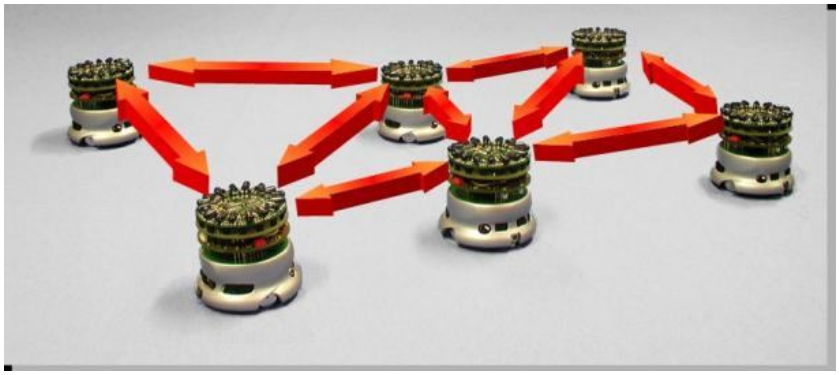
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

PROJECT REPORT

Mobility Modeling of Swarm Robots



Submitted by

Madhur Behl
Student ID 051188

Under the Guidance of

Dr. Deepak Bagai
Faculty coordinator
Dept. E&EC
Punjab Engg. College

Dr. Merkourios Karaliopoulos
Project Advisor
Communication Systems Group
TIK, D-ITET, ETH Zurich

Prof. Dr. Bernhard Plattner
Project Supervisor
Communication Systems Group
TIK, D-ITET ETH Zurich

Submitted to
Department of Electronics and Electrical Communications Engineering
Punjab Engineering College, Chandigarh, INDIA
(Deemed University)

(Semester Project February to July 2008)

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

PROJECT REPORT

Mobility Modeling of Swarm Robots

Submitted by

Madhur Behl
Student ID 051188

Under the Guidance of

Dr. Deepak Bagai
Faculty coordinator
Dept. E&EC
Punjab Engg. College

Dr. Merkourios Karaliopoulos
Project Advisor
Communication Systems Group
TIK, D-ITET, ETH Zurich

Prof. Dr. Bernhard Plattner
Project Supervisor
Communication Systems Group
TIK, D-ITET, ETH Zurich

Submitted to
Department of Electronics and Electrical Communications Engineering
Punjab Engineering College, Chandigarh, INDIA
(Deemed University)

(Semester Project: February to July 2008)

To,

My parents;

In loving memory of my grandparents;

And

to all those who enjoy learning

ACKNOWLEDGEMENTS

This is perhaps the easiest and hardest chapter that I have to write. It will be simple to name all the people that helped to get my semester thesis done, but it will be tough to thank them enough. I will nonetheless try…

Foremost I would like to thank my advisor, my mentor, and my friend Dr. M. Karaliopoulos. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make the entire thesis fun for me. Throughout my thesis period, he provided encouragement, sound advice, good teaching, good company, and lots of good ideas. I would have been lost without him.

It is difficult to overstate my appreciation to Prof. Dr. Bernhard Plattner, who provided me with this valuable opportunity to work as a guest student and carry out this thesis. I express my gratitude to him for the faith he showed in my abilities and thank him for the same. I also thank him for making the financial and laboratory resources available.

A vote of thanks also goes to the entire Communication Systems Research Group at the Computer Engineering and Networks Laboratory at ETH. Every interaction with its members was a great learning experience for me and contributed to my work.

I also want to thank the entire faculty at my home university, Punjab Engineering College for making this semester training possible in the first place.

I cannot finish without saying how grateful I am with my family: sisters, brothers, uncles, aunts, and my nephew all have supported me throughout. Most importantly, I wish to thank my parents, Mr. Anil Behl and Mrs. Suman Behl. They have always supported and encouraged me to do my best in all matters of life.

TABLE OF CONTENTS

INTERNSHIP PROFILE

The project was carried out at the Computer Engineering and Networks Laboratory (TIK) in the Department of Information Technology and Electrical Engineering (D-ITET) at Swiss Federal Institute of Technology, in Zurich (ETHZ).

TIK is part of the Information Technology and Electrical Engineering Department and consists of three research groups active in the areas of:

- Communication Systems (CSG)
- Computer Engineering, including Systems Optimization and Speech Processing
- Distributed Computing (DCG)

The *Communication Systems Group (CSG)* conducts research on the design and implementation of communication systems. In particular, the group focuses on self-organizing networks and network security.

The core research topics are:

- Wireless and ad hoc networking,
- Future Internet architecture and protocols, and
- Network measurement and vulnerability analysis.

The CSG is headed by Prof. Dr. Bernhard Plattner and includes three senior researchers and nine graduate students.

My advisor, Dr. Merkourios Karaliopoulos is a Senior Researcher in the group. The project is being supervised by both him and Prof. Dr. Bernhard Plattner himself.

Project Overview

Mobility modeling of swarm robots

Outline:

Autonomous robotics is an area that attracts significant research interest in the robotics knowledge domain. The use of swarm robots for communication and networking purposes is one of their possible applications.

The standard algorithms controlling the trajectories of these robot swarms aim at maintaining a given swarm formation in the 2D or even 3D space. Collision avoidance and repulsion are the main operations coming under the control of those algorithms.

When looking at these robots from a purely networking point of view one question that becomes relevant is how could this structured movement of the swarm robots be best simulated and to what extent could it be approximated by existing or to-be-developed random group mobility models.

The project builds on work on:

- **Group mobility modeling in 2D and 3D space:** constraints are different than the movement within buildings, which has been the main focus of the rather limited 3D mobility modeling work so far
- **Robot swarm movement control:** here basic control algorithms addressing collision avoidance and repulsion in 2D space may have to be extended to also address movement in 3D space.

Project tasks

1.) Reviewing task. Effectively covers three areas:

- Robot-swarm algorithms
- Group mobility modeling

2) Simulation work

- Implementation in simulator (potentially ns2) of selected existing control algorithms and their validation
- Implementation (reuse where possible) of control algorithms for 3D space

Chapter 1

Introduction

As technology rapidly progresses, diverse sensing and mobility capabilities will become more readily available to devices. Many modern robots are already equipped with multiple sensing capabilities. Robots that are agile and autonomous are well suited for jobs that are “dull, dirty and dangerous”. Future uses for such robots include 3D landscape mapping, reconnaissance and surveillance capabilities, establishing emergency communication networks using mobile agents as relay nodes for search and rescue are some of the novel applications that one can propose.

Once mobility becomes feasible, one can envision networks of such mobile autonomous agents, performing various important tasks. Communication will undoubtedly be one of the essential functionalities of these mobile networks. The project explores the existing control algorithms from a networking point of view. Standard algorithms for maintaining a swarm exist, from networking point of view; robots could be data sources or relays in an ad-hoc network. The main difference is the capability to control mobility unlike the random mobility patters that have been used to model ad-hoc networks for decades.
It becomes necessary, to see how controlled mobility is different and what advantages can it is offer.

Our aim is to enhance a packet-level simulator with the capability to simulate mobility control algorithms. The simulator we have used is ns-2 (version 2.27). Even though ns2 is largely versatile and well suited for various simulations, it does not cater directly for mobility control algorithms. Therefore, the project work includes developing new modules for network robotics related simulations, or enhancing already existing modules, thus adding to the utility of the simulator as a tool for the investigation of networked robotics.

The latter is a research area that has gained importance in recent years. Our project work is intended as an important contribution towards the better study of fundamental trade-offs in this area, which bridges the data communication with the robotics world.

Another part of the project aims at exploring mobility for 3D space. The present ad-hoc mobility models and robot swarm algorithms are applicable and have been proposed for 2D cases only. Therefore, we explore mobility in 3D and further enhance the simulator with the functionality of simulating 3D mobility control algorithms as well.

Chapter 2

Mobility Models and Swarm Algorithms: Introduction

What are Mobility Models?

To simulate a new protocol we use a mobility model that represents mobile nodes (MNs) that will use the protocol.
In an ad hoc wireless network, nodes may move freely within the field. For a pair of nodes to communicate, a route must be formed between intermediary nodes. For this type of network, it is very important to model node movement, as transmitter range is generally fairly small in relation to the size of the field. One can classify the mobility models as two types: traces, where actual node movements have been logged and can be replayed in a simulation. The other is synthetic models, which do not use traces.
Another classification is the entity mobility models and the group mobility models. In the entity/independent models the movement of each node is modeled independently of any other nodes in the simulation. Furthermore, within the category of entity models, some models introduce correlation in the mobility patterns of the nodes, whereas others do not. In the group mobility models, there is dependence in the movement of a group of nodes throughout the cells or field.

Classification of Mobility Models

Entity-based models

1) **Random walk mobility model:** In this model, a random speed and a direction are chosen by a node from the intervals [minspeed, maxspeed] and [0, 2 * pi], respectively. Each movement of a node occurs either for a constant time, or for a constant distance. On reaching a boundary of the simulation area the node reflects back depending on the angle at which it struck the wall.
2) **Random Waypoint:** In this model, the node pauses for a fixed amount of time, during which it chooses a new destination and then also chooses a random speed from the set [minspeed, maxspeed]. After the pause time is exhausted it continues to move to the destination and upon reaching the destination the same procedure is repeated.
3) **Random direction**. In this model a node chooses a random direction and then travels to the border of the simulation area in that direction. On reaching the boundary it again chooses another direction to follow.

