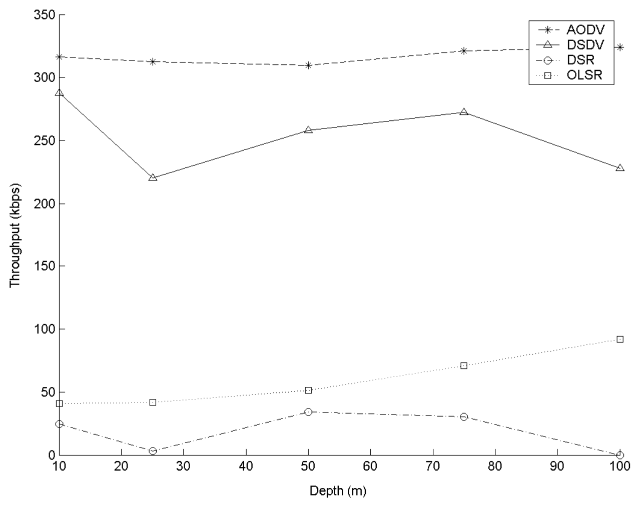


Figure 16: Throughput for different node depths

4.3.3 Throughput for different number of nodes

As shown in Figure 17, throughput for DSDV and OLSR decreases with the increase in number of nodes, but after a certain number of nodes throughput improves slightly.

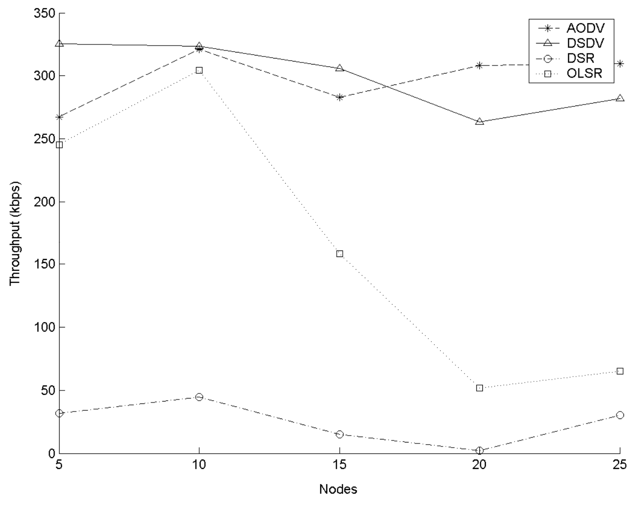


Figure 17: Throughput for different number of nodes

Throughput for OLSR decreases by a factor of 6 when number of nodes is increased from 10 to 20 and is improves by a factor of 1.3 when node count is further increased to 25. This is expected by proactive protocols, as nodes are increased more and more routing packets are generated thus affecting the actual throughput. Initially with lesser nodes this impact is higher and very visible, but afterward this effect gradually norms out and throughput takes a steady shape. The other reason behind this behavior, which is also true for reactive protocols, is more nodes mean more routes are available to the destination and thus contributing towards the increase in throughput. We can observe this for AODV and DSR in Figure 17, throughput for 25 nodes is actually slightly better than the throughput for 20 for each protocol. Throughput for AODV and DSR is not steady and is rather fluctuating in nature. This is due to their reactive nature, as number of nodes is changed the routes are changed which in turn affects the packet delivery ratio. A change in packet delivery ratio directly affects the average throughput of the nodes.

4.4 Routing Overhead

All routing protocols send packets for activities related to route discovery and route maintenance. These routing packets are not the actual pay load of the network and hence treated as an overhead. This routing overhead may not be a serious concern in case of wired network where we have an abundance of bandwidth with high propagation speeds. Routing overhead is major concern in case of wireless networks and especially in the case of underwater acoustic networks. Due to the nature of the acoustic propagation in underwater, we have limited bandwidth and high signal to noise ratios, the routing over head is a major concern. We would like to have a routing protocol with as low routing overhead as possible but at the same time not compromising the routing efficiencies.

4.4.1 Routing Overhead for different traffic conditions

Overhead of routing protocols as a function of number of data connections is shown in Figure 18. OLSR and DSDV have almost a constant overhead for all traffic conditions and do not change abruptly with the increase in data traffic.

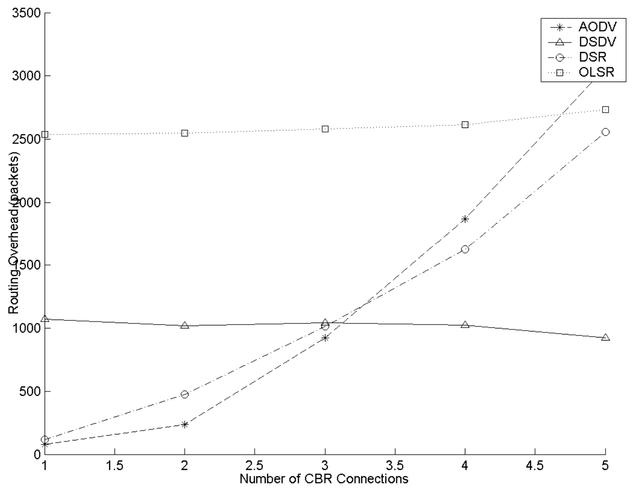


Figure 18: Routing overhead for different traffic conditions

This is expected from proactive routing protocols. Because the routing protocol produces almost the same amount of routing packets irrespective of the traffic conditions as long as the number of nodes and topology is kept constant. OLSR has significantly higher routing overhead when compared against DSDV. Roughly 1.5 times more routing is traffic is generated by OLSR than DSDV. The normalized routing load for each routing protocol for different traffic conditions is show in Figure 19. From the graph it is evident that reactive protocols are highly affected by the increase in number of connections. NRL for 5 CBR connections increases many folds for AODV and DSR. The NRL for DSDV decreases with the increase in traffic but increases slightly again for 5 CBR connections.

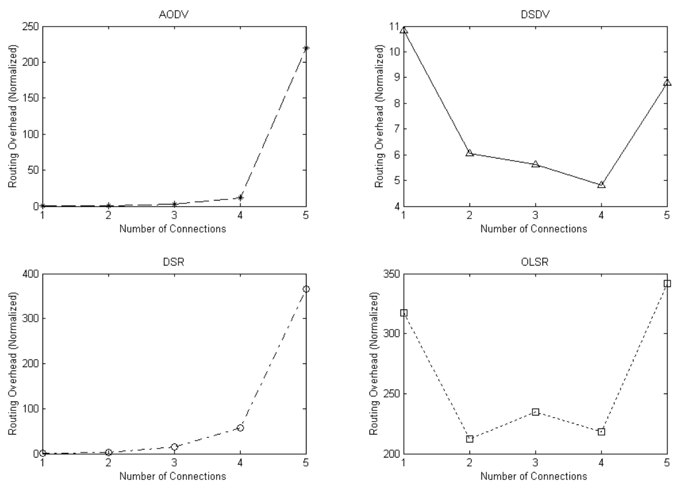


Figure 19: NRL for different traffic conditions

Routing overhead for AODV and DSR increases as the traffic is increased. This is because both protocols are reactive in nature and more routing packets are transmitted over the network as the data traffic increases because more route requests are initiated by sending and forwarding nodes. Overall routing overhead is highest for OLSR. AODV and DSR have lower overheads for lesser number of connection but overhead increases exponentially as the number of nodes increase.

4.4.2 Routing Overhead for different node depths

In Figure 20, it is observed that routing overhead for OLSR, DSR and DSDV remains almost constant irrespective of change in depth. OLSR again has the highest overhead among all protocols and then DSDV and then AODV.

