Investigation of NS-2 Movement File Generation for a Curved Path

by

Hawra Alseef

Bachelor of Computer Science
King Faisal University, 2007

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor(s): Przemyslaw R. Pochec, PhD, Computer Science
John DeDourek, Prof., Computer Science

Examining Board: Rodney Cooper, Prof., Computer Science, Chair
Joseph Horton, PhD, Computer Science

This report is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

July, 2013

©Hawra Alseef, 2013
Abstract

MANET is a decentralized wireless network, having no fixed infrastructure or designated routers. In MANT, mobile nodes move dynamically and follow tracks that determine the mobility pattern in the network. Mobility of mobile nodes significantly affects MANET performance. The simulation of large mobile networks depends on movement generation, such as the Setdest utility in NS-2. In this research, we investigate the standard Setdest generator that specifies the motions along straight-line paths. We propose and implement a new method for movement generation in the NS-2 that specifies the motions along curved paths. The new generator was successfully tested with the NS-2 simulator. From the results we conclude that only the node speed significantly affects MANET performance, and not the shape of the path taken by a node.
Dedication

I would like to dedicate this work to:

My mother, “Seedigah”. Mummy you inspired my whole life; you are the only school that I will never graduate from. God bless you.

My father, “Bader”, there is no doubt in my mind that without your continued support and counsel I could not have achieved this level of education. Daddy you are the greatest man in my life. God bless you.

My oldest sister, “Dr. Nahj”, for her understanding and endless loves through the duration of my studies and for her support and encouragement throughout my life as well as the lives of my other brothers and sisters (Mustafa, Murtada, Dr. Gufran, Dr. Duua, and Sajedah).

To all those special and precious people in my life, I could not have completed this effort without their assistance and enthusiasm. Thank you all, I am very fortunate to have you in my life.
Acknowledgements

Indeed, all praise and thanks due to Allah for the continuous support and guidance he has bestowed on me throughout my entire life and my studies. I would like to express my heart-felt gratitude to every member of my family, for their great support and encouragement, particularly my parents, who endured my study abroad.

I am truly indebted and thankful for the tremendous support from the individuals who helped and guided me throughout my temporary residency and study in Canada.

I am sincerely and heartily grateful to my supervisors Prof. John DeDourek and Prof. Przemyslaw Pochec for their motivations, suggestive views, great knowledge, and patience during the period of working with them.

Last but not least, I dedicate this work to the Ministry of Higher Education of Saudi Arabia through the Saudi Arabian Cultural Bureau in Canada for their sponsorship.
Table of Contents

Abstract ii
Dedication iii
Acknowledgments v
Table of Contents ix
List of Tables x
List of Figures xii
Abbreviations xiii

1 Introduction 1
  1.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Objectives and Motivations . . . . . . . . . . . . . . . . . . . 2
  1.3 Project Description . . . . . . . . . . . . . . . . . . . . . . . . 3
  1.4 Organization of the Report . . . . . . . . . . . . . . . . . . . . 4

2 Background 6
2.1 Networking ................................................. 6
  2.1.1 Wired Networks ..................................... 7
  2.1.2 Wireless Networks ................................. 7

2.2 Mobile Devices Network .............................. 8
  2.2.1 Wireless Mesh Network (WMN) ................. 10
  2.2.2 Wireless Sensor Networks (WSN) ............... 10
  2.2.3 Mobile Ad-Hoc Networks ......................... 11
    2.2.3.1 Routing Protocols in MANET ............... 12

2.3 Movement Patterns in MANET ....................... 19
  2.3.1 Deterministic Movement Pattern ............... 19
  2.3.2 Semi-Deterministic Movement Pattern ........... 20
  2.3.3 Random Movement Pattern ....................... 21
  2.3.4 Example of Networking Research on the Impact of Curved Paths .......................... 21

2.4 Investigating Network Performance .................. 22
  2.4.1 Empirical Study .................................. 23
  2.4.2 Analytical Modeling .............................. 23
  2.4.3 Simulation ....................................... 23

3  Network Simulator-2 ................................ 25
  3.1 Network Simulator-2 ................................. 25
  3.2 Main Features of NS-2 .............................. 26
  3.3 NS-2 Limitations .................................. 27
### Design of the New Movement File Generator

#### 4.1 Fractals in Nature
- 4.1.1 The History of Fractals
- 4.1.2 The Characteristics of a Fractal
- 4.1.3 Examples of Fractals Found in Nature

#### 4.2 The Von Koch Curve
- 4.2.1 The Construction of the Koch Fractal
- 4.2.2 Properties of The Koch Fractal

#### 4.3 Outline of the Implementation of the New Movement Generator

#### 4.4 Fractal Transformation

#### 4.5 Fractal Movement Modifications

#### 4.6 Some Special Cases of the Fractal Transformation

#### 4.7 Fractal Movements Restrictions

#### 4.8 Visualization Tool

### Investigation of the New Movement File Generator

#### 5.1 Performance Metrics
- 5.1.1 Network Throughput
- 5.1.2 Delivery Ratio
- 5.1.3 Packet Loss

#### 5.2 Simulation Environment
5.2.1 Basic simulation parameters of TCL Script . . . . . . . 59
5.2.2 Experiment Set up for Traffic Connection . . . . . . . 60
5.3 Measuring Performance in the Simulation Experiment . . . . 61
5.4 Analysis and Results . . . . . . . . . . . . . . . . . . . . . . 64
  5.4.1 Base case scenario Setdest . . . . . . . . . . . . . . . . 64
  5.4.2 Fractal Motion Scenario . . . . . . . . . . . . . . . . . . 65
  5.4.3 Comparing the Fractal and Linear Motions at Low Speed 66
  5.4.4 Comparing the Fractal and Linear Motions at High Speed 69
  5.4.5 Comparing the Linear Motion at High and Low Speed 70
  5.4.6 Comparing the Fractal Motion at High and Low Speed 72

6 Conclusion and Recommendation 74
  6.1 Result of the Research . . . . . . . . . . . . . . . . . . . . . 75
  6.2 Recommendations for Future Research . . . . . . . . . . . . 76

Bibliography 77

Vita
### List of Tables

3.1 Network Simulator-2 Models [31] .......................... 28

5.1 The basic settings of NS-2 simulation ................... 60

5.2 Experiment set up ......................................... 60
# List of Figures

- 2.1 Mobile devices cooperation [19] ........................................ 8
- 2.3 A network with A, B, and C hosts [22] ............................... 14
- 2.4 DSR algorithm routing process [23] ................................. 16
- 2.5 The urban traffic model [28] ............................................ 20
- 2.6 A mobile node following the column model [28] ............... 21
- 2.7 A node in random motion [28] .......................................... 22
- 3.1 Simplified user’s view of NS-2 [30] ................................. 27
- 3.2 Sample of an original movement file ................................. 30
- 3.3 The probability distribution for the random waypoint mobility model [33] ................................................................. 32
- 4.1 Step 1 of Koch fractal construction ................................. 37
- 4.2 Step 2 of Koch fractal construction ................................. 37
- 4.3 Step 3 of Koch fractal construction ................................. 37
- 4.4 Converting of a single movement to 4 movements ........... 41
- 4.5 A single movement divided to 4-segments .......................... 42
4.6 Sample of an original movement file .......................... 47
4.7 Sample of the fractal movement file .......................... 48
4.8 Sample Of original movement file forms a triangle ......... 50
4.9 The fractal movements after 3 transformations ............. 51
4.10 First time transformation ..................................... 52
4.11 5-Times transformation ...................................... 52
4.12 Sample of original movement file forming a triangle ...... 54
4.13 Sample of fractal movement file having a value out of move-
ment area boundary ............................................ 54

5.1 Screen shot of NAM Animation with two stationary nodes 23
and 24 .......................................................... 62
5.2 Linear Motion Result ......................................... 64
5.3 Fractal Motion Result .......................................... 65
5.4 Comparing the Linear and Fractal Motions at Low Speed .. 67
5.5 Differences between Fractal and Linear Motions at Low Speed 68
5.6 Averages of Linear and Fractal Motions at Low Speed ...... 68
5.7 Comparing the Linear and Fractal Motions at High Speed .. 69
5.8 Differences between Fractal and Linear Motions at High Speed 70
5.9 Comparing Linear Motion at High and Low Speed .......... 71
5.10 Averages of Linear Motion at Low and High Speed ........ 72
5.11 Comparing Fractal Motion at High and Low Speed .......... 73
**List of Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AODV</td>
<td>Ad-hoc On-Demand Distance Vector</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CBQ</td>
<td>Class-Based Queuing</td>
</tr>
<tr>
<td>CIDR</td>
<td>Classless Inter-Domain Routing</td>
</tr>
<tr>
<td>DSDV</td>
<td>Destination Sequence Distance Vector</td>
</tr>
<tr>
<td>DSR</td>
<td>Dynamic Source Routing</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad-Hoc Network</td>
</tr>
<tr>
<td>NAM</td>
<td>Network Animation Tool</td>
</tr>
<tr>
<td>NS</td>
<td>Network Simulator</td>
</tr>
<tr>
<td>OTcl</td>
<td>Object Tool Command Language</td>
</tr>
<tr>
<td>RED</td>
<td>Random Early Detection</td>
</tr>
<tr>
<td>RWPM</td>
<td>Random Way Point Model</td>
</tr>
<tr>
<td>TCL</td>
<td>Tool Command Language</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>Th</td>
<td>Throughput</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VBR</td>
<td>Variable Bit Rate</td>
</tr>
<tr>
<td>WPAN</td>
<td>Wireless Personal Area Network</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WMAN</td>
<td>Wireless Metropolitan Area Network</td>
</tr>
<tr>
<td>WMN</td>
<td>Wireless Mesh Network</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter presents an introduction of the research in section 1.1. Also, it states the objectives and motivations of this research in section 1.2. More precisely, section 1.3 specifies the project description. Finally, section 1.4 presents the organization of the report.