4) **Boundless simulation area**: In this model a relationship exists between the previous direction and velocity of a node to its current direction and velocity. There are functions to determine the new velocity, direction and hence position. All these are updated after a fixed time interval. If a node reaches the boundary of a simulation area then instead of reflecting back from it, it continues travelling and reappears from the opposite side of the area.
5) **Gauss markov mobility model**: Here again the current velocity and direction are dependent on the previous values of velocity and direction, and are changed after fixed intervals of time using some functions. Moreover the nodes do not remain near the edge of the simulation area for long and are forced away from the edge once they reach certain distance from the edge. There are no sharp turns as the past velocities and directions influence the future paths.
6) **Probabilistic Random walk**: A probability matrix is used to determine the position of a particular node in the next step. There are three states: current position, previous position and next position. The behavior of is more probabilistic rather than random. Choosing the appropriate parameter for the matrix is quite challenging.
7) **City section model:** Quite realistic model in which the simulation area is like the section of a city. It may be in the form of a grid in which some lines have higher speed limits while some have comparatively lower speeds. The node chooses a random destination and then its movement is based on the shortest time it will take to reach that destination; nodes cannot move freely but have to take into account the traffic/path regulations.

Group Mobility models

In the group mobility models, there is some relationship among the nodes and their movements throughout the cells or field. So the movement of a MN is no longer independent but has a relationship with other nodes as well.
Some basic group mobility models were reviewed, including:

1) **Column** group mobility model
2) **Nomadic** community mobility model
3) **Pursue** Mobility model:
4) **Reference Point Group Mobility Model**: It is a more general model and the above mentioned group models can be derived from RPGM itself.

More on RPGM

Each group has a logical center. The motion of the entire group is determined by the path taken by the logical center. Within each group there are several mobile nodes; each of these nodes have their respective reference points about which they can exhibit random motion using any one of the entity models. Each group has a motion vector. The new position is calculated using the motion vector; first, the reference point of a node moves to the new location and, then, the node's new position is calculated by adding another random vector to the new reference point position.

The path a group follows can be defined by a sequence of check points corresponding to given time interval. This model provides us with a flexible framework to describe patterns which are time bound and task oriented.
Possible applications of the RPGM include its use as:

1) **In-place mobility model**: The entire area is divided into regions and each region has a different group each having its own motion pattern.
2) **Overlap mobility model**: In this model the same region is available to all the different groups with different motion patterns.
3) **Convention Mobility model**: Again there are different zones with a group in each zone, in addition to that there is one group that roams from zone to zone and can move around anywhere in the area

Robot Swarm Algorithms

Swarm algorithms are used to disperse a group of autonomous robots in a bounded region using inter robot communication and sensing techniques, while addressing issues like collision avoidance and repulsion

Directed Dispersion:

Uniform dispersion:
In this a maximum dispersion distance between the robots and physical boundaries is used to help spread the bots evenly. The algorithm works by calculating the positions of its closest neighbors and then moving away from them, under the constraint that a minimum safe distance has to be maintained among them at all times.

Frontier guided Dispersion:
In this, robots are guided into the areas which have not yet been explored using the frontier robots. A robot can occupy any of the three positions – Wall, Frontier, and Interior. Wall robots are those that detect a boundary or an obstacle. Frontier robots are the ones who are on the edge of an open space while the remaining are interior robots. The method used to find frontier robots is that all robots calculate the largest angle between itself and adjacent robots which are within a certain distance from it. The robots, for which this angle is 180° or greater are termed as frontier robots.

Once the robots know their positions in the network the frontier bots move while other bots follow. It is possible to let the frontier bots 'pull' the swarm behind them. However this may cause newly developed frontiers to pull bots away from the previously explored areas thus fracturing the connectivity of the network. So, instead in order to build a reliable network a robot can only move if they are in contact with at least two children (other bots) in the frontier tree.

Gas expansion and guided growth models:

This technique is inspired from the biological phenomenon of pheromones. Virtual pheromones are messages that belong to a robot and are locally transmitted without specifying a recipient. They provide navigational information to other bots within the region. The robots use attraction and repulsion forces to avoid collisions and at the same time staying within the communication range of each other.

Gas expansion model is similar to uniform distribution model except the fact that instead of using position vectors of other bots, it uses attraction and repulsion forces to maintain distances between them.

Guided growth model works well when there are large numbers of bots to cover the entire region. It uses barrier robots to help spread the swarm. Barrier bot emits a barrier pheromone signal, preventing all other bots from coming close to it. As it moves away from others, they follow it to fill the space while maintaining connectivity. It also emits a growth inhibitor pheromone, which prevents the formation of other barriers. Once the barrier cannot move any further, it stops emitting the growth inhibitor and a new barrier helps spread the swarm further.

Shapebugs

It is an algorithm to spatially organize the group into arbitrary shapes. Each member has a map of the shape to be constructed. This algorithm works in two parts: trilateration and movement. Trilateration is the process of calculating the position of a bot, based on the position of its neighbors. Once the position is known, the algorithm execution proceeds with the movement phase. The robots that are unaware of their position move randomly until they reach a position that is within the shape. Robots who think they are inside the shape should ensure that however they move, they must not get out of the shape and they should also be capable of absorbing new members into the shape. The movement phase has a main goal of maintaining uniform density, just like uniform pressure in a closed container of gas. So the movement vectors of the members are inversely dependent on distance, thus moving them away from a more dense area to a lower one.

Chapter 3

MATLAB Coding

The following Mobility Models were implemented in MATLAB

- Random Walk Mobility Model
- Random direction mobility Model
- Random waypoint
- Reference Point Group Mobility

First, the results were obtained for one single node, and later for multiple nodes.

Writing these programs required good knowledge of MATLAB commands, functions and plotting techniques. So, considerable time was spent on those.
Also, a major challenge while programming these models on MATLAB was the incorporation of the boundary conditions. In order to ensure that the nodes always remain within the target bounded area and get reflected off appropriately at its borders.

Some of the outputs obtained are presented below:

Random Walk Mobility Model

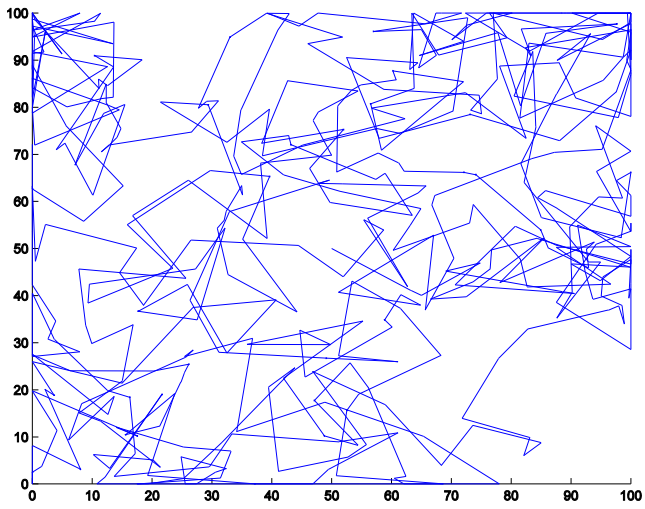


Figure 1 : node movement in Random Walk

Random Direction Mobility Model

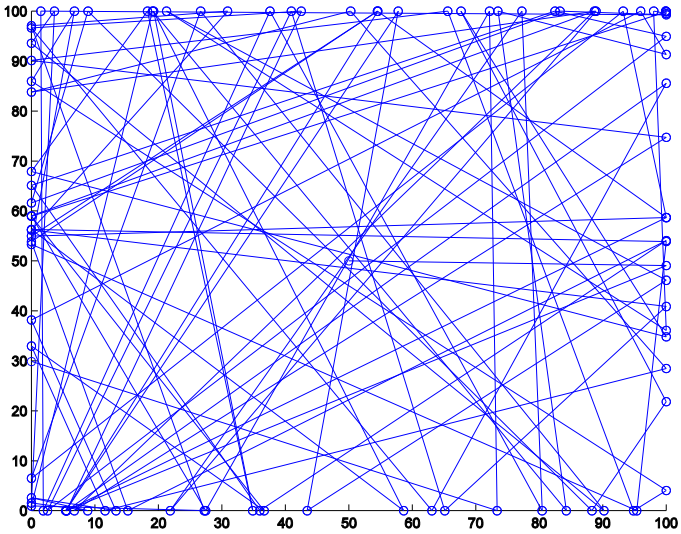


Figure 2 : Node Movement for Random Direction Mobility Model

Random Waypoint Mobility Model

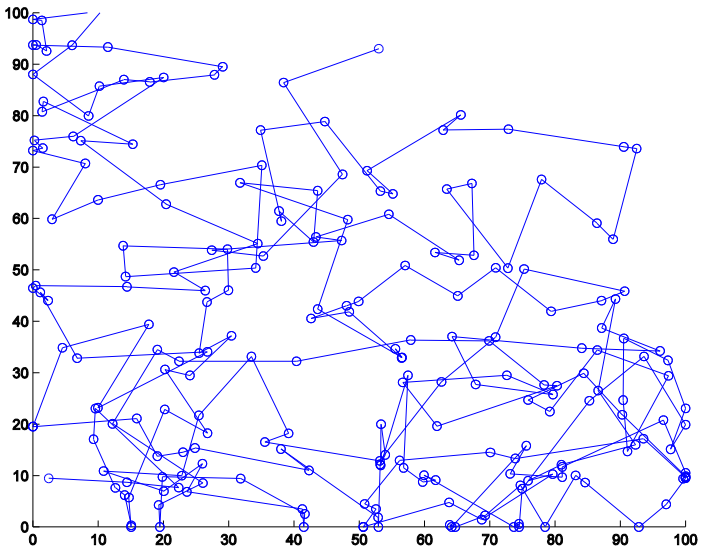


Figure 3 : Node Movement for Random Waypoint Mobility Model

Reference Point Group Mobility Model

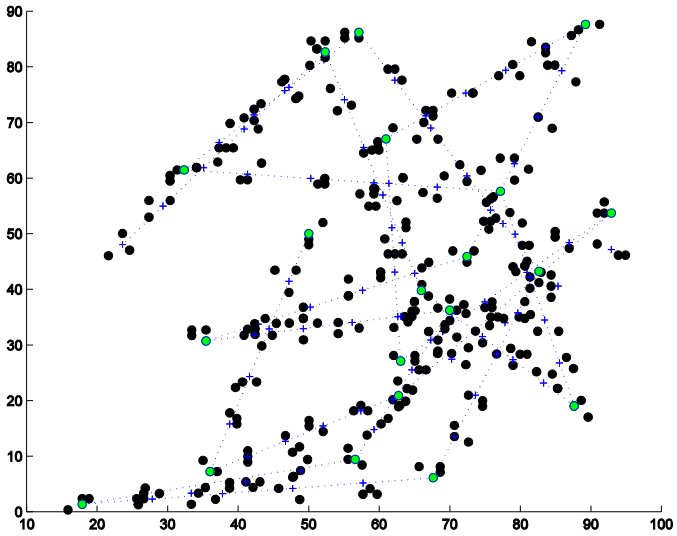


Figure 4: Reference Point Group Mobility Model

.

Each of the nodes in the group moves randomly about the reference point at all times while remaining 'within its range'.

Chapter 4

ns-2: An overview

The network simulator 2 (ns-2) is a popular and powerful simulation environment, and the number of ns-2 users has increased greatly in recent years. Although it was originally designed for wired networks, ns-2 has been extended to work with wireless networks, including wireless LANs, mobile ad hoc networks (MANETs), and sensor networks.
ns-2 is a discrete even simulator targeted at network research.

It was developed at UC Berkeley. It is maintained at ISI, Information Sciences Institute
[http://www.isi.edu/nsnam/ns/].

Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. It is the de facto packet level simulator for network research and allows the user to collect information about every packet in the simulation.

Network Simulator ns-2

- Maintained through VINT project – Open Source Project
- NS2 :collaborative simulation environment
  - ✓ Freely distributed and open source
  - ✓ Supports NT research and education
    - ✓ Protocol design, traffic analysis etc.
  - ✓ Provides common reference

- Open source! advantages:
  - facilitates building upon other work
  - allows others to use your work
- Disadvantages:
  - huge and complicated code
  - partially missing, somewhat poor documentation

User's Perspective

From the user's perspective, ns 2 is an OTcl (Object oriented Tool Command Language) interpreter that takes an OTcl script as input and produces a trace file as output.

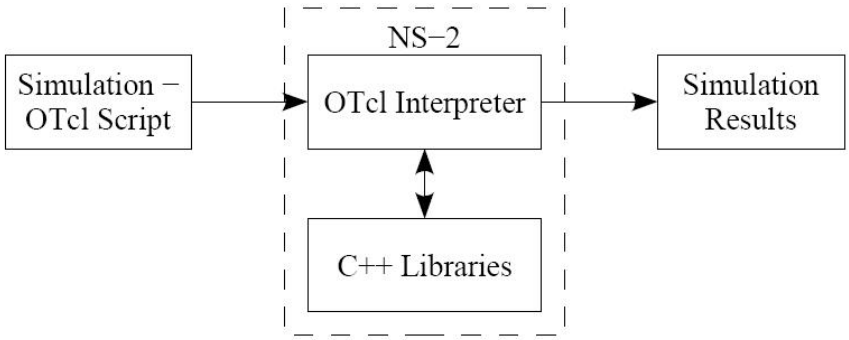
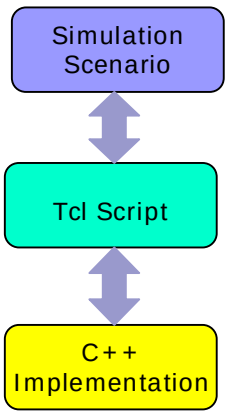


Figure 5 : ns-2 Interface

Discrete Event Simulator

- Physical activities are translated to events
- Events are queued and processed in the order of their scheduled occurrences
- Time progresses as the events are processed

ns-2 Environment



ns-2 Directory Structure

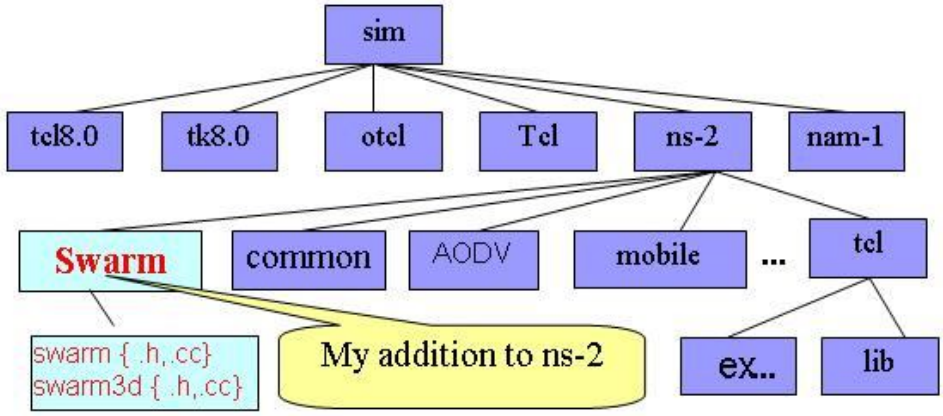


Figure 6 : ns-2.27 directory structure and our extensions

ns-2: C++/ Tcl

- NS-2 Code contains two sets of languages, namely C++ and OTcl.
    - C++ is used for the creation of objects because of its speed and efficiency.
    - OTcl is used as a front-end/interpreter to setup the simulator, configure objects and schedule events because of its ease of use.

Needless to say, ns-2 was chosen due to the access to simulator kernel code, a necessity for extending the simulation environment.

Therefore, our goal was to enhance this simulator with the ability to implement mobility control algorithms for both 2D and 3D scenarios

Chapter 5

Mobility in ns-2

How does a mobile-node move in ns-2?

ns-2 has the provision for running wireless simulations, hence, it was necessary to find out what does the simulator do with respect to generation of movements for nodes. The task was to explore the already existing mobility model (if any) modules in ns-2.

Our findings were that, *ns2 only uses the Random Waypoint Mobility* (RWM) algorithm for creating node movements in a wireless scenario. The RWM model that it uses is time bound (and not destination bound), in which the node changes its direction after moving for a fixed interval of time.

It was also important to note how much of the mobility can be controlled, i.e. How many parameters can the user change?

The random movement can be enabled or disabled by the user from the simulation script while creating the mobile node.
The code for the generation of movements is contained in the files MobileNode { .h,.cc} under the directory ~/ns-2.27/common/

Another directory wherein user can create movement pattern files is using the *setdest* function. By specifying the parameters like speed, pause times and coordinates etc, the *setdest* function calculates the position of the nodes for all instances of time.

In order to better understand and validate the existing RWP model we simulated a scenario in ns2, wherein nodes move randomly in accordance with the inbuilt RWM model.

