A Thesis

entitled

A Novel Swarm Intelligence based IWD Algorithm for Routing in MANETs

by

Jyothirmye Vaddhireddy

Submitted to the Graduate Faculty as partial fulfillment of the requirements for

The Master of Science Degree in Electrical Engineering

_____

Dr. Devinder Kaur, Committee Chair

_____

Dr. Henry Ledgard, Committee Member

_____

Dr. Jackson Carvalho, Committee Member

_____

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

December 2011

An Abstract of

A Novel Swarm Intelligence based IWD Algorithm for Routing in MANETs

by

Jyothirmye Vaddhireddy

Submitted to the Graduate School as a partial fulfillment of the requirements for
The Master of Science Degree in Electrical Engineering

The University of Toledo
December 2011

In this thesis a new routing protocol for Mobile Ad-Hoc Networks (MANETs) has been developed and simulated. The protocol is named IWDHocNet. With the explosion of technology, the networks are becoming increasingly diverse and heterogeneous. MANETs do not require a fixed infrastructure whereas simple wireless networks require an infrastructure and access points connected to a backbone. In MANETs all the nodes act as routers and participate in discovery and maintenance of routes. These features of MANET pose extra challenges for routing. IWDHocNet addresses the challenges of MANET. IWDHocNet protocol takes its inspiration from how the swarm of water drops moves through the rivers to find the optimum path.

The protocol was simulated in NS-2 simulator under a variety of network conditions by varying the node mobility and data traffic. The performance of the protocol was compared with two other established routing protocols such as AODV and DSDV. The comparisons were made based on three performance metrics – packet delivery ratio, average end-to-end delay and average routing load.

We have found when the mobility of network is not very fast, the performance of the network is comparable to DSDV and AODV. However, for dynamic network with highly mobile nodes, AODV outperformed IWDHocNet, although the performance was still comparable with DSDV in some situations.

IWDHocNet is proactive in its current form. For future work, it is proposed to adapt the IWDHocNet routing protocol to be reactive like AODV as it may improve the performance.

This thesis is dedicated to my family and my fiancé.

Acknowledgements

I would like to thank in the first place my advisor, Dr. Devinder Kaur, for her constant encouragement and support in completing my thesis. She taught me to be optimistic always. I would also like to thank UT Electrical Engineering and Computer Sciences department for partially funding my Master's degree.

I would like to thank my parents, Prathap Reddy and Padma and my brother, Raghu for their constant love, understanding and encouragement. My parents have been a source of inspiration and motivation for me. I am very grateful to them for their sacrifices and efforts that made this thesis possible. Big thanks to my fiancé, Anil Sadhu. He taught me to face the problems and not to try to run away from them.

Finally, with no less enthusiasm, I would like to thank my friends in UT – Sirisha, Sravani, Priya and Sampath, for making my stay in Toledo wonderful. They are my stress-busters. I want to thank my boss (at my part-time job) Daniel Miller, for being very flexible with my schedule at work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

One of the most important developments in the recent years is the increased use of wireless communication service. With the technological advances in computers and mobile devices, the need for data transfer is soaring and the mobile computing is going through a rapid growth.

Wireless communication between the devices can be done in two different approaches. The first approach is to follow cellular network infrastructure, which uses a central base station to transmit the data from one device to another. This approach has its problems as it can be set up only when there exists such a cellular network infrastructure.

The second approach is to form an ad-hoc network. An ad-hoc network does not have any fixed infrastructure and a central base station to relay the packets. In this network, all nodes participate in the transmission of data from one node to another. The nodes in the network can be static or mobile. If the nodes are mobile, such network is

called a Mobile Ad-hoc Network, MANET. The absence of a fixed infrastructure and the free mobility of the nodes in a MANET pose several types of challenges. One challenge is *routing*. Routing is the method of directing data packets from a source to its destination along a certain path within the network. In this thesis, we focus on the problem of routing in MANETs.

Swarm Intelligence[1] is a nature inspired computational model based on the collective behavior seen in biological systems such as ants, rivers, termites, etc. This technique uses swarm particles to optimize a given problem. There are numerous applications of swarm intelligence; one among them is routing in communication networks. Routing based on swarm intelligence provides a promising alternative to the traditional routing approaches. Since it uses mobile software agents for network management, which have the ability to adapt, cooperate and move intelligently from one location to the other. Many of the Swarm Intelligence algorithms have been previously applied to MANETs. But, application of the Intelligent Water Drops (IWD) algorithm to the routing is a novel concept. The IWD algorithm is a relatively new swarm-based optimization algorithm inspired by the processes that happen between the water drops and the soil as a river navigates its flow to the ocean. The work presented in this thesis explores the adaptation of the IWD algorithm for routing in MANETs.

1.2 Our Contribution, a new Routing Algorithm

In this thesis, an intelligent water drop (IWD) algorithm has been proposed to solve the routing problem in MANETs with an objective of optimizing the route setup time and. The proposed Routing algorithm derives its method of operation from how the

water drops in the river take an optimal route in their journey to the oceans (or ponds, lakes).

The goals of this thesis are as follows:

- Get a general understanding of mobile ad-hoc networks.
- Develop a new routing protocol for the MANETs, based on nature inspired Intelligent Water Drops (IWD) algorithm.
- Implement the routing protocol.
- Analyze the protocol theoretically and verify through simulation.
- Do the performance analysis of the algorithm.

1.3 Thesis Organization

This thesis is divided into 7 chapters. Chapter 2 gives an overview of MANET technology and Chapter 3 gives in insight into routing algorithms. Chapter 4 gives introductory information of IWD algorithm and Chapter 5 describes how IWD algorithm can be adapted for MANETs. The Simulation results and analysis are presented in Chapter 6. Finally in Chapter 7, conclusions are drawn and possible future work is presented.

Chapter 2

Mobile Ad-Hoc Networks

Mobile ad-hoc networks (MANETs) have a profound impact in the world of computer networks. Characterized by anytime/anywhere establishment of a wireless network, the MANET infrastructure enables location-independent services.

2.1 What are MANETs

Mobile Ad-hoc Networks, MANETs, are a class of wireless networks that are an active area of research. A MANET is a self-configuring network of mobile devices that communicate with each other over wireless links. The network is said to be self-configuring as it does not require any network infrastructure for its deployment, unlike static networks. In other words, a MANET is a collection of mobile nodes with dynamic network infrastructure forming a temporary network. The primary objective of designing a MANET is to achieve network “anywhere and anytime” [2]. These networks do not have any central server or base station, and mobile devices are designed to have all required network intelligence inside them. A cellular network has a much larger range

than mobile ad-hoc networks. However, MANETs have the advantage of being quickly deployable.

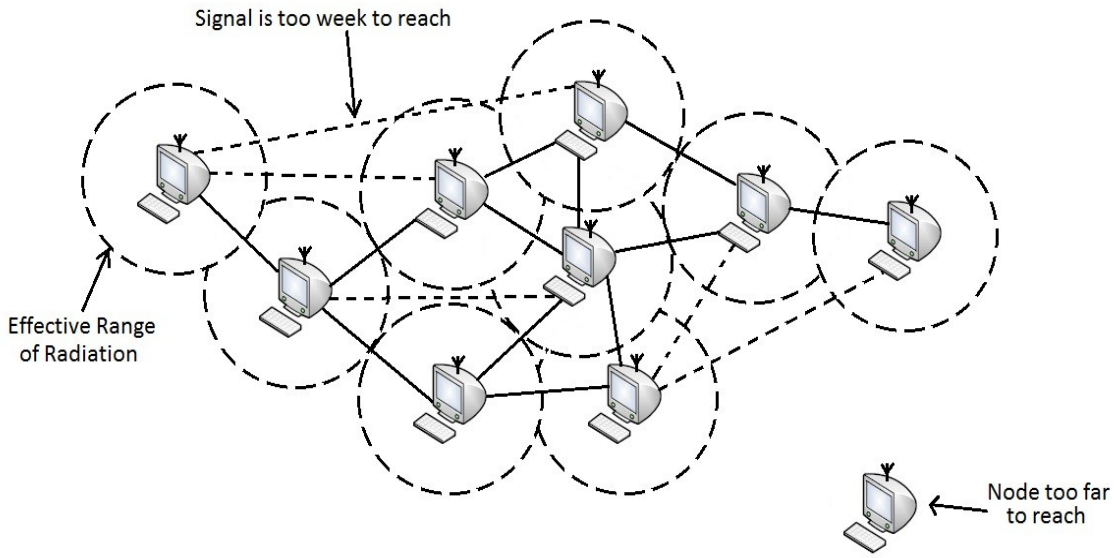


Figure 2-1: Overview of an ad-hoc network

Figure 2.1 gives an overview of an ad-hoc network, where wireless mobile nodes have formed a network, with one node too far to reach. In the figure, the circle around each mobile user represents the effective range of radiation of the particular mobile user; dashed lines represented that there could be a possible communication link that could not be established because the signal is too weak to reach the other node; solid lines are the active communication links that can be used. Communication between two non-reachable nodes uses nodes which are in the effective range as routers to transmit the data.

In a MANET, nodes (mobile devices) are free to move in any direction, thus changing the links frequently. Also, any node can join or leave the network at any time. Each node acts as a routing node and takes part in discovery and maintenance of routes to

other nodes in the network. Data packets to the neighboring nodes are usually sent by a direct link and remote nodes are sent by a self-discovered, multi-hop fashion, as defined by the ad-hoc routing protocol.

2.2 Application of MANETs:

An ad-hoc network finds its applications in several fields ranging from military to commercial purposes.

- Search and Rescue Operations. In an emergency public disaster, such as earthquake, MANETS can be set up at any location, where there is a lack of installed infrastructure or when the equipment has been destroyed,
- Military. MANETs were originally developed to help in military operations, such as battle zones, where setting up an infrastructure is almost impossible.
- Law enforcement and security operations.
- Home networks.
- Conferencing.
- Sensor Networks.

2.3 Characteristics and Challenges of MANETs:

Ad-hoc networks must possess several unique features. One of such is automatic discovery of available services. As explained, in a MANET, every node has the freedom to join or leave the network at any time, because of this the nodes should automatically configure to the new nodes that are joined into the network. Also, as the network does not

provide centralized administration, it must be able to prevent the network from collapsing when a node leaves the network.

The absence of a fixed infrastructure in a MANET poses several types of challenges and one among them is *routing* described in chapter 3. The most important challenges in designing routing algorithms for MANETs are:

i. ***Mobility***: Rapid and unpredictable mobility of the nodes results in continuously evolving new topologies, which requires the routing algorithms to discover or update the routes by making quick decisions. Thus, the routing algorithms should have a small control overhead.

ii. ***Limited battery capacity***: Ad-hoc network is usually made by battery-powered devices with low capacity and thus the nodes have only limited resources. This requires the packets to be distributed on multiple paths, which would result in the depletion of the batteries of different nodes at an equal rate and consequently increasing the lifetime of the network. Low battery power also leads some of the nodes to die and consequently leaving the network.

Therefore an important challenge for MANETs is to design a routing algorithm that is not only energy efficient but also a one that establishes reliable routes with less control overhead. Routing protocol, in MANET, should also have the ability to automatically recover from any problem in a finite amount of time without human intervention.

Chapter 3

Routing Protocols

This chapter walks through the concept of Routing and gives a brief note on some of the traditional Ad-Hoc routing protocols implemented so far.

3.1 Routing In Ad-Hoc Networks

Routing is the method of directing data flow from sources to destinations along a certain path within the network. It is at the core of all network activities. The goal of a routing algorithm is to calculate the best path between any two nodes in the network. A good routing algorithm always strives to maximize the performance of a network.

The challenges with implementation of MANETs, as described in section 2.3, are the lack of a backbone infrastructure, mobility of the nodes and limited node resources. These challenges make routing in mobile ad-hoc networks a challenging task. Ad-hoc networks should rely on special routing protocols that can adapt to the frequent topological changes. Identifying mobile nodes and discovering a correct routing of packets to and from each node while moving is certainly challenging.

3.2 Classification of Routing Protocols

Ad-hoc routing protocols can be classified into two broad categories:

1. *Centralized* versus *Distributed*. In centralized routing protocols, the routing decision is made at the central node whereas in distributed routing protocols, the routing decision is made by all the network nodes [2]. Centralized algorithms cannot make any decisions without manual intervention and thus not very suitable for efficiently designed ad-hoc networks. Most of the routing protocols are designed to be distributed so as to increase the reliability of the network. By making a routing protocol distributed, the nodes can enter or leave the network anytime without disturbing the entire network.

2. *Static* versus *Dynamic*. Static routing, as the name suggests, is a fixed path routing and it assumes the network to be time-invariant, which is rigid and does not handle traffic conditions and link failures. In ad-hoc networks, we need dynamic/adaptive routing, in which the routes may change to adapt the changes in the environment.

Independent of whether the routing protocol is a centralized/ distributed or static/dynamic, it can again be categorized as either *table driven* (*proactive*) or *source initiated* (*reactive*).

3.2.1 Table-Driven Routing Protocols

Table-driven or *proactive* routing protocols finds routes to all possible destinations ahead of time. In this protocol, the nodes maintain a table with list of destinations and their routes which are updated by periodical distribution of the routing tables. A table-driven protocol requires one or more tables in every node to store updated routing information about other nodes. The routes to the destinations are recorded in the nodes' routing tables and are updated either periodically or in response to change in network topology so as to maintain consistent and up-to-date routing information. The advantage of these protocols is, they are faster in decision making and does not need a route-discovery procedures to find a route. The drawback is that they need more time to converge to a steady state, causing problems in continually changing networks and/or dense networks. Also, maintaining routes can lead to a substantial messaging overhead consuming large portion of network bandwidth and power for the non-productive control packets.

Examples of pro-active algorithms are: Destination Sequenced Distance Vector routing (DSDV), Wireless routing protocol (WRP), Fisheye State Routing protocol (FSR), Optimized Link State Routing protocol (OLSR) [2].

3.2.2 Source-Initiated Routing Protocols

Source-initiated or *reactive* or *on-demand* routing protocols are on-demand procedures and create routes only when requested by the source nodes [2]. When the source node needs to send a data packet to the destination node, it requests a route by

initiating a *route-discovery process* in the network, which is completed once a route is discovered. After the route has been discovered it is maintained by a *route-maintenance procedure,* until the route is no longer needed or the destination node is not reachable to the source. This means that the network reacts only when needed and does not broadcast information periodically. However, a node may experience long waiting times before it can transmit the data packets as it would not know the next hop to forward the packet due to dynamic network topology.

The main advantage of these protocols is that overhead messaging is less. The main drawbacks of these protocols are: High latency time in route finding and; Excessive flooding leading to network clogging.

Examples of Reactive routing protocols are: Dynamic Source Routing (DSR), Ad-hoc On-Demand Distance Vector routing (AODV) and Temporally Ordered Routing Algorithm (TORA) [2].

3.3 Important routing algorithms for MANETs

The following section describes some of the most representative routing algorithms for MANETs. These routing algorithms will be used as a benchmark to compare the proposed *IWDHocNet* algorithm.