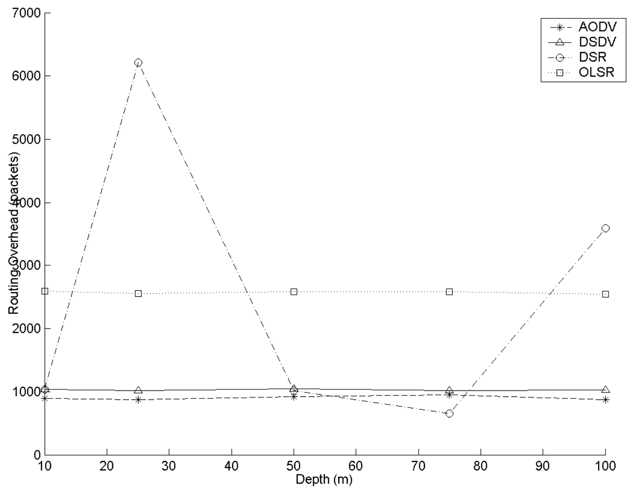


Figure 20: Routing overhead for different node depths

Again DSR shows a fluctuating behavior by having variable routing overhead at variable depths. The normalized routing load for different depths is given in Figure 21.

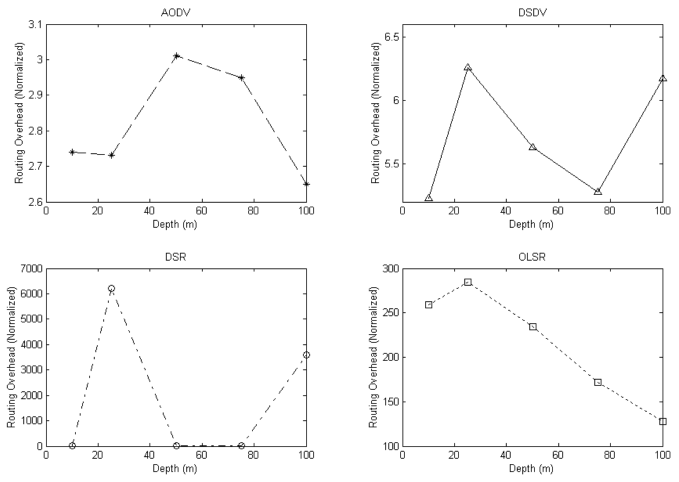


Figure 21: NRL for different node depths

The two higher peaks observed for DSR in Figure 21 is due to the fact that DSR had almost zero PDF at the depths of 25 and 100 meters. The depth of 25 meters seems not suitable for DSDV, DSR and OLSR when analyzed with respect to routing overhead. Such depths are referred as "*shallow waters*". Communication in shallow water is highly effected by the manmade and ambient noise.

4.4.3 Routing Overhead for different number of nodes

Figure 22 shows that the routing overhead increases with increase in number of nodes for OLSR, AODV and DSDV. This because as more nodes added to the network, more routing control information needs to exchanged among more number of nodes. The reason DSR shows a fluctuating behavior is because it is a source based routing protocol. And a small relative change in number of nodes actually helps the protocol in discovering and maintaining routes as more nodes have the probability of having a valid route

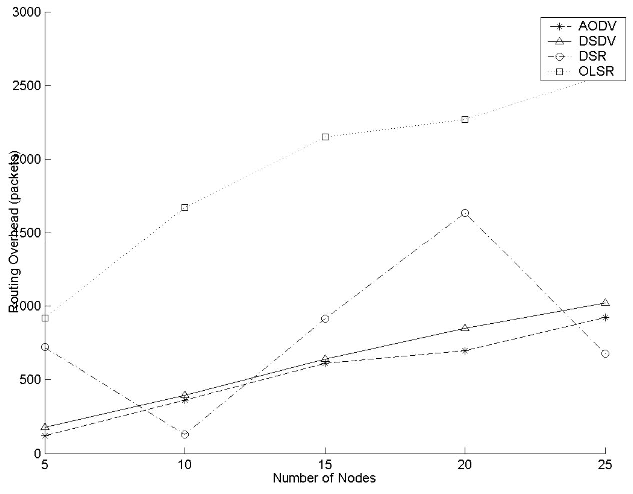


Figure 22: Routing overhead for different number of nodes

. OLSR has the highest routing overhead, roughly 5 times higher than AODV, more than 3 times higher than DSDV and 2.5 times higher than DSR. DSR performs better than AODV with 10 and 25 nodes and worse than AODV and DSDV with other number of nodes. The normalized routing load for all the protocols with different number of nodes is shown in Figure 23. We can see the general trend of NRL is that it increases with the increase in number of nodes, except for the case of DSR.

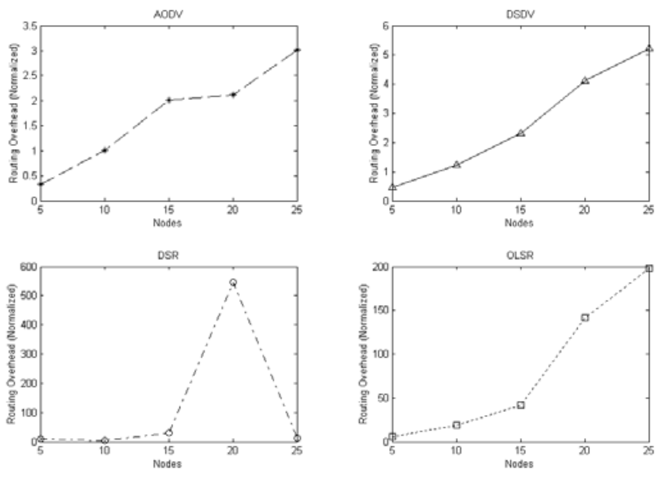


Figure 23: NRL for different number of nodes

4.5 Energy Consumption

Energy consumption of the nodes is a major concern in underwater networks in general. Usually it's not feasible to change or charge the batteries of the underwater sensors very often. So it is highly expected by a routing protocol to be light on the energy consumption needs.

4.5.1 Energy consumption for different traffic conditions

Average energy of nodes for different traffic conditions is shown in Figure 24. It is evident that OLSR has the highest rate of energy consumption irrespective of the number of connections. This explains the low PDF and throughput exhibited by OLSR in earlier results because nodes are getting depleted sooner than other in case of other protocols. OLSR is proactive in nature and has the highest routing overhead among the discussed protocols; this is a major reason for this high energy consumption. More number of routing packets means more send and receive operations by the nodes which cause them to lose their energy sooner. It is also observed for all protocols that more energy is consumed as the traffic increases. At the end of the simulation for AODV with 1 CBR connection, the average energy of the nodes is more than the 50 percent of their average initial energy. However as the number of CBR connection is increased to 5, average energy of the nodes at the end of the simulation drops to less than 10 percent of their initial value.

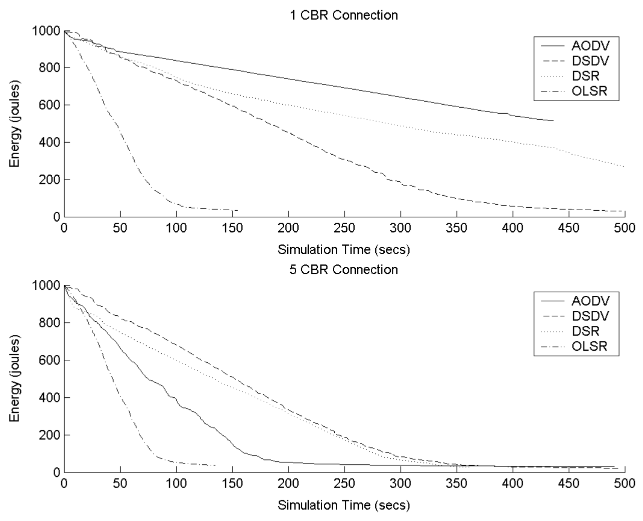


Figure 24: Energy consumed by nodes for different traffic conditions

A similar trend is observed for DSR where average energy of the nodes drops from 30 percent to less than 10 percent by changing the traffic condition from 1 CBR to 5 CBR connections.

