AD HOC PACKET ROUTING SIMULATION AND TACTICAL PICTURE DISPLAY TOOL FOR NAVY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR AYMAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

AUGUST 2004

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certified that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ahmet Coşar
Supervisor

Examining Committee Members

Prof. Dr. Asuman DOĞAÇ	(METU Ceng)	_____
Assoc. Prof. Dr. Ahmet Coşar	(METU Ceng)	_____
Prof. Dr. Faruk POLAT	(METU Ceng)	_____
Assoc. Prof. Dr. İsmail Hakkı Toroslu	(METU Ceng)	_____
Arif Selçuk ULUAĞAÇ	(Dz.K.K.lığı)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Onur AYMAK

Signature :

ABSTRACT

AD HOC PACKET ROUTING SIMULATION AND TACTICAL PICTURE DISPLAY TOOL FOR NAVY

Aymak, Onur

M. S., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ahmet Coşar

August 2004, 70 Pages

The importance of communication is vital in wartime. The capability of having all the position information of the allied and enemy forces in a single Tactical Information Display System (TIDS), maintains a great advantage for deciding what to do before the enemy reacts. A Naval Information Distributing System (NIDS) is developed for building an effective communication infrastructure between the war ships. In the designed network, besides the mobile platforms (ships), some fixed platforms (land stations) are used to transfer the information coming from these mobile platforms to all the other platforms. To demonstrate the performance and effectiveness of the Naval Information Distribution System, a discrete event simulation model is developed on a Geographic Information System. The goal of this thesis is to describe and experimentally evaluate an effective and feasible information sharing and routing system for Navy.

Keywords: Broadcast routing, sink tree, ad hoc on demand distance vector routing, geographical coordinate system, Navy, information distributing.

ÖZ

DENİZ KUVVETLERİ İÇİN AD HOC PAKET YÖNLENDİRME SİMÜLASYONU VE TAKTİK RESİM OLUŞTURMA ARACI

Aymak, Onur

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Danışman: Asst. Prof. Dr. Ahmet Coşar

Ağustos 2004, 70 sayfa

Savaş zamanı, muhabere hayati bir önem arz etmektedir. Bütün müttefik ve düşman unsurları tek bir Taktik Bilgi Gösterim Sistemi'nin üzerinde izleyebilmek, bir sonraki adımı düşmanın reaksiyonundan önce değerlendirebilmek için büyük bir avantaj sağlayacaktır. Deniz Bilgi Aktarım Sistemi, savaş gemileri arasında efektif bir muhabere sağlamak amacıyla geliştirilmiştir. Kurulan network üzerinde, mobil istasyon olan savaş gemileri ile birlikte sabit olarak tasarlanan ve savaş gemilerinden gelen bilgilerin diğer birliklere aktarımını sağlayan kara istasyonları da kullanılmıştır. Deniz Bilgi Aktarım Sistemi'nin performansını ve efektifliğini gösterebilmek maksadıyla Coğrafi Bilgi Sistemi üzerinde çalışan bir simülasyon geliştirilmiştir. Bu tezin amacı, Deniz Kuvvetleri için etkin ve kullanışlı olacak şekilde, bilgi aktarımının sağlanması ve bu aktarım için uygun yolların bulunmasıdır.

Anahtar Kelimeler: Yayınlama yönlendirmesi, batış ağacı, geçici ve isteğe bağlı uzaklık vektör yönlendirmesi, coğrafi koordinat sistemi, Deniz Kuvvetleri, bilgi dağıtımı

To My Family

ACKNOWLEDGEMENTS

I express sincere appreciation to Assoc. Prof. Dr. Ahmet Coşar, for his guidance and encouragement throughout the research. I offer sincere thanks to my family for their emotional support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

NID (S) : Naval Information Distribution (System)
TIDS : Tactical Information Display System
GIS : Geographical Information System
AODV : Ad hoc On demand Distance Vector routing
VLF : Very Low Frequency
LF : Low Frequency
MF : Medium Frequency
HF : High Frequency
VHF : Very High Frequency
UHF : Ultra High Frequency
SHF : Super High Frequency
EHF : Extremely High Frequency
AM : Amplitude Modulation
FM : Frequency Modulation
RREQ : Route Request packet
RREP : Rouet Reply packet
INFPAC : Information packet
NTPAC : New Track Packet
RPF : NIMA's Raster Product Format
MVC : Model View Controller
DMS : Degree Minute Second format
TCG : Türkiye Cumhuriyeti Gemisi
NM : Nautical Miles (1 NM is 1852 meters)

CHAPTER 1

INTRODUCTION

Naval military operations must be fully exercised and fastidiously planned. Especially in wartime, these operations should be decided and completed in a very short period of time. By executing exercises on simulations which could be very similar to the real life situations, the success of the operations can be obtained. This success directly depends on the ability of executing operations before the enemy's response.

Like all military forces, Navy has two main goals in wartime. First, detecting the enemy's position and then making the decisions about attacking or escaping. The most difficult one is the first goal. Detecting the enemy from one platform can be thought as easy but it is not so easy to transfer this information and merge all the position information of the enemy in a single platform. When a ship detects or receives position information about an enemy platform, the commander has a short period of time to make decision about what to do before the enemy detects him and starts reaction. To increase this time period, some information distribution systems are currently beeing depeloped.

The Naval Information Distribution (NID) system is developed to maintain usable, easily configured, flexible and extensible information distribution. As current information distribution systems are frequently not capable of meeting the Navy's expectations, the proposed NID system brings on a new view to information sharing.

As the most important components of a Navy, the war ships have a Tactical Information Display System (TIDS), which displays the positions of the allied ships and the other (hostile or neutral) kinds of platforms using a Geographic Information System (GIS). This information enables a ship to check for a safe space or to find the uncontrolled area(s). The position information of all the allied ships is maintained and shared by using the NID system and displayed using the TIDS. The most important difference of the proposed NID system is its use of permanently connected land-based

stations, besides the mobile platforms (i.e. war ships), in the formation of the communication network.

1.1 Problem Statement

As previously discussed the currently available naval information distribution systems seem not to be easily configured and do not make use of all of the currently available communication technologies. Since such systems have been developed mostly by foreign countries, the Tactical Information Display systems that are being used by our Navy communicate information only through ships. This thesis aims to do below tasks;

- develop a discrete event simulation of the designed NID system that;
  - uses broadcast, ad-hoc on demand vector routing, and sink tree broadcast algorithms in routing/broadcasting the information packets,
  - creates a desired number of simulated platforms, which work independently moving to their given positions,
  - shows the information coming from the other units in a geographical space.
- demonstrate the correctness of the NID system
- experimentally evaluate the performance of the NID system

1.2 Approach Taken

In order to demonstrate the usability and performance of the designed NIDS, we decided to build a simulation. By using two available simulation tools, the information distribution simulation can be controlled and displayed.

The study is performed in three phases: literature survey, development of the model, and simulation design & testing. In the literature survey, network routing algorithms and the specifications of Geographic Coordinate System are examined. In the second phase, the NID system is introduced and its properties are described. In the last phase, construction of the simulation is defined, the properties of the simulation tools are explained, the components of NID system are introduced and the results of the

implementation are presented. Besides, some scenarios are given and the correctness and effectiveness of the NID system is examined.