3.3.1 Destination-Sequenced Distance-Vector Routing

The Destination-Sequenced Distance-Vector (DSDV) [3] routing algorithm was the first routing algorithm for Ad-Hoc wireless mobile networks. This is a table-driven

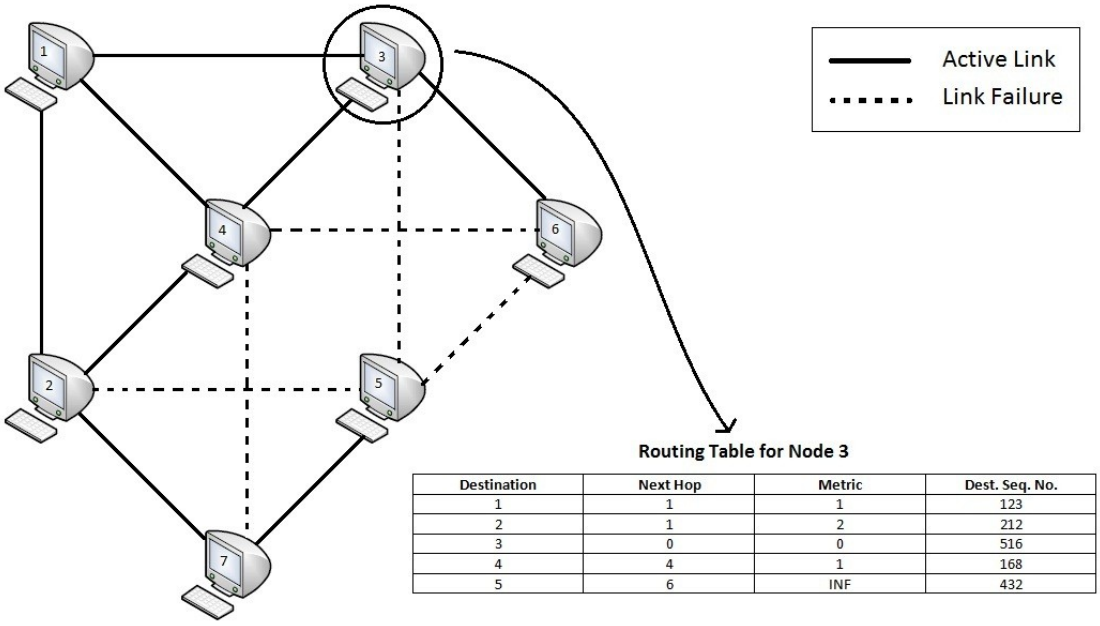
routing algorithm based on the idea of the classical Bellman-Ford routing algorithm [2, 3] with certain improvements [4]. This is a hop-by-hop distance-vector routing protocol and requires each node to inform its neighbors of topology changes periodically.

In DSDV algorithm, each node maintains a routing table that stores the next-hop and number of hops for all the available destinations and also a sequence number assigned by destination node. The sequence number which is a tag to each route distinguishes the fresh routes from the stale routes and thereby establishing loop-free routes. The routing table updates are periodically transmitted by the nodes to their immediate neighbors to maintain the consistency. The nodes also transmit its routing table when it seems a significant change occurred in its table, like a link failure. Thus, it can be called both event-driven and time-driven. But, sending this huge updates can increase large amount of network traffic; to reduce the amount of information in these packets, the updates are sent in two ways – a *full dump* or an incremental update. A full dump carries all available routing information to its neighbors and could need many packets while sending. On the other hand, an incremental update sends out only the entries with a metric change since the last update and it must fit in one packet. Any space left in the incremental update packet is filled with those entries, for which the sequence number has changed. The DSDV nodes maintain an additional table for advertising these incremental packets.

The routing information sent contains a new sequence number, the address of the destination and number of hops to the destination node. On receipt of an update packet, the route labeled with the highest sequence number is used and discards the old sequence number. In the event the sequence numbers are identical, the one with least cost (better

metric) is chosen to optimize the path. This process continues until all the nodes' routing tables are updated. In case of duplicate updated packets, the node considers only the least-cost metric one and discards the rest.

Figure 3-1 shows a routing table for node 3, whose neighbors are 1,4 and 6. The dashed lines indicate broken links or no links, i.e. no communication between the nodes. Therefore, node 3 has no information about node 5. It can also be seen that a cost of ∞ metric is assigned to node 5 to ensure no communication to node 5, as the link is broken.



Destination	Next Hop	Metric	Dest. Seq. No.
1	1	1	123
2	1	2	212
3	0	0	516
4	4	1	168
5	6	INF	432

Figure 3-1: A DSDV Routing Table

Because DSDV requires periodic broadcast of its routing tables, it needs a small amount of bandwidth even when the network is idle and also needs some time to converge before a route can be formed or used. This convergence time is very less in a static wired network, where topology changes are not frequent. However, in an ad-hoc

network, the changes in topology are very random, thus needing frequent broadcasts of the routing tables and causing a slow convergence of routes as packets are dropped and nodes move about.

3.3.2 Ad-Hoc on Demand Distance Vector (AODV)

Ad-hoc On-demand Distance Vector Routing (AODV), as the name indicates is also a distance-vector routing protocol. This is an improvement over the DSDV algorithm described in the previous section. AODV is different from DSDV for it establishes a route request only when it is needed as opposed to creating complete list of routes. Thus, it is named *On-Demand*. According to the authors of the AODV [2, 5], this protocol adapts quickly to dynamic link conditions and offers low processing, memory overhead and network utilization.

In AODV, the network remains idle till the necessity for a connection is created. When the need arises, the source node checks if it has a valid route to the destination and when it does not find one, it initiates a path discovery process for the destination node. It broadcasts a Route Request (RREQ) packet with the destination address to its immediate neighbors. The other node (neighbor) forwards this request further till it reaches a node that has recent route information to the destination or the destination node itself. This process can be seen in Figure 3.2. This node upon finding a route, then sends a Route Reply, RREP, message packet backwards to the source node, along the same path which it received the RREQ packet as seen in Figure 3-3. (For this reason, AODV always uses only symmetrical links so that it can hear back the route). The source node then selects

the route that has least number of hops, comparing the routes received from the other nodes as well.

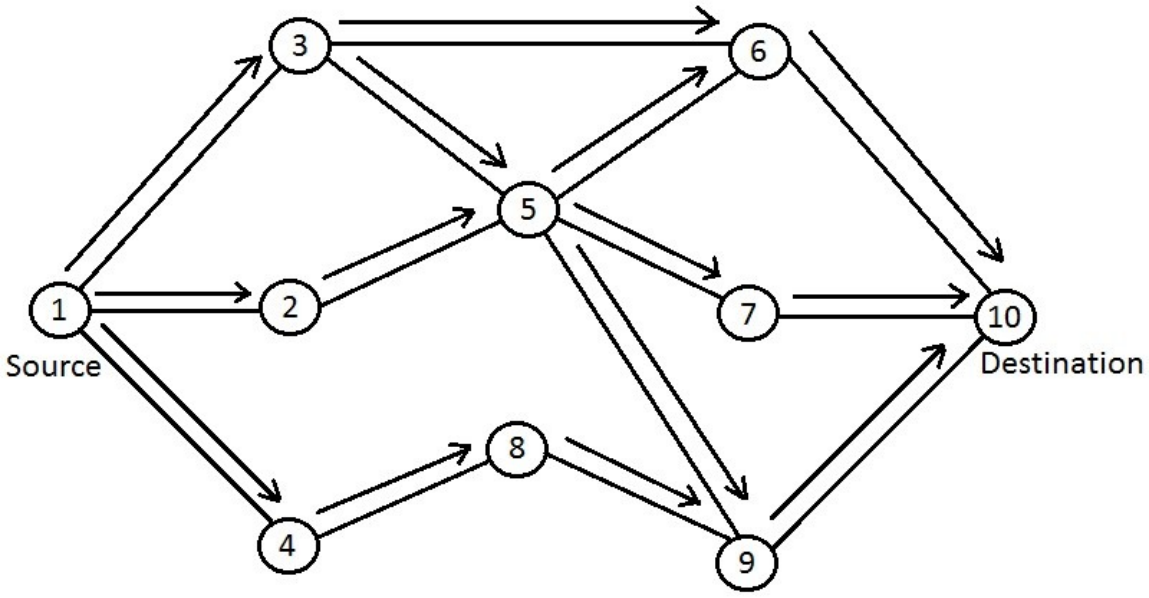


Figure 3-2: Broadcast of Route Request (RREQ) packet

The AODV protocol also takes many factors into consideration before deciding on the route to avoid loops. For example, the nodes discard a route request packet that it has already seen and each request for a route has a sequence number. The nodes use this sequence number to avoid duplicate route requests. Another feature is that the route requests are set with a *time to live* to avoid retransmission of the route requests, between the nodes.

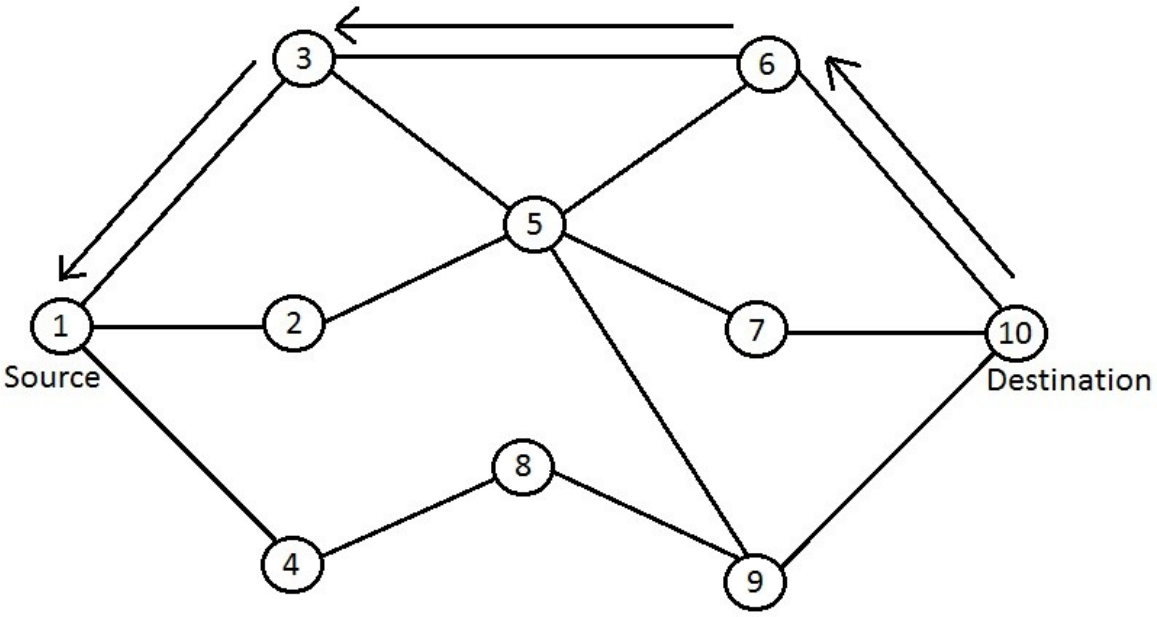


Figure 3-3: Path taken by the Route Reply (RREP) Packet

The disadvantage with AODV is that it only supports one route for each destination and thus when a route breaks the node needs to reinitiate a route request query. The number of route requests increases with the mobility in the topology. Another drawback with AODV is it does not support unidirectional links. When a node receives a RREQ, it will setup a reverse route to the source by using the node that forwarded the RREQ as the next hop. This means that the route reply is uni-casted back the same way the route request used and thus needs bidirectional links.

3.4 Conclusion and Routing based on Swarm Intelligence

One of the most important challenges faced by the MANETs is to continually adapt itself to the changes in the network. The primary objective behind the development of Artificial Intelligence (AI) is to make to system adaptable to learn and build the

solution on its own. An AI technique, based on swarm intelligence, can offer a viable alternative to the traditional link state or distance vector routing protocols.

Routing algorithms based on Swarm Intelligence techniques could provide some better performance results than the traditional current routing algorithms as the latter do not seem to be adequate to tackle the increasing complexity in MANETs. Swarm Intelligence is different from the other algorithms as it utilizes mobile software agents for the network management. These agents have the intelligence to act on their own (being autonomous) and are capable to move from one node to another node to establish a route. The aim of expanding the idea of implementing swarm intelligence to routing is to adapt to the dynamic traffic and changes in topology quickly creating no network overloads.

Many researchers have already applied different swarm intelligence techniques to the routing problem and showed better results. For an example, the authors in [6] applied ACO (*Ant Colony Optimization*) algorithm, an algorithm inspired from ant colony behavior, for routing in MANETs. According to the authors in [6], the algorithm outperformed the AODV algorithm. It showed "much better effectiveness than AODV, in terms of average delay, delivery ratio, and jitter" [6] and the difference increased with density and size of the network. Though AODV exhibited better efficiency in terms of routing overhead, the variance was rather small. Moreover, the algorithm seemed to be quite beneficial than AODV for larger networks. The results given by ACO algorithm interested many researchers and made many improvements later, [7] is one of them. Another example is the application of *BeeAdHoc* algorithm [8] which is inspired from the foraging principles of honey bees. This algorithm also has showed better results in terms of the energy consumption of the nodes than the other state-of-the-art routing algorithms.

The simplicity of the algorithm resulted in "substantially smaller number of control packets sent, as a result, the algorithm is energy efficient" [8].

The results from the other swarm intelligence routing algorithms inspired the work in this thesis to explore more swarm intelligence techniques for their application to Routing and came up with a relatively new algorithm called IWD (Intelligent Water Drops), which is based on the flow of the water drops in rivers. The algorithm is explained in detail in the following sections.

Chapter 4

Intelligent Water Drops

4.1 Introduction

The use of computers to model nature, and study of nature to improve the usage of computers has become a challenging and interesting research area. Many algorithms came out from this research and Intelligent Water Drops (IWD) is one such algorithm developed in the recent times. It is a nature-inspired optimization algorithm inspired from the natural water drops which change their environment to find the near optimal or optimal path to their destination, ocean or pond. The processes that happen between the water drops of a river and the soil of the river bed formed the basis for this algorithm. IWD algorithm falls in the category of Swarm-Based Optimization algorithms. The IWD algorithm was first introduced by Dr. Shah-Hosseini in the year 2007 [9]. The algorithm so far was successfully implemented to the Traveling Salesman problem, n-Queen puzzle, Multidimensional Knapsack problem (MKP) [9], Smooth trajectory planning [10], Robot Path planning [11], Vehicle routing problem [12] and Economic Load Dispatch problem [13].

4.2 Natural Water Drops

Water drops are found almost everywhere in nature, flowing mostly in rivers. The rivers can be considered as huge moving swarms of water drops. The path of the river, in fact, is created by this swarm of water drops. As the water drops move, they try to change the environment along the path they are flowing. Interestingly, the environment also has some impact on the flow of the water drops; i.e., the environment and water drops both are affected by each other. The environment here is the soil on the bed of the river (flow of the water drops). For example, the faster moving water drops changes the environment (soils) more by picking up more soil than the slower moving ones and the environment having hard soils resists the flow of the water more than the soft soils. "In fact, a natural river is the result of a competition between water drops in a swarm and the environment that resists the movement of water drops" [14].

Figure 4-1 shows the flow of river Amazon. As we can see in the figure, the path that the river follows is full of twists and turns. It can be said that the paths that the water drops take are not a smooth ride always or it can also be said that they are never smooth, but still the water drops manage to get to their destination. According to basic physics, the earth's gravitational force pulls everything straight toward the center of the earth.

Figure 4-1: Amazon River in South America, which is considered to be among the longest rivers in the world, connecting smaller rivers to Atlantic Ocean. Figure adapted from [15].

Thus, with no obstacles on the way the water drops with the help of the gravitational pull should reach their destination following the ideal path that is the shortest path (straight line connecting source and destination) from source to the destination, which is ideally the earth's center. The gravitational pull can also increase the acceleration of the water drops as they near the center of earth. Nevertheless, in reality, due to different kinds of obstacles and constraints, the pathways the water drops follow cannot always be straight and have to go through a lot of twists and turns, and also the destination is not the earth's center but a pond, lake, sea or ocean. During this process, to reach their destination, the water drops always try to change the real path to make it a better path in order to approach the ideal path. With time, the path the river

passes by is changed by the water drops. Interestingly, the constructed path of the river always seems to be optimum in terms of distance from the destination and the constraints of the environment.

One feature of a water drop flowing in a river is its velocity [9]. It is assumed that each water drop, while flowing, can carry some amount of soil. Therefore, the water drop is able to transfer a certain amount of soil from one place to another place it flows through. The carried soil is usually transferred from fast parts of the path to the slow parts. The water flows over the soil and in course of time due to the velocity of the water drops the soil gets transferred from a high velocity beds and gets deposited in a low velocity beds. The soil transfer makes the high velocity beds deeper by removing soil from there and these beds can also accumulate more water because of their increasing depth and thereby attract more water to pass through. Thus, the environment affects the movement of the water drops and they themselves change the environment in which they flow. It should be noted that there are other mechanisms as well, which are involved in the river system but are not considered for this study.

Assume an imaginary water drop flowing from a point of river to the next point in the front, as shown in Figure 4-2. As the water flows from point *a* to point *b*, three obvious changes happen during the transition:

- Velocity of the water drop increases.
- Soil contained by the water drop increases.
- Soil of the river's bed between the two points decreases.