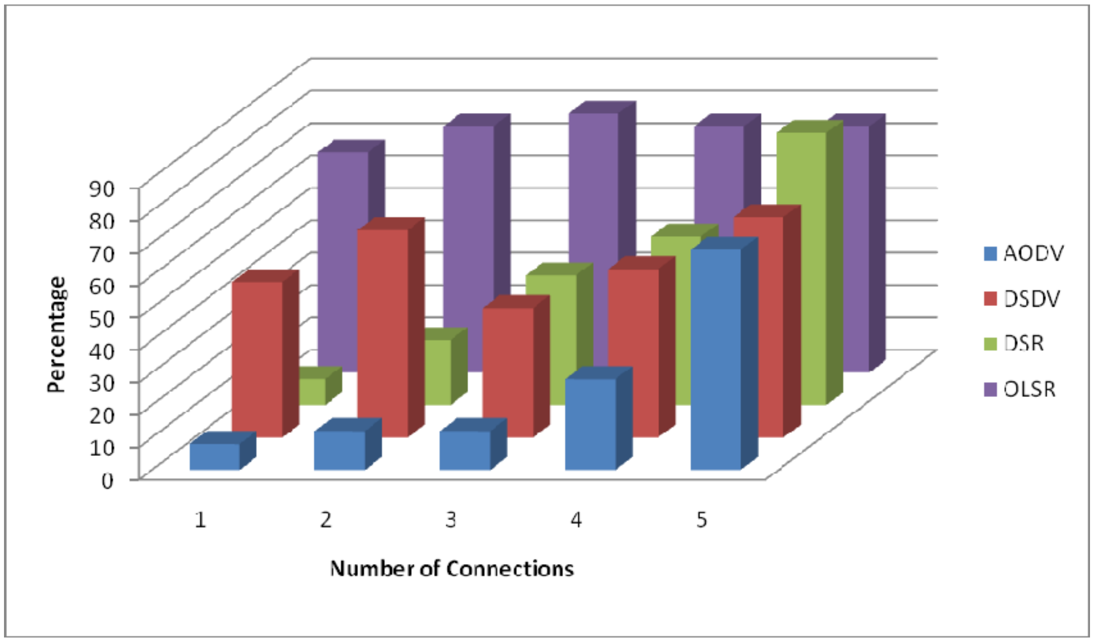


Figure 25: Percentage of node failures for different traffic scenarios

This is because both AODV and DSR are reactive in nature and as the traffic increases more route discovery requests are initiated which causes nodes to lose more energy as compared to lesser traffic scenarios. As DSDV is a proactive protocol and routing overhead does not change drastically with the increase in traffic, the average energy consumption at the nodes also remains unaffected, thus making it suitable for high traffic conditions. Figure 25 shows the percentage of node failures due to energy depletion for different traffic scenarios. For highest traffic conditions, DSDV has least percentage node failures and DSR has the highest percentage of node failures.

4.5.2 Energy Consumption for different node depths

The average energy of the nodes remains almost unaffected and unchanged for AODV, DSDV and OLSR with respect to change in node depth. This is shown in Figure 26. Again OLSR has the highest rate of energy consumption. AODV has the lightest effect on the energy level of the sensor nodes.

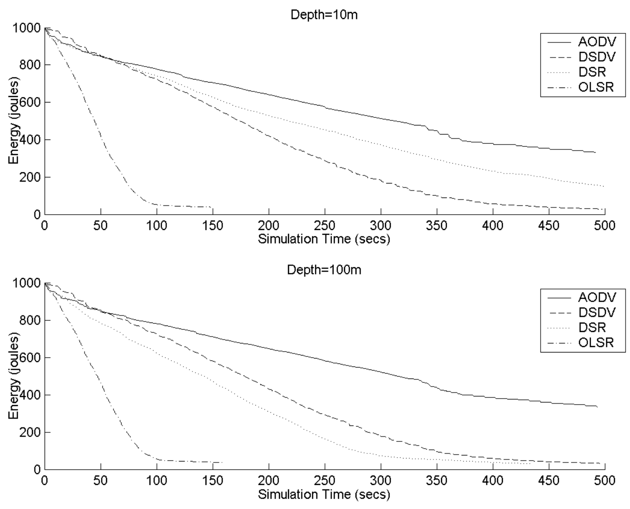


Figure 26: Energy consumption of nodes for different node depths

However DSR consumes approximately 10 times more energy at 100m depth than at 10m depth. The percentage of node failures due to energy depletion at different depths is presented in Figure 27 The variations in the percentage node failures for different depths is not much in AODV, DSDV and OLSR. However, DSR experiences relatively greater node failures at the depths of 25 and 100 meters.

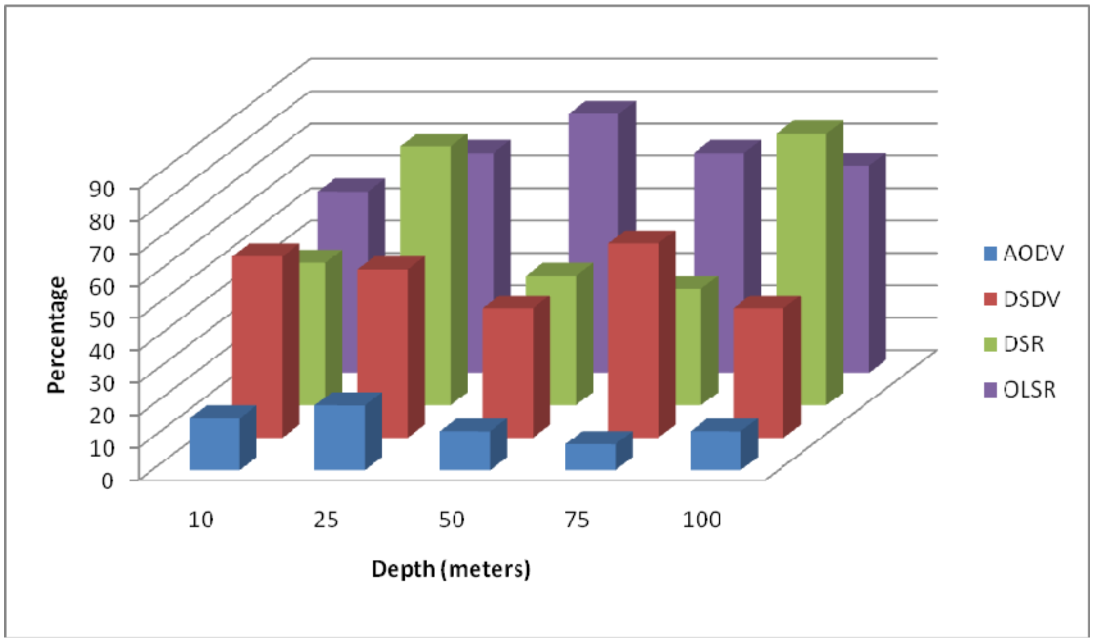


Figure 27: Percentage of node failures for different depths

4.5.3 Energy Consumption for different number of nodes

Average node energy decreases with the increase in number of nodes for all routing protocols. This is observed in Figure 28. With 5 nodes, the average energy of the nodes for all protocols is higher than 80 percent of the initial energy level at the end of the simulation. This figure drops to 50, 30, 10 and 10 percent for AODV, DSR, DSDV and OLSR respectively when nodes are increased to 25. This is expected from all protocols, as number of nodes increase, more packets are generated by each routing protocol and more packets are exchanged among them. This increased transmission drains the energy of the nodes. Again OLSR causes the nodes to deplete sooner as compared to other routing protocols. Figure 29 shows the percentage of node failures due to energy depletion for different number of nodes.