1.1 Introduction

A Mobile Ad-hoc Network (MANET) has become a focus of increasing interest for many researchers in the area of wireless networking. MANET is a decentralized wireless network, having no fixed infrastructure or designated routers, where mobile nodes communicate with each other via dynamically set up connections between the nodes. The mobile nodes move dynamically and follow tracks that determine the mobility pattern in the network. Mobil-
ity of mobile nodes significantly affects the performance of a MANET. There are many mobility patterns for mobile ad-hoc networks [1][2]. The standard way used in MANET simulation for movement generation for popular mobility models is a movement along a straight line between the current location and the designated destination point.

NS-2 is a discrete event simulator used in networking research [4]. None of the existing mobility models support movement along curved paths for mobile nodes in NS-2 simulation. In this research, I propose a new way of modeling motion in wireless network simulation, specifically of generating movement files for NS-2 simulation that specify the motion along curved paths. The new tool is then used for experimenting with different movement patterns and comparing their impact on packet delivery.

1.2 Objectives and Motivations

Controlling the motion of mobile devices is one concern that MANET researchers seek to accomplish. Generally, the node mobility significantly impacts connectivity among the nodes and affects the performance of networks resulting in a non-deterministic nature of the network topology in ad-hoc networks. NS-2 can represent different scenarios and models that include wired and wireless networks. It simulates the networks behavior and provides a trace file that can be examined to evaluate the performance. Also, NS-2 is able to simulate the node movement by using Setdest utility that
automatically generates the movement file of a large network simulation. This tool is used in MANET simulation for movement generation of simple mobility models that specifies a movement along a straight line between a starting point and a destination. This research objective is to model the motion of mobile networks in more complex way that specifies the motion along a curved path. I intend to develop an appropriate simulation method where mobile node movement is specified to form a fractal path from the original starting point to the destination that are generated dynamically by the Setdest utility. Indeed, this research aims at the investigation of a way that manipulates a NS-2 movement file in deterministic way. In other words, we are to investigate the following question experimentally: Is a straight-line path always optimal for modeling motion in MANET simulation? Is a fractal path for modeling motion in MANET better or worse than a straight-line path? At the end, simulation outcomes of comparison between the original (straight line) movement file generated by the Setdest utility and the new (fractal) movement file are shown.

1.3 Project Description

The main objectives for this report are:

1. Implementing a different movement pattern that specifies movements along curved paths.

2. Developing a visualization tool that plots the movements of mobile
nodes on the screen.

3. Comparing the performance of the new movement generator of curved paths with the Setdest movement generator as a base case under the packet delivery metric.

1.4 Organization of the Report

- Chapter 2 introduces a brief background about networking and provides a description of some types of wireless networks. Additionally, different movement patterns used in MANET are introduced and followed by a section describing different ways of investigating network performance.

- Chapter 3 discusses the Network Simulator-2 as the simulator used for this research. It presents different aspects related to NS-2, such as features, models, limitations, and Setdest utility of NS-2. Also, the Random Waypoint Mobility Model algorithm that NS-2 used is discussed briefly, as well as how to interpret a movement file generated by the Setdest utility that specifies the motion along straight line.

- Chapter 4 mainly introduces the implementation of a new movement file generator that specifies the motion along a curved path. Also, the Von Koch fractal, which is the selected curve for our implementation is described. Then the fractal transformation is presented, which explains the methodology used in reaching the aim of the research, which is
converting the motion along a straight line to a curved path. Some fractal movement modifications that are required for the transformation as well as some restrictions are stated in this chapter. Finally, the visualization tool that is developed to plot the movement on the screen is described.

- Chapter 5 presents the new movement generator of curved-path motion used to investigate the effect on network performance when changing the shape of individual movements in mobile networks. Also, some performance metrics are introduced and followed by the description of the simulation environment used for our experiment analysis. A comparison between the base case that is the Setdest utility and the new fractal movement generator is presented. Finally, the results are evaluated in this chapter.

- Chapter 6 represents a summary of the observations from this research and some recommendations for the future researches
Chapter 2

Background

This chapter introduces a brief background about networking in section 2.1 and provides a description of some types of wireless networks in section 2.2, including extra details of the common sample routing protocols used in MANET. These protocols are DSR and AODV. Additionally, different movement patterns used in MANET are introduced in section 2.3, and different ways of investigating network performance are described in section 2.4.

2.1 Networking

A network refers to a collection of computers and other hardware interconnected with each other for the purpose of sharing data and information. Networks can be classified based on which hardware and associated software
technology is used among the different devices in the network. There are two main categories of communication networks, which are wired and wireless [3].

2.1.1 Wired Networks

This type of network connects a number of devices connected via cables and routers to allow transferring data between two or more applications or computers. Wired networks have a limited mobility due to the existence of cables and routers.

2.1.2 Wireless Networks

The other type of communication network is wireless. Wireless networks are designed to overcome the obstacles of wired networks such as the limitation of mobility by the elimination of cables. In wireless networks, computers are connected with each other wirelessly and more easily. Wireless networks employ electromagnetic (radio) waves to maintain communication channels between devices, such as laptops, mobile phones, etc. These devices can, for example, connect to the web via a Wi-Fi hotspot. Consequently, wireless is gaining increasing popularity in the area of networking, and its techniques are improving rapidly. There are some advantages that can be obtained by using wireless networks. For instance, convenience of access, increased productivity, effective setup installation, and increased security. Technologically, there
are six types of wireless networks. These types are wireless personal area networks (WPAN), wireless local area networks (WLAN), wireless metropolitan area networks (WMAN), wireless mesh networks, wireless wide area networks, and finally, mobile devices networks which will be discussed in more detail [4].

2.2 Mobile Devices Network

![Figure 2.1: Mobile devices cooperation](image)

The phone has become the most effective communication tool available today and one of the necessities in our lives. Cellular network is a radio wave transmission network that can be installed over a territory using multiple cells. Each cell uses various frequencies, which are different from those of others cells in order to increase both coverage and capacity. In addition, these cells
are connected to one or more fixed-location transceivers called base stations. A base station is a wireless communications station that handles all communications. Through the base station the portable transceivers communicate with each other as well as with fixed transceivers and telephones anywhere in the network. There are several benefits associated with mobile networks, such as more flexibility, lower transmission power, larger coverage area, and increased capacity [5][6][19].

Many current and future applications of wireless communication networks rely on a type of wireless network called ad-hoc. Ad-hoc networks are developed to meet the many needs of wireless network users. Particularly, the most common use is to connect devices that must frequently change locations. Moreover, one of the main advantages of an ad-hoc network is that it overrides the need for a wireless router. Ad-hoc networks are networks that do not rely on a pre-existing infrastructure in which the network nodes are formed to maintain a dynamic interconnection topology between mobile devices. More precisely, the path of transferring data from the users node to the destination node is provided dynamically by other users devices acting as routers [7][18]. There are several features that tend to favour ad-hoc networks. For instance, they are independent from central network administration, self-configuring, self-healing where the network will update its configuration and identify new paths for connection if any changes occur in the network, and scalable to accommodate additional nodes within the connection range [9]. Furthermore, ad-hoc networks can provide access to the Internet from many different lo-
cations. There are three main applications associated with wireless ad-hoc networks. These applications are wireless mesh networks (WMN), wireless sensor networks (WSN), and mobile ad-hoc networks (MANET)[7].

2.2.1 Wireless Mesh Network (WMN)

A wireless mesh network is a fully connected network, which allows a number of interconnections between nodes throughout the network. The network can use either flooding or routing propagation techniques. A wireless mesh network has various characteristics. One of them is that WMN is a self-configuring network where adding, removing, and moving nodes can be easily and immediately managed. This characteristic adds another feature to the mesh network, namely self-healing. If a node loses connectivity, the network automatically finds the fastest and most reliable path to send data. Moreover, transferring data in wireless mesh networks is de-centralized; data can travel between nodes without being sent back to a central server. Also, it is worth pointing out that a wireless mesh network provides a high level of privacy and security [10][11].