Each water drop can carry a certain amount of soil with it, therefore the amount of the soil of the water drop increases and also the soil of the river bed is decreased because of it. Here the IWDs soil is increased by removing some soil from the river bed.

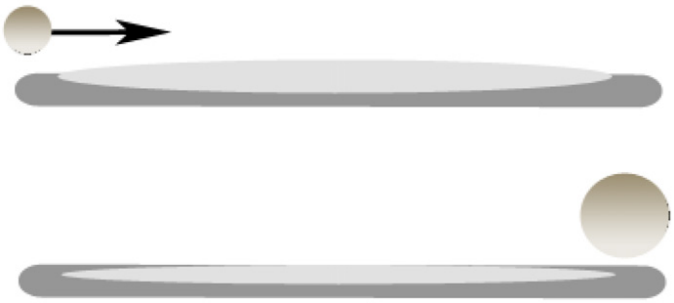


Figure 4-2: Depiction of IWD flowing from left to right, while removing soil from the river bed and adding it to its own soil and also increasing its velocity. Figure adapted from [10].

If we consider two water drops with same amount of soil and different velocities, the water drop with higher velocity gathers more soil in its journey than the other one. Thus, the velocity of an IWD flowing over a path determines the amount of soil that is removed from the path. In contrast, the velocity of an IWD depends on the amount of the soil on the path. A path with little amount of soil increases the velocity of the IWD flowing through it than the path with large amount of soil. Combining the above said two statements, IWD chooses a path with little soil, because this (less soil) is the path of least resistance which will allow IWD to gain speed and thus gather more soil. The path with large soil is not preferred as it lets the IWD gather less soil and gain less speed and thereby resisting the flow of IWD through it. It is obvious that a water drop chooses an easier path to a harder path and thus an IWD prefers the paths with low soils on its beds to the paths with higher soils.

Based on all the properties explained so far in this section, the following three properties can be assumed for a water drop:

➔ A high speed water drop gathers more soil than a slower water drop.

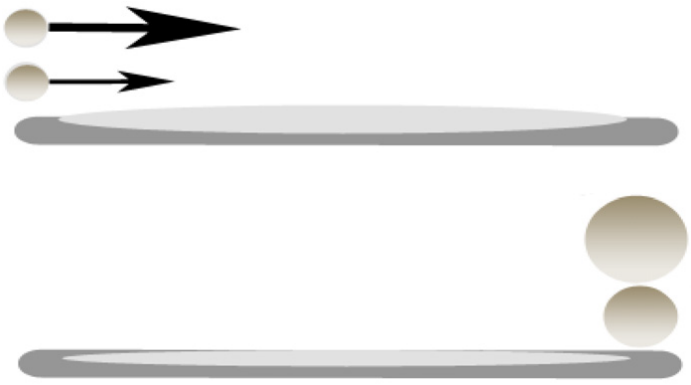


Figure 4-3: The faster IWD gathers more soil that the slower IWD while both flowing from the left side of the river bed to the right side. (The size of the IWD shows its carrying soil). Figure adapted from [10]

Therefore, the water drop with bigger speed removes more soil from the river's bed than another water drop with smaller speed. The soil removal is thus related to the velocities of water drops.

➔ The velocity of a water drop increases more on a path with low soil than a path with high soil.

The velocity of the water drop is changed such that on a path with little amount of soil, the velocity of the water drop is increased more than a path with a considerable amount of soil. Therefore, a path with little soil lets the flowing water drop gather more soil and gain more speed whereas the path with large soil resists more against the flowing water drop such that it lets the flowing water drop gather less soil and gain less speed.

Therefore, a path with little soil lets the flowing water drop gather more soil and gain more speed whereas the path with large soil resists more against the flowing water drop such that it lets the flowing water drop gather less soil and gain less speed.

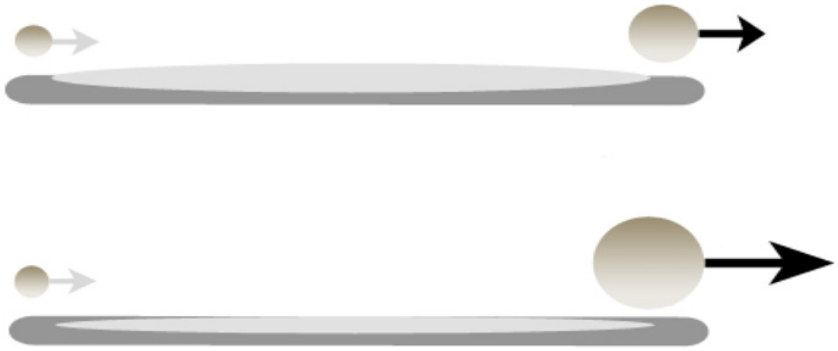


Figure 4-4: Two identical IWDs flowing in two different rivers. The IWD that flows in the river with less soil gathers more soil and gets more increase in speed. Figure adapted from [10]

Another property of a natural water drop is that when it faces several paths in the front, it often chooses the easier path. Therefore, the following statement may be expressed:

➔ A water drop prefers a path with less soil than a path with more soil.

The water drop prefers an easier path to a harder path when it has to choose between several branches that exist in the path from the source to destination. The easiness or hardness of a path is denoted by the amount of soil on that path. A path with more soil is considered a hard path whereas a path with less soil is considered an easy path.

4.3 Intelligent Water Drops

Rivers take lots of twists and turns along their way as they move forward to the oceans. The questions which wonder us are why these twists have been created and is there any logic behind them? And if there is intelligence behind it can we use this mechanism and design and develop an intelligent algorithm based on them? The IWD algorithm is a step to achieve this.

We can develop an artificial water drop which possesses the same functionality as a natural water drop. This artificial water drop can be termed as Intelligent Water Drop or IWD for short. The IWDs in the IWD algorithm are created with two main properties:

- The amount of soil it carries, Soil (IWD).
- The velocity that it possesses, Velocity (IWD).

Both of these properties change as the IWD flows in its environment. (The environment here is the river bed that will be changing.) In engineering perspective, an environment is what represents the problem at hand. Each IWD flows from source towards the destination in this environment. There can be numerous paths from the source to the desired destination. The location of the destination is sometimes known and sometimes not. If the position of the destination is unknown, the goal will be to find the optimum destination in terms of quality. If the destination's position is known, finding an optimal (often the shortest) path to the destination becomes the goal.

Consider an IWD moving in discrete finite-length steps in its environment, from its current location *i* to its next location *j*, the IWD velocity, velocity (IWD), is increased by an amount Δvelocity (IWD), which is nonlinearly proportional to the inverse of the soil between the two locations *i* an *j*, soil(*i*, *j*), as shown in equation 4.1:

$$\Delta velocity(IWD) \propto^{NL} \frac{1}{soil(i,j)} \tag{4.1}$$

Here, nonlinearly proportionality is denoted by $\propto^{NL}$. One possible formula, according to [14] is given in eq. (4.2) in which the velocity of the IWD denoted by $vel^{IWD}(t)$ is updated by the amount of soil, $soil(i,j)$, between the two locations *i* and *j*:

$$\Delta vel^{IWD}(t) = \frac{a_v}{b_v + c_v.soil^{2\alpha}(i,j)} \tag{4.2}$$

Here, a_v, b_v, c_v and α are constant velocity updating parameters that are set for a given problem. The updated velocity of the IWD, $vel^{IWD}(t+1)$ after reaching node *j* will be equivalent to $vel^{IWD}(t) + \Delta vel^{IWD}(t)$.

The amount of soil carried by the IWD, soil(IWD), is increased by removing some soil from the path joining the two locations *i* an *j*. The amount of the soil added to IWD, Δsoil(IWD), and the amount of the soil removed from the path Δsoil(*i,j*), is inversely and nonlinearly proportional to the time needed for the IWD to reach from *i* to *j*, time(*i,j*;*IWD*).

$$\Delta soil(IWD) = \Delta soil(i,j) \propto^{NL} \frac{1}{time(i,j;IWD)} \tag{4.3}$$

The author in [14], suggests an example for the above formula, which is given in eq. (4.4). *time(i, j;vel^{IWD})* is the time taken for the IWD with velocity *vel^{IWD}* to move from point *i* to *j*. $\Delta soil(i,j)$ is the soil added to the IWD.

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s.time^{2\theta}(i,j;vel^{IWD})} \tag{4.4}$$

Again the parameters in this equation, a_s, b_s, c_s and θ are user-selected parameters just like the parameters in eq. (4.2).

This motion can be considered as a linear motion and thus the duration of time for the IWD can be calculated by simple laws of physics for linear motion. Thus, the time taken is inversely proportional to the velocity of the IWD, *velocity(IWD)*, and also to the distance between the two positions, *d(i,j)*.

$$time(i,j;IWD) \propto \frac{1}{velocity(IWD)} \tag{4.5}$$

One example of the above formula is given in eq. (4.6) with which the time taken for the IWD to travel from *i* to *j* with *vel^{IWD}* can be calculated.

$$time(i,j;vel^{IWD}) = \frac{HUD(i,j)}{vel^{IWD}} \tag{4.6}$$

For a given problem, there should be a local heuristic function HUD(.,.) defined which would measure the undesirability of an IWD to move from one node to another node.

The amount of the soil that has been removed from the path, the IWD flows, is dependent on the velocity of the moving IWD. Velocity is inversely proportional to the time taken to travel; Thus, the amount of the soil removed is inversely proportional to the time the IWD needs to pass the path between the nodes. This justifies what is discussed earlier that a fast IWD picks up more soil than a slower IWD does. Faster rivers can make the river beds deeper as they remove more soil from their beds unlike slower rivers which does not have enough strength to remove the soil from the bed.

According to the properties observed in section 4.2, the soil removed from the visited path between positions *i* and *j* is proportional to the amount of soil removed by the IWD flowing on the path. Eq. (4.7) is framed based on this.

$$soil(i,j) \propto \Delta soil(i,j) \tag{4.7}$$

Ideally, $\Delta soil(i,j) = \Delta soil(IWD)$ should hold true but because of the other disturbances that occur in nature, the equation is limited to the one in (4.8). *soil(i,j)* is the updated soil on the path connecting locations i and j.

$$soil(i,j) = \rho_0 . soil(i,j) - \rho_n . \Delta soil(i,j) \tag{4.8}$$

In the above equation, ρ_0 and ρ_n are positive numbers between zero and one. In the original algorithm for Traveling Salesman problem [9], $\rho_0 = 1 - \rho_n$ is considered.

The soil removed from the path, $\Delta soil(i,j)$ is added to the soil contained by the water drop, $soil^{IWD}$ as shown in eq. (4.6).

$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j) \tag{4.9}$$

Another mechanism that was seen in the behavior of an IWD is that it prefers the paths with low soils on its beds to the paths with higher soils on its beds. This preference is made by calculating and assigning a probability to each path from the current node to other possible nodes. To implement this, a uniform random distribution is used among the soils of the available paths such that the probability of the IWD to move from location *i* to *j* denoted by *prob(i,j;IWD)* is inversely proportional to the amount of soils on the available paths.

$$prob(i,j;IWD) \propto soil(i,j) \tag{4.10}$$

The path with lower soils has a better chance that it is selected by the IWD to move through it than the paths with higher soils; the lower the soil, the better the probability that the IWD chooses. One such formula based on Eq. (4.7) has been used in which the probability of choosing location *j* is given by:

$$prob(i,j;IWD) = \frac{f(soil(i,j))}{\sum_{k \notin v_c(IWD)} f(soil(i,k))} \tag{4.11}$$

Where, $f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))}$. ε_s is a constant parameter, which is kept small and positive to prevent a possible division by zero in the function *f(.)*. The set $v_c(IWD)$denotes the nodes the IWD should not visit (eg. the visited nodes in the Traveling Salesman problem) to keep satisfied the constraints of the problem.

The usage of function *g(soil(i,j))* is to shift the *soil(i,j)* of the path connecting nodes *i* and *j* toward the positive values to avoid possible cancellations of the positive and negative soil amounts while calculating. *g(soil(i,j))* is computed by the Eq. (4.9).

$$g(soil(i,j)) = \begin{cases} soil(i,j) & if \ \min\left(soil(i,.)\right) \geq 0 \\ soil(i,j) - \min(soil(i,.)) & else \end{cases} \quad (4.12)$$

The function min(.) returns the minimum value of its arguments.

The IWDs work in an organized way to find the optimal solution to a given problem. “The problem is encoded in the environment of the IWDs, and the solution is represented by the path that the IWDs have converged to” [14].

Chapter 5

Modified IWD Algorithm for Routing:

IWDHocNet

In this section we discuss the adaptation of the IWD algorithm meta-heuristic for Mobile ad-hoc networks and describe the *IWDHocNet* (Intelligent Water Drops based optimization algorithm for mobile ad-hoc NETworks). IWDHocNet is inspired by how the water drops in a river find their way to the oceans/lakes.

5.1 Algorithm Description

IWDHocNet is a proactive type of routing algorithm, in the sense that it updates the routing information by sending the control packets at regular intervals as long as the communication session continues. This algorithm can also be categorized as a Table Driven Routing Algorithm, as each node maintains a routing table. The Routing tables are used to store the routing information, with which the control packets and data packets

are forwarded in a stochastic way. This section describes the general functioning of the algorithm in words. Section 5.2 will further detail the algorithm.

In IWDHocNet, the routing information is structured in the routing tables. (The routing information is the next hop node id and the amount of soil on the link.) Each network node maintains two tables: routing table and neighbor table.

- The routing table, a two-dimensional matrix, is organized on a per-destination basis. Each entry in the table stores the routing information. The records are 3-way tuples, consisting of destination node id d, next hop n and the amount of soil on the link connecting the node and the hop, S_{nd}, to travel towards d. The soil amount, S_{nd}, expresses the relative goodness (desirability) of relaying a packet over node n in order to reach destination d. The lower the soil, S_{nd}, the higher the probability to choose n. The routing table plays a vital role, as the control packets refer to this table before forwarding any data or control packets to it.
- The neighbor table is used to keep track of the neighbors, to which the nodes has an active wireless link. This list is updated by sending and receiving periodic acknowledgement/HELLO packets, which is explained later.

The algorithm, like most of the other routing algorithms, comprises of three phases: *Route Discovery*, *Route Maintenance* and *Link Failure Handling*, which are explained below.

5.1.1 Route Discovery Phase

During the Route Discovery Phase, new routes to the destination are created. To discover the routes this algorithm uses two kinds of Agents: Forward IWD (FIWD) and Reverse IWD (RIWD). FIWDs are used to explore a route to the destination and RIWDs are used to inform the source node about the established route to the destination and update the soil on the links. The mobile agents FIWDs and RIWDs play the role of control packets in this algorithm. These packets are very small and thus they do not create much network congestion as the data packets do.

As the communication session starts, at regular intervals, each network node of the session launches mobile agents, FIWD packet out over the network towards a randomly selected destination node. The FIWDs are launched in concurrent with the data packets. Each agent moves in a step-by-step fashion towards the destination. Each node that receives the FIWD packet checks if it is the packet's destination, if not it forwards the packet to the next node towards its destination. The FIWDs store the full array of nodes they have visited, and the trip times to the nodes, on their journey to the destination. At each intermediate node, the packet chooses the next node based on a probability calculation; the probability based on the amount of soil on the link and the cost associated with the link. The cost of the link is the route cost metric which can be: number of hops, end-to-end delay, a combination of hop count and end-to-end delay, signal-to-interference-and-noise ratio, etc. [7, 16]. The End-to-End delay is considered as the cost of the link in this algorithm. End-to-End delay is the expected delay experienced

by a data packet in reaching a node. The probability calculating equation is given in the next section, 5.2.

Once the FIWD reaches its destination, it is converted into RIWD (Reverse IWD). The RIWD packet, as the name says travels in reverse direction to go back to the source node retracing the same path the FIWD (forward IWD) followed. As the RIWD travels, at each intermediate node, it establishes a track to the destination node. It updates the local routing tables of the intermediate nodes and also the source based on the goodness of the path. The method of updating the routing tables is explained in detail in the section, 5.2. With this the route discovery phase completes.