1.3 Thesis outline

This thesis develops a graphical simulation of ship movements that allows the users to visualize the effects of the movement and other data on a geographical display.

To avoid the reliance on closed, proprietary systems, the software developed in this thesis is built using open source software. The thesis is organized as follows:

Chapter I gives the main information about the thesis, why there is a need for information distribution.

Chapter II includes some brief information about the radio waves and presents routing algorithms that could be used in a NID system. Also this chapter gives information about the Geographic Coordinate System.

Chapter III gives detailed explaination of the Naval Information Distribution system. In this chapter properties of the ships and land stations are defined, methods to prepare packets and the communication algorithms are explained.

Chapter IV presents the simulation system. First, a brief information about the used simulation tools is given, then the main classes used in the simulation are explained, the process of the simulation is defined. At the end of the chapter the algorithms used for the NID system network are explained and the pseudo-codes of the algorithms are given.

Chapter V presents information about the implementation results and experimental observations of the NID system. Methods used for finding the optimal positions for the land stations and ships are mentioned and implementation details of the simulation are explained with examples.

Finally, in Chapter VI, the conclusions and the possible future works are discussed.

CHAPTER 2

BACKGROUND

In this chapter, radio waves and the radio equipments of the war ships are explained. Then the routing algorithms, in the presence of both mobile and fixed hosts are examined, at the end of the chapter, a brief information about the Geographical Coordinate System is given.

2.1 Radio Waves and War Ship UHF Equipments

The frequencies between 3kHz and 300 GHz are called radio frequencies. The frequency spectrum of radio waves is divided into bands, each band is 10 times greater than the one below it. This arrangement maintains a good way of remembering the ranges of each. Table 1 shows this divided spectrum.

Table 1: Radio Waves Spectrum

DESCRIPTION	ABBREVIATION	FREQUENCY
Very Low	VLF	3 to 30 KHz
Low	LF	30 to 300 KHz
Medium	MF	300 to 3000 Khz
High	HF	3 to 30 MHz
Very High	VHF	30 to 300 MHz
Ultra High	UHF	300 to 3000 MHz
Super High	SHF	3 to 30 GHz
Extremely High	EHF	30 to 300 GHz

The period of a radio wave is the amount of time required for the completion of one full cycle. If a sine wave has a frequency of 2 hertz, each cycle has a duration, or period, of one-half second. If the frequency is 10 hertz, the period of each cycle is one-

tenth of a second. Since the frequency of a radio wave is the number of cycles that are completed in one second, as the frequency of a radio wave increases, its period decreases. A wave lenght is the distance between the identical points of two waves. It is also explained as the space occupied by full cycle of a wave.

The velocity of a radio wave is equal to the speed of light (300.000 kilometers per second) in a vacuumed special environment. Because of various factors, such as barometric pressure, humidity, molecular content, etc., radio waves travel inside the Earth's atmosphere, the velocity of a wave is less than the speed of light. But, practically the velocity of radio waves, referred as its speed at which radio waves travel in vacuumed environment. [5]

The war ships currently use VLF, LF, HF, VHF and UHF radio frequencies to communicate with eachother. Since the ionosphere layer on the atmosphere reflects the radio signals (with frequencies less than 30 MHz), VHF and UHF signals have smaller ranges. But because of non-reflecting of these signals, the loss of the signal can be thought as minimal.

The main UHF communication equipment of the Navy is AN/WSC-3 UHF Radio Set (version 7 and 11). It uses the frequencies between 225 and 399.975 MHz. It has 7000 channels with 25 KHz increments and having the capability of data communication. Its maximum output RF power is 30W for AM, and 100W for FM). The maximum range can change depending on the weather circumstances but practically the range can be thought to be more than 40 nautical miles (NM) and less than 80 NM [6].

Table 2: AN/WSC-3 Radio Sets Properties

VER	**# OF CHANNELS**	**MIN POWER**	**MAX DISTANCE (min power)**	**MAX POWER** (AM/FM)	**MAX DISTANCE**
VII	7000	1 W	5 NM	30/100	80 NM
XI	7000	Under 1 W	3 NM	30/100	80 NM

2.2 Routing

In a network, when a packet is prepared or received from another unit, the main problem arises: which one of the many possible routes will be used to send that packet. The decision must be given in a short period of time. The algorithm that prepares the routing tables and makes the routing decision is called routing algorithms. There are many routing algorithms for mobile hosts but most of them are for mobile users in a network of fixed routers. In an extreme the routers can also be mobile. In this case two choices of routing algorithms can be considered in network with mobile hosts. Broadcast routing and Ad hoc On-Demand Vector routing.

2.2.1 Broadcast Routing

In some applications there are some messages, which need to be received by all other hosts. Sending a packet to all destinations simultaneously is called broadcast routing. There are many algorithms to make efficient broadcast routing but in this thesis two types of broadcast routing are used. The broadcast for mobile hosts and the broadcast using sink tree algorithm. [1]

2.2.1.1 Broadcast for Mobile Hosts

In a mobile environment, hosts are not static and they could move anytime. So, the other hosts wouldn't be sure about each other's positions. This situation makes the one-to-one communication harder. If any host wants to send a message to all other hosts, the basic routing solution is the broadcast routing. In the designed system, with broadcast routing, any station, which has a message to every host, just sends it with a bidirectional antenna (In wired network this can be simulated by sending the message on all of the outgoing connections). A station, which receives a message from any other host, broadcasts it with the hope that some stations didn't receive that packet and they will receive the forwarded message. Of course this kind of routing wastes a lot of bandwidth, time and energy (especially for wireless transmissions). The first problem arises when any host starts sending a message while another host in its range, is also transmitting. This problem can be easily solved with a channel sensing system. When a station prepares a packet and becomes ready for transmission, it first senses the channel and if the channel is idle then it starts transmitting. However sensing the channel idle does not guarantee that all receiving stations within a transmission's range, are currently idle.

Think about a station “X” that wants to send a message. At that time, any station “Y”, in the range of “X”, should be receiving another message from station “Z” which is not in the range of “X”. In that situation, station “X” hears no transmission and starts sending its packet. Station “Y” hears both of the packets and this would cause a collision. In Figure1, we give a picterial representation of this situation, where circles represent transmission ranges for stations.

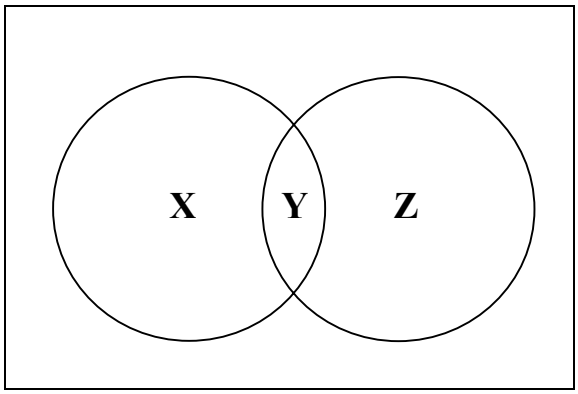

Figure 1: Conflict between “X” and “Z”

Especially in military applications, since the speed of the message delivery can be vital, this kind of conflict must be handled. In a fleet at sea, the solution is to use different frequencies for every ship. Therefor ships have many receiving antennas and they can receive as many messages as their receiving antennas, simultaneously.