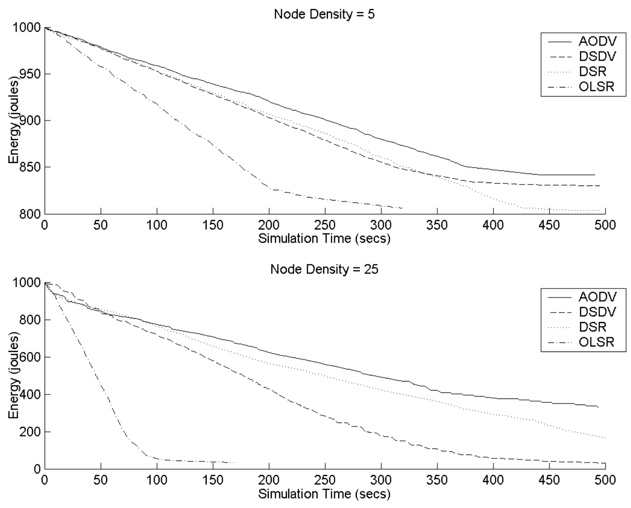


Figure 28: Energy consumption of nodes for different number of nodes

For 25 nodes, the percentage of node failures is actually lesser than the percentage of node failures for 20 nodes. This is because the optimal number of nodes in the network ensures the availability of alternate routes to the destination. These alternate routes distribute the load of forwarding packets to different nodes, thus not letting any specific node starve for energy. This effect is highly visible and prominent in case of reactive dynamic protocols like DSR.

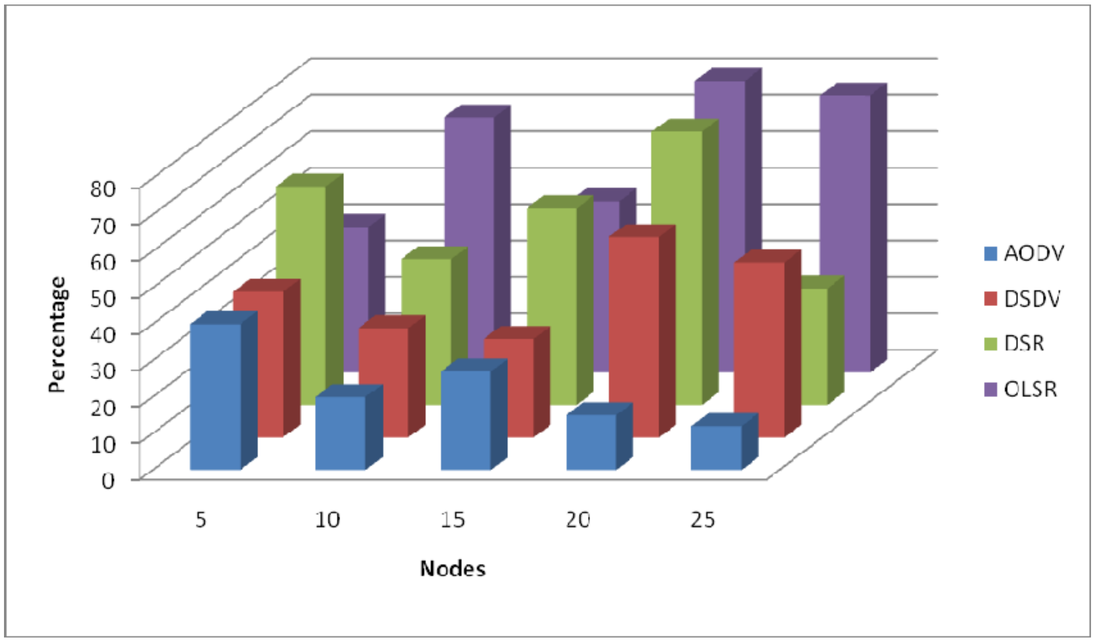


Figure 29: Percentage of node failures for different number of nodes

Chapter 5

Conclusion & Future Work

5.1 Conclusion

In this paper we made an effort to examine four popular ad-hoc routing protocols in underwater acoustic networks environment. After analyzing the gathered results, we concluded that OLSR is not suitable for underwater networks due to its high rate of energy consumption. Energy consumption of sensor nodes is always a major concern in underwater networks. OLSR is also not suitable because of its high routing overhead as compared to other routing protocols. DSR will also not be suitable for underwater environment because it shows very low packet delivery ratios, although DSR consumes lesser energy as compared to OLSR. DSR is also not recommended because throughput, end-to-end delay and routing overhead sharply changes as number of connections, depth or number of nodes is changed. AODV and DSDV on the other hand tend to perform better but both have some tradeoffs. AODV has higher delivery ratio for lesser number of connections than DSDV and DSDV has higher delivery ratios for more number of connections. This directly reflects on throughput. AODV has higher throughput for lesser connections than DSDV but throughput decreases sharply as traffic increases. Both AODV and DSDV have steady delivery ratios and end-to-end delays and do not fluctuate much with the change in number of nodes and depth. Overall delay is observed lesser in case of AODV as compared to DSDV. AODV has very little routing overhead for less traffic but increases multiplicatively with the increase in traffic. This also effects the energy consumption of the nodes. Nodes run out of energy sooner in high traffic conditions for AODV. DSDV, more or less, has the same routing overhead for all traffic conditions. This may be a higher figure for lesser traffic conditions but as the traffic increases this is evened out. Energy consumption for DSDV is actually better than AODV in higher traffic conditions. To conclude this paper, AODV is recommended for denser underwater networks but with less traffic. DSDV is suitable for higher traffic conditions with optimal number of nodes. In continuation of this research work, it will be interesting to evaluate MAC layer protocols and suggest modifications required for their optimal working in underwater acoustic sensor networks.

References

[1] David B. Johnson, David A. Maltz, and Josh Broch, "*Underwater acoustic sensor networks: research challenges,*" Ad-Hoc Networks, Volume 3, Issue 3, Pages 257-279, May 2005.

[2] James Preisig, "*Acoustic propagation considerations for underwater acoustic communications network development*". In WUWNet '06: Proceedings of the 1st ACM international workshop on Underwater networks, pages 1–5, New York, NY, USA, 2006. ACM Press.

[3] D. Pompili, T. Melodia, and I. F. Akyildiz, "*Deployment Analysis in Underwater Acoustic Wireless Sensor Networks*," in Proc. of ACM International Workshop on UnderWater Networks (WUWNet), Los Angeles, CA, September 2006.

[4] David B. Johnson, David A. Maltz, and Josh Broch, *"DSR:The dynamic Source Routing Protocols for Multi-Hop Wireless Ad Hoc Networks,"* in Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5, pp.139-172, Addison-Wesley, 2001.

[5] C. E. Perkins, and E. M. royer, "*Ad-Hoc On-Demand Distance Vector Routing,*" in proceedings of the 2nd IEEE Workshop on Mobile Computing systems and Applications, New Orleans, LA, pp. 90-100, February 1999.

[6] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "*Optimized Link State Routing Protocol for Ad-Hoc Networks*," in Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001),2001.

[7] Perkins, Charles E. and Bhagwat, Pravin,"*Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*," ACM, 1994.

[8] The Network Simulator, NS-2, *Available from www.isi.edu/nsnam/ns*

[9] F. Harris and M. Zorzi. "*Modeling the underwater acoustic channel in ns2,*" In ACM International Workshop on Network Simulation Tools. (NSTools), 2007

[10] Jorjeta G. Jetcheva and David B. Johson,"A Performance Comparison of On-Demand Multicast Routing Protocols for Ad Hoc Networks", School of computer science, CS Dept, Pittsburgh, December 15, 2004.

[11] David Oliver Jorg, "*Performance Comparison of MANET Routing Protocols in Different Network Sizes*", Computer Science Project, Institute of Computer Science and Applied Mathematics, University of Berne, Switzerland, 2003.

[12] Uyen Trang Nguyen and Xing Xiong," *Rate-adaptive Multicast in Mobile Ad-hoc Networks,*", IEEE International Conference on Adhoc and Mobile Computing, Networking and Communications 2005 (WiMob 2005),Montreal, Canada, August 2005.