5.1.2 Route Maintenance Phase

The second phase, Route Maintenance phase improves the routes during the communication session. Once a route has been constructed, data packets and other FIWD control packets start using the route to reach the destination. The algorithm then starts the phase of maintaining the routes to extend and improve the established routes (routing information) during the communication. No special packets are needed for the route maintenance as the *IWDHocNet* maintains its routes by using the IWD control packets. The soils adjusted in the initial route discovery phase are further adjusted by the other control packets as they explore the network, throughout the communication session. Thus within a small converging time the algorithm can suggest the best (minimal cost) routes to the data packets and can deliver more data packets successfully.

5.1.3 Link Failure Handling

This phase of the routing algorithm handles any route failures caused by the link failures. Link failures are usually caused by node mobility, which is a very salient feature of a MANET. A node usually, does not stay idle in a MANET. Link failures can be detected through many ways: a missing acknowledgement from the transferred data/control packets, with the help of periodic HELLO messages [17], etc.

This algorithm, IWDHocNet borrows neighbor discovery protocol from AODV algorithm [5] and Zone Routing Protocol (ZRP) [18, 19]. It uses HELLO messages to identify the link failures. HELLO messages are very short messages that are periodically broadcasted from a node in the network. Reception of a HELLO message from a certain node indicates the presence of an active link to that node. The nodes constantly listen to the HELLO messages they receive and on receiving one, the node looks for the sender in its active neighbor list; if it does not find, it adds the sender of the message to the list and also updates the routing table by adding an entry for the *Next Hop* for each destination listed (the soil is given a default value). If the sender is already present in its list, it updates the time it last heard from that node with the current time. Each node periodically also checks its own neighbor list if they are still available. This is done with the help of timeout. The neighbors are given an *expire* time. After the timer times-out, and if the node still did not hear anything from that node, it considers it to be inactive and the neighbor is deleted from its list. After realizing that a neighbor is not active anymore, it deletes the corresponding entries of the neighbor in the routing table too. It should be noted that the IWDHocNet algorithm does not support repairing of the routes, in case of

link failures. The nodes only delete the routing table entries for the failed node to prevent data packet drops.

5.2 Detailed descriptions

This section gives a detailed description of the routing algorithm. The data structures maintained by the nodes in IWDHocNet is described first followed with a step by step description of the route discovery, maintenance and link failure handling.

5.2.1 Data Structures

Each network node maintains two data structures: Routing table (for routing information), Neighbor table (to maintain the list of active neighbors).

Routing Table

Each node in IWDHocNet maintains a routing table $\mathcal{T}_k$ - a two-dimensional matrix which is organized on a per-destination basis. The table holds entries for each possible destination *d* and for each neighbor node *n*. Each entry $\mathcal{T}_{kn}^{d}$ in the table provides the information to route from node *i* to destination node *d* via node *n*. The entry $\mathcal{T}_{kn}^{d}$ stores the amount of soil S_{nd} on the link connecting to the node *j* (hop/neighbor) to reach destination *d*. The soil amount S_{nd} expresses an estimate of the relative goodness (desirability) of relaying a packet over the node *n* in order to reach destination *d*. The lower the soil, S_{nd}, the higher is the probability to choose *n*. This soil amount is updated by the *iwd* control packets as they explore the routes to destinations. The updating process of the soils is explained in the next subsection.

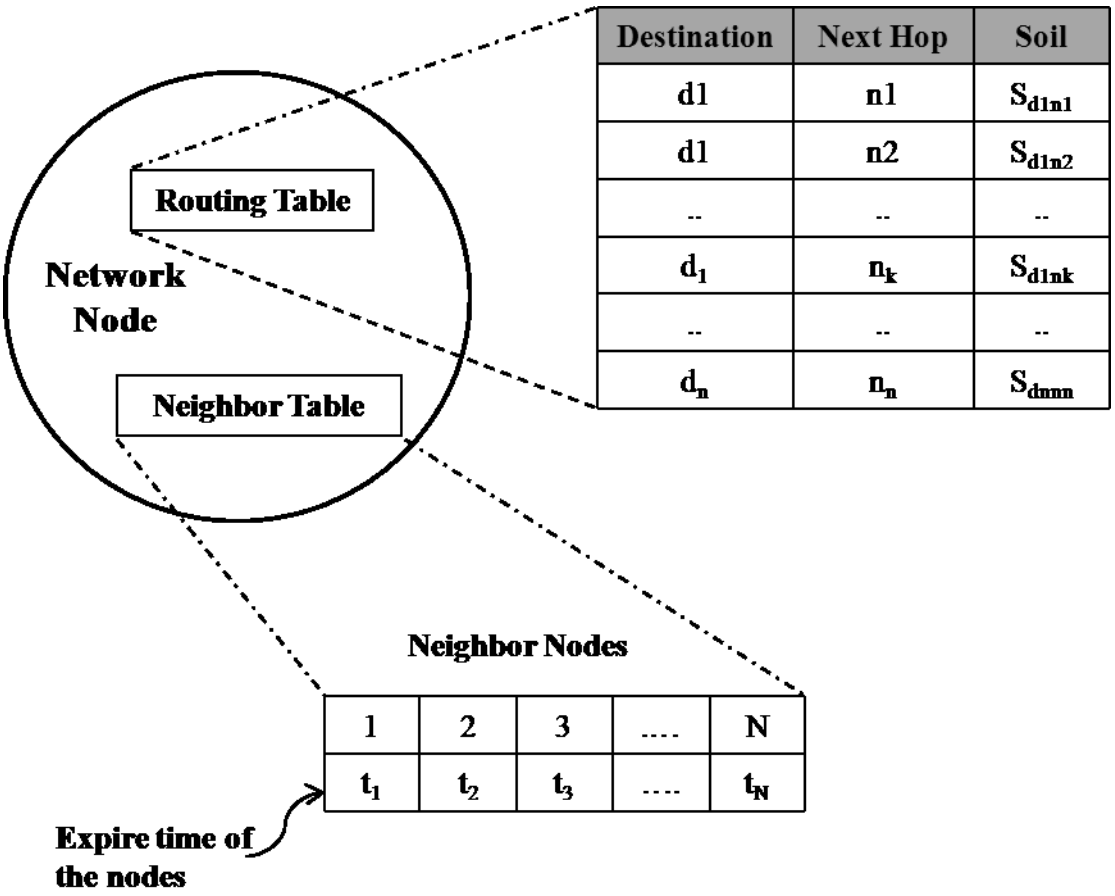


Figure 5-1: Data structures of a node in IWDHocNet: Routing table and Neighbor table.

Neighbor Table

The neighbor table $\mathcal{N}_i$ is used to keep track of the active neighbors. It is also a two dimensional matrix containing the node ids of the neighbors and a time value t_n indicating when the entries should be expired. The time value t_n helps to determine which of the nodes are still active and in the range of communication and which are not. When an entry is added to the neighbor table, i.e. when a node is added, the t_n is set to be

equal to a default expire time (2 seconds is used in this algorithm). This list is updated by sending and receiving periodic acknowledgement/HELLO packets.

5.2.2 Step by Step Procedure of the Algorithm

Before starting the algorithm the static and dynamic parameters used in the algorithm have to be initialized.

Initialization of static parameters: Set the velocity updating parameters, a_v = 10.00, b_v = 0.01 and c_v = 1. Soil updating parameters are set as a_s = 10.00, b_s = 0.01 and c_s = 1. Each link to the neighboring node is set with an initial amount of soil, *InitSoil* = 100. The initial velocity of the IWD's originating is set to be equal to *InitVel* = 10.

Initialization of Dynamic parameters: For every IWD, a memory stack *Memory(IWD)* is created and set to empty. This memory stack is used to save the order of visited nodes and the corresponding trip times (time taken to travel from one hop to the next).

The algorithm is explained in detail for the three phases.

5.2.2.1 Route Discovery and Route Maintenance Phase

1. At regular intervals, from every node *s*, a FIWD (Forward Intelligent Water Drop) $FI_{s\rightarrow d}$ is launched with its goal set to find a path to a randomly selected destination node, *d*. (Gaussian function is used to select a node on random from the list of possible destinations.) These FIWDs are launched asynchronously, so as to avoid traffic overload that might occur by sending too many packets at the same instance of time. The FIWDs are set with a 'time to live' parameter as they are

initialized, which defines the lifetime of the packet, it is set to be equal to twice the number of nodes in the network. The FIWDs as explained in section 5.1 are small packets; they are serving as control packets whose role is only to establish a path. These packets are launched concurrently with the data packets.

2. The launched FIWD travels in a hop-by-hop fashion in order to discover a path to the destination node. While traveling, the IWDs update its memory with the address of each visited node N_v and also the time taken to reach the node (trip time).
3. At each node *i*, the FIWD chooses the next hop neighbor, *j* from the list of nodes in its neighbor list ignoring the ones which were already visited. The node *j* is chosen with a probability which is calculated based on the amount of soil on the link connecting the current node *i* and node *j* and also the cost of the link. The probability is calculated based on the equation, (5.1).

$$prob(i,j;IWD) = \frac{f(soil(i,j))}{\sum_{k \neq N_v} f(soil(i,k))} \tag{5.1}$$

In equation 5.1, *f(soil(i,j))* is a parameter which defines the goodness of the path, it is defined in equation 5.2. In this equation, ε_s is a small positive number to prevent division by zero (ε_s = 1.0 is taken for this algorithm), the function *g(soil(i,j))* (defined in equation 5.3) is used to shift the *soil(i,j)* values to prevent any cancellations of the positive and negative numbers, and *Cost(i,j)* represents the cost of the link connecting *i* and *j*, which in this algorithm is taken to be the expected travel time to reach node *j*.(There are different performance metrics that are considered for calculating the cost)

$$f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j)) + Cost(i,j)} \quad (5.2)$$

Where,

$$g(soil(i,j)) = \begin{cases} soil(i,j) \ \ if \ \min(soil(i,l)) \geq 0 \\ soil(i,j) - \min(soil(i,l)) \ \ else \end{cases} \quad (5.3)$$

The cost of the link, *Cost(i,j)* given in equation 5.4 [7] is calculated by multiplying the number of packets awaiting in the queue to reach node *j* (Queue length) and the last trip time recorded to reach *j* from node *i* in milliseconds.

$$Cost(i,j) = ((Qlen_{ij} + 1) * t_{ij}) \quad (5.4)$$

4. It should be noted that if the neighbor list is empty or if there are no nodes in the set neighbor list, such that ($k \neq N_v$) to forward the FIWD to, the packet, FIWD is dropped, because there is no route ahead for the packet.
5. The control packet FIWD is then forwarded to the neighbor chosen based on the probability calculated in the previous step, with the destination still set to *d*.
6. At each and every step before processing the received control packets, the 'time to live' parameter of the packet is checked to see if the packet lasted longer than its lifetime; if it does the packet is destroyed to avoid stale packets.
7. The steps 2-6 are repeated till the FIWD reaches the destination node. Once the destination node is reached, the forward IWD $FI_{s\rightarrow d}$ is converted to a reverse IWD $RI_{s\rightarrow d}$ which simply inherits the stack in the memory of the $FI_{s\rightarrow d}$. The generation of the RIWD (reverse IWD) is important because of the lack of a centralized node

in MANETs which can communicate with all the nodes. The RIWD acts like a 'Route Reply' message and is sent back to the source node.

8. While moving along the path, the RIWD $RI_{s\rightarrow d}$ updates the amount of soil in the routing tables of the nodes for the corresponding entries of the destination node. For example, consider a path 1->2->3->4 has been discovered from node 1 to node 4; the RIWD updates the soil amount, S_{nd}, in the routing table entry, $\mathcal{T}_{kn}^{d}$, for 1, 2 and 3. To update the route from node 2->4, the soil amount associated with next hop 3, S_{34} is decreased. The updating process of the soils is explained in the next step.
9. The $RI_{s\rightarrow d}$ on its way back to the source node, adjusts the amount of soil on the links along the discovered path and the velocity with which the IWD is moving. The velocity of the IWD is increased based on the equation 5.5. $vel^{IWD}(t+1)$ in the equation is the updated velocity and $vel^{IWD}(t)$ is the old velocity of the IWD; a_v, b_v, c_v are the static velocity updating parameters explained at the start of the algorithm.

$$vel^{IWD}(t+1) = vel^{IWD}(t) + \frac{a_v}{b_v + c_v soil(i,j)} \tag{5.5}$$

Arriving at a node k coming from a node h, the $RI_{s\rightarrow d}$ decreases the soil S_{kh} in the routing table T_k by an amount $\Delta soil(i,j)$.

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s . time^2(i,j; vel^{IWD}(t+1))} \tag{5.6}$$

Such that,

$$time(i,j;vel^{IWD}(t+1)) = \frac{Queue\ Length(i,j)}{vel^{IWD}(t+1)} \tag{5.7}$$

The soil of the path from node *i* to *j*, and the soil carried by the IWD, $soil^{IWD}$ is updated based on equation 5.8 and 5.9 respectively. In the equation 5.8, ρ_n is the static soil updating parameter set to 0.9 and N_{hops} is the number of hops it took to reach destination *d* via node *j*; if the number of hops is more, very less amount of soil is removed from the link, this is done to make the amount of soil displaced to be inversely proportional to the number of hops so as to look for more routes with less number of hops.

$$soil(i,j) = (1-\rho_n).soil(i,j) - \rho_n.\frac{\Delta soil(i,j)}{N_{hops}} \tag{5.8}$$

$$soil^{IWD} = soil^{IWD} + \frac{\Delta soil(i,j)}{N_{hops}} \tag{5.9}$$

10. The RIWD inherits the memory stack till it reaches the source node and the steps 8 and 9 are repeated at each node in the stack. After reaching the source node, the control packet IWD is 'freed' as its purpose has been served.

Figures 5-2 and 5-3 are the flowchart representations of the work flow followed by FIWD and RIWD packets.