Another problem is receiving and re-transmitting the same packet. This wastes a lot of bandwidth. If the receiving host detects that a packet has been processed before, it would ignore that packet. In this case the packet should be compared with a list of previously received packets. This comparison could require a large memory. Inserting time-stamps on each transmitted message can be a better solution; assuming sufficiently synchronized clocks could be maintained on all ships.

In military applications, for extremely important and urgent messages the broadcast routing is used. If the messages are very long and there would be frequent messaging, broadcast routing reduces the efficiency of channel bandwidth utilization.

2.2.1.2 Broadcast Using Sink Tree

In a wired network if the number of hosts and connections increases, broadcasting a message becomes very inefficient because of the duplicate packets. As an example, consider we have 5 nodes connected to each other as in Figure 2. If one of them tries to send a packet by broadcasting method and if there were no checks for duplicate transmission of packet, it would be re-transmit forever. Several algorithms have been developed to solve this problem. One of them is using a sink tree and sending the packets along branches of the sink tree.

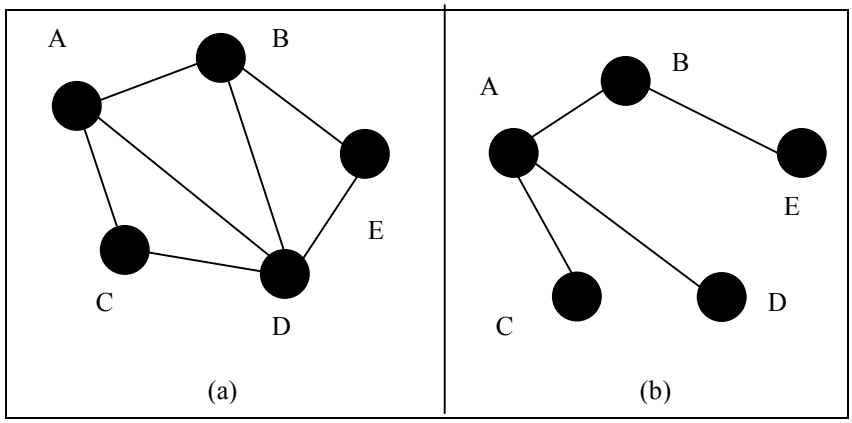

Figure 2: A connected network and the sink tree of it

A sink tree is composed of nodes and some branches, since it is a tree, there are no cycles (loops) on it. If each node knows which of the outgoing lines are belonging to the sink tree, an incoming message would be sent only to those lines except the one that the message arrived from. In Figure 2 (b), a possible sink tree is shown for the network in Figure 2 (a).

By use of a sink tree, the number of times the same packet is received restricted to be only one. So there will be no redundant transmissions. But, the sink tree broadcasting requires all of the hosts to know the sink tree. Creating tables for each host can solve this problem [1].

2.2.2 Ad hoc On-demand Distance Vector Routing (AODV)

The Ad hoc On-demand Distance Vector (AODV) routing protocol is intended for use by mobile nodes in an ad hoc network (which is very suitable for ships at sea). AODV is capable of both unicast and multicast routing. It is an on demand algorithm, meaning that it builds routes between nodes only as desired by source nodes. It maintains these routes as long as they are needed by the sources. It has loop free property and can support large numbers of mobile users [1].

2.2.2.1 Discovering Route to a Mobile Host (Route Request Packet)

AODV builds routes by broadcasting a route request packet and processing (if any) received route reply packets. It maintains routing tables in each node. When a source node desires a route to a destination for which it does not already know a route, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information on their routing tables for the source node (thus automatically building routes to the source of broadcast) and set up backwards pointers to the source node in the routing tables.

The format of the RREQ packet is shown in Figure 3.

Source adress	Request ID	Destination adress	Source Sequence #	Destination Sequence #	Hop count

Figure 3: Route Request packet format

The source and destination adresses are the IP adresses of the sender and destination, they identify who is looking for whom. The request id is a counter for sender's requested messages. Both sorce adress and the request id identify the RREQ packet uniquely. So any host, who has received the RREQ packet having the same sender's adress and same request id, would simply ignore the packet. The source sequence number and destination sequence number fields are used for recording the most fresh route information. After a RREQ packet arrives at a host, its destination sequence number is compared with the destination sequence number locally stored in the table. If the table's destination sequence number is greater than the packet's, then a fresher route is known and the receiving host could answer the RREQ packet; saying "use me". With the source sequence number the receiving host updates the source sequence number in

its table. Hop count indicates the distance from the source. If a fresh route for the destination with less hop count is known then that route will be used to reach the destination and the table will be updated. [1]

2.2.2.2 Responding with a Route (Route Reply Packet)

When a RREQ packet arrives at a station, the receiving station processes the message by checking the following conditions:

1.The source adress and the request id are checked from the history. If the message has been processed before (duplicate message), the process would be terminated and the receiver would ignore that message. If not, it would be written to the history.

2.The receiver looks up the sender's row in its table. As the RREQ packet arrives, the receiver understands that it can reach the source with the packet's hop count steps. If in the routing table, the source distance(hop count) is larger than the packet's hop count, the information in the routing table will be updated.

3.In the next step, the receiver looks for destination adress in the packet. If a fresh route (destination sequence number of the table is greater than the packet's destination sequence number) is known it will unicast a RREP packet back to the source.

4.If the destination route is unknown by the receiver or the destination sequence number of the packet is greater than that in the table, then the receiver will increment the hop count field of the RREQ packet and broadcast it.

The format of the RREP packet is shown in Figure 4.

Source adress	Destination adress	Destination sequence #	Hop count	Lifetime

Figure 4: *Format of the Route Reply packet*

The source adress and the destination adress are taken from the RREQ packet, the destination sequence number is copied from the routing table's destination sequence number. If the RREP packet is prepared by the destination, than the hop count will set to

0, otherwise the hop count field will be copied from the destination distance from the routing table. The life time field controls how long this route will be valid.[1]

Each intermediate host receiving the RREP packet, increments the hop count field and forwards the packet until it reaches the sender of the RREQ packet. Also each host receiving the RREP packet updates its routing table if one of the following conditions occurs.

1.The route to the destination is unknown

2.The destination sequence number of the RREP packet is greater than the routing table

3.The destination sequence numbers are equal but the RREP packet's hop count is smaller than the distance field of the routing table

Updating is done by setting the distance field to the hop count of the RREP packet and setting the destination sequence field to the destination sequence of the RREP packet in the destination row of the routing table

2.2.2.3 Maintenance of Routing Tables

Since every host prepares and update the routing tables, the way to send a message to any other host is known. But especially in military area the intermediate stations can move out of the range or some of them becomes damaged so that they can not transmit any packets. In some case, the messages will not reach their destinations even though the routing tables are being updated periodically. An example of ships at sea is shown in Figure 5, and the routing table of F245 is shown in Table 3.