The results for a wireless scenario for 50 nodes are shown:

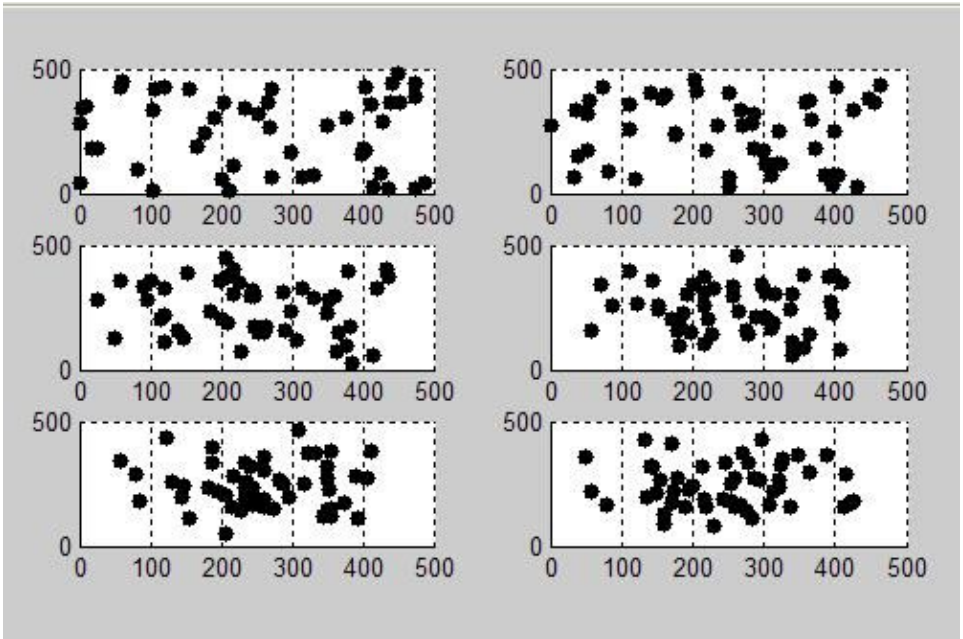


Figure 7 : Node distribution by Random Waypoint Model present in ns-2

Data plotted here (in MATLAB) was obtained from a trace file generated by simulation of the existing RWP mobility algorithm in Ns2.

One noticeable feature in the above figure is the clustering of nodes in the center of the simulation area as time progresses. In fact this was something that is expected to happen because in case of random waypoint model, the probability of a node to choose a destination which is near to the center rather than the edges is high, and so it leads to the clustering of nodes [6].

Certain problems with Random Waypoint Model:

Even though RWM has been used for so long now (even in ns-2), to carry out performance evaluations of protocols, studies indicate that this model fails to provide a steady state i.e. the average nodal speed consistently decreases over time and therefore unreliable results can be obtained by using this model.

This problem has been discussed in the following paper:

Jungkeun Yoon. Mingyan Liu. and Brian Noble. *Random Waypoint Considered Harmful*. [6]

The problem is inherent to random waypoint model that is destination bounded. The model chooses a destination and a speed for a node at random, and the node will keep moving at that speed until it reaches the destination. Therefore, a node

with slow speed and a far-away destination may take a long time to finish the trip or may never reach the destination. Thus on sampling the speeds of nodes at various times, this node will tend to lower the average nodal speed and eventually more and more node will tend to get 'trapped'

One easy solution to this problem is to have a minimum speed defined, so that nodes do not get trapped at all. This restriction on the minimum speed is imposed in our module for swarm node.
Another solution is to use time bound random waypoint model instead of a destination bound. I the time bound RWP model all nodes change directions after fixed time intervals, thus preventing the nodes from getting trapped for longer intervals. The model that we implement in ns-2 is also time bound.

Another model is proposed as a solution to this problem and is described in the paper:
J.-Y. Le Boudec and M. Vojnovic, "*Perfect Simulation and Stationarity of a Class Of Mobility Models*" [7]

Chapter 6

Towards Control Mobility in ns-2

As already stated, ns-2 only uses the Random Waypoint Mobility algorithm for generating node movements. So it becomes necessary to understand

- How RWP model is implemented in ns-2?
- How much node movement, can a user control?

ns-2 does not cater at all to mobility control algorithms. So, in order to implement a mobility control algorithm in ns-2 and answer the questions stated above, it was required to dig deep into the existing modules that generate random node movements.

The generation of node movements is done through functions present in a separate class, called MobileNode, which is a kind of Node having all the features of a normal node but also has an additional feature that it can move (randomly)

These functions written in C++ randomly assign a destination and move the node towards it in fixed time iterations (Random Waypoint Mobility Model)

The project required, for us to *create a Swarm Node* similar to a Mobile node with the added utility that the motion of the swarm node is controlled instead of being purely random.

Control Mobility Algorithm

The first step to do so was to define 'control mobility'.
What sort of control are we planning to do?

For this an existing control algorithm was selected for implementation in Ns2.

The selected algorithm for the simulation in NS-2 has been proposed in the following research paper-

- Towards Mobility as a Network Control primitive: *David K. Goldenberg, Jie Lin, A. Stephen Morse, Brad E. Rosen and Y. Richard Yang* [8]

This serves as the reference algorithm for my work.

They propose a mobility control scheme for improving communication performance in networks

In brief, the algorithm states that:

For achieving greater efficiency in communication the optimal positions of the relay nodes must lie entirely on the line between the source and destination. Furthermore the relay nodes must be evenly spaced along the line.

This assertion is validated through simulations and mathematical proofs.
Also, two variants of the algorithm are presented:-

- Synchronous
- Asynchronous

The algorithm works by calculating a mobile node's destination based on the position of its adjacent neighbors (hops) within the route and then moving the node to its destination, to finally obtain an optimum configuration.

Therefore the mobility of a node is controlled instead of being random.

Having thus chosen a reference algorithm, the task that lay ahead was its implementation in ns-2, thereby enhancing the packet level simulator with the capability to run a mobility control algorithm.

Chapter 7

Extending the Ns code:

New Module

In order to implement the control algorithm, the creation of a separate module was required, which had all the features of a mobile node, plus the added functionality that its motion was controlled rather than being random.

It was important to figure out how a node can communicate with its neighbors and get data regarding the position of its adjacent neighbors, and then use this data to calculate its destination.

In some cases the existing functionality of code already present in Ns for mobile node was used, but more often new functions had to be written for our case. This involved defining a new class of nodes, namely '*SwarmNode'*, which inherits from an existing class MobileNode. It is not enough to define this inheritance in C++, but the inheritance of a class, also needs to be defined in the static class definition in Tcl.

In order to implement the control algorithm

- A new class of nodes called 'SwarmNode' was created. { which inherits from MobileNode}
- An object of this class had all the functionality of a regular Node and MobileNode plus, an add on feature that *its mobility was controlled rather than random.*
- The 'SwarmNode' was linked with the AODV routing agent { necessary to get next and prev hop info }
- Defined inheritance in Tcl to allow user to configure the node from the script itself.

An outline of the added code is presented below: This code is contained in a separate module in the files *~ / ns2/swarm/ swarm {.cc, .h}*
Also, additions had to me made in already existing files like *ns-lib.tcl , mobilenode.h , aodv{ .h,.cc}* to ensure that proper functionality is added to the swarmnode module.

New Classes and functions

- *Class SwarmNode*: It is an inherited class from MobileNode and includes the declarations of various functions and variables and other attributes.

- *Static class SwarmnodeClass*: Inherits from TClClass, and is used in order to create an instance of a TClObject of the type SwarmNode.

- *Class HandlePosition*: Updates the position of the Swarmnode after a fixed time interval by scheduling events for the scheduler to execute.

Functions:

- command():
    - Creates a 'Tcl' instance for the SwarmNode object
    - Compares strings which serve as commands for the swarm node object
    - inputs values of parameters that the user passes from Tcl scripts, Ex: the user can set the value of control motion to be switched on or off for the swarm node

- destination():
    - initializes the topography for the simulation
    - assigns initial random positions to distribute nodes
    - assigns random speed to each node
    - ensures that speed assigned is always greater than a certain minimum value

- calc_destination():
    - using the IDs of the adjacent neighbors, it calculates their current position
    - using this information then the destination for the swarm node is calculated
    - if the node is unable to find its neighbors, it is moved randomly for the next time interval and repeats the procedure again

- move_to_dest():
    - moves the node towards the destination calculated previously
    - the movement is done in steps
    - on reaching the destination, check again for the position of the neighbors

- move_random():
    - assigns a random destination to the node
    - ensures that the destination is bounded within the topography assigned by the user
    - calculates unit vectors in the direction, in which the node is required to move

- update_location():
    - It is a function similar to update_position() in MobileNode
    - It is in this function that the position of the node is updated and it moves towards the assigned destination
    - The node does not reach the destination in one go, but moves in steps according to the speed it was randomly assigned

- The user can specify from the script (while configuring the node) whether he wants mobility control to be 'ON' or 'OFF' (see figure)
- Everything else remains the same
    - The nodes are created as before (if user specifies 'ON' then a SwarmNode is created)
    - The traffic is also created like for any other scenario (but only AODV is supported).

```
# configure node

    $ns_ node-config -adhocRouting $val(rp)
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -swarmNode ON \
                -channel $chan_1_ \
                ……..
```

Mobility control can be set from the script via node-config

How 2D control algorithm works in ns-2?

Initially all the 'SwarmNodes' are created and distributed randomly.

```
for { set i 0} { $i < $opt(nn)} { incr i} {
set n_($i)  [$ns_ node]
$n_($i) control-motion 1
$n_($i) spread
}
```

After the node is created (set n_($i) [$ns_ node]) the 'spread' calls destination() function, which assigns random position and speed to the SwarmNode created.