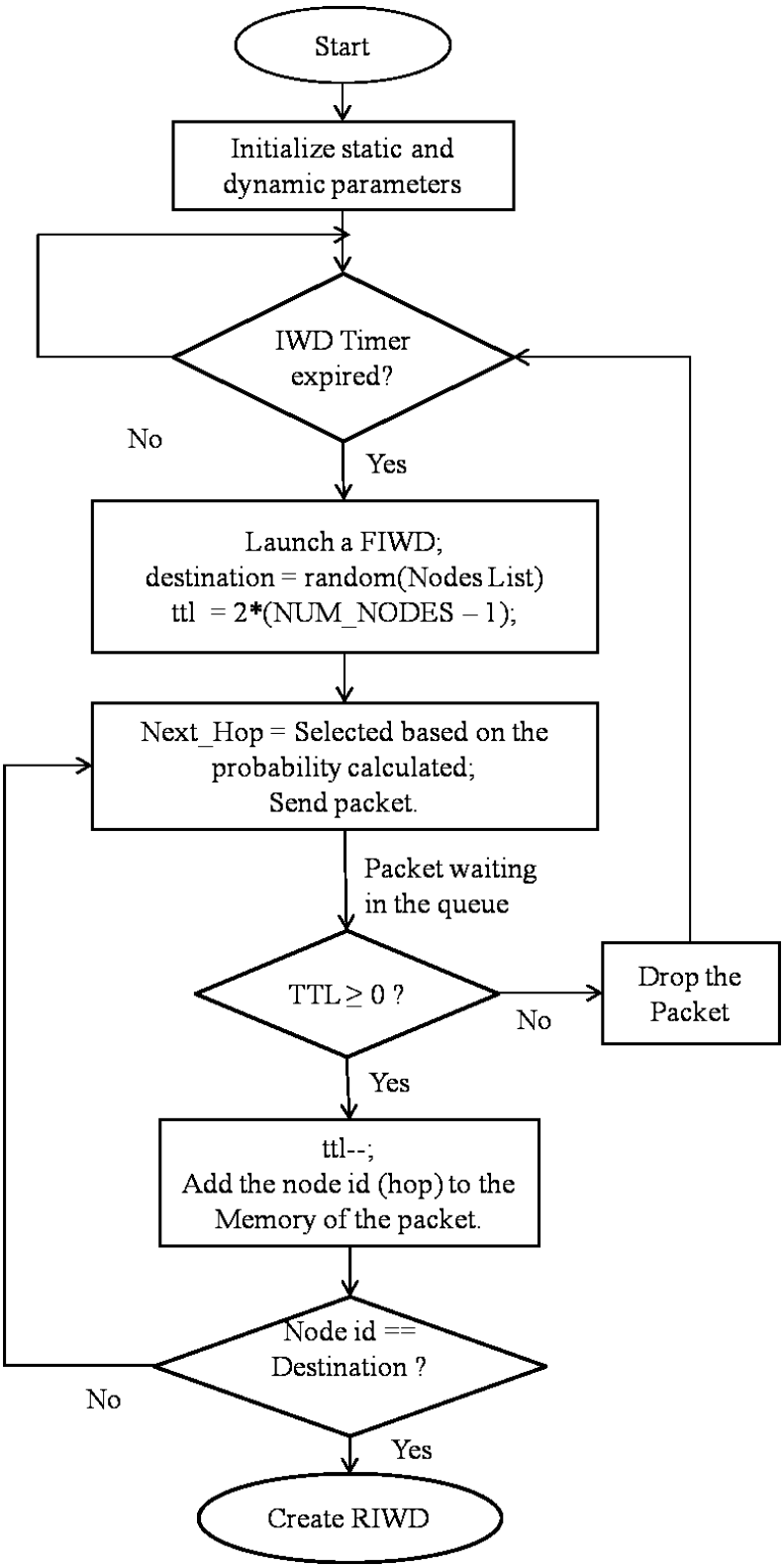


Figure 5-2: Flowchart of FIWD

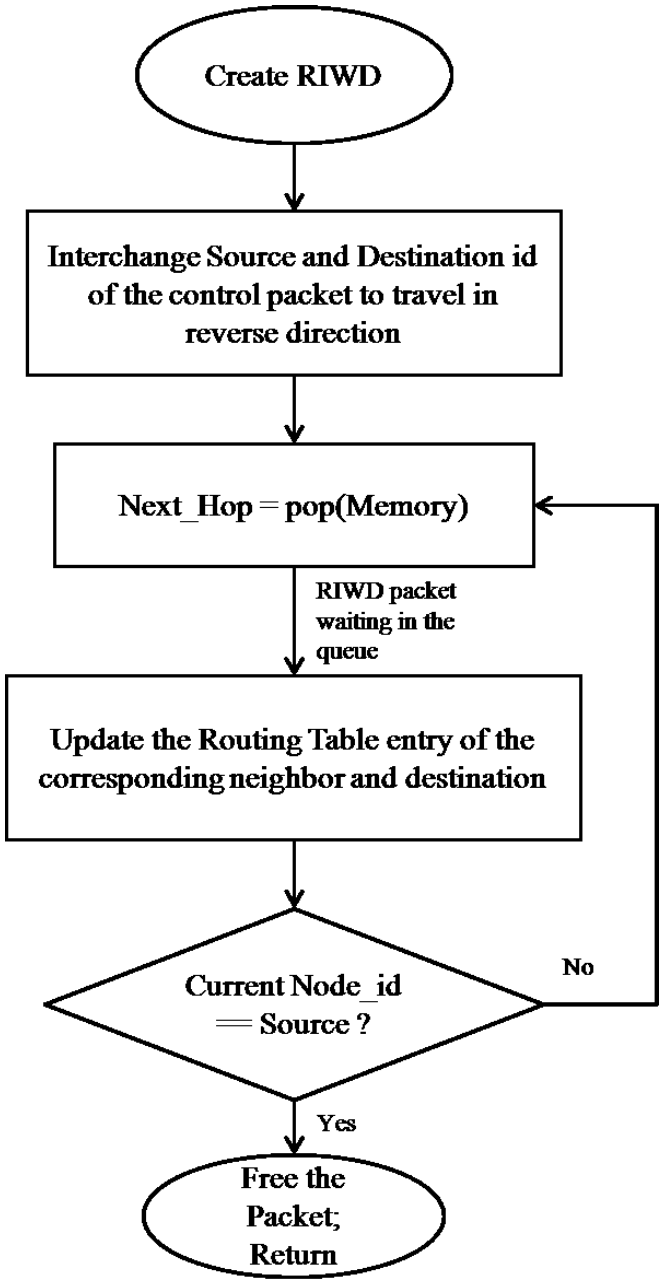


Figure 5-3: Flow of the Reverse IWD (RIWD) packet

```
t     = Current time;
t_end = Time Length of the Simulation;
t_iwd = Average time interval between generation of IWDs;
while (t≤t_end)
   foreach (Node) /* Concurrent activity over the network*/
     𝒩 = Neighbor table of the node;
     𝒯 = Node routing table;
     Δt = Random(t_iwd ± 25%); /* To generate asynchronous FIWDs */
     in_parallel
        if (t mod Δt)
          destination_node = RandomNode(Node_List);
          next_hop_node = CalculateNextHop(Node, destination_node, 𝒯);
          ℳ = Initialize Memory Stack;
          Push(ℳ, Node);
          LaunchFIWD(Node, destination_node, next_hop_node);
        end if
        foreach FIWD in Queue
           while (current_node ≠ destination_node)
              if (non-visited nodes in Neighbor Table = NULL)
                   Drop_FIWD_Packet; break;
              Push(ℳ, next_hop_node);
              next_hop_node = CalculateNextHop(Node, destination_node, 𝒯);
              ForwardFIWD(current_node, next_hop_node);
              WaitOnQueue(current_node, next_hop_node);
              Current_node = next_hop_node;
           end while
  /*FIWD reached the Destination node after the loop*/
  ChangeToReverseIWD(destination_node, Node, ℳ);
  foreach RIWD in Queue
      while (current_node ≠ source_node)
          next_hop_node = Pop (ℳ);
          ForwardRIWD(current_node, next_hop_node);
          WaitOnHighPriorityQueue(current_node, next_hop_node);
          UpdateRoutingTable(𝒯,current_node, destination_nodes, HopCount);
      end while
  end foreach; end in_parallel; end foreach; end while /* end of while (t≤t_end) */
```

Figure 5-4: Pseudo-code of the Route Discovery and Route Maintenance phase in IWDHocNet algorithm

At the start of the algorithm, since the amount of soil on all the paths will be the same, the probability of choosing any neighbor for the next hop will also be equal for all the nodes. But, after few iterations the initial routes will be setup and better routes are found or the routes are further strengthened based on the quality of the solution.

Data Packet Forwarding:

Data packets in IWDHocNet are forwarded in the same way as the FIWD control packets, i.e., the routing decisions are made hop-by-hop based on the amount of soil and the queue length on the link, except they are given high priority while processing. The data packets are forwarded to their destinations by following the steps 3-6 described in the step-by-step procedure in this section. But, when the neighbors list is empty (step 4), instead of dropping, the protocol sends the data packets into a local queue till its 'time to live' expires. The protocol initiates a 'data purge' procedure every two seconds to check if there are any packets waiting for a route in the local queue. If there are any packets, the protocol again searches for a route, till it finds a route or the data packet expires or the queue becomes full.

5.2.2.2 Link Failure Handling

Link failures in IWDHocNet are detected using the traditional method of broadcasting HELLO messages [17, 19] in the network. HELLO messages, as the name suggests are small messages broadcasted to advertise its presence. HELLO messages help the nodes to maintain an updated Neighbor table. The neighbor discovery works in parallel with the route discovery and route maintenance phase.

1. Each node in the protocol broadcasts HELLO messages asynchronously. HELLO messages are very small messages which do not contain any additional information other than the source id, it has been created. The messages are sent with a random periodicity in the range, [0.75*HELLO_INTERVAL, 1.5*HELLO_INTERVAL]. The reception of a HELLO message from a node indicates the node's availability.
2. The nodes continuously listen to the HELLO messages. When a node receives a HELLO message from a node, it checks if the node is already in the neighbor table. If the node is present, the expire time is increased. If not, the node is added to the Neighbor table, and an entry is created for the neighbor in the routing table, for all the possible destinations. (The structure of the routing table is explained in section 5.2.1). The amount of soil in the table entries is assigned to be equal to *InitSoil* as the route is freshly created.
3. Also for every HELLO_INTERVAL seconds, the protocol initiates a 'neighbor purge' procedure in which the nodes check if its neighbors are still available for transmission. This is checked with the help of *timeout.* As explained the nodes maintain a Neighbor table in which each neighbor is assigned with an associated expire time (section 5.2.1). If node *k* has not received any HELLO messages from node *n* and the current time is greater than the expire time, neighbor *n* is considered to be dead or moved somewhere out of the communication range and is removed from the neighbor table. The corresponding entries of the neighbor in the routing table are also deleted.

Figure 5.5 shows the neighbor discovery process used in IWDHocNet protocol.

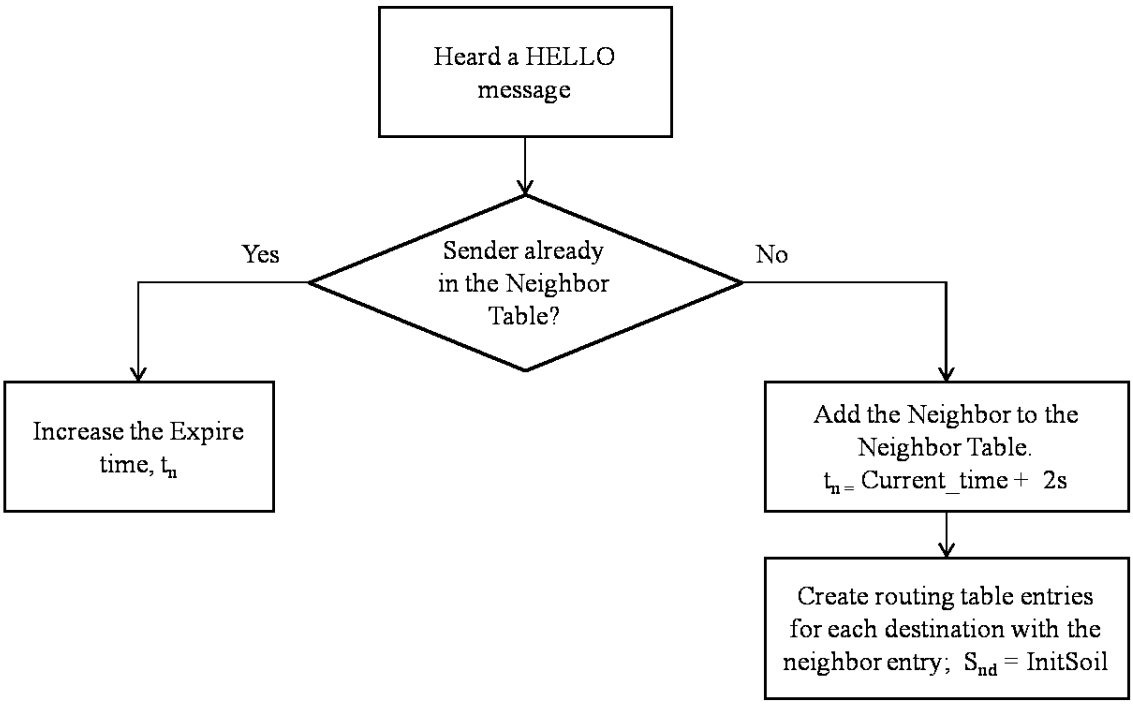


Figure 5-5: Flowchart showing how the received HELLO messages are processed.

Chapter 6

Simulations and Results

The developed routing algorithm, *IWDHocNet* is implemented on a network simulator to analyze the performance. Since the implementation of IWD algorithm to routing in networks is an experiment, the algorithm is first tested on a simulator before implementing it in real-time applications. The first section of this chapter, 6.1 gives a very brief description of the simulator that is used for testing; section 6.2 describes the simulation methodology and the performance metrics used for the performance analysis and section 6.3 presents the simulation results and compares the IWDHocNet protocol with AODV and DSDV protocols.

6.1 NS-2 Simulator

To test and compare the performance of the proposed *IWDHocNet* routing algorithm, we used the network simulator NS-2, version 2.34 [20]. NS is a discrete event simulator targeted at networking research. It is an object-oriented network simulator developed as a part of the VINT (Virtual InterNetwork Testbed) project at the University

of California in Berkeley[21]. It provides support for simulation of TCP, routing, and multicast protocols over wired and wireless; local and satellite networks. It is heavily used in ad-hoc networking research and is a standard experiment environment being followed by many in the research community.

NS-2 simulator is a UNIX based simulator. It is written in C++ and OTcl, an object oriented version of Tcl. Users with this simulator can define their own routing protocols and arbitrary network topologies composed of nodes, routers, links, data sources etc. Though the development of routing protocols in NS-2 simulator is quite difficult and challenging, it is chosen for this thesis because of its popularity in academia research. With the help of some documentation [22-25] about designing a routing algorithm in MANETs, the algorithm IWDHocNet is developed using C++ language.

6.2 Results

The routing algorithm, IWDHocNet, is evaluated in a number of simulation scenarios. Under each scenario, the algorithm is compared with the well-known state-of-the-art protocols, AODV and DSDV, which are explained in chapter 3.

6.2.1 Experimental Setup

The network model used in the simulation is composed by mobile nodes and wireless links that are considered bidirectional. The mobility model uses *Random Waypoint Model* (RWP)[26] to create the movement patterns of independent nodes for the simulation scenarios needed. RWP is one of the most widely used random-based

synthetic mobility model in performance analysis of ad-hoc networks. In this model, the mobile nodes start their journey from a random location and move to a random destination without any restrictions, the velocity with which the nodes move is randomly selected from a uniform velocity distribution. After reaching a random destination the node will pause (wait) before moving to the next destination. Several scenarios were obtained from RWP by varying the velocity of the nodes and the "pause times".

Table 6.1: Simulation Parameters

Protocols	: IWDHocNet, AODV, DSDV
Number of nodes	: Between 5 and 50
Dimensions of area	: 500 x 500
Simulation Time	: 500 seconds
Transmission range	: 250 m
Mobility Model	: Random Waypoint Model
Physical Layer	: IEEE 802.11
Varying Speed	: (0-50) m/s
Varying Pause Time	: (0-500) seconds
Traffic Generator	: CBR
Size of packet	: 512 bytes
Varying Packet Rate	: (1- 8) packets/sec
Varying Connections	: (2-25)

The simulation parameters configured are shown in the Table 6-1. A base simulation setting is created in which, 25 mobile nodes are randomly placed in a rectangular area of 500m x 500m. The nodes move with a maximum velocity of 50m/s (about 110 mph) which is the speed of a fast moving car. The velocity is varied from 0m/s to 50m/s to create different scenarios. Each simulation is run for a length of 500 seconds. The pause times of the nodes are varied from 0 seconds (continuous mobility) to 500 seconds (stationary). The data traffic is generated by CBR (Continuous Bit Rate) traffic generators. The data packet sizes are 512 bytes. The number of traffic generators (or connections) and the packet generation rates are varied to test under different load conditions. We used the 802.11 protocol at the MAC layer. The radio propagation range of the nodes is set to 250 meters.

6.2.2 Performance Metrics

Three key performance metrics are considered for the analysis of the results[27]. They are:

i. Packet Delivery Ratio: Ratio of the number of data packets delivered to the destination to the number of packets generated by the source. *"Total packets received/Total packets sent"*

ii. Average End-to-End Delay: The average time taken by a data packet to travel from the source to the destination. *"Sum(delay of packet i)/Total number of packets transmitted"*

iii. Average Routing Load: Number of routing (control) packets sent in one second. "*Total Routing Control Packets/Simulation time"*

6.2.3 Comparisons to other Routing Algorithms

In this section, we analyze the results obtained from IWDHocNet and compare the protocol with state-of-the-art protocols, AODV and DSDV. The comparison is based on four parameters: the effects of changing pause time, mobility, number of connections and bit rates on the performance metrics described in the previous subsection.

6.2.3.1 Varying the pause time for RWP mobility

Here, the node mobility is varied by changing the pause time between the node movements; the lower the pause time the higher the mobility in the network. By increasing the pause time in RWP, the nodes become less mobile and consequently the network becomes less dynamic and the scenarios become less difficult and vice versa. The simulations are performed with 8 different pause time values: 0, 20, 50, 100, 200, 300, 400 and 500 seconds. This experiment is carried out with 25 nodes which are set to move at a maximum speed of 5m/s between the pause time intervals; the bit rate of the data traffic is set at 2packets/second with 12 traffic generators. The following results in figures 6.1, 2 and 3 show the performance metrics as a function of node pause time.