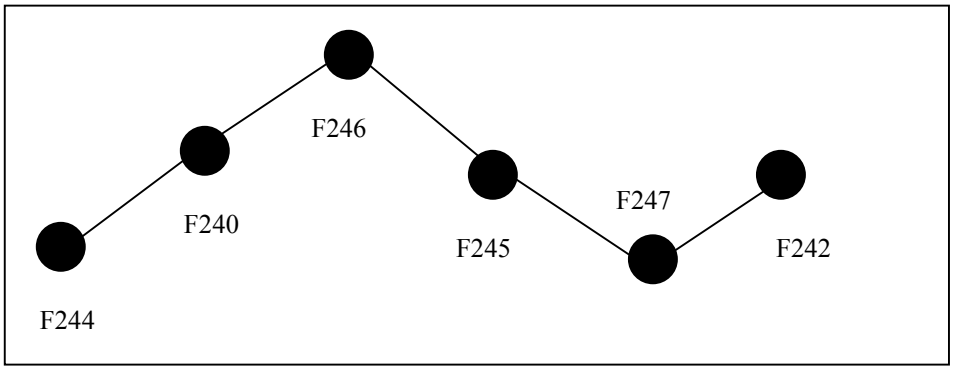

Figure 5: An example of ships at sea

Table 3: A sample routing table for "F245"

Destination	Next hop	Distance	Active neighbours
F247	F247	1	F246,F244
F244	F246	3	F242
F246	F246	1	
F242	F247	2	F244
F240	F246	2	

The *"Active neighbours"* field shows that some of the hosts, which are actively using the table's owner (F245) to send packets at their destinations. If F245 become demaged or moves out of the range, the routes (F244-F240-F246-F245-F247), (F246-F245-F247), (F242-F247-F245), (F246,F240,F244) will be inactive, but F242, F244 and F246 wouldn't know that F245 was gone.

In order to maintain the routing table to be correct each host periodically sends a "hello" message to its neighbours. If all the neighbours reply with "hello" messages, all routes should be active. But, if any neighbour does not reply to the "hello" message, that means it is unreachable from the sender of the "hello" message. Then, the sender must inform the active neighbours of the missing host. In this example if F245 sends "hello" message and if it doesn't receive a reply message from F247 than it would inform F244 and F246(active neighbours of which destination's next hop field is F247) that it could not reach to F247. The route topology changes step by step and routing tables are updated.

2.3 Geographic Coordinate System

2.3.1 Latitude and Longitude

The geographic coordinate system is a spherical coordinate system. It defines two angles measured from the center of the Earth. One angle, called the latitude, measures the angle between any point and the equator. The latitude lines are placed parallel to the equator and divide the earth into 180 equal portions from equator to north or south. The reference latitude is the equator and each hemisphere is divided into ninety equal portions, each representing one degree of latitude. In the northern hemisphere degrees of

latitude are measured from zero at the equator to ninety at the North Pole. In the southern hemisphere degrees of latitude are measured from zero at the equator to ninety degrees at the South Pole. Wherever you are on the earth's surface, the distance between lines of latitude is the same (60 nautical miles or 111.120 meters), so anywhere in the world the distance between the two points can be calculated by using the distance between two latitudes. The lengths of latitudes are not the same; they become smaller towards to the poles [2], [3].

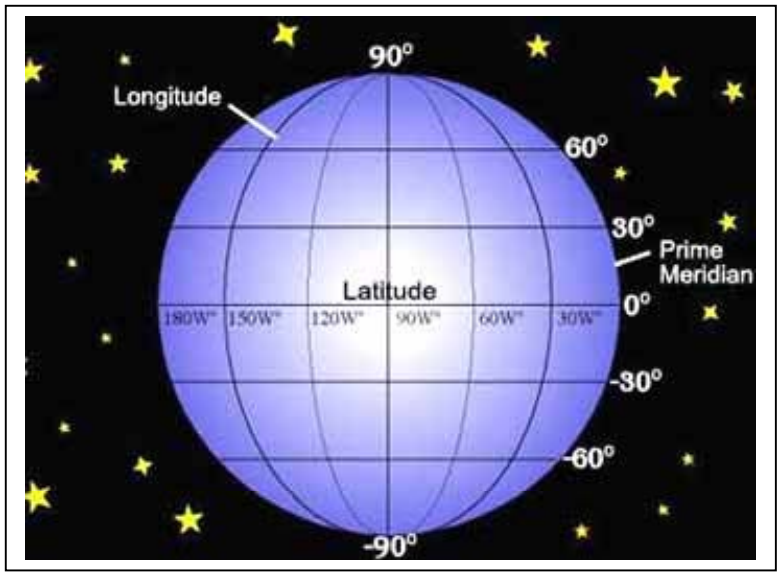

Figure 6: Latitude and Longitude

The other angle, called the longitude, measures the angle along the Equator from an arbitrary point on the Earth. Greenwich (in England) is the accepted zero-longitude point in most modern societies. On the other hand, the distance between the lines of longitude is not static throughout the world. Lines of longitude run perpendicular to the equator and converge at the poles so all the longitude lines have the same length [2].

The reference line of longitude (the prime meridian) runs from the North Pole to the South Pole through Greenwich, England. Subsequent lines of longitude are measured from zero to 180 degrees east or west (values west of the prime meridian are assigned negative values for use in digital mapping applications) of the prime meridian [2], [4].

2.3.2 Distance Over Coordinate System

By combining these two angles, any location on Earth can be specified. The equator is obviously an important part of this coordinate system; it represents the zeropoint of the latitude angle, and the halfway point between the poles. The equator is the fundamental plane of the geographic coordinate system. All spherical coordinate systems define such a fundamental plane.

Lines of constant latitude are called parallels. They trace circles on the surface of the Earth, but the only parallel that is a great circle is the equator (latitude=0 degrees). Lines of constant longitude are called meridians. The meridian passing through Greenwich is the Prime Meridian (longitude=0 degrees). Unlike parallels, all meridians are great circles, and meridians are not parallel: they intersect at the north and south poles [2].

At the equator and only at the equator the distance represented by one degree of longitude is equal to the distance represented by one degree of latitude. As moving towards the poles, the distance between lines of longitude become progressively less until, at the exact location of the pole; all 360° of longitude are represented by a single point. So, by using the geographic coordinate system, grids of lines divide the earth into squares that cover 3600 square nautical miles (or 12347,65 square meters) at the equator [3].

As this range is too long to calculate the location, a map grid is divided into small sections and it is described with an acceptable level of accuracy for the locations of the points on it. Degrees are divided into minutes (') and seconds ("). There are sixty minutes in a degree, and sixty seconds in a minute (3600 seconds in a degree) [3].

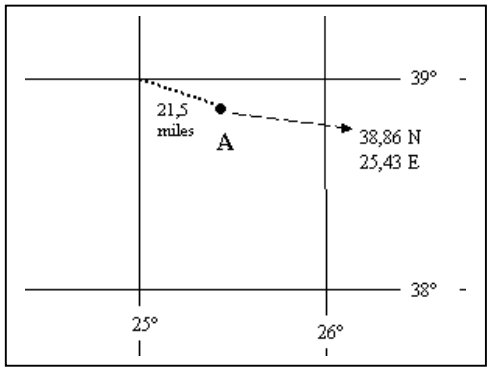

Figure 7: The distance of the point "A"

However, because the accuracy of the fourth decimal place is often uncertain, decimal degree coordinates are often rounded to three decimal places. As an example of measurement the point “A”, which locates at the position of 38°51’36" N, 25°25’45" E is shown in Figure 7. Using decimal degree notation, the location of the point “A” could be written as 38.8600° N, 25.4292° E. This point is approximately 21.5 miles from the point: 39°00’00" N, 25°00’00" E. [3].