Next, the source and destination nodes are fixed, (they do not move during the simulation). The user can specify their coordinates or can even fix them at random positions.

```
$n_(0) fix
$n_(0) set X_ 50.0
$n_(0) set Y_ 50.0
$n_(0) set Z_ 0.0

$n_(9) fix
$n_(9) set X_ 400.0
$n_(9) set Y_ 400.0
$n_(9) set Z_ 0.0
```

Next, the traffic flow between the source and destination is started

```
#Setup traffic flow between nodes

#TCP connections between node_(0) and node_(9)

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $n_(0) $tcp
$ns_ attach-agent $n_(9) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 10.0 "$ftp start"
```

After letting the traffic run for some time, so that AODV find the optimum route, the algorithm is then started

```
for { set i 0} { $i < $opt(nn)} {incr i} {
$ns_ at 20.0 "$n_($i) start"
}
```

Each node checks if it is a part of the traffic or not

- If so, then it gets the prev and next hop information from the AODV agent and calculates its destination as the mid point of the current locations of the adjacent hops.
- If not, then it moves randomly for a fixed time interval and then checks again

The algorithm is 'synchronized' i.e. all relay nodes make the check and change directions at the same time. This is important in order to prevent the 'trapping' of nodes problem that was discussed earlier.

The algorithm converges with time as nodes are placed at their optimum positions.

Finally, the nodes at reset at a desired time, the simulation is stopped and the traffic and movement data is dumped into the trace file

```
$ns_ at 101.0 "stop"
$ns_ at 101.01 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
   global ns_ tracefd
   $ns_ flush-trace
   close $tracefd
}

if { $opt(lm) == "on" } {
  $ns_ at 10.0 "log-movement"
}
```

Chapter 8

Results - 2D Control Algorithm in ns-2

After running the simulation the trace generated contains data for both the traffic and node movement. In order to validate the working of the algorithm we need a visualization tool that can show us the swarm node distribution with time

Unfortunately, the network animator (NAM) for ns-2 has not been extended for wireless visualization.
Therefore MATLAB is used to generate time snaps of swarm nodes.

However, the format of the trace file is not compatible with MATLAB and cannot be directly used for plotting

```
s 16.982763444 _19_ AGT  --- 293 ack 40 [0 0 0 0] ------- [19:0 0:0 32 0] [139 0] 0 0
r 16.982763444 _19_ RTR  --- 293 ack 40 [0 0 0 0] ------- [19:0 0:0 32 0] [139 0] 0 0
s 16.982763444 _19_ RTR  --- 293 ack 60 [0 0 0 0] ------- [19:0 0:0 30 7] [139 0] 0 0
r 16.992862034 _19_ AGT  --- 269 tcp 1060 [13a 13 7 800] ------- [0:0 19:0 27 19] [140 0] 4 0
s 16.992862034 _19_ AGT  --- 294 ack 40 [0 0 0 0] ------- [19:0 0:0 32 0] [140 0] 0 0
r 16.992862034 _19_ RTR  --- 294 ack 40 [0 0 0 0] ------- [19:0 0:0 32 0] [140 0] 0 0
s 16.992862034 _19_ RTR  --- 294 ack 60 [0 0 0 0] ------- [19:0 0:0 30 7] [140 0] 0 0
M 17.00000 0 (0.00, 0.00, 0.00), (0.00, 0.00, 0.00), 2.21
M 17.00000 1 (239.75, 93.89, 244.94), (0.00, 0.00, 0.00), 3.19
M 17.00000 2 (92.72, 250.90, 49.24), (0.00, 0.00, 0.00), 3.20
M 17.00000 3 (34.14, 257.92, 127.10), (0.00, 0.00, 0.00), 4.04
M 17.00000 4 (104.62, 77.43, 122.61), (0.00, 0.00, 0.00), 2.59
```

Figure 8 : Trace file with both traffic and movement data for swarm nodes

Therefore, in order to plot data in MATLAB the movement related data has to be filtered out of the trace files into a format that MATLAB recognizes
For this we use a text programming language called ***'awk'.***

'Awk' is a string parsing language for performing text processing tasks. It classifies the text into columns and each column is for a separate field. The scripts written in 'awk' are simple logical statements based on fields, which work by searching through the entire trace file.

As a result of the filtering of the trace, a data file is generated which can be loaded into MATLAB for plotting purposes:

```
17.00000 10 271.7 73.16 0.00, 0.00 2.02
17.00000 11 125.5 594.5 0.00, 0.00 4.73
17.00000 12 597.2 75.06 0.00, 0.00 2.98
17.00000 13 224.8 82.48 0.00, 0.00 3.34
17.00000 14 172.1 468.1 0.00, 0.00 2.72
17.00000 15 587.3 175.4 0.00, 0.00 2.77
17.00000 16 285.2 18.46 0.00, 0.00 3.92
17.00000 17 634.3 517.7 0.00, 0.00 3.47
17.00000 18 119.7 653.9 0.00, 0.00 2.46
17.00000 19 650.0 250.0 0.00, 0.00 2.66
18.00000 0 10.00 10.00 0.00, 0.00 2.21
18.00000 1 50.00 600.0 0.00, 0.00 4.08
18.00000 2 157.1 428.0 0.00, 0.00 4.18
18.00000 3 109.9 28.67 0.00, 0.00 3.20
18.00000 4 76.25 576.0 0.00, 0.00 2.12
18.00000 5 117.1 63.07 0.00, 0.00 4.04
18.00000 6 233.6 172.9 0.00, 0.00 2.04
18.00000 7 347.4 312.3 0.00, 0.00 3.46
18.00000 8 577.5 562.6 0.00, 0.00 4.26
18.00000 9 600.0 600.0 0.00, 0.00 4.52
18.00000 10 271.7 73.16 0.00, 0.00 2.02
18.00000 11 125.5 594.5 0.00, 0.00 4.73
18.00000 12 597.2 75.06 0.00, 0.00 2.98
```

Figure 9 : Data file compatible with MATLAB after filtering

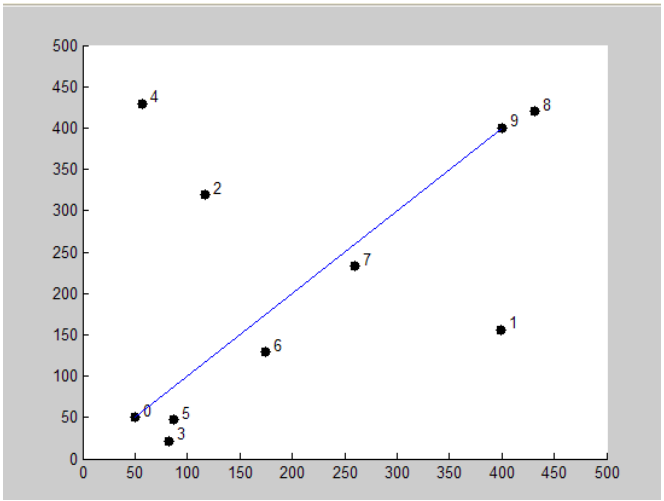
The data is plotted via MATLAB scripts that searches the entire ‘matrix’ row by row and plots the node positions at different times.

Some results obtained for the 2D implementation in ns-2 are presented next.

Different scenarios were simulated by varying the range of a swarm node and for different densities as well as different types of flows. The results obtained are:

2D Simulation 1: default range

Traffic: Singleflow, AODV; Topology: 500 x 500m; Source node: 0; Destination node: 9; Range: 250m (default range)



Initial

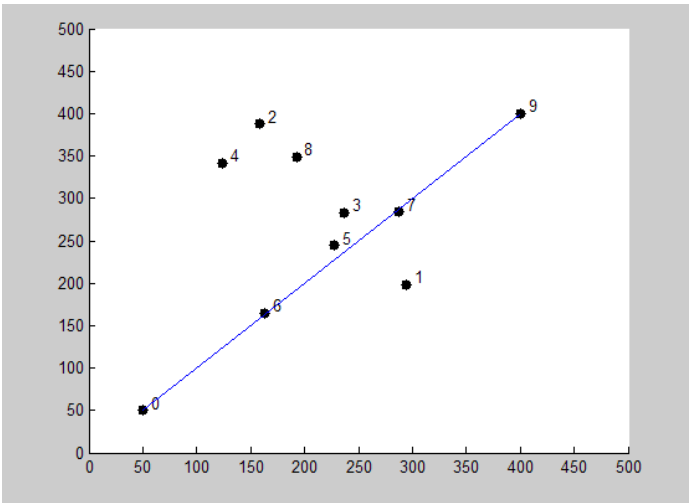
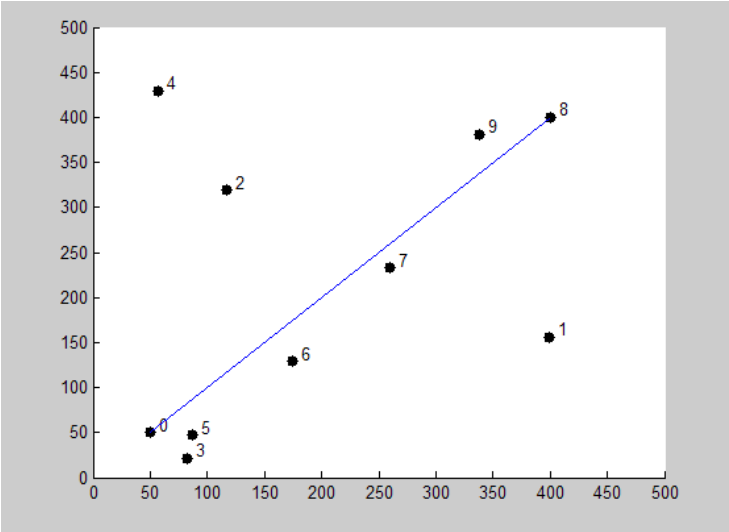