2.2.2 Wireless Sensor Networks (WSN)

A wireless sensor network is a collection of nodes connected to each other in a network where each node is connected to one or more sensors in addition to a microcontroller, wireless transceiver, and energy source. Basically, wireless
sensor networking can be used for monitoring weather conditions, industrial processes, and medical devices, as well as in military applications. There are many benefits associated with wireless sensor networks, such as less power consumption, inexpensive components, and stability of the coverage area can be easily adapted to any expansion or reduction in the network. Ad-hoc networks also have some disadvantages including an insufficient speed of communication and less security [12][20].

Figure 2.2: Wireless Sensor Network [20]

2.2.3 Mobile Ad-Hoc Networks

The Mobile Ad-Hoc Network (MANET) is a collection of mobile nodes that communicate with each other wirelessly and without relying on fixed infrastructure. The interconnections between nodes change constantly where all the connecting nodes are free to move dynamically within the coverage range
of the wireless network. Each node works as a host or a router where it plays a major role in transferring data from a source to a destination point. As MANET is a subcategory of ad-hoc network, it also has the most notable feature that facilitates the mobility among the connecting nodes in the network. Without a doubt, nodes mobility has an impact on the network where it affects the network performance positively or negatively [1]. Here are five examples of mobility models in MANET. The first type is the Exponential Correlated Random Mobility Model. The mobility of this model is based on motion equations where there is a given position of a node or group of nodes at a specific time. The second type is the Column Mobility Model. Nodes mobility of this model follows a given line where each node is allowed to follow the other in a specific direction. The third type is the Nomadic Community Mobility Model. The node mobility of this model formed a group that follows specific realistic scenarios. The fourth type is the Pursue Mobility Model. Also the mobility of this model is formed by a group of mobile nodes that track a target based on the acceleration function to pursue the target appropriately. The fifth type is the Reference Point Group Mobility Model. This model is formed by a group of mobile nodes that has a leader to control the motion [13][17].

2.2.3.1 Routing Protocols in MANET

The infrastructureless and dynamic nature of the Mobile Ad-Hoc networks provide efficient end-to-end communication where we can employ, for exam-
ple, the traditional TCP/IP structure that supports end-to-end communication between nodes. Indeed, each node in mobile Ad-hoc networks communicates with each other using only their sole transmitter-receiver. Therefore, every node voluntarily participates in transmitting the packet that flow to and from different nodes. Each node follows a routing algorithm to route different packets. Routing is the process of exchanging information between different hosts in a network. In other words, routing is the mechanism of transmitting a packet towards its destination by determining the most efficient path. The measurement of an efficient path can be based on various metrics, such as the number of hops, traffic, security, etc. [23]. Routing plays a major role in network operation. In ad-hoc networks, the necessity of efficient routing protocols is very critical due to frequently changing network topology in this type of wireless network. If the network contains only two hosts and they are located closely to each other, there would be no need for a routing protocol. Indeed, in many ad-hoc networks, two hosts may not be able to communicate directly because they are not within wireless transmission range of each other. In this case, they could communicate if other hosts between them in the same ad-hoc network would voluntarily participate to forward packets for them [21][22].

For example, in the network illustrated in Figure 2.3, A and C cannot exchange packets between them because mobile host C is not within the range of host A’s wireless transmitter and host A is not within the range of host C’s wireless transmitter. Therefore, host B would participate to
forward packets for them where B is within the overlap between A’s range and C’s range. In fact, in a real ad-hoc network, the routing problem is more complicated due to the inherent non-uniform propagation characteristics of wireless transmissions as well as the possibility of any hosts involved to move at any time [22].

Figure 2.3: A network with A, B, and C hosts [22]

As MANET is a self-configuring network, each host node acts as specialized router itself. A number of routing protocols have been proposed for MANET, two of them are prominent: the Ad-hoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR) [23].

2.2.3.1.1 Dynamic Source Routing Protocol (DSR)

DSR is an on-demand source routing protocol, which is a type of reactive protocol that forms a route on-demand when transmitting packets between nodes in an ad-hoc network. If a node wants to communicate with another
node to which it has an unknown route, this type of routing protocol will try
to establish such a route. In other words, every node in this routing protocol
will be able to communicate by maintaining information of only active routes
to the destination nodes. In the DSR communication, the route path to the
destination will be discovered after the source node sends the first packet.
The routing process of DSR can be implemented in two phases: the first
phase is route discovery and the second is route maintenance. In the route
discovery phase, the completed route from the source to destination should
be determined first by the sender itself and included in the packet [22]. The
process of determining the source route requires piling up the addresses of all
intermediate nodes between the source and destination. As it is illustrated
in figure 2.4, X, Y, Z, V, and W form ad-hoc network where X is the source
node and Z is the destination node. In the phase of route discovery, X trans-
mits a Route Request Packet with the address of destination node Z. All the
intermediate nodes V, W, and Y receive the Route Request Packet from X.
Receiving nodes V, W, and Y append their own address to the Route Re-
quest Packet and transmit the packet further. Finally, the destination node
Z receives the Route Request packet. The address information of all interme-
diate nodes on the path from the source node X to the destination node Z are
now included in the Route Request packet. One of the most notable features
of DSR is that the route discovery phase is based on determining the most
viable route. The route maintenance function is initiated by discovering the
errors at a node on the route, and is important for monitoring active routes.
The detected hop will be removed from the node’s route cache; all routes containing the hop are felled at that point [21][23][25].

![Figure 2.4: DSR algorithm routing process](image)

**Advantages of Dynamic Source Routing Protocol**

Some of the advantages of DSR are listed below [23][24].

- Routes in DSR are discovered on an on-demand basis.
- DSR provides multiple routes to destination.
- DSR is a reactive protocol that eliminates the need to periodically flood the network; therefore, nodes can enter sleep mode in order to conserve their power.
- DSR is a bandwidth-efficient protocol.
- DSR is an adaptive protocol for any topological changes caused by node mobility.
DSR requires less network communication overhead (Promiscuous overhearing).

Disadvantages of Dynamic Source Routing Protocol
There are some disadvantages of DSR making it not efficient for large networks [23][24].

- DSR has a high response time (Large delays).
- DSR has scalability problems due to source routing and flooding.

2.2.3.1.2 Ad Hoc On-Demand Distance Vector (AODV)
The second routing protocol for mobile ad-hoc networks is AODV. AODV is a reactive routing protocol that only establishes a route to a destination on demand. Both DSR and AODV are routing protocols for MANET but each employs different techniques that significantly affect the networks performance [24]. DSR and AODV can be compared and evaluated based on various metrics, such as the packet delivery ratio, normalized MAC load, normalized routing load, and average end-to-end delay by altering the number of sources, rate of speed, and pause time [39]. However, the main difference between DSR and AODV arises in the source routing feature where the source routing is a mechanism in which the sender identifies the route (from the source to destination) in the packets header so that the intermediate node can be identified by the address on the way to the destination node [27]. In the AODV protocol, when a node needs to start a connection, first it broad-
casts a request for a connection. Other nodes rebroadcast this message and record the node that they received an answer from. Then, they establish a reverse route to the sender node. When a node receives such a message and already has a route to the destination node, it sends a message backwards through a reverse route to the requesting node. The sender node then begins using the route that has the least number of hops through other nodes [26].

In conclusion, there are some characteristics that tend to favour the AODV protocol. One of them is that AODV shares many prominent features of DSR, such as on-demand route discovery, route maintenance, and hop-by-hop routing of Destination Sequence Distance Vector (DSDV) algorithms. Therefore, MANETs takes advantages of obtaining the routes on-demand and uses the AODV protocol [4].

Advantages of AODV Routing Protocol

Some of the advantages of AODV are listed below [26][27].

- AODV has better performance than DSR in higher-mobility scenarios.
- AODV provides quick convergence when the networks topology is changed.
- AODV provides higher reliability of data delivery.

Disadvantages of AODV Routing Protocol

Some of the disadvantages of AODV are listed below [26][27].

- AODV requires more time to establish a connection.
• AODV has heavy overhead control over the network.

• AODV does not have an efficient route maintenance technique

2.3 Movement Patterns in MANET

There are three categories of mobility pattern in MANET. These patterns are deterministic, semi-deterministic, and random [1][28].

2.3.1 Deterministic Movement Pattern

The characteristics of the deterministic mobility pattern are specifically determined. More precisely, the movement is formed based on several parameters, such as the source, destination, direction, and velocity. The deterministic mobility pattern can be classified as the most predictable type of motion as well as the most simplistic of all mobility patterns. Additionally, one more aspect of this pattern is that the deviation in the movement of the node is zero. A common example that illustrates this pattern would be cars moving in an urban traffic area, where the speed of the cars and direction are predefined and where cars move in straight lines or turn at cross lights. Below is figure 2.5 that shows the urban traffic model of this pattern.
2.3.2 Semi-Deterministic Movement Pattern

This pattern is non-random motion but the node follows a mobility pattern that is not so deterministic. This pattern is also described as a partially deterministic pattern where a given parameter or components of the node movement can be known with a certain probability, and all the other components are known deterministically. An example of this pattern is a number of birds flying in the sky together. Generally, they do not follow a specified path, but they all move in the same direction. They form a column, and they do not move out of it. This mobility pattern is known as a column model. Figure 2.6 shows a mobile node following the column model.