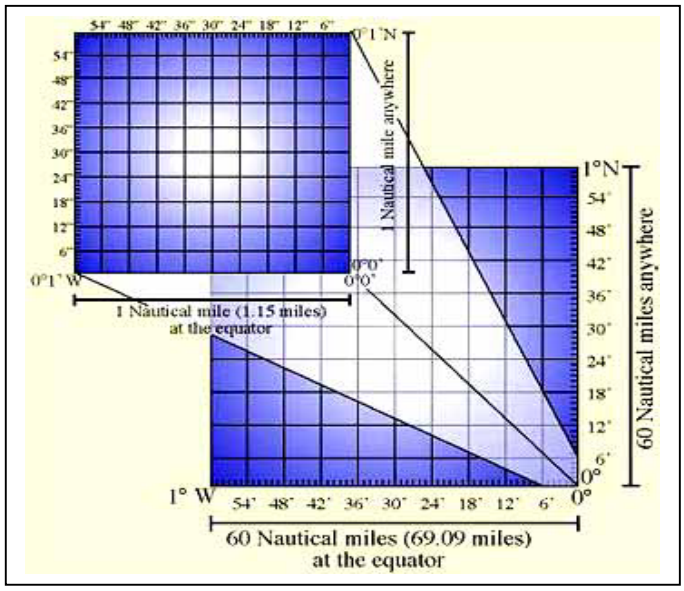

Figure 8: Distance over coordinate system:

CHAPTER 3

NAVAL INFORMATION DISTRIBUTION SYSTEM

In military applications the main priciples are safety, speed and accuracy. In wartime the military forces have two main goals: Discovering the enemy's position, making it ineffective or destroying it. If any unit can detect the enemy's position, the first thing to do is sharing this information with the allied units. The communication between the military units could be even vital. Naval Information Distribution System maintains and shares all the friendly unit position information and detected hostile position information, drawing a tactical picture on geographic spatial display to decide what to do next.

3.1 Why Information Distribution

As a main component of the military, Naval Forces have to work harder to achive the mission by using the most recent technology. This approach gives confidence to the allies and fear to the enemies. As one of the main goals of defending, every allied unit should monitor all of the friendly and the hostile platforms positions, and this information must be distributed in a safe, fast and reliable manner.

3.2 Definition of the NID System

Naval Information Distribution System (NID) is basically a wireless network system, which has mobile and fixed hosts. The units of the NID system are:

1. War Ships (mobile)
2. Land Stations (fixed)

As the ship positions may change, the (wireless) network between ships has no permanent links. The Ultra High Frequency (UHF) radio range restricts the communications between the ships. UHF range cannot be determined exactly as a certain

distance. This range depends on weather conditions and geographic properties of the current location.

3.3 Properties of the NID System

As mentioned above, the NID system contains a wireless mobile network and a fixed network made up of ships and the land stations, respectively. Each platform type has a different property. Their common task is to transport the received packets to all of the ships.

3.3.1 Packets

The communication is maintained via packets. There are two types of packets. One of them indicates the allied ship information. This type is called information packet (INFPAC), the other type of packet indicates the hostile platforms. The second type is called new track packet (NTPAC).

3.3.1.1 Information Packet (INFPAC)

All allied ships prepare this type of packet periodically. The time period is adjustable by the ship, and if it is selected to be very short, it increases the network traffic and may cause congestion. But if the time period is selected too long, then the information accuracy will not be dependible.

The format of the INFPAC is shown in Table 4.

Table 4: An example of information packet of "F246"

Message #	Track id #	Track Position	Course	Speed
21	F246	39,02 N - 25,96 E	122	12

The message number is incremented by sender each time a new packet is prepared. The track identification number, which indicates the sender of INFPAC, position, course and the speed values are written in the packet respectively. The message number and the track identification number uniquely identify the INFPAC. If two INFPACs have the same track ids but different message numbers, the packet, which has the larger message number, shows the fresh information about the track.

3.3.1.2 New Track Packet (NTPAC)

As one of the properties of the war ships, they use their radars to search for the area around them. If any ship detects a track, it first copies the position of this new track, gives track identification number to the track and tries to calculate its course and speed. In a few seconds the calculation should be finished and the ship prepares a new track packet. The new track packet's format is shown in Table 5.

Table 5: An example of new track packet of "F247"

Message #	Type	Sender Id	Target Id	Position	Course	Speed
1	Hostile	F247	NT01	38,72 N 24,95 E	067	15

In this packet type, message number, sender identification number and the target identification number uniquely define the message. So, if the same sender sends two NTPACs about the same target then the NTPAC with the smaller message number will be ignored.

Please note that this thesis doesn't attemp to identify whether two track reports (e.g. same enemy ship could be reported by two or more allied ships) actually correspond to one one real object. Also if a track is lost and re-discovered it is identified as a new enemy ship.

3.3.2 War Ships

Figure 9: A war ship

The most important property of the NID system is having mobile, orientable platforms. The war ships are not only the most effective force on water; they also maintain a great mobility freedoom. The basic properties of the war ships in the NID system are as follows

1. Moving to the given coordinates to control some area

2. In a short time period preparing & sending information packets, which inform the identification number, position, course and speed of the war ship

3. Searching the area around within the range of its radars (surface radars, air radars etc.).

4. Giving an identification number and trying to identify it if it detects any track. Also preparing a new track packet declaring the id, position, type, speed and course of the new track.

5. Building a routing table, expressing where the other ships are currently located and how to reach them.

6. Receiving the information packets from the other allied ships to maintain the routing table. Marking the incoming id numbers, positions, courses and speeds into the Tactical Information Display System (TIDS). Broadcasting these packets in order to be received by all allied neighbouring ships.

7. Marking the id, position, speed and the course of the new target into the TIDS, if any new track packet is received. Re-broadcasting the packet using a broadcast algorithm.

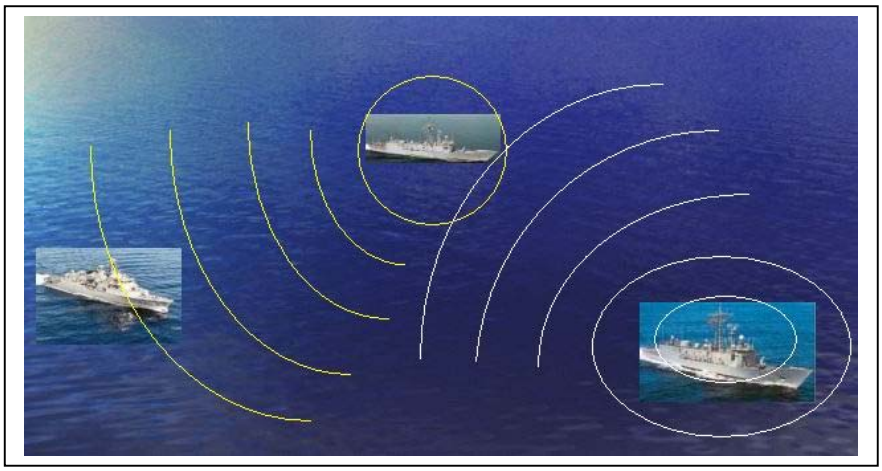

Figure 10: Transporting of a packet

3.3.3 Land Stations

The fixed part of the NID system consists of land stations. These stations are located beyond the coast of the land. Only allied ships know their positions. As long as they are working silently (no wireless transmissions), detecting any land station is a very difficult task. The basic properties of the land stations:

1. They have fixed positions and they are connected to each other with fiber optical cables.

2. They listen to the war ships' packet transmitting channels.

3. If any information packet or new track packet is received from ships, the packet is distributed to every other land station with a sink tree-broadcasting algorithm.