Figure 6-1 illustrates the varying packet delivery ratio of the three routing protocols with pause time. At a higher pause time setting, i.e. when the nodes are static, IWDHocNet was able to achieve almost 100% packet delivery ratio like AODV and DSDV. IWDHocNet algorithm outperformed DSDV algorithm by showing a slight increase in the packet delivery ratio at higher pause times but failed to maintain its stability as the pause time decreased (as the network became more dynamic).

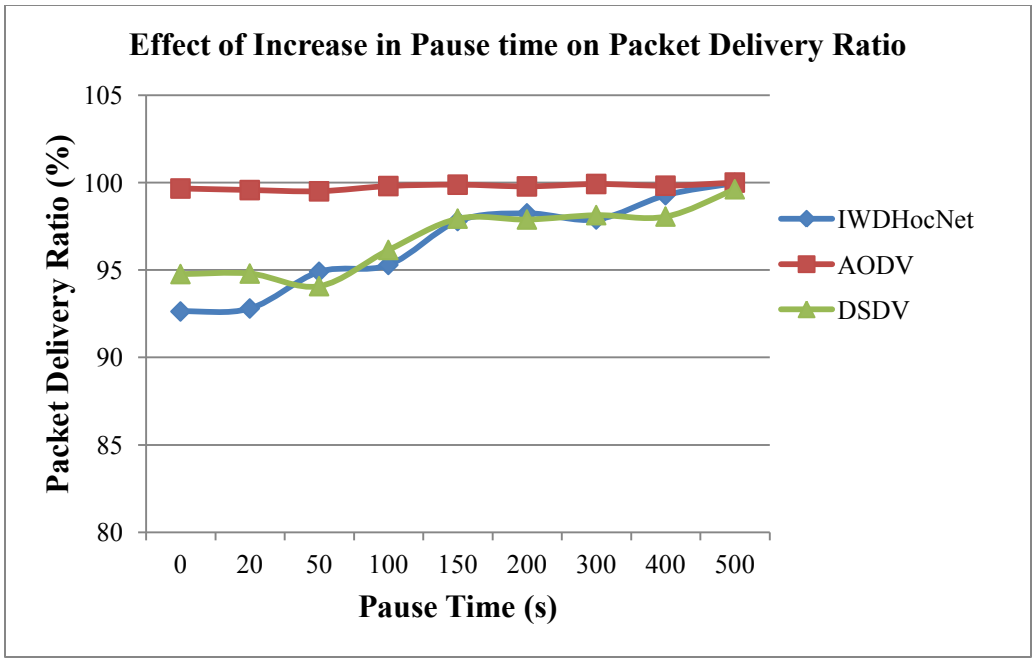


Figure 6-1: Packet Delivery Ratio decreases in IWDHocNet as the nodes pause for less time

The purely reactive, AODV algorithm was able to deliver more than 99% of the data in all the cases. The proactive algorithms, IWDHocNet and DSDV algorithms packet delivery rate decreased with decrease in pause time. On an average, AODV delivered 99.7% of the data followed by DSDV with 96.8% and IWDHocNet with 96.5%.

The degradation in the packet delivery rate could be because of the proactive nature of the IWDHocNet and DSDV algorithm, since a stale entry in the routing table of the nodes could point the data packets towards a broken link. AODV algorithm on the other hand initiates a fresh route discovery when a data packet arrives (only after checking that the route did not expire) and thus it was able to deliver the data packets even under dynamic conditions.

DSDV and AODV protocols, both maintain only one route per destination. On the other hand, IWDHocNet protocol provides multipath routing and thus it is expected to show better performance than the other two protocols. However, after analyzing the trace files it is found that IWDHocNet needs some time to establish the optimal routes initially when new links are created and old links are broken and also because of its random nature in selecting a destination to establish the paths. In other words, the algorithm is unable to make quick (correct) decisions when the node mobility is high. Thus most of the data packets took wrong or imperfect paths and got dropped.

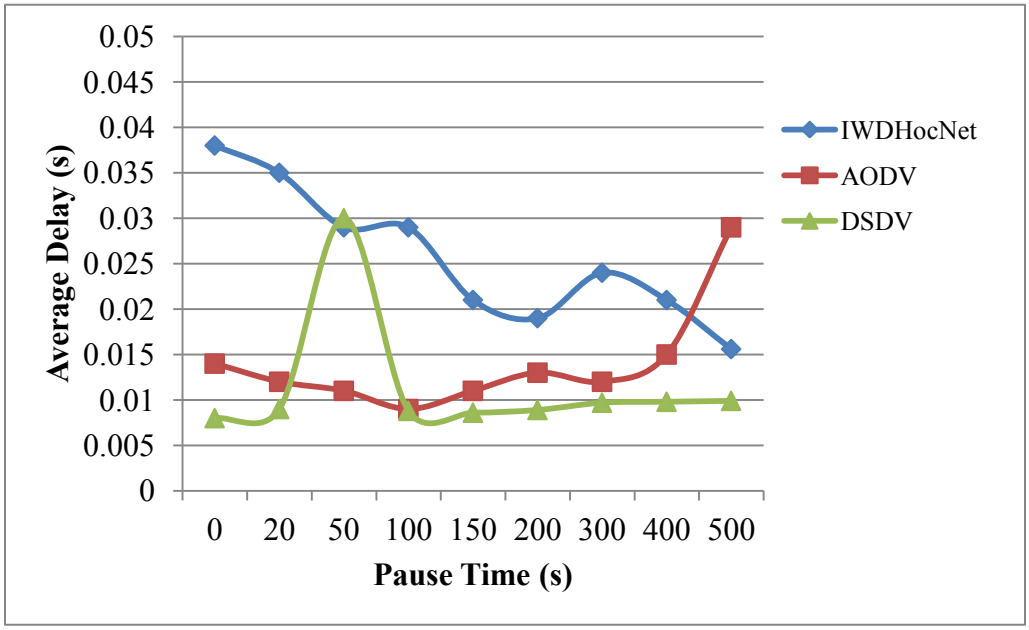


Figure 6-2: The average End-to-End delay of IWDHocNet algorithm increases when the pause time in RWP decreases.

Figure 6-2 shows the effect of varying pause time on the end-to-end delay the data packets experience. The average end-to-end delay for IWDHocNet protocol almost doubled as the nature of the network changed from static to very dynamic. The reason is

the same as explained in the last paragraph that the protocol is unable to make quick decisions when new links are created and old links are lost. The data packets took a long route instead of a shorter route and thus the delay increased. Also, as explained in chapter 5, the data packets wait in the queue if it does not find any hops from any node, which also results in a delay in delivery.

The average delay seen in IWDHocNet is more when compared to the DSDV and AODV algorithms, this is because of the data calculations the algorithm executes to find a route. IWDHocNet algorithm calculates the next hop on the fly unlike the other two algorithms and thus needs more time to reach destination. But even though the delay is more, the IWDHocNet is able to maintain a good packet delivery ratio, as seen in figure 6-1.

Figure 6-3 shows the effect of increasing pause time on the routing load of the three protocols. The routing load for IWDHocNet is comparable with that of AODV algorithm and sometimes even better, as seen in the figure. DSDV algorithm outperformed IWDHocNet and AODV algorithm in terms of routing load. The routing load for IWDHocNet and DSDV does not change with the node mobility. IWDHocNet and DSDV algorithms being proactive in nature, continuously sends routing packets at regular intervals unlike AODV algorithm which sends the routing packets only when there is a need to find a route.

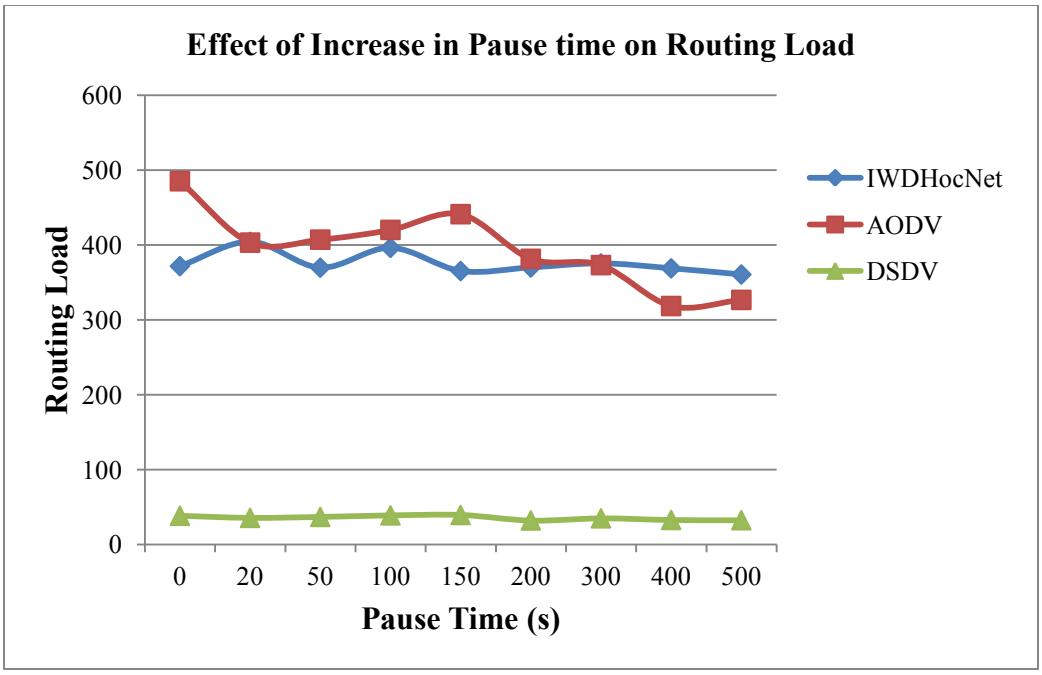


Figure 6-3: Routing load for IWDHocNet and AODV protocol is much higher when compared to that of DSDV protocol.

Though, DSDV and IWDHocNet are proactive routing protocols, the average routing load is higher with IWDHocNet algorithm when compared to DSDV. There are two reasons for this: IWDHocNet broadcasts HELLO messages, at an average interval of 1 second for neighbor discovery; and the time interval of sending the forward IWD (FIWD) control packets is much smaller (3 seconds) when compared to the time interval of control packets of DSDV (10 seconds).

6.2.3.2 Varying the maximum node speed for RWP mobility

In the second set of experiments, the node mobility is changed by increasing the maximum node speed in the random waypoint model. The speed is varied from 1m/s, which corresponds to a leisure walk, to 50m/s, the speed of a fast moving car. Increase in

speed of the nodes results in increase in the complexity of the network for obvious reasons. The experiment is carried out with six different node speeds: 1, 2, 5, 10, 20 and 50.

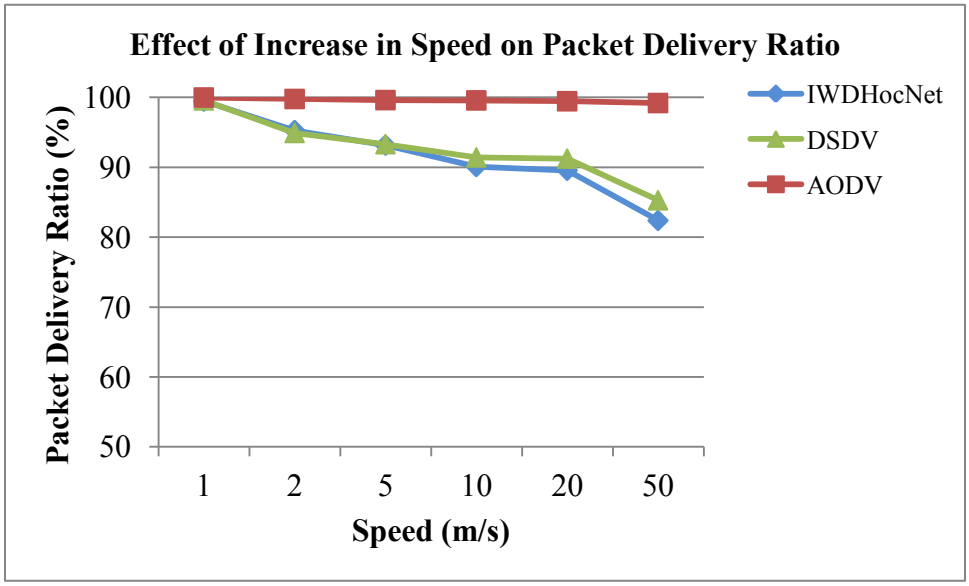


Figure 6-4: The performance in terms of Packet Delivery Ratio dropped by 15% as the speed of the nodes in the network increased for IWDHocNet.

Figure 6-4 shows the variation in packet delivery ratio with increasing speed of the nodes. The performance of IWDHocNet algorithm is close to DSDV but does not outperform it. At lower speeds, the IWDHocNet showed slighter better delivery rate than DSDV. However, DSDV traded off its better performance at higher speed with the average end-to-end delay as illustrated in figure 6-5. AODV algorithm continues to deliver better performance even under high node mobility conditions. However, there is a slight decrease seen in the AODV algorithm too as the node mobility increased.

The reason for degradation in the performance of IWDHocNet in a higher mobile network is the same that the algorithm cannot find optimal routes quickly. Another reason is, as one node moves away from a node, the node waits for several seconds (3 seconds) before confirming the failure of the link and modifying its routing table. Any data packets diverted towards this broken link will be lost due to this delay. Decreasing the length of the wait time for this link failure handling would not help as it increases the load on routing, also any delay caused in the reception of HELLO messages would end up giving wrong results. This can be handled by implementing local route repair techniques, as a part of the future work.

The lower packet delivery ratio is not just due to the node mobility. A large number of data packets got dropped at the end of simulation because of the delay in calculating the next hop for the data packet.

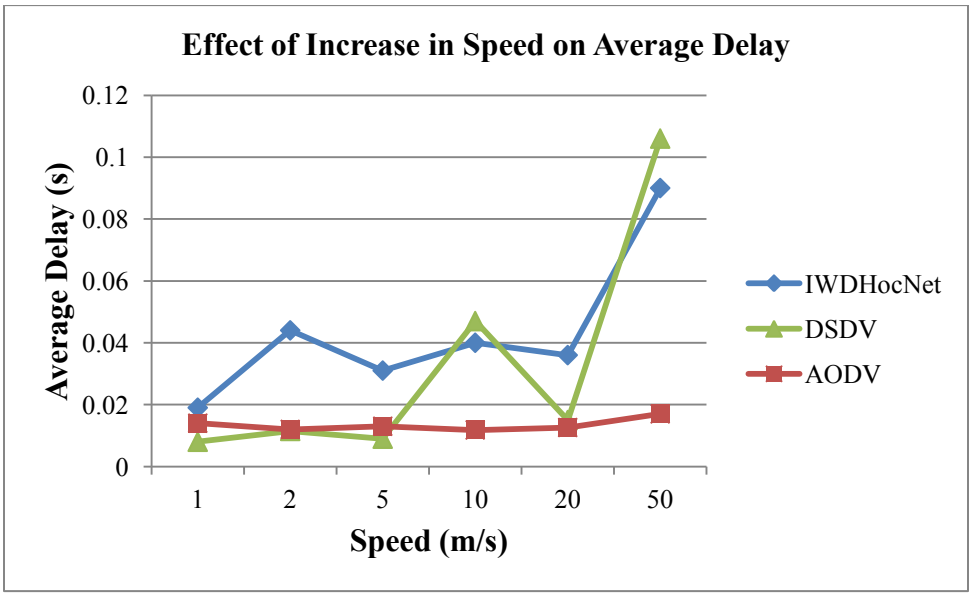


Figure 6-5: The Average end-to-end delay experienced by the data packets, in IWDHocNet protocol, increased with increase in node mobility.

In figure 6-5, the average end-to-end delay is measured at varying speed of the nodes. At higher node mobility, above 10m/s, IWDHocNet protocol outperformed DSDV protocol. Though DSDV started out with better results at lower speeds, IWDHocNet finishes with lower delay at higher speeds. AODV algorithm showed better results in case of higher mobile conditions.