Figure 10 Simulation final at convergence (t=100) route 0->6->7->9

As is clear from the figure, nodes that are a part of the traffic are aligned on the line joining the source and destination

2D Simulation 2: reduced range

Traffic: Singleflow, AODV; Topology: 500 x 500m; Source node: 0; Destination node: 8; Range: 100m (Reduced range)



Initial

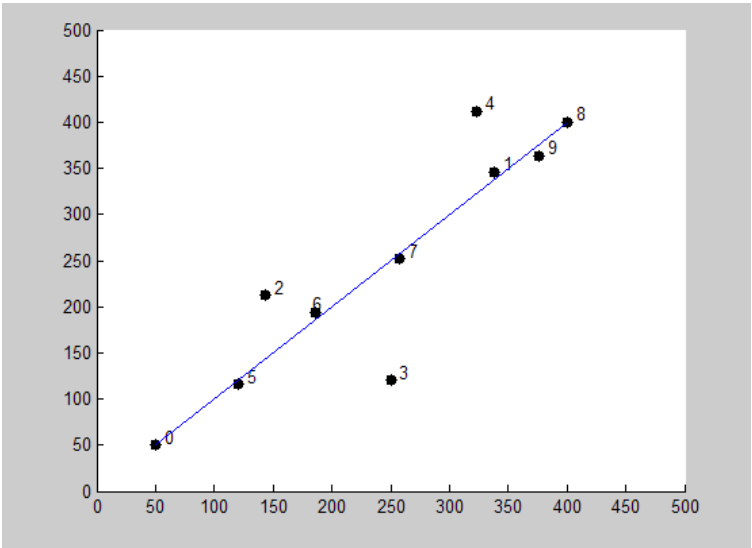
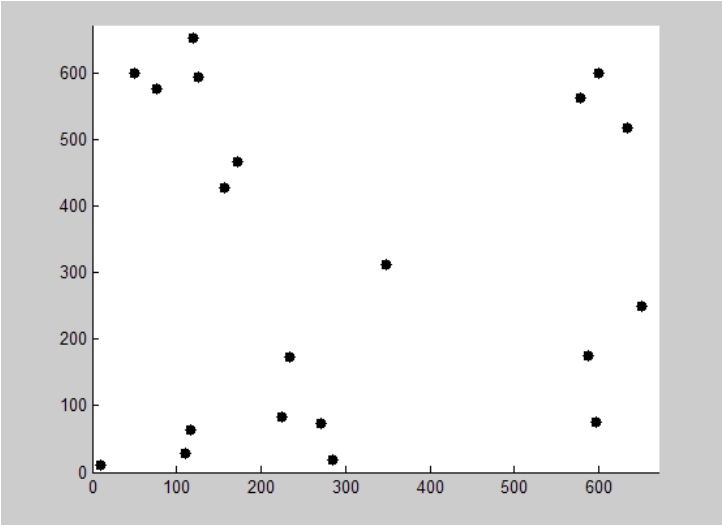


Figure 11 Simulation 2 Final: route 0->5->6->7->1->8

On reducing the range, more number of nodes become a part of the traffic; and are eventually aligned along the line joining the source and destination

2D Simulation 3: multiflow

Traffic: Multiflow, AODV; Topology: 670 x 670m; No. of Nodes: 20
Range: 250m (default range)
Flow 1: (10, 10) -> (650,250); Flow 2: (50,600) -> (600,600)



Initial

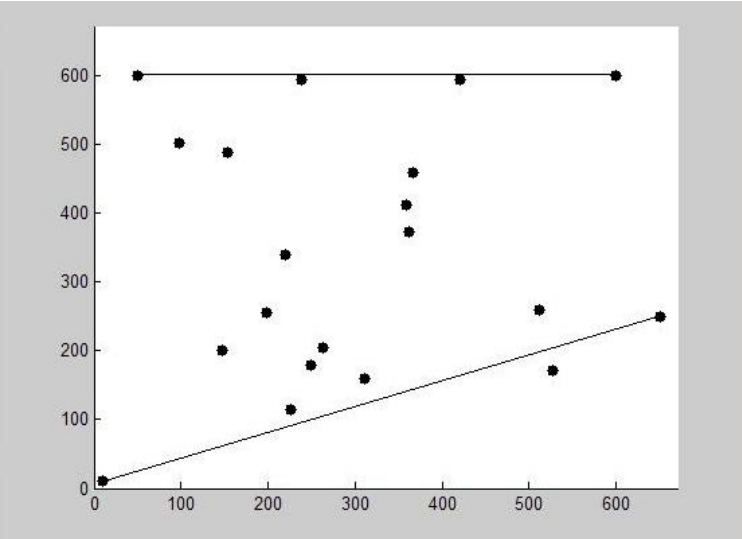


Figure 12 Simulation 3 final convergence

Even for multiple flows (with increased node density) the algorithm appears to converge to optimum configuration.

Chapter 9

Controlled Mobility Modeling for 3D Space

Introduction

2D mobility models have been traditionally used to represent cellular or ad hoc movement patterns for moving nodes. These models, which include the random and the group mobility models, are critical for network traffic and protocol analysis. Similar and more sophisticated models exist for maintaining and controlling the mobility of a robot swarm, but these too are applicable to a 2D distribution only.

Node movement in 3D space increases the mobility of the node as it can now move along an extra dimension as well. Applications include ground surveillance by flying robots/UAVs, establishing communication networks in case of emergencies etc. Therefore; it becomes increasingly important to model control node movement in three dimensions.
Unfortunately, unlike 2D, no set algorithms are available for mobility control in 3D. So, as a result very limited work has been done for implementation of 3D mobility control algorithms. We first explore some of the mobility models that have been proposed for 3d environments and later present an implementation of a 3D control algorithm in ns-2.

Related Work

The literature available on mobility control for 3D space is quite limited. One can classify few different categories from where the contributions have come in this area. One such category is the navigation of robots in 3D environment; the focus of which has been largely restricted to indoor environments rather than free space. Then there is another category of papers that considers path planning and controlled motion for flying robots/UAVs for various applications like search and rescue, surveillance etc., however they too consider a specific problem rather than presenting a general model. Contributions have also been made from the communication and networking community in terms of mobility models for protocols that work in 3D.

S.Kim *et al* discuss mobility modeling and traffic analysis for three dimensional indoor environments [12]. In their paper they extend mobility modeling in 1D and 2D space to 3D indoor building environments by considering the proper boundary conditions on each floor and analytically modeling mobility. The model proposed by them targets a particular scenario and is therefore very specific (even though it

is quite realistic). Moreover the mobility model they propose is based on random selection of velocity and direction based on probability distributions, so it is not a controlled mobility scenario.

The next categories of papers describe models for path planning and controlled motion of flying robots/UAVs. All of them, however, present results for an approximated 2D case.
The paper [9] by Corke, Peterson and Rus describes a control algorithm for the navigation of a flying robot. Instead of focusing more on the mobility of the flying robot this paper can be seen as an application of a sensor network to navigate a flying robot. The nodes forming the ground sensor network are equipped with sensors, limited memory and processing capabilities and communication capabilities. Each node in the network has access to the global state via communication within the network. Robot guidance and navigation is achieved by the interaction between the robot and local nodes. The reverse is also true, i.e. the robot (flying) may inject data into the network. Inspite of using a robot with flying capabilities, they only consider path planning in 2D, i.e. no indications are given for when should the flying robot drop or gain altitude. However, an interesting approach that they have described is the localization of ground based sensors with the help of the flying robot, which is very useful from the point of view of communication and information exchange of flying robots with stationary or moving ground objects.

The paper [10] by Kuiper and Tehrani also proposes a mobility model for UAVs. The problem they have considered is that of scanning a specific area periodically using limited number of UAVs. To achieve this they propose two basic mobility models 1) A random model (very similar to random walk but incorporates probabilities to move in certain directions) and 2) A pheromone based model: Here positional information is exchanged between UAVs and their path depends on the position of others. Both the models, especially the pheromone based, are well suited for the purpose of scanning. Similar to the earlier paper [9] by Corke, Peterson and Rus, a noticeable assumption that they have also made is that all UAVs are flying at nearly the same altitude. Under this assumption, this clearly becomes a case of 2D mobility, even though the nodes (vehicles) are flying in a 3D environment. This fact is also evident from the 'random action table' that the UAV uses to decide its next action. The discrete probabilities given in this table are only for moving in a 2D area and do not take into account the motion of the UAV in the vertical plane. However, in contrast to the model proposed by Corke, Peterson and Rus [], the pheromone model described, controls the mobility of the UAVs without using ground based navigation. This is important from the application point of view, since we would want the nodes/UAVs to determine their path on their own via communication with local nodes/UAVs, rather than providing some sort of an external aid.