4. If any packet is received from another land station, they broadcast the packet to the ships around them. This could be done by using antennas installed at different locations than land stations, so that such transmissions don't reveal land stations's locations to the enemy. If security is even more important, then mobile platforms could also be employed to carry antennas.

The number of land stations required to cover all of the Aegean coast of Turkey was estimated to be five and using this prototype NID simulation system this was verified.

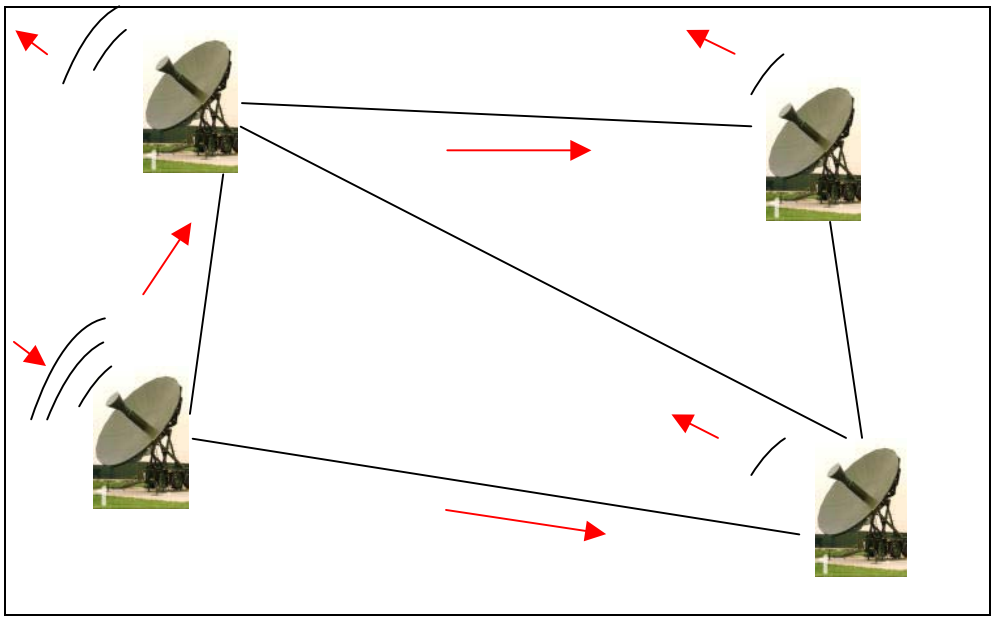

Figure 11: Land Station packet transferring

3.3.4 Communication between the Platforms

Since the INFPACs and NTPACs are confidentional, they must be encrypted. The war ships have the subsystems for encrypting. Normally the messages between the war ships are encrypted by using different types of encryptors.

After the packets are prepared, the ships modulate them to the UHF frequency. After encrypting the packets with UHF encryptors, ships broadcast them by using UHF antennas. Any ship, receiving the INFPAC or NTPAC, has three tasks:

1. Decrypting the packet, unmodulating it and controlling the packet whether it has been processed before. If it has been processed, terminate the subsequent tasks for that packet (i.e. ignore the packet).

2. Recording the information in the Tactical Information Display System and the routing table.

3. Re-transmitting the packet by using the broadcast algorithm.

The land stations have a little different job to maintain the communication. After receiving a packet they do these tasks:

1. Decrypting the packet, unmodulating it and controlling the packet whether it has been processed before. If it has been already processed, terminate the subsequent tasks for that packet (i.e. ignore the packet).

2. Broadcasting the packets that are received from the ships around.

3. Transmitting the packet to the other land stations by the "Broadcast Using The Sink Tree" algorithm (If the land station is not the last host of the sink tree, then continuing to transmit the packet to the branches of the tree).

After these tasks are completed, every other ship, which is close enough (within UHF range) to each other, would receive any INFPAC or NTPAC prepared by any ship. Also if one of these ships is close to any land station, then the packets will be distributed to every land station and to the ships, which are close to these land stations.

3.3.5 Tactical Information Display System

Tactical Information Display System (TIDS) combines the inputs coming from different kinds of systems (such as surface radar, air radar, information coming from other units), allows the user to control any geographic area to decide what to do [5].

Figure 12: Tactical Picture Display System

3.3.6 Assumptions and Limitations

- Since the UHF signal is strong enough, the signal loss in the UHF channels are thought to be minimal. This loss can be tolerated by employing the error correction methods.
- The UHF range can change due to the weather conditions and geographic area. In a sunny, dry weather condition this range could be 80 NM, in a bad weather this range could be 40 NM. We have selected 40 NM as UHF range and believe that this will give realistic results. [6]
- In the NIDS model, only war ships have the ability of searching the area using their radars and identifying the other platforms. The land stations are only used for transferring the packets.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF NAVAL INFORMATION DISTRIBUTION MODEL

In this chapter, the developed simulation tool is described. The simulation's basic classes, which are developed using java programming language and the algorithms used in these classes, are explained with pseudo codes.

4.1 General Overview of the Simulation Tool

4.1.1 OpenMap GIS Toolkit

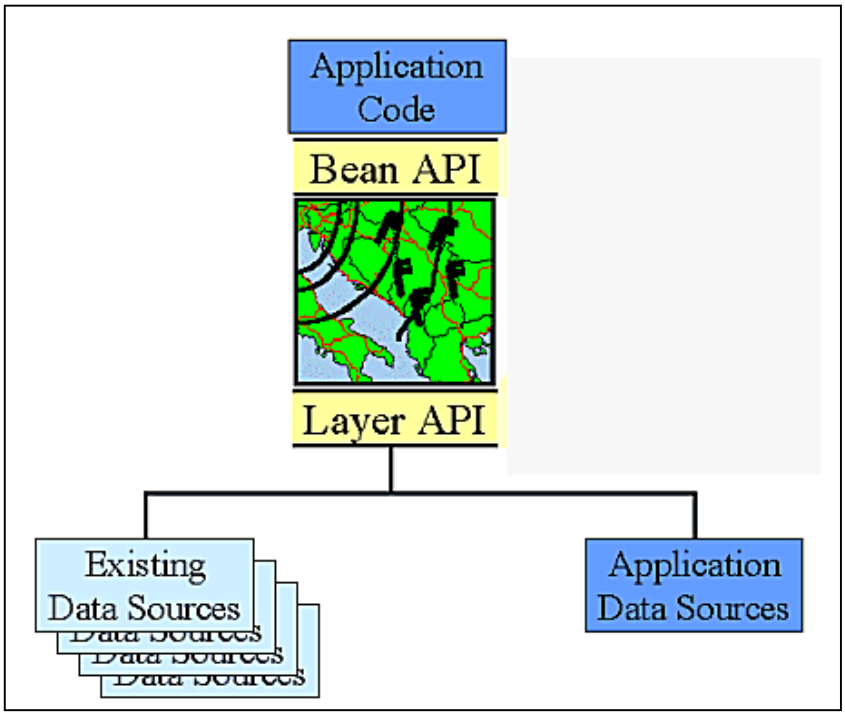

Figure 13: OpenMap map displaying structure [9]

OpenMap is a toolkit for building applications and applets on geographic information system. It is based on Java Beans. By using the Swing components in

OpenMap, it is possible to use geographic coordinates, displaying map data, and handling user input events to manipulate that data. The maps consist of graphical objects , which reacts to user inputs. OpenMap has some Java Swing components called "layers", which are responsible for drawing themselves as part of the whole map. The "layers" are independent and this freedom helps them for accessing their data sources and creating their graphics for the map. Layers can act as clients, creating graphics from data received from a server, or simply displaying graphics acquired from a server [8].