Figure 2.5: The urban traffic model [28]
2.3.3 Random Movement Pattern

This pattern refers to node mobility in which the motion is completely non-uniform or random motion. In other words, all the parameters that control the motion are unspecified. Furthermore, there is no max deviation which the nodes can take up for their next movement. Figure 2.7 shows a node moving in random motion.

2.3.4 Example of Networking Research on the Impact of Curved Paths

Similar research investigating the impacts of curved-path motion has been done by Yun W., Yoon Kah L., and Jun Y. in 2009 [29]. This research poses the question whether a straight-line path is always the best for intrusion detection in wireless sensor networks. This research also addresses the net-

Figure 2.6: A mobile node following the column model [28]
work configuration problem for detecting an intruder within a pre-specified time/distance threshold where the previous research assumed that the intruder usually takes a straight-line intrusion path. This research takes into account that the straight-line intrusion path is often not the case in reality. An intruder can invade the network following a curved path or even a random walk in order to improve its attacking probability. In other words, the research describes the effects of different paths of the intruder on the intrusion detection probability in an arbitrary wireless sensor network. Outcomes of their observations are shown to validate their analysis and conclusion.

2.4 Investigating Network Performance

It is preferable to evaluate the network before it is actually put to use. By evaluating the network, errors will be detected and corrected and this will

Figure 2.7: A node in random motion [28]
definitely lead to avoiding such situations in the future. In fact, there are three approaches that can be followed to investigate the network performance: empirical study, analytical modeling, and simulation [1][4].

2.4.1 Empirical Study

In this approach, the entire or a portion of the network is experimented with under a controlled environment. For this purpose, test users are needed as they behave as real users. In order to measure the performance, the network needs to be observed when it is in use. The most notable disadvantage of this approach is that it entails building the entire or a portion of the network and that might be difficult and expensive as well [1][4].

2.4.2 Analytical Modeling

Analytical modeling is a technique that uses mathematical formulae to calculate interactions between the different network entities thereby allowing us to predict the network performance. In analytical modeling, tests should be done using one-dimensional and two-dimensional topology with both static and dynamic situations [1][4].

2.4.3 Simulation

Simulation is testing or mimicking an actual behavior of a network by using a computer program. Simulation is a useful tool that allows experimentation
without exposure to risk and, where real systems are too complex to model with less cost. Furthermore, it helps determine how a real network would function and helps predict the networks behavior. Simulation includes protocol simulation, movement simulation, and traffic generators. There are several programming languages and simulators for simulating a network, such as Java, C, or NS-2 [4][14]. Network Simulator-2 is presented in the next chapter in more detail.
Chapter 3

Network Simulator-2

Chapter 3 discusses the Network Simulator-2 as it is the simulator used for this research. It presents different aspect related to NS-2, such as features, models, limitation, and the Setdest utility of NS-2 in various sections. Also, the Random Waypoint Mobility Model algorithm that NS-2 used is presented briefly, as well as how to interpret a movement file generated by the Setdest utility that specifies the motion along straight line.

3.1 Network Simulator-2

The network simulator is a powerful program for network simulation. NS-2 is the second version of NS. It is the most popular open-source based simulator that uses two languages (OTcl/C++) in order to create a simulation environment. There are some general characteristics that tend to favour the
NS-2 simulator. One of them is that both wired and wireless networks are supported in the NS-2 simulator. In addition, this simulator implements a number of network components and protocols that provide an efficient simulation environment. As a wireless network facilitates the mobility among the nodes, NS-2 provides Setdest commands that are used for mobility features whereby it depicts the movement of nodes and the process of transferring packets. Furthermore, it provides the ability for a mobile node to move at a certain time from its current location to a desired destination at a constant speed. It also has a network animation tool (NAM) that visualizes the network simulation and packet trace data where it helps in understanding the complex behavior of network simulation. The NAM trace file contains information which includes the topology set up, nodes, links, queues, node connectivity, and packet trace information. Additionally, NS-2 provides an Xgraph tool, which is used for graphical plotting. Moreover, NS-2 has pre-processing components: traffic and topology generators as well as a post-processing component which is a simple trace analysis, often in Awk, Perl (mostly), or Tcl. Therefore, NS-2 can provide insight into the operations and interactions of networking protocols [4][14][15].

3.2 Main Features of NS-2

The first and most important feature is the usage of two programming languages, which are C++ and OTcl. These two languages are powerful and
have many functional characteristics. The basic network component objects and the event scheduler are written and compiled using C++ in order to reduce packet and event processing time. C++ is responsible for implementing the detailed protocol of network simulation. OTcl works in parallel with C++ to provide an efficient simulation environment [41]. An Object Tool Command Language interpreter is used to implement a user’s command scripts. It is used to control the simulation scenario where the network structure and the topology are defined and schedule the events. The OTcl script is used to initiate the event scheduler, set up the network topology, and tell the traffic source when to start and stop sending packets through the event scheduler. A simplified user’s view of NS-2 is shown in figure 3.1. Therefore, the combination of these two languages makes NS-2 powerful [30].

![Figure 3.1: Simplified user’s view of NS-2 [30]](image)

### 3.3 NS-2 Limitations

Some of NS-2 limitations are listed below [32].
Table 3.1: Network Simulator-2 Models [31].

<table>
<thead>
<tr>
<th>Traffic models and applications</th>
<th>Web, FTP, Telnet, Constant-Bit Rate (CBR), Ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport protocols</td>
<td>Unicast: TCP (Reno, Vegas), UDP Multicast</td>
</tr>
<tr>
<td>Routing and queuing</td>
<td>Wired routing, Ad-Hoc routing</td>
</tr>
<tr>
<td>Queuing protocols</td>
<td>Random Early Drop (RED), Drop-Tail, CBQ</td>
</tr>
<tr>
<td>Physical media</td>
<td>Wired (point-to-point, LANs), wireless, satellite</td>
</tr>
</tbody>
</table>

- NS-2 does not support some protocols, for example Classless Inter-Domain Routing (CIDR) and subnetting.

- Support for Variable Bit Rate (VBR) is not provided in NS-2 where VBR is used for audio/video streaming in real networks.

- Although NS-2 supports different delays like propagation, queuing, and transmission, but it does not support processing delay.

- NS-2 has limited functions when simulating a large network; it is slow.

3.4 NS-2 Utility for Generating Node Movement

The NS-2 simulator is a robust program that has the capability to support both wired and wireless networks. Indeed, wireless networks may include a large number of devices, and their motions are mostly not stationary. Therefore, a mechanism is needed to create the motions of these mobile nodes and
evaluate the networks performance in different ways. The NS-2 simulator aims at providing a tool, Setdest that generates the movements of these mobile nodes at a certain time to specific coordinates from its current location at a specific constant speed. Setdest is a CMU’s node-movement generator that can be used to create random waypoint node movements for mobile nodes. This tool first creates the initial position of nodes within specified boundaries and then uses the Random Waypoint algorithm to generate random movements for these mobile nodes by running a command in TCL in NS-2. The Setdest tool is run with the following parameters:

```
./ setdest v 1 -n number of nodes -p pause time -M max speed 
-t simulation time -x max x -y max y > output file name.
```

The node-movement generator is a part of NS-2 installation and is usually available under ns/indep-utils/cmu-scen-gen/setdest. The movement file will be generated based on the specified parameters in the command line where n is the number of nodes, p is the pause time, M is the Maximum speed of the nodes, t is the simulation time, and X and Y are the maximum values of the allowed node movement rectangle area. This tool generates the initial position of mobile nodes in a two-dimensional area.

For example, executing

```
./ setdest v 1 -n 2 -p 3.00 -M 1000 -t 500 -x 300 -y 300
> output-file
```

29
gives the movement file shown in figure 3.2.

```
#
# nodes: 2, pause: 3.00, max speed: 1000.00, max x: 300.00, max y: 300.00
#
$node_(0) set X_ 50.00000000000
$node_(0) set Y_ 50.00000000000
$node_(0) set Z_ 0.00000000000
$node_(1) set X_ 150.00000000000
$node_(1) set Y_ 100.00000000000
$node_(1) set Z_ 0.00000000000

$ns_ at 0.100000 "$node_(0) setdest 450.00000000000 50.00000000000 1000.00000000"
$ns_ at 0.600000 "$node_(0) setdest 250.00000000000 450.00000000000 1000.00000000"
$ns_ at 1.300000 "$node_(0) setdest 50.00000000000 50.00000000000 1000.00000000"
```

Figure 3.2: Sample of an original movement file

The sample movement file in figure 3.2 has the following Setdest command:

```
$ns_ at 0.100000 "$node_(0) setdest 450.00000000000 50.00000000000 1000.00000000"
```

This line specifies that at time 0.100000s, node0 starts to move from its current location towards the destination (450,50) at a speed of 1000m/s.