[13] Technical Guides - Speed of Sound in Sea-Water, *Available from* *http://resource.npl.co.uk/acoustics/techguides/soundseawater/*

[14] Römer, Kay; Friedemann Mattern (December 2004). "*The Design Space of Wireless Sensor Networks*". IEEE Wireless Communications 11 (6): 54–61. doi:10.1109/MWC.2004.1368897

[15] D. Pompili and T. Melodia, "*An Architecture for Ocean Bottom Underwater Acoustic Sensor Networks*," Poster Presentation, Proc. of Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Bodrum, Turkey, June 2004

[16] P. Xie, J. Cui and L. Li, "VBF: Vector-Based Forwarding Protocol for Underwater Sensor Networks," 2005.

[17] N. Shenoy, Y. Pan and V. G. Reddy, "Quality of service in internet MANETs," in 2005, pp. 1823-1829 Vol. 3.

[18] M. Lee and V. W. S. Wong, "An energy-aware spanning tree algorithm for data aggregation in wireless sensor networks," in 2005, pp. 300-303.

Appendix A: Sample TCL Simulation Script

```
# Author: Naveed Qadri
# Date:   11/27/2009
# Email: nbqadri@nbqadri.com
# Web: http://www.nbqadri.com

# ======================================================================
# Define options
# ======================================================================

set val(chan)           Channel/UnderwaterChannel;# channel type#
set val(prop)           Propagation/Underwater   ;# radio-propagation model
set val(netif)          Phy/UnderwaterPhy        ;# network interface type
set val(mac)            Mac/802_11               ;# MAC type
set val(ifq)            CMUPriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         10                         ;# max packet in ifq
set val(nn)             25                         ;# number of mobilenodes
set val(rp)             DSR                        ;# routing protocol
set opt(energymodel)    EnergyModel                ;# using ns-energy model
set opt(initialenergy)  1000.0                  ;# Initial energy in Joules
set rx 350                         ;# receive energy in mW  avg 500 mW
set tx 2500                        ;# transmit energy in mW avg upto
2.5W for greater distances

# ======================================================================
# Overriding NS-Defaults
# ======================================================================

Phy/UnderwaterPhy set CPThresh_ 10.0;
Phy/UnderwaterPhy set CSThresh_ 0.284
Phy/UnderwaterPhy set RXThresh_ 4.0;
Phy/UnderwaterPhy set bandwidth_ 2e5 ;
Phy/UnderwaterPhy set Pt_ 65 ;
Phy/UnderwaterPhy set freq_ 300;
Phy/UnderwaterPhy set L_ 1.0
Mac/802_11 set CWMin_        4;
Mac/802_11 set CWMax_        32;
Mac/802_11 set SlotTime_     0.020;
Mac/802_11 set SIFS_         0.010        ;
Mac/802_11 set PreambleLength_       144             ;# 144 bit
Mac/802_11 set PLCPHeaderLength_     48              ;# 48 bits

# ======================================================================
# Main Program
# ======================================================================
# Initialize Global Variables

set ns_           [new Simulator]
set tracefd     [open uwtrace-5con_DSR.tr w]
set nf [open nam-uw-5con_DSR.nam w]
set f1 [open uw-5con-flow1_DSR.tr w]
set f2 [open uw-5con-flow2_DSR.tr w]
set f3 [open uw-5con-flow3_DSR.tr w]
set f4 [open uw-5con-flow4_DSR.tr w]
set f5 [open uw-5con-flow5_DSR.tr w]
```

```
$ns_ use-newtrace ;# new trace format for Wireless

#$ns_ flush-trace
# must remove later.. shud clear memory
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $nf 15 15

# ===========    set up topography of object    =========================

set topo       [new Topography]
$topo load_flatgrid 15 15


# Create God

create-god $val(nn)

# =================  Configure node  ===============================

        $ns_ node-config -adhocRouting $val(rp) \
                   -llType $val(ll) \
                   -macType $val(mac) \
                   -ifqType $val(ifq) \
                   -ifqLen $val(ifqlen) \
                   -antType $val(ant) \
                   -propType $val(prop) \
                   -phyType $val(netif) \
                   -channelType $val(chan) \
                         -energyModel $opt(energymodel) \
                   -rxPower $rx \
                   -txPower $tx \
                   -initialEnergy $opt(initialenergy) \
                   -topoInstance $topo \
                   -agentTrace ON \
                   -routerTrace ON \
                   -macTrace ON \
                   -movementTrace OFF

      for {set i 0} {$i < $val(nn) } {incr i} {
            set node_($i) [$ns_ node]
            $node_($i) random-motion 0          ;# disable random motion
      }


# == Provide initial (X,Y, for now Z=15) co-ordinates for mobilenodes =====

$node_(0) set X_ 0.250
$node_(0) set Y_ 0.250
$node_(0) set Z_ 0.05

$node_(1) set X_ 0.054
$node_(1) set Y_ 0.350
$node_(1) set Z_ 0.05

$node_(2) set X_ 0.305
$node_(2) set Y_ 0.440
$node_(2) set Z_ 0.05

$node_(3) set X_ 0.455
```

```
$node_(3) set Y_ 0.420
$node_(3) set Z_ 0.05


$node_(4) set X_ 0.126
$node_(4) set Y_ 0.400
$node_(4) set Z_ 0.05

$node_(5) set X_ 0.038
$node_(5) set Y_ 0.173
$node_(5) set Z_ 0.05

$node_(6) set X_ 0.175
$node_(6) set Y_ 0.475
$node_(6) set Z_ 0.05

$node_(7) set X_ 0.371
$node_(7) set Y_ 0.126
$node_(7) set Z_ 0.05

$node_(8) set X_ 0.250
$node_(8) set Y_ 0.350
$node_(8) set Z_ 0.05

$node_(9) set X_ 0.161
$node_(9) set Y_ 0.045
$node_(9) set Z_ 0.05

$node_(10) set X_ 0.325
$node_(10) set Y_ 0.080
$node_(10) set Z_ 0.05

$node_(11) set X_ 0.500
$node_(11) set Y_ 0.310
$node_(11) set Z_ 0.05

$node_(12) set X_ 0.010
$node_(12) set Y_ 0.495
$node_(12) set Z_ 0.05

$node_(13) set X_ 0.150
$node_(13) set Y_ 0.250
$node_(13) set Z_ 0.05

$node_(14) set X_ 0.350
$node_(14) set Y_ 0.250
$node_(14) set Z_ 0.05

$node_(15) set X_ 0.170
$node_(15) set Y_ 0.360
$node_(15) set Z_ 0.05

$node_(16) set X_ 0.250