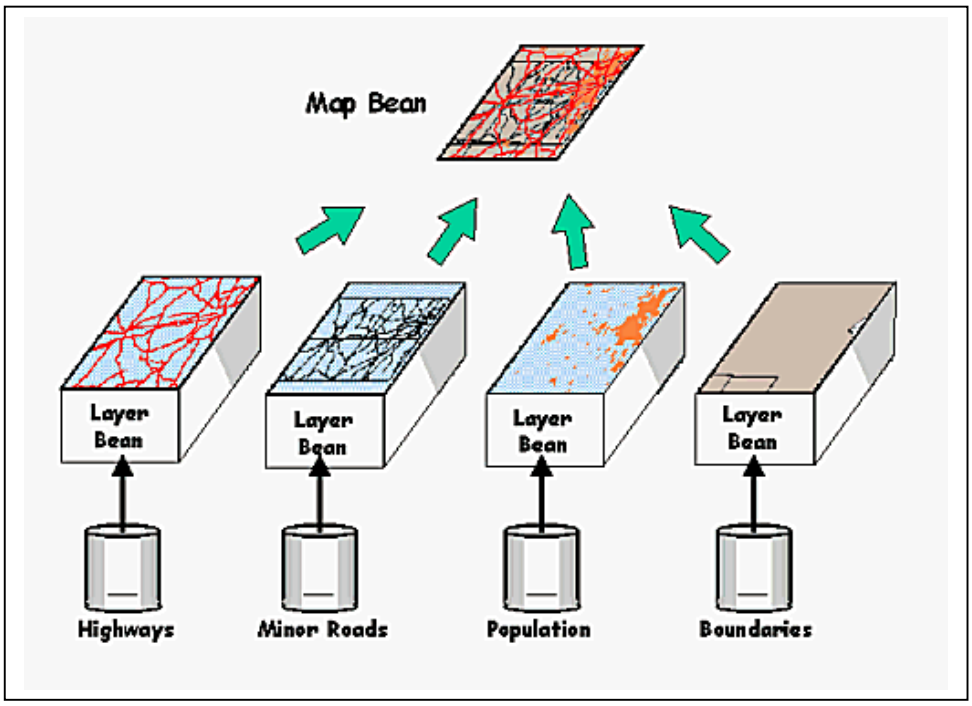

Figure 14: MapBean Architecture [9]

The OpenMap architecture has the mechanisms to listen to mouse and keyboard events for the Layers that want to receive them. Each Layer has the capability to change a graphic's appearance, add or delete graphics, or provide more information about a graphic. The graphic's information can be displayed, via the Information Delegator, in a Web browser, a text line section, or a pop-up window [9].

MapBean is the main part of the OpenMap architecture. It is a Swing component that is a map window. To define what the map should look like, the MapBean needs a "Projection". The Projection has a scale, a center latitude and longitude, a window pixel height and width, and a projection type. All of these attributes

are done for describing the map in the MapBean window. To place graphics on the map, layers need to be added to the MapBean. Layers create graphics from a data source and they are awaked when any attribute of the MapBean's projection is changed. After the projection is changed, they are expected to modify their list of graphics according to the parameters of the projection, and pass the Java Graphics (received in the layer's paint method) object to the graphics so they can draw themselves into it. The layers control how and (approximately) when their contributions to the map are drawn [8], [9].

4.1.1.1 OpenMap Components

4.1.1.1.1 MapBean Component

The MapBean, is a Java Swing component that displays a map window. It holds a reference to the projection object. In this reference, the description of how the map should be drawn (latitude/longitude location, scale, projection type, pixel height and width of the map) is defined. The MapBean is also the parent class to the layer objects, which act as child components to the MapBean.

The map can be changed by modifying the projection or by changing the layers of the MapBean. The layers listen for any changes to the projection in the MapBean, update their graphics accordingly, and then redraw themselves [8].

4.1.1.1.2 Projections

OpenMap supports Mercator, Orthographic, and Gnomic projections and also a "cadrg" projection, an Albers Equal Arc projection that is compliant with the pixel spacing defined in NIMA's Raster Product Format (RPF) specification. [8]

The Projection interface allows users read-only access to the current MapBean projection. MapBean updates all the Layers and other Projection Listeners when the view changes. A Projection object is defined with:

- Latitude and longitude of the center point of the MapBean canvas
- Scale

- Height and width of the MapBean canvas

- Projection type (Mercator, Orthographic, etc) [8].

4.1.1.1.3 Information Delegator

The Information Delegator is the object which maintains the communication with the user. To achieve this task, it can get inputs from the text lines on the map, and it can give information to the user by bringing up a message window. It controls the activated layers, and displays status lights that located at the bottom of the map. The status lights show the activation of the layer and they are on if the layer is sending status updates to the Information Delegator [9].

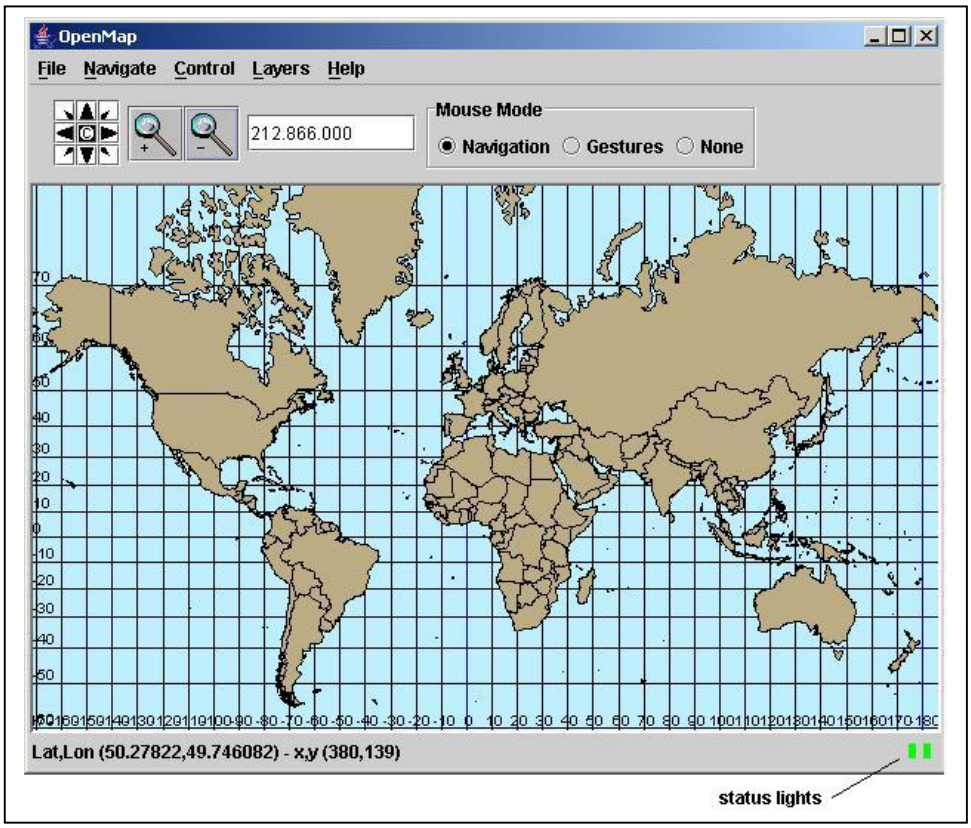

Figure 15: OpenMap Viewer Application

4.1.1.2 OpenMap GUI

The OpenMap Viewer application starts with a map of the Earth with a 10° graticule, as shown in Figure 15. The following user interface items are available. The main apperance of menu is shown in Figure16.