### 3.5 The Random Waypoint Mobility Model Algorithm (RWPM)

There are different types of random mobility models that are used in MANET. Each model follows a certain algorithm in which the nodes move. These
models are Random Walk, Random Waypoint, and Random Direction. Set-
dest utility generates the node movements following the Waypoint algorithm,
which will be discussed in detail.

The way nodes move from their starting position towards the destination
at the Waypoint algorithm is randomly generated with random direction and
velocity. When the simulation begins, each node selects one point within the
simulation area randomly, as a target point. Nodes start to travel towards
the destination and pause for some time after they reach that target point,
and then start moving randomly in the ranges of velocity between min-speed
and max-speed towards a newly chosen destination within the movement area
dimension. This process will be repeated until the end of the simulation by
the NS-2. One notable aspect of the Waypoint algorithm is that the random
node movements appear to have a probability distribution concentrated in the
centre regions of the network [33]. Figure 3.3 shows the resulting probability
distribution for the random waypoint mobility model in an experiment.
Figure 3.3: The probability distribution for the random waypoint mobility model [33]
Chapter 4

Design of the New Movement

File Generator

This chapter introduces the implementation of a new movement file generator that specifies the motion along a curved path. It presents a brief background of fractal in nature in section 4.1. Also, the Von Koch fractal, which is the selected curve for our investigation is described in section 4.2 and followed by the description of the implementation in section 4.3. Fractal transformation is presented in section 4.4 that concludes explaining the methodology used in reaching the aim of the research by converting the motion along a straight line to a curved path. Some fractal movement modifications are required for the transformation, and are introduced in section 4.5. Also, some special cases of fractal transformation are discussed and illustrated in section 4.6 as well as some restrictions are stated in section 4.7. Finally, the visualization
tool that is developed to plot the movement in screen is described in section 4.8 of this chapter.

4.1 Fractals in Nature

The term fractal was invented by mathematician Benoit Mandelbrot in 1975. It refers to the Latin word fract meaning "broken". A fractal is an object that displays self-similarity, on all scales. Self-similar typically means the object that has the same structure when viewed from near as from far. In other words, fractals are the same at different scales [34][35].

4.1.1 The History of Fractals

The historical path of fractals is outlined in the following [34]:

- The term "fractional exponents" was used for the first time in the 17th century when the mathematician Gottfried Leibniz made clear the importance of paying attention to the notion of recursion and the property of self-similarity. It is worth mentioning that "Geometry" did not yet know about fractals at that time.

- In 1872, Karl Weierstrass was the first to present the definition of a graph that is based on a particular function, nowadays known as fractals. These graphs represent the feature of being everywhere continuous but nowhere differentiable.
• In 1883, Georg Cantor introduced the Cantor sets, which are examples of subsets of the real line that have unusual properties and are now considered fractals.

• In the last part of that century, Felix Klein and Henri Poincare discovered a number of fractal patterns that are known as "self-inverse" fractals.

• In 1904, Helge Von Koch, introduced the famous fractal, the Von Koch curve. This fractal will be implemented for this research.

• Many fractal curves have been described later by different researchers.

4.1.2 The Characteristics of a Fractal

Some of the fractal characteristics are listed below [34].

1. Self-similarity.

2. Recursion.

4.1.3 Examples of Fractals Found in Nature

One of the things that tend to favor fractals is their ubiquity. Fractals are not limited to geometric patterns, but can also be found anywhere in nature. A number of natural fractals are listed below [43]:

• Trees and Leaves

• Rivers and Mountain ranges
• Snowflakes
• Crystals
• Clouds
• Stalagmites and Stalactites
• Lightning
• DNA and Germs

4.2 The Von Koch Curve

This curve is named after the Swedish mathematician Helge Von Koch who first designed them in 1904. He used the Koch Snowflake to prove that it is possible to have curves that are continuous everywhere but differentiable nowhere. In fact, these Koch curves are amongst the most important fractal objects that allow numerous variations and have inspired many artists to produce amazing pieces of art [36].

4.2.1 The Construction of the Koch Fractal

The construction of this curve is quite simple. It basically starts with a straight line that will be converted to the Koch curve.

• The straight line is divided up into three equal segments. The middle segment is removed and replaced by two segments having the same
length to generate an equilateral triangle, leaving us with the first iteration of the Koch Curve. Applying such a 4-sides generator to a straight line leads to the shape in figure 4.2.

- This process is then repeated for the 4 segments generated at the first iteration, leading to the following drawing in figure 4.3 for the second iteration of the building process.

- Applying the steps of the Koch curve will generate The Koch Snowflake, which is an example of a fractal that is self-similar, as it looks the same

Figure 4.1: Step 1 of Koch fractal construction

Figure 4.2: Step 2 of Koch fractal construction

Figure 4.3: Step 3 of Koch fractal construction
4.2.2 Properties of The Koch Fractal

Some of the properties of the Koch fractal are listed below [36].

1. Number of Sides (n)

For each iteration, every segment of the figure from the previous iteration will be converted to four segments in the following iteration. Since we begin with three sides, the formula for the number of sides in the Koch curve is

\[ n = 3 \times 4^a. \]  

(4.1)

Where \( a \) indicates the number of iteration. For iterations 0, 1, 2, and 3, the numbers of sides are 3, 12, 48, and 192 respectively.

2. Length of Sides (L)

In every iteration, the length of a side is \( \frac{1}{3} \) the length of a side from the previous iteration. If we begin with an equilateral triangle with side length \( x \), then the length of a side in iteration \( a \) is

\[ L = x \times 3^{-a}. \]  

(4.2)

For iterations 0 to 3, length = a, a/3, a/9, and a/27.
3. Perimeter (p)

The key features of the Koch curve lies in having the same length of all sides in each iteration, this leads to a perimeter, which is simply the number of sides multiplied by the length of a side

\[ p = n \times length. \]  \hspace{1cm} (4.3)

From the previous formula, we get

\[ p = (3 \times 4^a) \times (x \times 3^{-a}). \]  \hspace{1cm} (4.4)

In the same manner, for the first 4 iterations (0 to 3) the perimeter is 3a, 4a, 16a/3, and 64a/9.

We notice that, the perimeter increases by 4/3 times for each iteration, so we can rewrite the formula as

\[ p = (3a) \times \left(\frac{4}{3}\right)^a. \]  \hspace{1cm} (4.5)

4.3 Outline of the Implementation of the New Movement Generator

The main objective of this research is to implement a new method for movement generation in MANET simulation in NS-2. Indeed, the standard way for movement generation is to use the Setdest utility that generates a set of
Setdest commands that are then "executed" in the NS-2 simulator. Setdest commands generate a movement along a straight line between the current location and the designated destination point. This research aims at providing a new tool for modifying the simulation environment by modeling motion in wireless network simulations, specifically for generating movement files for NS-2 simulation that specify the motion along curved (fractal) paths. Typically, defining the node movements needs to be done ahead of the NS-2 simulation. In general a curved path can be approximated by a series of short line segments, which determine the final shape of the curve. Therefore, a Java program was implemented that reads the movement file with random movements generated, for example, by the Setdest utility. Then, as each movement in the movement file is specified by a separate Setdest command, we will replace each one of these Setdest commands, each specifying a movement along a straight line, with a series of Setdest commands specifying the movement along a curved path (fractal) as is illustrated in the coding example in figure 4.4. Once the new movement file is generated the NS-2 simulation can proceed in a standard way.
The sample movement file (movement) in figure 4.4 has the following movement statement:

\$ns_ \text{ at } 0.4000 \ "\text{node}_\text{(0)} \text{ setdest 100.00000 400.00000 1000.0252}\"

This line specifies that at time 0.40000s, node0 starts to move from the starting point (100,100) towards the destination (100,400) at a speed of 1000m/s (this can be one single random movement in a straight line). This single command in the movement file is then replaced by four new commands generating the movement along the path corresponding to the shape of the Koch fractal. The four movements are listed below:

\$ns_ \text{ at } 0.4000 \ "\text{node}_\text{(0)} \text{ setdest 100.00000 200.000 1333.36698}\"
$ns_ at 0.4749 "$node_(0) setdest 13.397459 250.000 1333.36698"
$ns_ at 0.5499 "$node_(0) setdest 100.00000 300.000 1333.36698"
$ns_ at 0.6249 "$node_(0) setdest 100.00000 400.000 1333.36698"

4.4 Fractal Transformation

The objective of this research to implement a Java program that employs the transformation of every single movement along a straight line between the current location and the designated destination point in the movement file generated by the Setdest utility, to a fractal movement that consists of four movements [37].