$node_(16) set Y_ 0.150
$node_(16) set Z_ 0.05

$node_(17) set X_ 0.360
$node_(17) set Y_ 0.370
$node_(17) set Z_ 0.05

$node_(18) set X_ 0.427
```

```
$node_(18) set Y_ 0.200
$node_(18) set Z_ 0.05

$node_(19) set X_ 0.497
$node_(19) set Y_ 0.015
$node_(19) set Z_ 0.05

$node_(20) set X_ 0.260
$node_(20) set Y_ 0.010
$node_(20) set Z_ 0.05

$node_(21) set X_ 0.125
$node_(21) set Y_ 0.125
$node_(21) set Z_ 0.05

$node_(22) set X_ 0.092
$node_(22) set Y_ 0.265
$node_(22) set Z_ 0.05

$node_(23) set X_ 0.001
$node_(23) set Y_ 0.005
$node_(23) set Z_ 0.05

$node_(24) set X_ 0.501
$node_(24) set Y_ 0.495
$node_(24) set Z_ 0.05
# ========== Keeping nodes at X meters apart. ===========================
$ns_ at 0.10 "$node_(0) setdest 0.250 0.250 0.50"
$ns_ at 0.10 "$node_(1) setdest 0.054 0.350 0.50"
$ns_ at 0.10 "$node_(2) setdest 0.305 0.440 0.50"
$ns_ at 0.10 "$node_(3) setdest 0.455 0.420 0.50"
$ns_ at 0.10 "$node_(4) setdest 0.126 0.400 0.50"

$ns_ at 0.10 "$node_(5) setdest 0.038 0.173 0.50"
$ns_ at 0.10 "$node_(6) setdest 0.175 0.475 0.50"
$ns_ at 0.10 "$node_(7) setdest 0.371 0.126 0.50"
$ns_ at 0.10 "$node_(8) setdest 0.250 0.350 0.50"
$ns_ at 0.10 "$node_(9) setdest 0.161 0.045 0.50"

$ns_ at 0.10 "$node_(10) setdest 0.325 0.080 0.50"
$ns_ at 0.10 "$node_(11) setdest 0.500 0.310 0.50"
$ns_ at 0.10 "$node_(12) setdest 0.010 0.495 0.50"
$ns_ at 0.10 "$node_(13) setdest 0.150 0.250 0.50"
$ns_ at 0.10 "$node_(14) setdest 0.350 0.250 0.50"

$ns_ at 0.10 "$node_(15) setdest 0.170 0.360 0.50"
$ns_ at 0.10 "$node_(16) setdest 0.250 0.150 0.50"
$ns_ at 0.10 "$node_(17) setdest 0.360 0.370 0.50"
$ns_ at 0.10 "$node_(18) setdest 0.427 0.200 0.50"
$ns_ at 0.10 "$node_(19) setdest 0.497 0.015 0.50"

$ns_ at 0.10 "$node_(20) setdest 0.260 0.010 0.50"
$ns_ at 0.10 "$node_(21) setdest 0.125 0.125 0.50"
$ns_ at 0.10 "$node_(22) setdest 0.092 0.265 0.50"
$ns_ at 0.10 "$node_(23) setdest 0.001 0.005 0.50"
$ns_ at 0.10 "$node_(24) setdest 0.501 0.495 0.50"

# ============ Setup traffic flow between nodes =======================

set udp [new Agent/UDP]
$udp set fid_ 1
```

```
set sink [new Agent/LossMonitor]

set udp2 [new Agent/UDP]
$udp2 set fid_ 2
set sink2 [new Agent/LossMonitor]

set udp3 [new Agent/UDP]
$udp3 set fid_ 3
set sink3 [new Agent/LossMonitor]

set udp4 [new Agent/UDP]
$udp4 set fid_ 4
set sink4 [new Agent/LossMonitor]

set udp5 [new Agent/UDP]
$udp5 set fid_ 5
set sink5 [new Agent/LossMonitor]
# ============ Attach sources and sinks in required nodes ============
$ns_ attach-agent $node_(24) $udp
$ns_ attach-agent $node_(0) $sink

$ns_ attach-agent $node_(6) $udp2
$ns_ attach-agent $node_(0) $sink2

$ns_ attach-agent $node_(11) $udp3
$ns_ attach-agent $node_(0) $sink3

$ns_ attach-agent $node_(4) $udp4
$ns_ attach-agent $node_(0) $sink4

$ns_ attach-agent $node_(23) $udp5
$ns_ attach-agent $node_(0) $sink5
#8/24 16/3 13/20 12/11 19/17

# ============== Connect source and sink together =====================
$ns_ connect $udp $sink
$ns_ connect $udp2 $sink2
$ns_ connect $udp3 $sink3
$ns_ connect $udp4 $sink4
$ns_ connect $udp5 $sink5

# ====== Define, Assign and start transmission protocols ==============

# Creating CBR Traffic
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 20
$cbr set interval_ 2.0
$cbr attach-agent $udp
$ns_ at 0.0 "$cbr start"

set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 20
$cbr2 set interval_ 2.0
$cbr2 attach-agent $udp2
$ns_ at 0.0 "$cbr2 start"

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 20
$cbr3 set interval_ 2.0
$cbr3 attach-agent $udp3
$ns_ at 0.0 "$cbr3 start"
```

```
set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ 20
$cbr4 set interval_ 2.0
$cbr4 attach-agent $udp4
$ns_ at 0.0 "$cbr4 start"

set cbr5 [new Application/Traffic/CBR]
$cbr5 set packetSize_ 20
$cbr5 set interval_ 2.0
$cbr5 attach-agent $udp5
$ns_ at 0.0 "$cbr5 start"

$ns_ at 0.0 "record"
# ==========  Tell nodes when the simulation ends  =====================

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 500.0 "$node_($i) reset";
}
$ns_ at 500.0 "stop"
$ns_ at 500.50 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
    global ns_ tracefd nf f1 f2 f3 f4 f5
#nf
    $ns_ flush-trace
    close $tracefd
    close $nf
    close $f1
    close $f2
    close $f3
    close $f4
    close $f5
    exec nam nam-uw-5con_DSR.nam &
exec xgraph uw-5con-flow1_DSR.tr uw-5con-flow2_DSR.tr uw-5con-flow3_DSR.tr
uw-5con-flow4_DSR.tr uw-5con-flow5_DSR.tr &
    exit 0
}

# ============= Recording Additions ====================================

proc record {} {
        global udp sink f1 udp2 sink2 f2 udp3 sink3 f3 udp4 sink4 f4 udp5
sink5 f5; # Getting sinks from above and file handlers

        #Get an instance of the simulator
        set ns_ [Simulator instance]

      #Set the time after which the procedure should be called again
        set time 0.1

      #How many bytes have been received by the traffic sinks?
        set bw1 [$sink set bytes_]
        set bw2 [$sink2 set bytes_]
        set bw3 [$sink3 set bytes_]
        set bw4 [$sink4 set bytes_]
        set bw5 [$sink5 set bytes_]


      #Get the current time
        set now [$ns_ now]
        #Calculate the bandwidth (in MBit/s) and write it to the files
        #puts $f1 "$now [expr $bw1/$time*8/1000000]"
        puts $f1 "$now [expr $bw1]"
      puts $f2 "$now [expr $bw2]"
      puts $f3 "$now [expr $bw3]"
            puts $f4 "$now [expr $bw4]"
            puts $f5 "$now [expr $bw5]"
      #Reset the bytes_ values on the traffic sinks
        $sink set bytes_ 0
        $sink2 set bytes_ 0
        $sink3 set bytes_ 0
        $sink4 set bytes_ 0
        $sink5 set bytes_ 0

        #Re-schedule the procedure
        $ns_ at [expr $now+$time] "record"
}
#==========================================================================

puts "Starting Simulation..."
$ns_ run
```