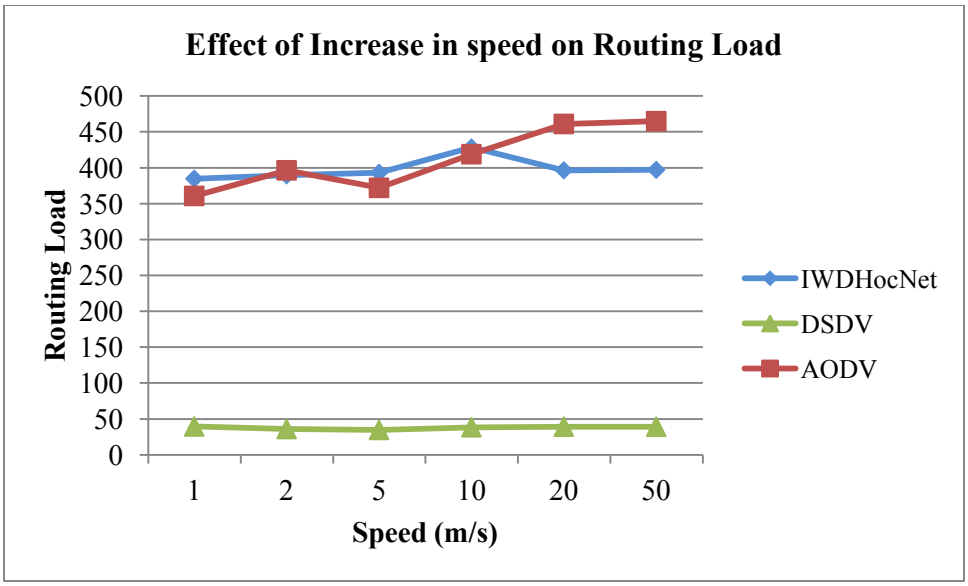


Figure 6-6: Routing load for AODV protocol increased by 30% as the speed of the nodes increased. IWDHocNet and DSDV maintained a constant routing load.

Figure 6-6 shows the variation of routing overload with node mobility. The routing load in AODV protocol increases with an increase in the speed of the node mobility. The routing load for IWDHocNet and DSDV does not vary. Ideally, the graph should be a straight line, but because of the asynchronous launching of the control packets (HELLO and IWD packets) the routing load is not constant. DSDV continues to

outperform both the protocols in terms of routing load. However, it traded this off with average end-to-end delay and packet delivery ratio.

6.2.3.3 Varying the number of connections for data traffic

Here, the experiments are conducted with varying number of connections or data sessions. Number of sessions indicates the number of sets of nodes between which the data is transmitted or a data communication has been set. The number is incremented in steps of 2 from 2 to 20. The network is configured for 25 nodes; the nodes are set to move at a maximum speed of 5m/s pausing for every 200 seconds (pause time is set to 200, as it is seen in the previous section that the IWDHocNet is able to give optimal results at that pause interval). 2 packets are sent per second from each data connection.

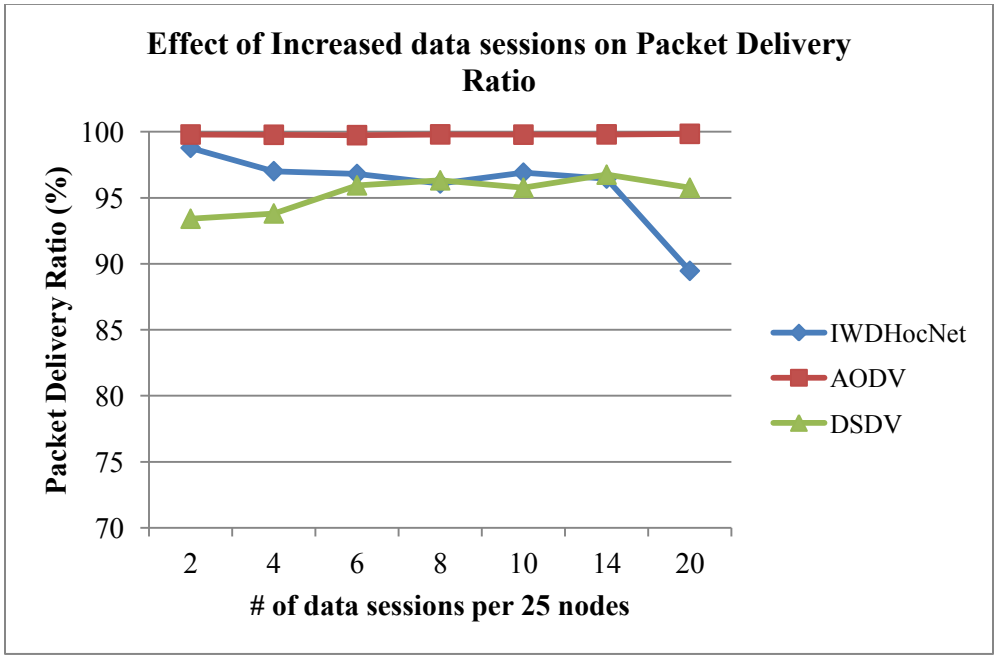


Figure 6-7: IWDHocNet protocol showed better performance than DSDV when number of data sessions is small. But, an increase in the number of data sessions caused the packet delivery rate to drop.

Figure 6-7 shows the measurement of packet delivery ratio as the number of data sessions increased. It is seen from the graph that IWDHocNet delivered more percentage of data packets than DSDV initially, but the packet delivery ratio dropped by 10%, as the number of data sessions increased from 14 to 20. However, it could not outperform AODV which delivered almost 100% of the data packets.

Analyzing the trace files from the NS-2 simulator showed that a large number of data packets got dropped during their mobility because the data queue was full (Queue length for the simulations is set to 50). The reason for the queue length to be full can be blamed on the complex calculations which are done in the IWDHocNet algorithm. The data packets wait in the queue when a routing decision is been made at the router level; the routing decision is made after some calculations and this caused the data packets to stay longer in the queue. The forward IWD control packets also share the same queue, which also creates a problem. Also, there were many packets that were dropped at the end of simulation run because of the same reason.

Figure 6-8 shows the variation of average end-to-end delay with the increase of data sessions. It is seen from the graph that IWDHocNet is unable to handle too many data sessions, as the delay increased substantially. Again this can be explained in terms of the complex calculations that are made during the data transmission.

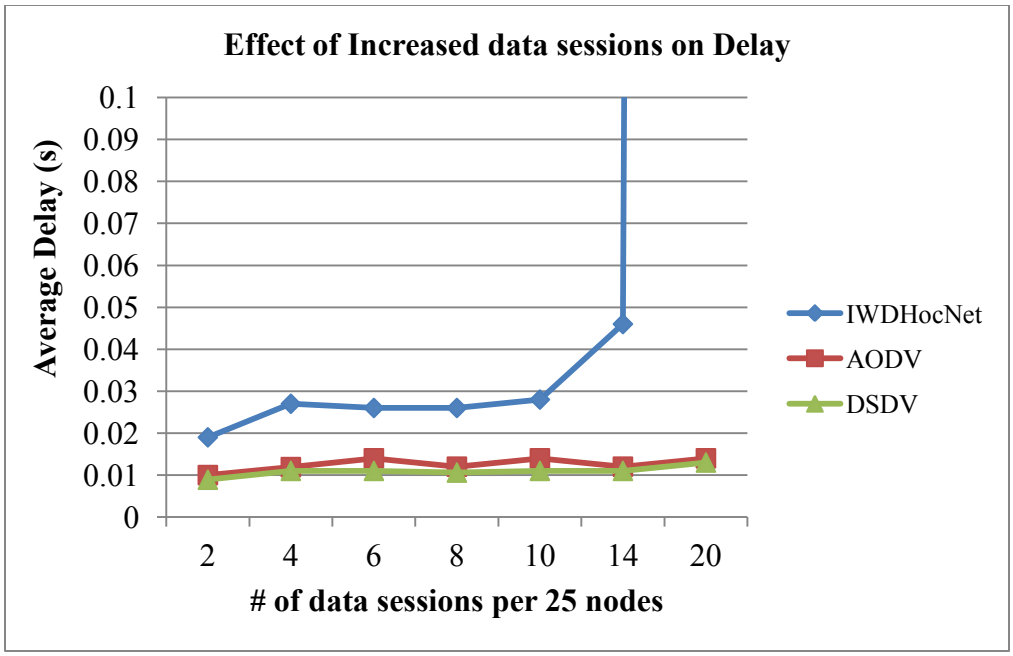


Figure 6-8: The Average end-to-end delay experienced by the data packets increased substantially as the number of data sessions increased in IWDHocNet.

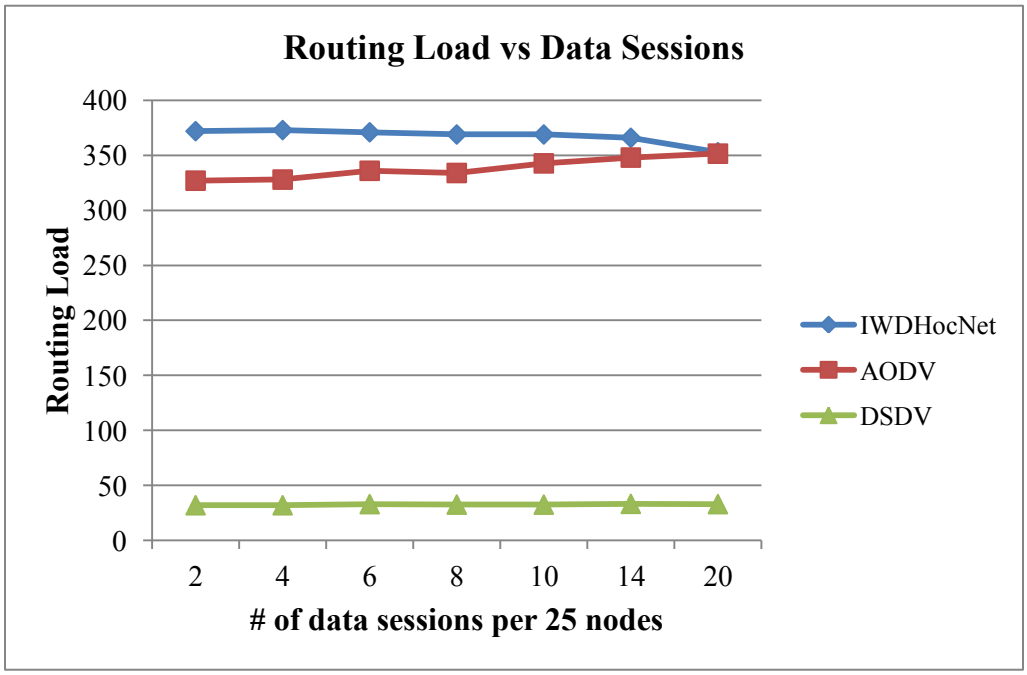


Figure 6-9: Routing Load remains constant for all the three routing protocols even though the data sessions increased.

Figure 6-9 plots the variation of routing load with the number of data sessions for all the three protocols. As explained in the previous sections the routing load for IWDHocNet and DSDV algorithm remains a constant independent of the node mobility, data traffic, etc. The routing load in the case of AODV algorithm also remained same because the network topology did not change in this experiment.

6.2.3.4 Varying BIT-RATE

In this experiment, lots of data is pumped into the network and the characteristics of each protocol are studied. The network is set with the same configurations as in section 6.2.3.3, except the number of connections is set to 12 and the rate of the data packets is increased from 0.5 to 8 packets/second.

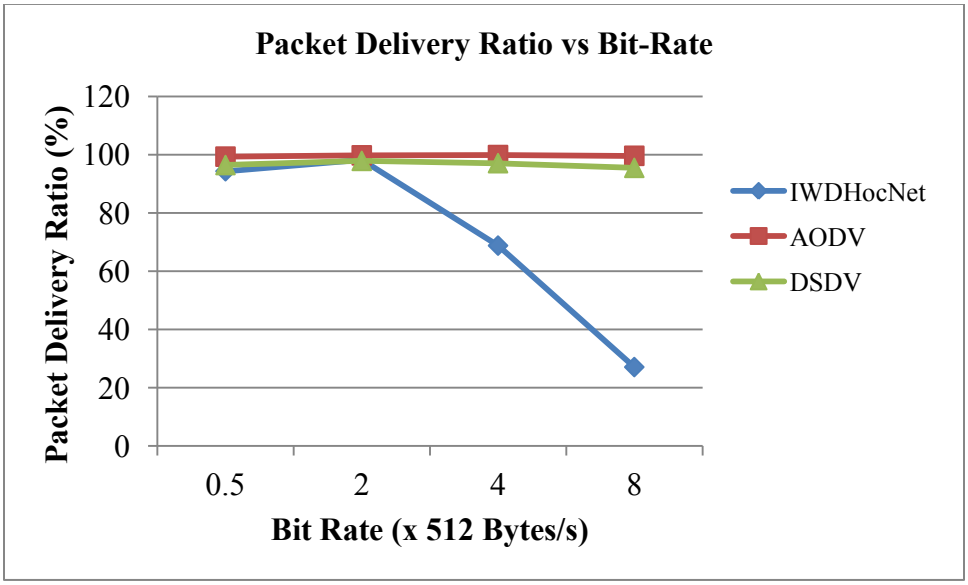


Figure 6-10: Packet delivery ratio decreased substantially as the data rate increased in case of IWDHocNet protocol.

Figure 6-10 shows that the packet delivery ratio dropped to 25% as the data sending rate increased, for IWDHocNet protocol. It is seen that the protocol is shown to give optimal results at a bit rate of 2 packets per second. The reason for this is also due to the queue dropping the data packets as explained in the previous comparison, varying the number of data sessions also applies to this case. Though, the delivery ratio decreased in all the three algorithms, the decrease seen in IWDHocNet is large.

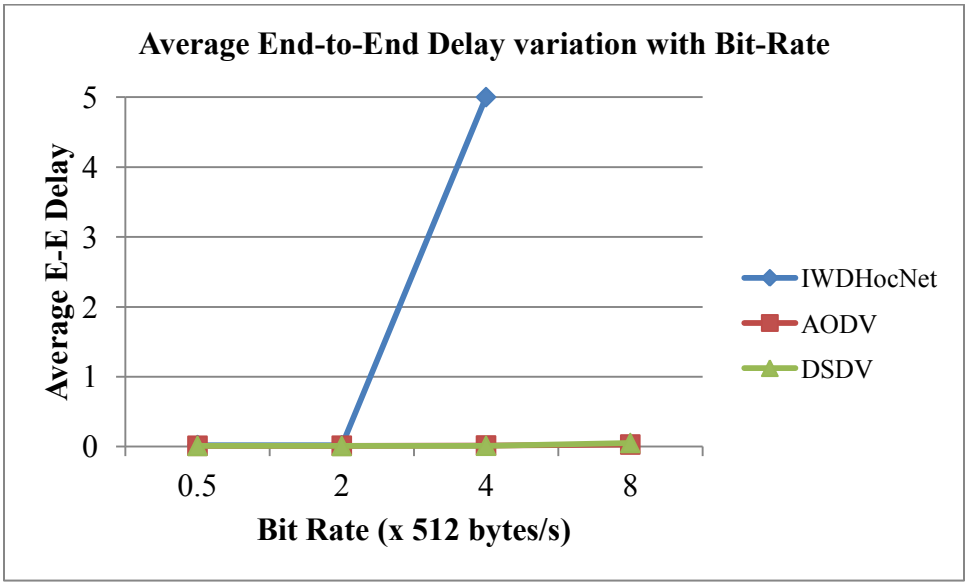


Figure 6-11: The average delay also increased as the bit rate of the data increased in case of IWDHocNet

Figure 6-11 also shows that the IWDHocNet algorithm is unable to manage too much of data traffic, as the delay increased from 0.022 seconds (at 2 packets/second) to 11 seconds (in case of 8 packets per second). Figure 6-12 measures the variation of the routing load with the bit rate. It is seen that the variation did not had any effect on the routing load for all the three protocols as the network topology remained the same in the experiment.