Let’s consider a single random movement from A as a starting point to B, which is the ending point. This single random movement will form the base segment which is AB, with A(Ax, Ay) and B(Bx, By), then the 4 sub-segments of Koch curve will be AP, PQ, QR, and RB as illustrated in Figure 4.5.

![Figure 4.5: A single movement divided to 4-segments](image)

In order to calculate the Q, P, and R points, we will apply the following:
First define two orthogonal vectors of same length:

\[ U(Bx - Ax, By - Ay), \]  
(4.6)

\[ V(Ay - By, Bx - Ax). \]  
(4.7)

Then the points will be calculated based on these two vectors:

\[ P = A + \left( \frac{1}{3} \right) U, \]  
(4.8)

\[ Q = A + \left( \frac{1}{2} \right) U + \left( \sqrt{\frac{3}{6}} \right) V, \]  
(4.9)

\[ R = A + \left( \frac{2}{3} \right) U. \]  
(4.10)

This way, the three points that form an equilateral triangle of the Koch curve will be calculated. Notice that the point + vector = point notation is similar to a translation.

### 4.5 Fractal Movement Modifications

In fact, applying the fractal transformations on the movement file requires some adjustments on the parameters on the Setdest commands. More precisely, the need of updating the time and speed in Setdest commands arises
when applying the fractal transformation. In order to make the fractal movements arrive at the destination at the same time that the original straight movement arrived, we need to do the following modifications.

1. Updating the Time

The four (fractal) movements should precede sequentially, each having a starting time after the previous movement ends. Let's consider the Setdest command below.

$\texttt{ns_ at T }$"$\texttt{node_# setdest X Y S}$"

(T) indicates the starting time that once the simulation time reach, the node will start moving towards the (X) and (Y) coordinates of the destination at specific speed (S). For each movement of the fractal path, we need to determine the starting time specified in the Setdest command. To this end, we need to compute the following: Calculating the Total Time of the straight line, which is

\[
\text{Total Time} = \frac{\text{Distance}}{\text{Speed}}, \quad (4.11)
\]

where

\[
\text{Distance} = \sqrt{(Bx - Ax)^2 + (By - Ay)^2}. \quad (4.12)
\]

Dividing the Total Time by 4
\[ SubTime = \frac{TotalTime}{4}. \]  

(4.13)

The fractal movements will be

\[
\begin{align*}
\text{$\texttt{ns\_ at TP "node\_(#) setdest X Y NewSpeed "}$} \\
\text{$\texttt{ns\_ at TQ "node\_(#) setdest X Y NewSpeed "}$} \\
\text{$\texttt{ns\_ at TR "node\_(#) setdest X Y NewSpeed "}$} \\
\text{$\texttt{ns\_ at TB "node\_(#) setdest X Y NewSpeed "}$}.
\end{align*}
\]

Updating the times for each movement as below:

\[
\begin{align*}
TP &= T(StartingTime), \\
TQ &= TP + SubTime, \\
TR &= TQ + SubTime, \\
TB &= TR + SubTime.
\end{align*}
\]

(4.14) \hspace{2cm} (4.15) \hspace{2cm} (4.16) \hspace{2cm} (4.17)

2. Updating the Speed

The other parameter required is to compute the total length of the fractal curve, which should be 4/3 of the straight line since all the four segments of the Koch curve have the same length. Therefore, we need to update the speed
there: the fractal speed should be four thirds of the speed in the straight line so that both movements arrive at the destination at the same time. In other words, the speed of fractal movements is 33% higher than the straight line. The formula below is used to update the speed in the Setdest command:

\[ \text{NewSpeed} = S \times \left( \frac{4}{3} \right). \]  

(4.18)

4.6 Some Special Cases of the Fractal Transformation

Indeed, the investigation of a new movement generation of fractal transformation on the movement file generated by Setdest utility imposes some constraints that need to be taken into account. The new movement generation is developed to represent the node movements along a fractal path on the original movement area boundary. The new fractal movement file captures the same properties of the original movement file. The properties are listed below:

- Nodes number
- Pause time
- Max speed
- Max X coordinate
• Max Y coordinate

• Initial Position of nodes

• Destination of node movements

Figure 4.6 is a sample of an original movement file generated by the Setdest utility and the new movement file that represents the fractal movements of the same file is shown in figure 4.7.

The new movement generator accommodates the key features of fractals, which are: Self-similarity, Recursion, and Infinity. In other words, applying the transformation on the same movement file over and over will produce more fractal movements in smaller scales where the individual movement will have the same starting and ending points but will consist of four different movements instead of one direct movement. To clarify this process, if the
movement file contains one single movement, each time we apply the fractal transformation, the number of the new movements will be multiplied by four. For iteration 1 to 4, the number of movements will be 4, 16, 64, and 256.

For example, consider the movement below as an individual movement at starting point (100,100), which is included in the movement file.

$ns_ \text{ at } 0.10765 \ "\$node_(1) setdest 500.207 \ 100.6797 \ 93.421220"$

Applying the fractal transformation for the first time on the above movement will result in:

$ns_ \text{ at } 0.107252 \ "\$node_(1) setdest 233.40200 \ 100.2265 \ 124.561"$
$ns_ \text{ at } 1.176138 \ "\$node_(1) setdest 299.90520 \ 215.8699 \ 124.561"$
$ns_ \text{ at } 2.249029 \ "\$node_(1) setdest 366.80033 \ 10.45317 \ 124.561"$

Figure 4.7: Sample of the fractal movement file
For the second time, will result in 16 different movements as below:

$ns_ at 0.90765 "$node_(0) setdest 415.470056 633.33 166.08227"
$ns_ at 1.37179 "$node_(0) setdest 415.470053 500.00 166.08227"
$ns_ at 1.83467 "$node_(0) setdest 299.999999 566.66 166.08227"
$ns_ at 2.29815 "$node_(0) setdest 300.000000 700.00 166.08227"
$ns_ at 1.97807 "$node_(0) setdest 415.470050 366.66 166.08227"
$ns_ at 2.44157 "$node_(0) setdest 300.000000 300.00 166.082271"
$ns_ at 2.90508 "$node_(0) setdest 300.000003 433.33 166.082271"
$ns_ at 3.36858 "$node_(0) setdest 415.470053 500.00 166.082271"
$ns_ at 3.04849 "$node_(0) setdest 184.529943 366.66 166.082271"
$ns_ at 3.51199 "$node_(0) setdest 184.529946 500.00 166.082271"
$ns_ at 3.97550 "$node_(0) setdest 300.000003 433.33 166.082271"
$ns_ at 4.43900 "$node_(0) setdest 300.000000 300.00 166.082271"
$ns_ at 4.11891 "$node_(0) setdest 184.529949 633.33 166.082271"
$ns_ at 4.58241 "$node_(0) setdest 300.000000 700.00 166.082271"
$ns_ at 5.04592 "$node_(0) setdest 299.999996 566.66 166.082271"
$ns_ at 5.50942 "$node_(0) setdest 184.529946 500.00 166.082271"
Repeating the transformation will result in increasing the number of movements that represent the fractal over and over. Figure 4.8 and 4.9 illustrate applying the fractal transformation on the movements that forms a triangle multiple times. One notable thing about the fractal transformation is that each time we apply the transformation, the path lengths and speed increase by 33%. As a result, after multiple transformations, the networks performance is likely to be affected by increase in the speed of node movement.

![Figure 4.8: Sample Of original movement file forms a triangle](image)

More Complex Scenario for Fractal Transformation

In some applications, moving each mode on a closed path may be required, for example in circles. Consider the case where the nodes initially move back and forth in a straight line: applying the fractal transformation will result in such a circular path (i.e. closed loop) of node movements.
For example, consider the piece of movement file that shows the initial position and movements of node (0), where it moves back and forth from (200,200) to (500,200).

$n$ode\(_{(0)}\) set \textit{X\_} 200.000000000000

$n$ode\(_{(0)}\) set \textit{Y\_} 200.000000000000

$n$ode\(_{(0)}\) set \textit{Z\_} 0.000000000000

$\text{ns\_ at } 0.100 \ "n$ode\(_{(0)}\) set\textit{dest} 500.000 200.000 \ 500.000"

$\text{ns\_ at } 0.400 \ "n$ode\(_{(0)}\) set\textit{dest} 200.000 \ 200.000 \ 500.000"

Applying the transformation for the first time, will output the figure 4.10.

Figure 4.9: The fractal movements after 3 transformations
Applying the transformation for 5 times, will output the figure 4.11.

4.7 Fractal Movements Restrictions

Applying the fractal transformation takes into account the starting point and ending point of the movement as it described in the previous section as well as the other properties. Therefore, there are some cases where the fractal transformation would not work, and they need to be identified.
1. Points Outside of The Movement Area Boundary

In some case where the original movement is located near to the movement area boundary, the fractal transformation will result in getting some points out of movement area dimensions. More precisely, the coordinates of the triangle points at the Koch fractal may have a value that are less or greater than the movement area boundary. For example, one of the movements that form a triangle at the movement file that has the movement area boundary (500,500) is located at the initial position (50,50), and moves toward the destination (50,450). Converting this movement to a fractal result in having the Q point of the triangle as a negative value.