Appendix B: AWK Analysis Scripts

```
# AWK Script for calculating basic paramters like drop, end2end delay etc


BEGIN {
     printf("\n..................Starting Basic 
Analysis.............................\n");
     nlines=0;   # number of trace lines
     nsends=0;
     nrecvs=0;
     ndrops=0;
     nfrwds=0;
     nmvmnt=0;
     nrpkts=0;   # number of routing packets
     nsndpkts=0; # number of send packets
     nrcvpkts=0; # number of rcv packets
     ndrppkts=0; # number of dropped packets
     ndrpbytes=0;# number of dropped bytes
     hpktID=0;   # highest packet ID occuring in trace
     e2edelay=0; # End-2-End delay
     tdelay=0;   # sum of all delays
     e2ercvd=0;  # recvd at end; for e2e calculation
     }

{  #-------------Getting Tokens-------------------------------------------
   action = $1;
   simtime= $3;   # simulation time
   trlevel=$19;   # trace level RTR, AGT, MAC
   pkttype=$35;   # -It packet AODV, DSR, Message
   pktsize=$37;   # -Il packet size
   pktID=$41;     # -Ii packet's unique ID



   node_id = $9;
   node_e=$17;# -Ne


   #-------------Processing---------------------------------------------
   nlines++;

   if (action == "s")
       nsends++;
   if (action == "r")
       nrecvs++;
   if (action == "d")
       ndrops++;
   if (action == "f")
       nfrwds++;
   if (action == "M")
       nmvmnt++;
   if (action != "s" && action != "r" && action != "d"  && action != "f" && 
action != "M" && action != "N")
       printf("Some other action is ... %s at line %f\n",action,nlines);

  # ----  Calculating Routing Overhead ----
```

```
   if ((action == "s" || action == "f") && trlevel == "RTR" && (pkttype
=="message"||pkttype =="DSR"||pkttype =="AODV"||pkttype == "UM-OLSR"))
       nrpkts++;

  # ----  Calculation for PDF ----
   if ( action == "s" &&  trlevel=="AGT"  &&  pkttype=="cbr" )
       nsndpkts++;
   if ( action == "r" &&  trlevel=="AGT"  &&  pkttype=="cbr" )
          nrcvpkts++;

  # ----  Packets/Bytes Dropped ----
   if ( action == "d" &&  pkttype=="cbr" && simtime > 0)
      {
      ndrppkts++;
      ndrpbytes=ndrpbytes+pktsize;
       }

  # ----  Calculations for Packet Delay ----
   if ( action == "s" &&  trlevel=="AGT"  &&  pkttype=="cbr" ){

   if ( pktstarttime[pktID] == 0 )
        pktstarttime[pktID] = simtime;
}

   if ( action == "r" &&  trlevel=="AGT"  &&  pkttype=="cbr" )
      {
        pktendtime[pktID] = simtime;
        #printf("%s\n",$9);
      }
      else
         {
           if(pktendtime[pktID]==0)
           pktendtime[pktID] = -1;
         }

   if ( pktID > hpktID)
        hpktID = pktID;

}
END {

for ( i in pktendtime )
{
    #printf("%s\n",i);
    packet_duration =pktendtime[i]- pktstarttime[i];
    if ( packet_duration  > 0 )
    {
    tdelay+= packet_duration ;
    e2ercvd++;
    }

}# for ends here


e2edelay=tdelay/e2ercvd;

NRL=nrpkts/nrcvpkts; # Normalized Routing Load.
PDF=(nrcvpkts/nsndpkts)*100; # Pkt delivery fraction

printf("\n..................Results.............................\n");
printf("\nTotal Lines are %d\n",nlines);
printf("\nTotal Send Lines are %d\n",nsends);
printf("\nTotal Recv Lines are %d\n",nrecvs);
printf("\nTotal Drop Lines are %d\n",ndrops);
printf("\nTotal Frwd Lines are %d\n",nfrwds);
printf("\nTotal Mvnt Lines are %d\n",nmvmnt);
printf("\..........................\n\n");
printf("\nTotal Routing packets are %d\n",nrpkts);
printf("\nNormalized Routing Load %.2f\n",NRL);

printf("\nTotal CBR packets sent are %d\n",nsndpkts);
printf("\nTotal CBR packets rcvd are %d\n",nrcvpkts);
printf("\nPacket Delivery Fraction is  %.2f\n",PDF);

printf("\nTotal packets dropped (CBR) %d\n",ndrppkts);
printf("\nTotal Bytes (CBR) %d\n",ndrpbytes);

printf("\nTotal Delay %.2f\n",tdelay);
printf("\nTotal Rcvd for delay are %d\n",e2ercvd);
printf("\nAverage E2E Delay is %.2f\n",e2edelay);

}
```

```
# AWK Script for calculating hops and tracking routing paths


BEGIN {
     printf("\n..................Starting Analysis for Next Hop
Details..................\n");
     nlines=0;          # number of trace lines
     nfwdnodes=0;       # number of forwarding nodes
     max_nodes=25;      # maximum number of nodes in simulation
     nfwdevnts=0;       # number of forwarding events
     }

{  #-------------Getting Tokens-------------------------------------------
   action = $1;
   simtime= $3; # simulation time
   thishop=$5;  # -Hs current hop
   nexthop=$7;  # -Hd next hop towards dest.
   nodeID=$9;   # -Ni Node ID
   trlevel=$19; # trace level RTR, AGT, MAC
   srcIP=$31;   # -Is source adddress
   dstIP=$33;   # -Id destination address
   pkttype=$35; # -It packet AODV, DSR, Message
   pktsize=$37; # -Il packet size
   flowID=$39;  # -If Flow ID
   pktID=$41;   # -Ii packet's unique ID


   #--------------Processing----------------------------------------------
   nlines++;
   if (action=="f" && trlevel=="RTR")
       {
       #printf("%s %s NodeID %s Src %s Dest %s FID %s NextHop
%s\n",action,simtime,nodeID,srcIP,dstIP,flowID,nexthop);
       fwdnodes[nodeID]=1;
       datafwd[nodeID,nexthop]++;
       nfwdevnts++;
       }
}
```

```
END {
printf("\n................Results............................\n");
printf("\nMax nodes set to %d",max_nodes);
printf("\nTotal forwarding events are %d",nfwdevnts);
printf("\nForwarding Nodes are  ");
for(i in fwdnodes)
 {
  printf("%s  ",i);
  nfwdnodes++;
 }
printf("\nTotal forwarding nodes are %d\n",nfwdnodes);

printf("\nNodes with their next Hops\n");

for(j=0;j<max_nodes;j++)
{
  for(k=0;k<max_nodes;k++)
    {
    if(datafwd[j,k]!=0)
    printf("%s --> %s \n",j,k);
    }
}

}

printf("\nPoints for MATLAB PLOT Src Vs Dest Vs # of forwards in MATRIX
FORM\n");
for(j=0;j<max_nodes;j++)
{
  for(k=0;k<max_nodes;k++)
    {
    printf("%.2f ",datafwd[j,k]);
    }
printf("\n");
}

}
```

```
# AWK Script for calculating throughput

BEGIN {
     printf("\n..................Starting Throughput
Calculations............................\n");
     nlines=0;   # number of trace lines
     recvdSize = 0
     startTime = 1e6 # some very high value
     stopTime = 0
     count=0;
     }

{  #----------Getting Tokens----------------------------------------
   action = $1;
   simtime= $3;   # simulation time
   node_id= $9;   # -Ni Node ID
   trlevel=$19;   # trace level RTR, AGT, MAC
   pktsize=$37;   # -Il packet size
   flowID=$39;    # -If flow ID
   pktID=$41;     # -Ii packet's unique ID
   #-----------Processing------------------------------------------
   nlines++;
   if (action == "s" && trlevel == "AGT" && pktsize >= 1) {
       if (simtime < startTime) {
             startTime = simtime;
             }
       }



  if (action=="r" && trlevel == "AGT" && pktsize >= 1) {
       if (simtime > stopTime) {
             stopTime = simtime;
             }

       # Rip off the header
       hdr_size = pktsize % 1;
       pktsize -= hdr_size;

       # Store received packet's size
       recvdSize += pktsize
       count++;
       }


}
END {


printf("\n..................Results............................\n");
printf("Number of recieved pkts %s and rcvd size is %s\n",count,recvdSize);
printf("Average Throughput[kbps] = %.4f e-3\t
StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTime-
startTime)*8),startTime,stopTime)

}
```

```
# AWK Script for tracking energy for source and sink nodes


BEGIN {
     printf("\n............Tracking Energy for source/sink
nodes..................\n");
     nlines=0;   # number of trace lines
     temp=0;
     interval=1; # stepping interval
     energyth=2; # threshold
     maxnodes=25;
      }
{  #-----------Getting Tokens-------------------------------------------
   action = $1;
   simtime= $3;   # simulation time
   trlevel=$19;   # trace level RTR, AGT, MAC