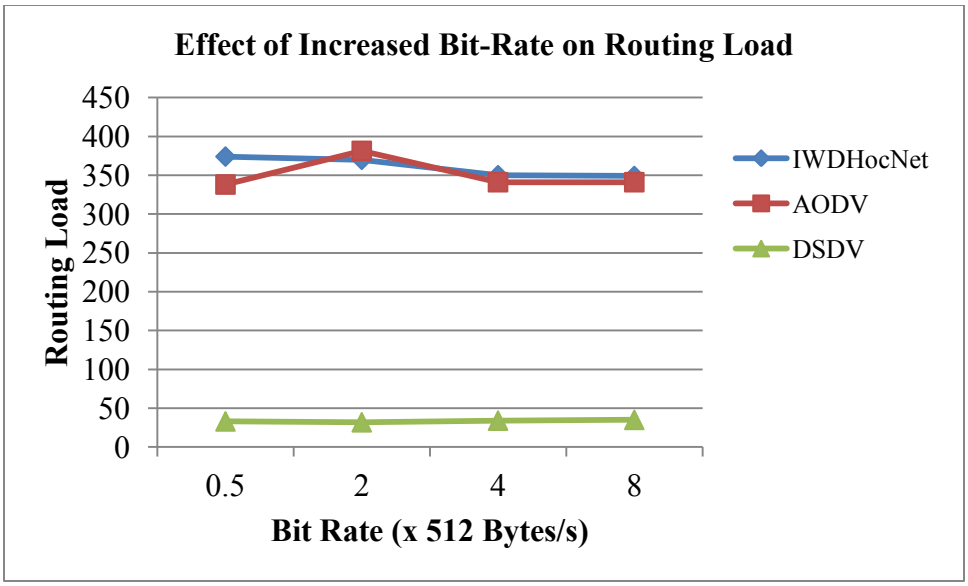


Figure 6-12: Routing load remains constant, independent of the data traffic for all the three protocols.

6.2.3.5 Effect of increasing node number in IWDHocNet

In this experiment, the number of nodes is varied from 5 to 50 and the performance of the IWDHocNet is compared for different cases. The IWDHocNet algorithm is not compared with the other state-of-the-art algorithms, as the purpose of this experiment is to study the behavior of the protocol under different network sizes. The settings of the experiment is same as that of the settings used in the experiment in section 6.2.3.1, except the number of data connections is set to be equal to half the number of nodes, in each case.

Figure 6-13 shows the variation of the packet delivery ratio with increasing number of nodes. When the number of nodes in the network is high, the topology would be dense and the connectivity would be rich, and therefore the data delivery rate is high. Thus, even in larger networks, the delivery ratio is maintained well. Though, it decreased

by 10% when the network became very mobile. The reason for this is explained in the earlier section (section 6.2.3.1).

In case of networks with fewer nodes, like a 5 node network the network suffers from partition and thus the packet delivery ratio dropped to 75% in some scenarios. This cannot be handled by any protocol because there is no way to transmit data when the nodes are not in the transmission range.

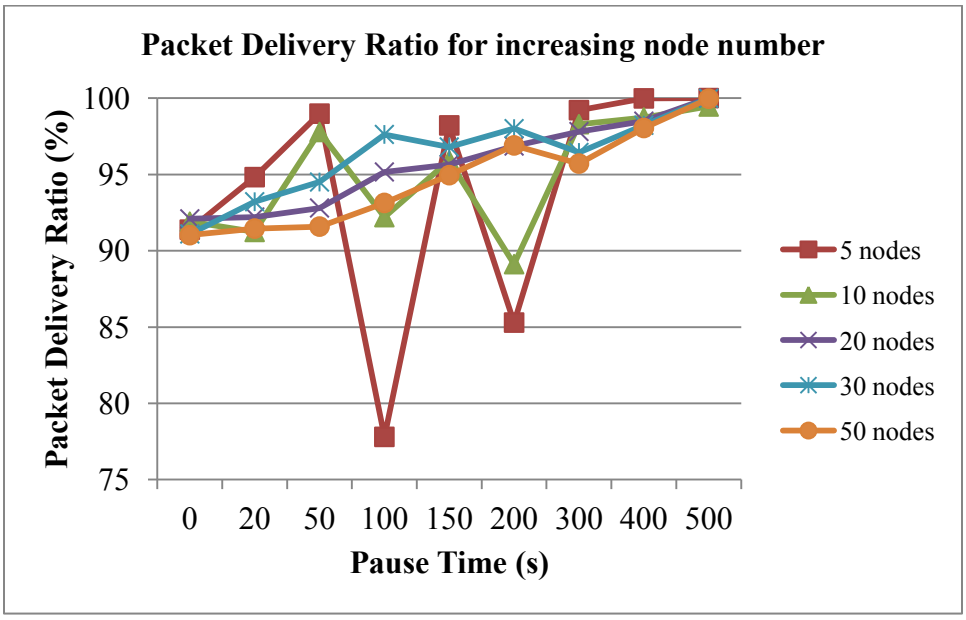


Figure 6-13: The IWDHocNet algorithm delivered 100% of the data packets under all the conditions. However, the delivery ratio decreased with the pause time.

Figure 6-14 shows the effect of increasing mobile node number on the end-to-end delay experienced by the data packets. It is seen that for a network with fewer nodes (take the case of 5 nodes), the delay is high at certain pause intervals. This is caused because of the network connectivity as explained in the last paragraph. It should be noted that, even

though there were no links to the destination node, the source node waited for some time for an active link to deliver the packets which increased the average delay. Thus, even though the delay is more, the delivery rate did not drop significantly (take the case of data points at pause time = 0, 20 and 50 seconds).

The average delay in different sizes of the network remained same, and did not change with the network size as the algorithm was able to find the optimal routes in all the cases.

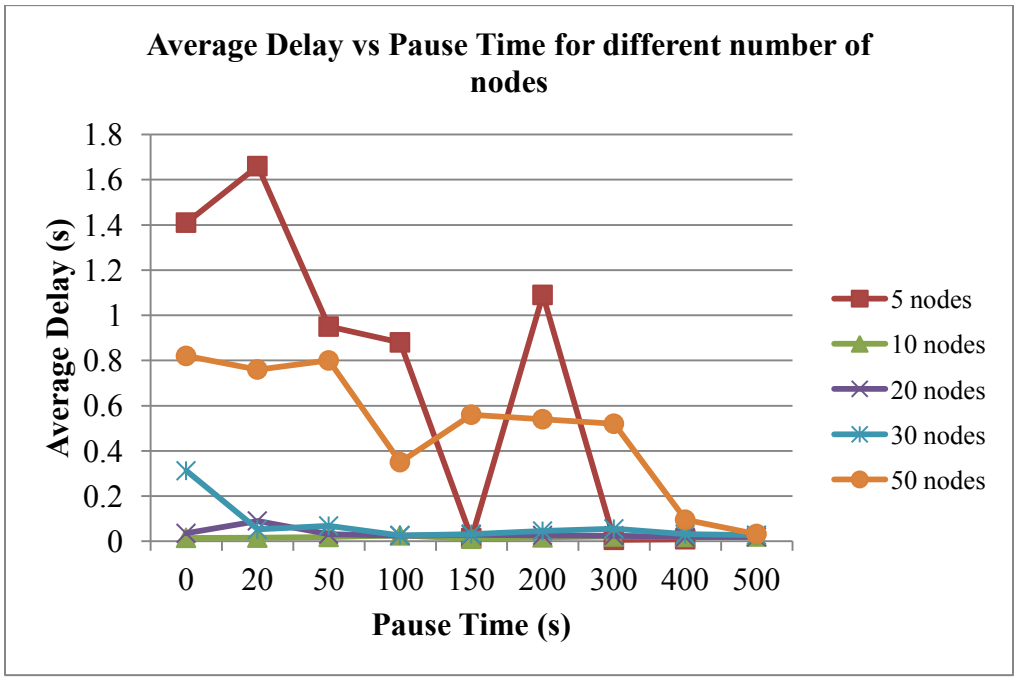


Figure 6-14: The Average delay in case of 5 nodes and 50 nodes increased with increase in node mobility, it is maintained well in case of a medium sized network.

Figure 6-15 shows that the amount of routing load varies with the number of nodes in the network. It is seen that the amount of routing load increases as the number of nodes increases in the network. This is because, as the number of nodes increase, the

number of routing packets (HELLO and IWD packets) sent on the whole also increases. It is observed from the graph that the routing load increased from 15 (5 node topology) to 1500 (50 node topology).

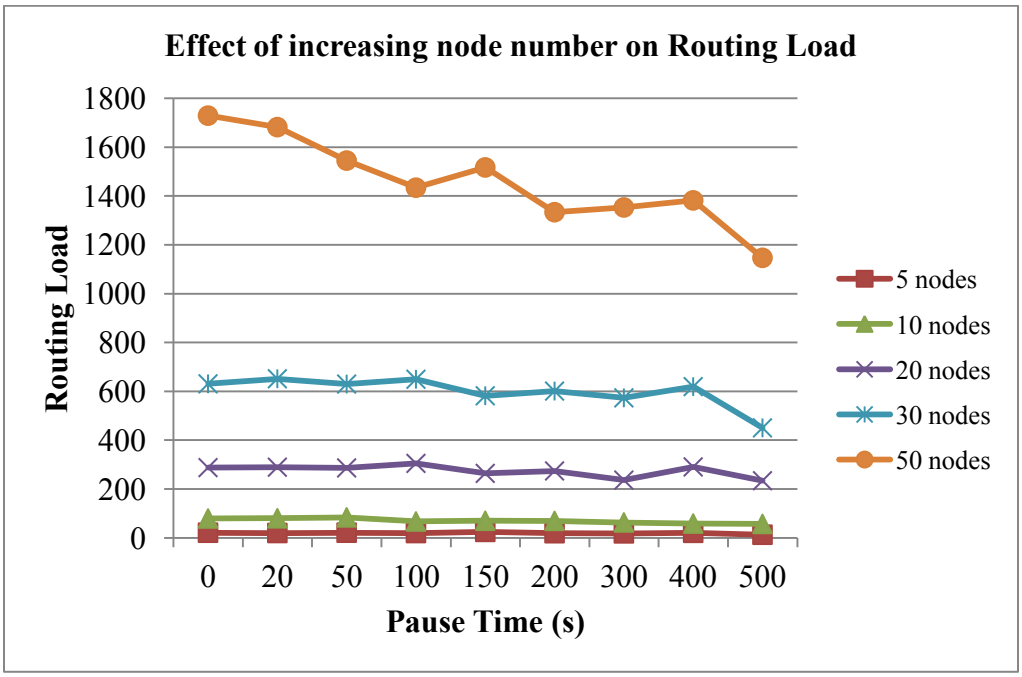


Figure 6-15: Amount of routing load increases with an increase in the number of nodes, though it remains constant with varying topology.

Chapter 7

Conclusion and Future Work

In this thesis, a new paradigm for routing in mobile Ad-hoc networks based on Intelligent Water Drops has been demonstrated. This protocol is the first effort to design intelligent water drops based routing protocol, which is named IWDHocNet. The algorithm has been demonstrated successfully on NS-2 Network Simulator.

The algorithm is compared with two other state-of-the-art protocols, AODV and DSDV. The following metrics were used to measure the performance of the protocol: packet delivery ratio, average delay and routing load. Experimental results showed that the protocol's performance in terms of packet delivery ratio is comparable with DSDV and AODV algorithms. In terms of routing load, IWDHocNet protocol is better than AODV though it could not reach the standards of DSDV.

The protocol, IWDHocNet, is just an example of successful application of the Intelligent Water Drops and there is a tremendous potential for extending the IWDHocNet. For future it is proposed to change the nature of the protocol to reactive from proactive and compare the results. Since, AODV being a reactive protocol showed

better performance when compared to DSDV which is proactive. This would also decrease the routing load experienced by the network. In future, it is worthwhile to implement additional route repair techniques in link failure handling to help the protocol perform better under high mobility conditions as well. The protocol is also expected to perform much better by simplifying the complex equations used in making the routing decisions. In order to explore the full potential of the protocol, it should be tested on other network models and other network platforms such as OPNET[28] and QualNet[29]. Moreover, the protocol should also be applied real time with PDAs and other devices to get real results.

References

1. Bonabeau, E., M. Dorigo, and G. Theraulaz, *Swarm Intelligence: from natural to artificial systems*. 1999: Oxford University Press.

2. Mir, N.F., *Computer and communication networks*. 2006: Prentice Hall.

3. Perkins C. and B. P. *Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers*. in *Conference on Communications Architectures, Protocols and Applications*. 1994. London, United Kingdom.

4. Misra, P. *Routing Protocols for Ad Hoc Mobile Wireless Networks*. Available from: http://www.cse.wustl.edu/~jain/cis788-99/ftp/adhoc_routing/.

5. Perkins C. and R. E. *Ad-hoc on-demand distance vector routing*. in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*. 1999.

6. Di Caro, G. and F. Ducatelle. *Swarm intelligence for routing in mobile ad hoc networks*. in *2005 IEEE Swarm Intelligence Symposium (SIS)*. 2005. Manno-Lugano, Switzerland: IEEE.

7. Ducatelle, F., *Adaptive Routing in Ad Hoc Wireless Multi-Hop Networks*. 2007, Università della Svizzera Italiana, Istituto Dalle Molle di Studi sull´Intelligenza Artificiale.

8. Wedde H.F., et al. *BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior*. in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. 2005. Washington DC, USA.

9. Shah-Hosseini H. *Problem solving by intelligent water drops*. in *Proceedings of IEEE Congress on Evolutionary Computation*. 2007. Swissotel The Stamford, Singapore: IEEE.

10. Duan H., Liu S., and W. J., *Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning.* Aerospace Science and Technology, 2009. **13**(8): p. 442-449.

11. Duan, H., S. Liu, and X. Lei, *Air robot path planning based on Intelligent Water Drops optimization*, in *IEEE International Joint Conference on Neural Networks, 2008*. 2008: Hong Kong. p. 1397-1401.

12. Kamkar, I., M.-R. Akbarzadeh-T, and M. Yaghoobi, *Intelligent Water Drops a new optimization algorithm for solving the Vehicle Routing Problem*, in *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*. 2010: Istanbul. p. 4142-4146.

13. Rao, S.R., *An Intelligent Water Drop Algorithm for Solving Economic Load Dispatch Problem.* International Journal of Electrical and Electronics Engineering, 2011. **5**(1): p. 43-49.

14. Shah-Hosseini H., *Optimization with the Nature-Inspired Intelligent Water Drops Algorithm*, in *Evolutionary Computation*. 2009, InTech.

15. *The Amazon River*. Available from: http://chullachaquiecolodge.com/TheAmazonRiver.html.

16. Baumann, R., et al., *A Survey on Routing metrics TIK Report 262*. 2007.

17. Moy, J. (1998) *OSPF Version 2*. STD 54, RFC 2328.

18. Haas J.Z. and P. R.M., *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. 2000.

19. Barbeau, M. and E. Kranakis, *Principles of ad hoc networking*. 2007: John Wiley and Sons.

20. *Network simulator, ns-2.*; Available from: http://www.isi.edu/nsnam/ns/.

21. Fall, K. and K. Varashan, *The ns-2 manual.* 2003.

22. Bernat, F.A., *Simulation of Ant Routing Protocol for Ad-hoc networks in NS-2*. 2006, Delft University of Technology.

23. Laxmi, V., L. Jain, and M.S. Gaur, *Ant Colony Optimisation Based Routing on NS-2*, in *International Conference on Wireless Communication and Sensor Networks (WCSN)*. 2006: India.

24. Pan, Y., *Design Routing Protocol Performance Comparison in NS2: AODV comparing to DSR as Example*.

25. Ros, F.J. and P.M. Ruiz, *Implementing a new manet unicast routing protocol in NS2*. December 2004, University of Murcia.

26. Johnson, D.B. and D.A. Maltz. *Dynamic source routing in ad hoc wireless networks*. in *Mobile Computing*. 1996: Kluwer Academic Publishers.

27. Bertsekas, D., P., and R. Gallager, *Data Networks (2. ed.)*. 2 ed. 1992: Prentice Hall.

28. *Inc OPNET Technologies. OPNET wireless network modeling and simulation. Software Package.*; Available from: http://www.opnet.com/.

29. *Scalable Network Topologies. QualNet network simulator. Software Package*. Available from: http://www.scalable-networks.com/.