Max :

X : 500.000000000000 Y : 500.000000000000

Starting point = X: 50 Y: 50

Ending point = X: 50 Y: 450

P : X : 50.000000000000 Y : 183.333300000000

Q : X : -65.470053837925 Y : 250.000000000000

R : X : 50.000000000000 Y : 316.666700000000

Therefore, we will not convert such a movement to a fractal and keep it as it is (movement along straight line). Figure 4.13 shows a more illustrative example of such a scenario where two of the movements were transformed into fractals while the movement near the boundary was left as a straight line.
Figure 4.12: Sample of original movement file forming a triangle

Figure 4.13: Sample of fractal movement file having a value out of movement area boundary
2. Speed Of Zero

The Setdest utility picks a random speed for each movement. In some cases, the speed chosen is zero; this happens when the current movement has the same destination that the previous movement does for a certain node. In this case, the fractal transformation will not work because updating the time and speed will have an error at speed of zero. In this case, the program will keep the movement as it is and will not convert it to a fractal.

4.8 Visualization Tool

As a part of this research we developed a tool that improves the NS-2 utilities. This tool does plotting the movements that are taking place during the simulation, where all the movements of the nodes are plotted on the screen. This tool works beside the NAM (Network Animator) tool that NS-2 provides. The difference between these two tools is that NAM uses animation to visualize the direction and the present movements of the nodes during the simulation, while the tool we developed plots the trace of the complete paths that each node followed during the simulation. This helps in understating the nodes mobility and their behavior in the network.

This tool reads the movement file as an input and extracts the starting and ending points of each node movement, store them in a list, and outputs the movements on the screen. The tool is implemented by ex-
tending the JPanel as the drawing container and overriding the PaintComponent(Graphics g) method to draw the movements by using the drawing method DrawLine that is provided by the Graphics class, it takes 4 parameters starting point’s (x, y) and ending point’s (x, y). Notice that this tool is used for plotting the node movements for both the original movement file and the new fractal movement file.
Chapter 5

Investigation of the New Movement File Generator

This chapter describes the methodologies used in order to investigate and evaluate the effects on network performance when changing the shape of individual movements in mobile networks. Performance metrics are discussed in section 5.1 and are followed by the description of the simulation environment of the experiment in section 5.2, including extra details about node configuration and traffic connection set up for our analysis. Section 5.3 presents the way we measured the network performance in the experiment. Section 5.4 includes all the analysis and evaluations of the research results. Furthermore, a comparison between the base case that is the Setdest utility and the new fractal movement generator is presented.
5.1 Performance Metrics

Several metrics can be defined to grade the performance of wireless networks. These metrics have been carefully chosen to give an idea of the behavior and reliability of the network simulations. Some performance metrics that are commonly used are listed below [4].

5.1.1 Network Throughput

Network throughput is a measurement of the amount of data transmitted from the source to the destination in a unit period of time, usually per second. Therefore, throughput is measured in total bits received per second. The throughput of a node can be measured by counting the total number of data packets successfully received at the node, and computing the number of bits received, which is finally divided by the total simulation time. In our experiments we recorded the number of packets received for a transmission between a single source and a single destination in a MANET. Therefore, the throughput can be calculated as:

\[
Th = \frac{\text{no.of BitsReceivedByNodeD}}{\text{ObservationDuration}}.
\]  

(5.1)

5.1.2 Delivery Ratio

Delivery ration is used to calculate the percentage of the transmitted data packets that are successfully received. It is an important metric, which can

58
be used to measure network performance. Typically, the delivery ratio is calculated as the percent of packets received to the packets transmitted.

5.1.3 Packet Loss

Packet loss is one performance metric that can be used to indicate the quality of service parameters of the network. It occurs when one or more packets of data transmitting across a network fail to reach their destination due to buffer overflow and congestion.

5.2 Simulation Environment

Each case of the network consists of a number of nodes roaming in a 800 x 800 meters square as movement area boundary. A higher number of nodes indicates a higher node density. Every packet has a size of 512 bytes. The buffer size at each node is 50 packets. Data packets are generated following a Constant Bit Rate (CBR) process. The experiments transmit packets from one source to one destination for every case of our strategies for 100 seconds. We ran the simulation 3 times for each node density and then we calculated the average of these runs.

5.2.1 Basic simulation parameters of TCL Script

The basic settings of NS-2 are fixed and defined for our experiment in Table 5.1 below.
Table 5.1: The basic settings of NS-2 simulation

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Type</td>
<td>Channel/Wireless channel</td>
</tr>
<tr>
<td>Network Interface Type</td>
<td>Phy/WirelessPhy</td>
</tr>
<tr>
<td>Mac Type</td>
<td>Mac 802 11</td>
</tr>
<tr>
<td>Radio-Propagation Type</td>
<td>Propagation, Two-ray ground</td>
</tr>
<tr>
<td>Interface Queue Type</td>
<td>Queue/Drop Tail</td>
</tr>
<tr>
<td>Link Layer Type</td>
<td>LL</td>
</tr>
<tr>
<td>Antenna</td>
<td>Antenna/Omni Antenna</td>
</tr>
<tr>
<td>Maximum Packet in ifq</td>
<td>50</td>
</tr>
<tr>
<td>Routing Protocols</td>
<td>AODV</td>
</tr>
</tbody>
</table>

Table 5.2: Experiment set up

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Time</td>
<td>100 sec</td>
</tr>
<tr>
<td>Node Movement Area</td>
<td>800 X 800</td>
</tr>
<tr>
<td>Number of Mobile Nodes (density)</td>
<td>5, 10, 15, 20, 25, 30, 40, 60, 80</td>
</tr>
<tr>
<td>Number of Source nodes and the position</td>
<td>One node at x=100 and y=500</td>
</tr>
<tr>
<td>Number of Destination nodes and the position</td>
<td>One node at x=700 and y=500</td>
</tr>
<tr>
<td>Traffic Type</td>
<td>(UDP) CBR</td>
</tr>
<tr>
<td>Packet Size</td>
<td>512 KB</td>
</tr>
<tr>
<td>Number of Simulation runs per experiment</td>
<td>3,10</td>
</tr>
</tbody>
</table>

### 5.2.2 Experiment Set up for Traffic Connection

For this experiment, the traffic connection is between the source and destination which are two stationary nodes having other mobile nodes between them or around. We measure how many packets are transmitted from the source and how many packets are received at the destinations. Below the
Tcl code that set up the CBR traffic connection between node (23) and node
(24).

set udp_(0) [new Agent/UDP]
$ns_attach-agent $node_(23) $udp_(0)
set null0 [new Agent/UDP]
$ns_attach-agent $node_(24) $null0
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.01
$cbr_(0) attach-agent $udp_(0)
$ns_connect $udp_(0) $null0
$ns_at 0.0 "$cbr_(0) start"

Figure 5.1 shows the NAM animation having the two stationary nodes (23) and (24).

5.3 Measuring Performance in the Simulation Experiment

The aim of this experiment is to compare the effectiveness of the new fractal
movement generator in terms of the base case movement pattern of Setdest
utility. There are two scenarios of our experiment. The first scenario uses the
original file generated by the Setdest utility. The second scenario uses the file
generated by our Java program; this file contains the fractal path movements instead of the original straight-line movements. Network simulation is then used to examine the differences in network performance caused by different movement patterns. Since we always have the same number of packets sent, we only need to compute the packets that are received at the destination node to do the evaluation. The packet delivery will be our performance metric for this experiment. We can count the number of packets received from the NS-2 trace file by using the following AWK and grep scripts: AWK script was used to count the receiving packets from the trace file.

For example, counting the sending packets from the source node (23). To include all sending data from the trace file named wireless-out.tr in the

Figure 5.1: Screen shot of NAM Animation with two stationary nodes 23 and 24
'sending-out' file, we run the command below:

    awk '/s -t/' wireless-out.tr> sending-out

To include from the sending-out file only the packets that are send from node (23) in separated file named 'out-23', we run the command below:

    awk '$9 == 23' sending-out > out-23

Count the cbr packets in out-23.

    cat out-23 |grep cbr| wc

The same technique used for the receiving packets at the destination node (24) To include all receiving data from the trace file named wireless-out.tr in the 'receiving-out' file, we run the command below:

    awk '/r -t/' wireless-out.tr> receiving-out

To include from the receiving-out file only the packets that are received by node (24) in the 'out-24' file, run:

    awk '$9 == 24' receiving-out > out-24
    cat out-24 |grep cbr| wc

Microsoft Excel sheet is used to create tables and graphs for the results in our experiment.
5.4 Analysis and Results

5.4.1 Base case scenario Setdest

As we defined before we used the Setdest utility in our experiment as the base case to generate the movement file for linear motions of the nodes. In the first set of experiments, we ran the simulation at three different speed settings: maximum speed 10 and 30. For each speed, we simulated a network with different number of mobile nodes: 5, 10, 15, 20, 25, 30, 40, 60, and 80. Each experiment was repeated three times and the average result measured in number of packets received at the destination was recorded.