   pkttype=$35;   # -Id packet AODV, DSR, Message
   pktsize=$37;   # -Il packet size
   pktID=$41;     # -Ii packet's unique ID

   node_id = $9;
   node_e=$17;# -Ne
   #---------------Processing--------------------------------------------
   nlines++;

   if (action != "s" && action != "r" && action != "d"  && action != "f" &&
action != "M" && action != "N" )
       printf("Some other action is ... %s at line %f\n",action,nlines);



if(action=="N" && $7>0)
  {
   if($5==0) {print(simtime,$7)>> "energy-node0.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==1) {print(simtime,$7)>> "energy-node1.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==2) {print(simtime,$7)>> "energy-node2.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==3) {print(simtime,$7)>> "energy-node3.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==4) {print(simtime,$7)>> "energy-node4.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==5) {print(simtime,$7)>> "energy-node5.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==6) {print(simtime,$7)>> "energy-node6.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==7) {print(simtime,$7)>> "energy-node7.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==8) {print(simtime,$7)>> "energy-node8.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==9) {print(simtime,$7)>> "energy-node9.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==10) {print(simtime,$7)>> "energy-node10.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==11) {print(simtime,$7)>> "energy-node11.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==12) {print(simtime,$7)>> "energy-node12.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==13) {print(simtime,$7)>> "energy-node13.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==14) {print(simtime,$7)>> "energy-node14.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==15) {print(simtime,$7)>> "energy-node15.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==16) {print(simtime,$7)>> "energy-node16.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==17) {print(simtime,$7)>> "energy-node17.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==18) {print(simtime,$7)>> "energy-node18.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==19) {print(simtime,$7)>> "energy-node19.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==20) {print(simtime,$7)>> "energy-node20.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==21) {print(simtime,$7)>> "energy-node21.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==22) {print(simtime,$7)>> "energy-node22.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==23) {print(simtime,$7)>> "energy-node23.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
   if($5==24) {print(simtime,$7)>> "energy-node24.tr"; if($7<energyth
&&time[$5]==0) {time[$5]=simtime;}}
```

```
  }
}   END {


for(i=0; i<maxnodes;i++){
print(i,time[i])>>"node-ran-out-of-power-at.tr";

}}
```

```
# AWK Script for tracking energy for source and sink nodes


BEGIN {
     printf("\n............Tracking Energy for source/sink
nodes...................\n");
     nlines=0;   # number of trace lines
     temp=0;
     interval=0.01; # stepping interval
     energyth=2; # threshold
     maxnodes=25;
     lastrecordtime=0;

lastenergy[0]=1000;lastenergy[1]=1000;lastenergy[2]=1000;lastenergy[3]=1000
;lastenergy[4]=1000;lastenergy[5]=1000;lastenergy[6]=1000;lastenergy[7]=100
0;lastenergy[8]=1000;lastenergy[9]=1000;lastenergy[10]=1000;lastenergy[11]=
1000;lastenergy[12]=1000;lastenergy[13]=1000;lastenergy[14]=1000;lastenergy
[15]=1000;lastenergy[16]=1000;lastenergy[17]=1000;lastenergy[18]=1000;laste
nergy[19]=1000;lastenergy[20]=1000;lastenergy[21]=1000;lastenergy[22]=1000;
lastenergy[23]=1000;lastenergy[24]=1000;
      }
{  #-----------Getting Tokens-------------------------------------
   action = $1;
   simtime= $3;   # simulation time
   trlevel=$19;   # trace level RTR, AGT, MAC
   pkttype=$35;   # -Id packet AODV, DSR, Message
   pktsize=$37;   # -Il packet size
   pktID=$41;     # -Ii packet's unique ID

   node_id = $9;
   node_e=$17;# -Ne



   #------------------------Processing------------------------------
   nlines++;

   if (action != "s" && action != "r" && action != "d"  && action != "f" &&
action != "M" && action != "N" )
       printf("Some other action is ... %s at line %f\n",action,nlines);


if(action=="N" && $7>0)
  {

if($5==0) { print(simtime,$7)>> "energy-node0.tr"; lastenergy[0]=$7; } else
{ print(simtime,lastenergy[0])>> "energy-node0.tr";}
if($5==1) { print(simtime,$7)>> "energy-node1.tr"; lastenergy[1]=$7; } else
{ print(simtime,lastenergy[1])>> "energy-node1.tr";}
if($5==2) { print(simtime,$7)>> "energy-node2.tr"; lastenergy[2]=$7; } else
{ print(simtime,lastenergy[2])>> "energy-node2.tr";}
if($5==3) { print(simtime,$7)>> "energy-node3.tr"; lastenergy[3]=$7; } else
{ print(simtime,lastenergy[3])>> "energy-node3.tr";}
if($5==4) { print(simtime,$7)>> "energy-node4.tr"; lastenergy[4]=$7; } else
{ print(simtime,lastenergy[4])>> "energy-node4.tr";}
if($5==5) { print(simtime,$7)>> "energy-node5.tr"; lastenergy[5]=$7; } else
{ print(simtime,lastenergy[5])>> "energy-node5.tr";}
if($5==6) { print(simtime,$7)>> "energy-node6.tr"; lastenergy[6]=$7; } else
{ print(simtime,lastenergy[6])>> "energy-node6.tr";}
if($5==7) { print(simtime,$7)>> "energy-node7.tr"; lastenergy[7]=$7; } else
{ print(simtime,lastenergy[7])>> "energy-node7.tr";}
if($5==8) { print(simtime,$7)>> "energy-node8.tr"; lastenergy[8]=$7; } else
{ print(simtime,lastenergy[8])>> "energy-node8.tr";}
if($5==9) { print(simtime,$7)>> "energy-node9.tr"; lastenergy[9]=$7; } else
{ print(simtime,lastenergy[9])>> "energy-node9.tr";}
if($5==10) { print(simtime,$7)>> "energy-node10.tr"; lastenergy[10]=$7; }
else { print(simtime,lastenergy[10])>> "energy-node10.tr";}
if($5==11) { print(simtime,$7)>> "energy-node11.tr"; lastenergy[11]=$7; }
else { print(simtime,lastenergy[11])>> "energy-node11.tr";}
if($5==12) { print(simtime,$7)>> "energy-node12.tr"; lastenergy[12]=$7; }
else { print(simtime,lastenergy[12])>> "energy-node12.tr";}
if($5==13) { print(simtime,$7)>> "energy-node13.tr"; lastenergy[13]=$7; }
else { print(simtime,lastenergy[13])>> "energy-node13.tr";}
if($5==14) { print(simtime,$7)>> "energy-node14.tr"; lastenergy[14]=$7; }
else { print(simtime,lastenergy[14])>> "energy-node14.tr";}
if($5==15) { print(simtime,$7)>> "energy-node15.tr"; lastenergy[15]=$7; }
else { print(simtime,lastenergy[15])>> "energy-node15.tr";}
if($5==16) { print(simtime,$7)>> "energy-node16.tr"; lastenergy[16]=$7; }
else { print(simtime,lastenergy[16])>> "energy-node16.tr";}
if($5==17) { print(simtime,$7)>> "energy-node17.tr"; lastenergy[17]=$7; }
else { print(simtime,lastenergy[17])>> "energy-node17.tr";}
if($5==18) { print(simtime,$7)>> "energy-node18.tr"; lastenergy[18]=$7; }
else { print(simtime,lastenergy[18])>> "energy-node18.tr";}
if($5==19) { print(simtime,$7)>> "energy-node19.tr"; lastenergy[19]=$7; }
else { print(simtime,lastenergy[19])>> "energy-node19.tr";}
if($5==20) { print(simtime,$7)>> "energy-node20.tr"; lastenergy[20]=$7; }
else { print(simtime,lastenergy[20])>> "energy-node20.tr";}
if($5==21) { print(simtime,$7)>> "energy-node21.tr"; lastenergy[21]=$7; }
else { print(simtime,lastenergy[21])>> "energy-node21.tr";}
if($5==22) { print(simtime,$7)>> "energy-node22.tr"; lastenergy[22]=$7; }
else { print(simtime,lastenergy[22])>> "energy-node22.tr";}
if($5==23) { print(simtime,$7)>> "energy-node23.tr"; lastenergy[23]=$7; }
else { print(simtime,lastenergy[23])>> "energy-node23.tr";}
if($5==24) { print(simtime,$7)>> "energy-node24.tr"; lastenergy[24]=$7; }
else { print(simtime,lastenergy[24])>> "energy-node24.tr";}

  }}
END {# No need to display. Write everything to file.}
```