S.Nanda and R.S. Gray, propose in their paper [11], a protocol that can be implemented in a three dimensional scenario. The paper presents an ad hoc

routing protocol called Multiple Location Aided and Routing, as an extension of Location Aided Routing[]. MLAR works effectively in both 2D and 3D.
They have also worked on extending ns-2 to support a 3D mobility model and routing protocols. Their primary contribution, from the point of view of 3D mobility, is the implementation of 3D Random Waypoint model in ns-2. The concept of Random Waypoint is used, as it is, to work for 3D. The only change being that instead of choosing random values of only X and Y coordinates the mobile node also makes a random selection for the Z coordinate. They also mention in the paper that the 3D Random Waypoint model that they implemented failed to achieve a steady state, a problem that is inherent to the Random Waypoint model. However they do highlight the possibility of extending the ns2 code to cater to 3D simulations.

Summary for 3D Mobility Models

Even though work has been done for mobility modeling and path planning for flying robots and UAVs which work essentially in a 3D environment, but the models presented have been approximated as a 2D scenario. The contributions from the robotic community are largely limited to specific scenarios like navigation and map building within an indoor environment rather than 3D space.
Therefore, on the whole, not much has been done for the case of mobility control for 3D space. There does not exist; a universal and generalized algorithm for 3D communications networks, like the Random Waypoint mobility model for 2D ad-hoc networks.
Therefore, this remains an open statement to be explored. There is possibility of future work largely oriented in direction, to identify special constraints for 3D motion, to maintain swarms of flying robots in 3D space etc.

Due to lack of a well defined control mobility algorithm for 3D, we decided to extend the 2D mobility control algorithm [8] to 3D.

Chapter 10

Extending ns-2 for 3D

3D provisions in ns- 2

The network simulator ns-2 has been used by the communication community for decades for performance evaluation of protocols, evaluating traffic etc. But, all these simulations have been restricted to 2D cases only. The version of ns-2 that we used was ns-2.27 (for all further discussions).

- Even though ns-2 has the provision for specifying the third coordinate of a node, it is seldom put to any use, and by default taken as zero. Moreover; majority of protocols available in ns-2, work only for 2D.
- However, the radio propagation models (i.e. FreeSpace Propagation Model) contain appropriate code, which takes into account all three coordinates of a node's position while making distance calculations.
- On the other hand, there are o provisions to define 3D topologies; the topography for a simulation can be defined only as a flat grid.

So this required many changes to the core ns-2 simulator code, in order to run 3d simulations.

3D extensions for ns-2

The 3D implementation of the mobility control algorithm involved extending all functions defined for the 2D case into 3D. In order to extend the mobility control algorithm into 3D the following changes had to be made:

- Under the swarm module another class was defined- 'SwarmNode3d' which also inherits from MobileNode class
- The functionality of a 'SwarmNode3d' object is similar to a 'SwarmNode' except that its motion is along all 3 axes
- Similar to 'SwarmNode' , the 'SwarmNode3d' class is also binded with Tcl and allows user to configure the 3D node
- The SwarmNode3d is also linked with the AODV routing agent in order to obtain information regarding the adjacent hops of a swarm node.

Similar to 2D, for the 3D case the user can set a parameter ‘motion3d_’ as ON or OFF to enable or disable the 3D motion respectively.

```
# configure node

      $ns_ node-config -adhocRouting $val(rp) \
                  -llType $val(ll) \
                 …..
                  -phyType $val(netif) \
                  -swarmNode OFF \
                  -motion3d_ ON\
                  -channel $chan_1_ \
                  -topoInstance $topo \
                  -
```

Setting ‘motion3d_’ parameter ON also ensures that mobility will be controlled, so the user does not have to explicitly set the control mobility parameter.

Also, our 3D extensions allow the user to create 3D topologies with the help of an added command ‘load-3dgrid’

```
# set up topography object
set topo    [new Topography]

$topo load_3dgrid 300 300 300
```

The rest of the functioning, i.e. setting the traffic, spreading the nodes randomly at start etc remain the same

By default, ns-2 does not provide complete 3D positional information in the trace file. So changes had to be made in order to generate trace files which contain information for all three coordinated of the node position.

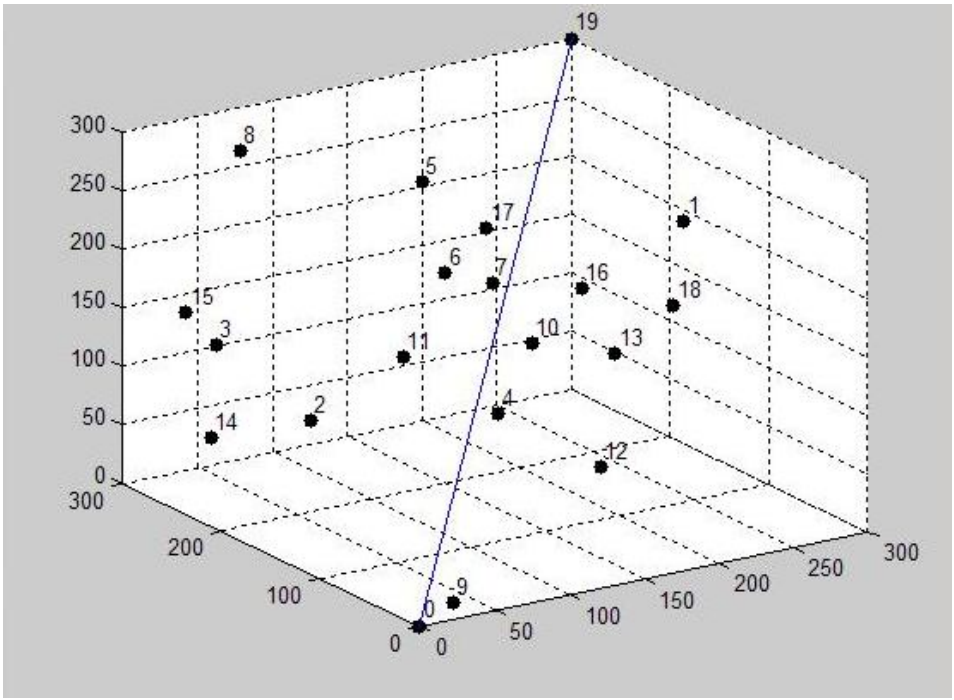
Chapter 11

Results: 3D control algorithm in ns-2

For the 3D case, the node density was kept higher than for 2D. Simulations were made for different values of the range.
The results for different scenarios for the 3D case are presented below:

3D Simulation 1:

- Traffic: Singleflow, AODV
- Source: 0, Destination: 19,
- Range 250 m (default)
- Topology: 3D grid: 300 x 300 x 300



Initial distribution

- First, the nodes are randomly distributed in 3d space
- Then the traffic is started, and the AODV protocol finds the optimum route
- Then the control algorithm is started

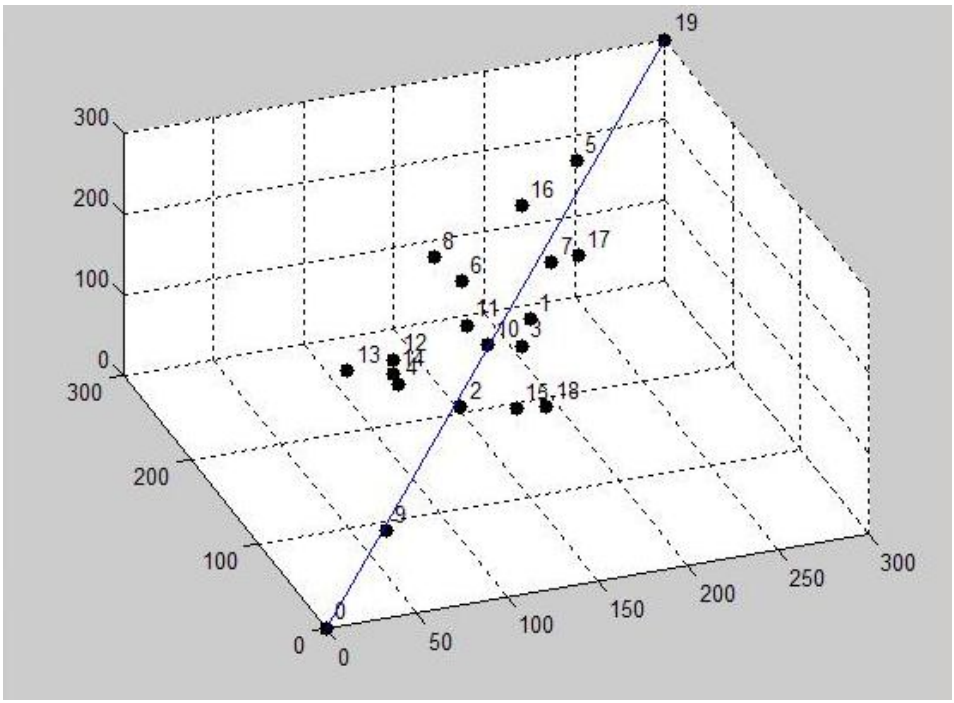
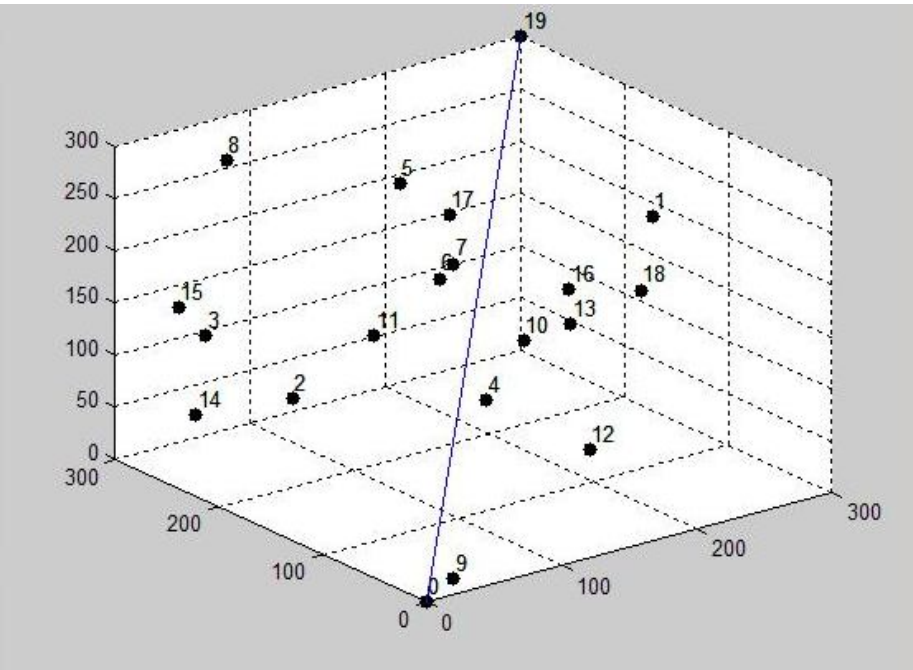