![Graph 5.2: Linear Motion Result](image)

Graph 5.2 shows the result from the base case scenario experiment, which is using the Setdest utility in NS-2 to generate the linear motions along straight-line paths. It shows two different curves that indicate the number of packets received at two maximum speeds. Furthermore, it appears that
the number of packets received per node starts to increase with high node density reaching the maximum level above 25 nodes. On the other hand, the number of packets received is lower at high maximum speed of 30 compared with the lower speed of 10.

5.4.2 Fractal Motion Scenario

For this experiment where nodes move along fractal paths, again we observe the number of packets received for each node density. To generate the fractal motion, we used the same movement files as in the previous set of experiments (with Setdest and linear motion), and applied to those movement files our fractal transformation discussed in chapter 4.

![Fractal Motion Result](image)

Figure 5.3: Fractal Motion Result

Graph 5.3 shows two different curves that indicate the number of packets received per node density at different maximum speeds. Note that the actual
speeds used in this scenario are 33 percent higher than the speed used in the base case. More precisely, while in the base case scenario nodes moved with speeds 10 and 30, in the fractal scenario these speeds are 13 and 39. The results for fractal motion also show that the number of packets received starts to increase with high node density (25 and more). Furthermore, comparing the linear motion in figure 5.2 and fractal motion in figure 5.3, we observe that at lower node densities, the number of packets delivered is low for both linear and fractal motions. On the other hand, at higher node densities, we observe a bigger drop in packet delivery at higher speed for fractal movement scenario than in the linear motion scenario. We can speculate that the reason behind this is that the speed of fractal movements is 33 % higher than in the linear motion and this affects the performance more significantly at the high speed, which is 39 for the fractal scenario. We also observe that we obtained better packet delivery with fractal motion at lower speed and at low node densities.

5.4.3 Comparing the Fractal and Linear Motions at Low Speed

Based on the initial experiments, we decided to investigate further the performance of the mobile network with the fractal motion of the nodes. We increased the number of simulation runs 10 to observe how the fractal motion affects the number of packets received at the destination. We generated 10
linear movement files by using Setdest in NS-2 having maximum speed of 10
and then converted them to fractal motion. As before, note that the fractal
motion has the maximum speed of 13. We calculated the average packet
delivery for each node density.

![Graph 5.4: Comparing the Linear and Fractal Motions at Low Speed](image)

Figure 5.4: Comparing the Linear and Fractal Motions at Low Speed

Graph 5.4 illustrates the difference in the number of packets received at
the destination when using the original movement files and the new fractal
movement files. It shows that most of the time the packet delivery for the
fractal movement is higher than the original linear movement. Although the
speed of the fractal path is higher than the original, we obtained a bigger
number of packets delivered at the destination at the speed of 13. Figure 5.5
shows the differences between the number of packets delivered for fractal and
linear motions at low speed. However, applying the t-test for the comparison
of two paired means representing the packets received in the linear motion
and the fractal motion experiments with 25 nodes gives 8%, which indicates that the observed difference is not statistically significant. Also, comparing the average packet delivery across all node densities does not show a significant difference (t-test value 49%) as it is illustrated in figure 5.6.

![Figure 5.5: Differences between Fractal and Linear Motions at Low Speed](image1)

![Figure 5.6: Averages of Linear and Fractal Motions at Low Speed](image2)
5.4.4 Comparing the Fractal and Linear Motions at High Speed

For this scenario, we observed the packet delivery at the destination for the linear and fractal movement at higher speed of 30. As before, we used 10 movement files generated by Setdest having maximum speed of 30 and converted them to fractals. In case of fractal movement files, the speed increased to 39 instead of 30. We ran the simulation 10 times for each node density and calculated the average packet delivery.

![Comparing the Linear and Fractal Motions at High Speed](image)

Figure 5.7: Comparing the Linear and Fractal Motions at High Speed

Graph 5.7 shows the packet delivery for linear and fractal motions. This time we observe a lower packet delivery for fractal motion recorded in most of the experiments. One possible explanation of lower performance with fractal motion is that the increase in movement speed of 9m/s, from 30 to 39, results in more frequent link disconnections and consequently lower packet delivery.
Figure 5.8 illustrates the differences between fractal and linear motions at high speed. Applying the t-test for the comparison of two paired means representing the packets received in the linear motion and the fractal motion experiments with 25 nodes gives 32 percent, which indicates that the observed difference is not statistically significant. Also, comparing the average packet delivery across all node densities does not show a significant difference (t-test value 60 percent).

![Figure 5.8: Differences between Fractal and Linear Motions at High Speed](image)

5.4.5 Comparing the Linear Motion at High and Low Speed

Based on the results from previous experiments, we decided to investigate the impact of changing the node speed on packet delivery both in a network with nodes with linear motion and with fractal motion.
Graph 5.9 illustrates the advantage of using lower speed in a network with linear motion. It shows that the packet delivery is consistently higher at low speed for almost all the node densities. Applying the t-test for the comparison of two means representing the packets received in the low and high speed experiments with 20 nodes gives 4%, which indicates that the observed difference is statistically significant. The average packet delivery for all node densities is 3176 at high speed and 3497 at low speed, and this difference in performance is statistically significant (t-test value 0.076%), as it is illustrated in figure 5.10.
5.4.6 Comparing the Fractal Motion at High and Low Speed

Graph 5.11 illustrates the advantage of using lower speed in a network with the fractal motion. The packet delivery is consistently higher at low speed for all node densities. Applying the t-test for the comparison of two means representing the packets received in the low and high speed experiments with 20 nodes gives 1%, which indicates that the observed difference is statistically significant. The average packet delivery for all node densities is 3186 at high speed and 3553 at low speed, and this difference in performance is statistically significant (t-test value 1.7%).
Figure 5.11: Comparing Fractal Motion at High and Low Speed
Chapter 6

Conclusion and

Recommendation

This research is a study of two different movement generators that can be used in Mobile Ad-hoc network simulation. First we investigated the linear movement generation by the Setdest utility used in the NS-2 network simulator. Then we investigated a new movement generation that specifies the motions along fractal paths. This chapter presents a summary of the observations from this research in section 6.1 followed by some recommendations for the future researches in section 6.2.
6.1 Result of the Research

We developed a method for manipulating NS-2 movement files. We built a visualization tool for plotting node traces based on standard NS-2 movement files. Also, we built a tool for transforming linear movements into fractal movements based on the Koch curve. The new tool reads a standard NS-2 movement file, decodes each movement, and replaces it with a series of new movements forming a fractal curve, and then outputs a new movement file. The newly generated movement file satisfies NS-2 specifications and can be used in NS-2 simulator. Both standard movement files generated with Set-dest and new movement files generated with the new fractal tool were used in simulating a MANET with varying number of nodes (i.e. with different node densities). We compared MANET performance in terms of packet delivery under two different motion scenarios and at different speeds.

We observed marginally higher performance of MANET with fractal motion at low movement speeds. However, the statistical tests show that the difference observed in our limited experiments is not significant. We observed that the packet delivery is lower at higher speeds for both motion types, and after the application of the t-test for the difference of the means, we concluded that the observed lower packet delivery at higher speed is statistically significant.

From our results, we conclude that only the node speed significantly affects MANET performance, and not the shape of the path taken by a node.
6.2 Recommendations for Future Research

We recommend further testing of both generators (standard Setdest and new fractal movement generator) with different simulation scenarios and parameters, such as simulation time, movement speed, data traffic intensity, etc. Also, it would be interesting to investigate other curves for modeling the motion, such as specifying the motion along sinusoidal paths for mobile nodes. As using fractal motion changes the shape of the path taken by the nodes, it would be interesting to investigate if this has any impact on the border effect observed in the standard Setdest generator. For this research, we developed a visualization tool that plots node movement paths based on the information stored in an NS-2 movement file; we strongly recommend developing a more comprehensive tool that plots the movements on the screen based on the data stored in trace files. Finally, using another type of MANET routing protocols can be used in comparing the movement patterns and their impact on MANET performance.
Bibliography


81


Vita

Candidate’s full name: Hawra Bader Alseef

University attended:

- King Faisal University, Kingdom of Saudi Arabia, 2003-2007, Bachelor of Computer Science.

- University of New Brunswick, Fredericton, NB, Canada, 2011, Diploma in University Teaching.

- University of New Brunswick, Fredericton, NB, Canada, 2010 - 2013, Masters in Computer Science.

Publications:

Hawra Alseef, Hanin Almutairi and Nada Alsalmi, Network Simulation: Investigation of NS-2 Movement File Generation for a Curved Path, Poster Presentation, Annual Research Exposition 2013 for the Faculty of Computer Science at University of New Brunswick, Fredericton, NB, Canada, April, 2013.

Contact Information: Sumer-22@hotmail.com.