Figure 13 3D simulation 1 final node distribution; route: 0->9->2->-7>19

It can be seen from the above figure that the nodes that are a part of the traffic are indeed aligned along the line joining the source and the destination.
Therefore, the mobility control algorithm that was proposed for 2D even functions for 3D.

3D Simulation 2:

- Traffic: Singleflow, AODV
- Source: 0, Destination: 19,
- Range 100 m (reduced range)
- Topology: 3D grid: 300 x 300 x 300



Initial distribution

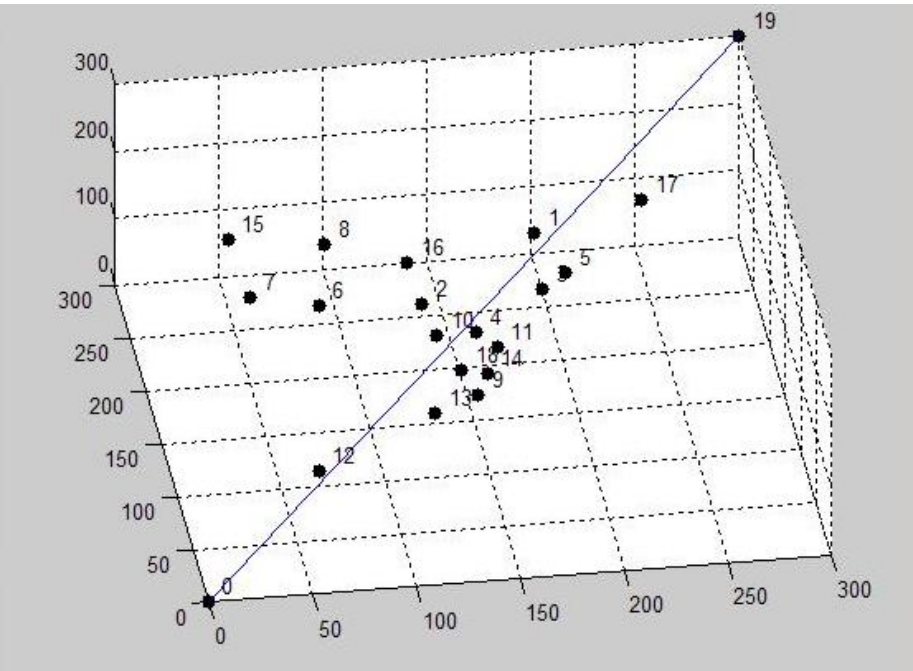


Figure 14 3D Simulation 2 Final distribution, route 0->12->10->1->17->19

Reduced range leads to more no. of hops in the optimum configuration.

Chapter 12

Summary – Contributions and Future Work

In the project, we implemented a mobility control algorithm [8] in the ns-2.27 packet level simulator, thereby enhancing the utility of the simulator as a tool for investigation in networked robotics. Our work allows the possibility to study better, controlled mobility in order to improve wireless communication network performance. We added a new module 'Swarm' to the existing ns-2 structure, that allows the user to run simulations with controlled mobility in both 2D and 3D.
We tested the control algorithm for its validation and presented results which highlight that the algorithm in ns-2 converges with time to the optimum configuration as is expected [8]. Our module supports both single and multiflow for 2D cases.
All our ns-2 extensions to the code were made following all conventions in the ns-2 structure. We also ensured to include maximum extended code in new module files and made only necessary and minimal changes to the existing code. Further our 'module' allows the user to easily configure the node from the script itself by just specifying few newly added parameters.
We further extended the mobility control algorithm from 2D to 3D space. 3D mobility is an area in which very limited work has been done. Our ns-2 3D extensions allow the user to create and model 3D networks with controlled mobility. This is a good first step towards modeling control algorithms in 3D space in a network tool (ns-2) which is very popular within the research community.

ns-2 has been largely used for simulations by the communication community for decades, the ability to simulate mobility control algorithm is a measure to extend the functionality and allow networked robotics simulations to be carried out as well. This interface between the communication and robotics/ control theory has been little explored until now and is sure to be a promising and important area for future exploration.
The future possibilities include some short term improvements in exploring and adding more mobility and swarm control algorithms into the simulator and proceed towards making it a more versatile complete research tool. Moreover, we have used AODV as a routing protocol; one can integrate the control algorithm with other protocols as well. Very little has been done in the direction to enhance ns-2 with interface to control-theoretic approaches so some long term improvements lie there.
We believe we have removed some of the limitations in the existing research, and hope to have made a contribution by sharing what we have learned.

References

[1] Tracy Camp, Jeff Boleng, Vanessa Davies: A Survey of Mobility Models for Ad-Hoc Network Research, *Published in Wireless Communication & Mobile Computing(WCMC):Special issue on mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483-502, 2002

[2] Xiaoyan Hong, Mario Gerla, Guangyu Pei and Ching-Chuan Chiang: A Group Mobility Model for Ad Hoc Wireless Networks, In *Proceedings of the ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM)*,August 1999.

[3] Siddharth Subhash Sail: MSc Thesis 'On the applicability of Random Mobility Models for Swarm Robot Movements' *Kate Gleason College of Engineering, Rochester Institute of Technology, Rochester New York, February 2007*

[4] Elizabeth M. Belding: Routing Approaches In mobile Ad-Hoc networks, *Mobile Ad Hoc Networking, Edited by Basagni, Giordano and Stojmenovic*, ISBN 0 471-37313-3 c 2004 Institute of Electrical and Electronics Engineers. Inc

[5] Network Simulator-ns2-http://www.isi.edu/nsnam/ns/tutorial

[6] Jungkeun Yoon. Mingyan Liu. and Brian Noble. *Random Waypoint Considered Harmful*. In Proc of Infocom. 2003.

[7] J.-Y. Le Boudec and M. Vojnovic, "*Perfect Simulation and Stationarity of a Class Of Mobility Models*", Best Paper Award Infocom 2005

[8] Towards Mobility as a Network Control primitive: *David K. Goldenberg, Jie Lin, A. Stephen Morse, Brad E. Rosen and Y. Richard Yang, Mobihoc'04 May, 2004, Roppongi, Japan*.

[9]Corke, P., Peterson, R., and Rus, D. 2003. Networked robots: Flying robot navigation using a sensor net. *In the 11th International Symposium of Robotics Research. Sienna, Italy.*

[10]E. Kuiper and S.N. Tehrani ' Mobility Models for UAV Group Reconnaissance Applications' *Wireless and Mobile Communications, 2006. ICWMC '06. International Conference on 29-31 July 2006*

[11] S. Nanda and Robert S Gray ' Multipath Location Aided Routing in 2D and 3D'

[12]KIM T. S., CHUNG M. Y., SUNG D. K., SENGOKU M.' Mobility modeling and traffic analysis in three-dimensional indoor environments' *IEEE transactions on vehicular technology ISSN 0018-9545*

[13] Nathan Michael, Calin Belta, Vijay Kumar. 'Controlling three dimensional swarms of robots'. Robotics and Automation, ICRA 2006. *Proceedings 2006 IEEE International Conference, May 15-19, 2006*

[14] Y.B. Ko, N.H. Vaidya,"Location-aided routing (LAR) in mobile ad hoc Networks", *Proceedings of the ACM/IEEE MOBICOM, 1998*

[15] Communications System Group, Institut TIK, D-ITET, ETH Zurich- http://www.csg.ethz.ch/

List of Figures