Figure 16: OpenMap Menu bar

4.1.1.2.1 OpenMap Menu Items

a.) File: There are four menu items contained here as shown in Figure 17. The "*About"* item provides information about the underlying OpenMap software which is written at beginning of the "openmap.properties" file. By the "*Open"* item, another "openmap.properties" file can be read if no user created properties file is loaded. The "*Save As"* menu allows to save the map in a picture format (i.e. jpeg). The last item is the "*Quit"* item, which terminates the application.

Figure 17: File menu item

b.) Navigate: Menu items found under the Navigate menu are shown in Figure 18. There are four items: the "*Coordinates"* item which locates the center position of the map to the given geographical coordinate and redraws the map; "*Projection"* which allows the user to set the map display projection; and the "*Zoom In/ Zoom Out"* controls which zoom the map by the specified amount.

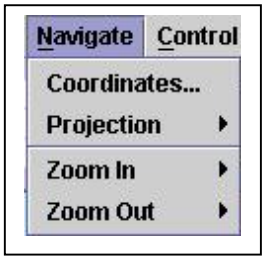

Figure 18: Navigate menu item

c.) Control: The Control Menu, has two items as shown in Figure 19. "*Mouse Mode"* changes the mouse behavior in the following manner. "*Navigate"* allows you to move around on the map. When any point in the map is clicked, the map is redrawn as taking the clicked point to the center of the map. "*Gestures"* passes mouse events through to layers, and "*None"* ignores all mouse clicks. The "*Redraw"* redraws the map.

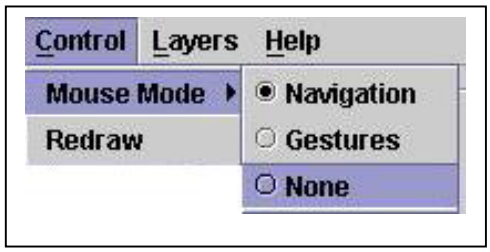

Figure 19: Control menu item

d.) Layer: The Layers menu allows the user to select which layers are to be manipulated. Figure 20 shows an example of Layers menu. They are arranged by the relative position of the layer in the display. The layer display status is controlled by selection of the check box. The "*Edit Layers"* item brings up the layer editor window.

Figure 20: Layers menu item

e.) Help: This menu contains a single item that will bring up help documents in a user specified web browser as shown in Figure 21. This specification should be given in the properties file.

Figure 21: Help menu item

4.1.1.2.2 OpenMap Toolbar Items

The following items can be found on the toolbar. "*Rosette",* the rosette pans the map in the specified direction and the middle button recenters the view to the starting point; "*Magnifying Glass", "*+" Zooms in 2X over the center of the map "-" Zooms out 2X over the center of the map; "*Scale Entry"* allows the user to enter the scale of the map*;* "*Mouse Mode",* Changes the mouse behavior [8], [9].

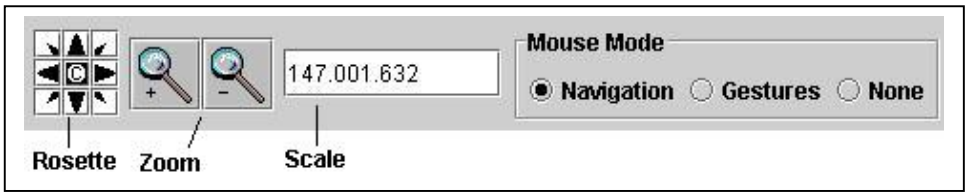

Figure 22: OpenMap tool bar

4.1.1.2.3 OpenMap Layer Editor Window

The layer editor window, shown in Figure 23, uses the following icons to manipulate layers in the display:

- Turn layer off/on;
- Move selected layer to top of map;
- Move selected layer up one level in map;
- Move selected layer down one level in map;

- Moveselected layer to bottom of map.

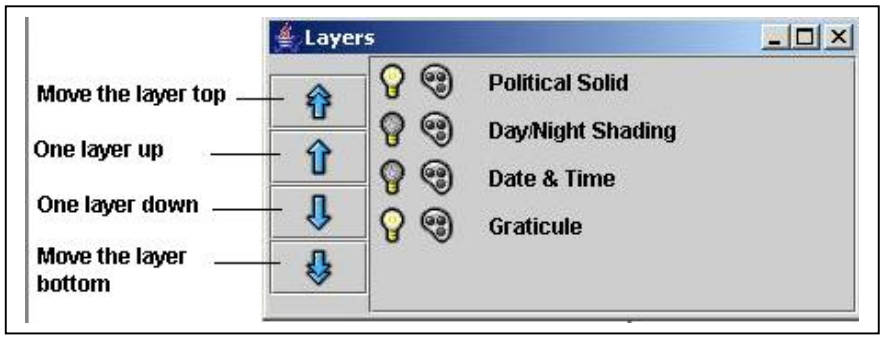

Figure 23 :Layer editor palette

4.1.1.2.4 Coordinates Window

The Coordinates Window, which is shown in Figure 24, is used to specify coordinates in decimal degrees or DMS format (Degrees, Minutes and Seconds). Once the data is entered the user clicks "*Apply*" and the map will be recentered over the position indicated.

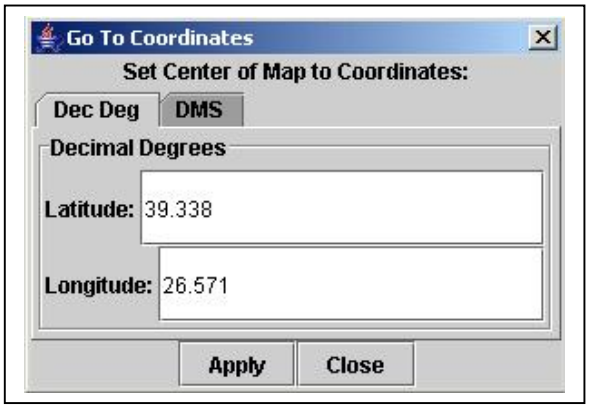

Figure 24: Coordinates editor window

4.1.2 Simkit Simulation Toolkit

As mentioned earlier the Simkit is used for designing discrete event simulation. Since the simulation outputs, which are created by the Simkit, are not visible, another tool called OpenMap is used to show the outputs of the simulation.

4.1.2.1 What Is Simkit?

The Operations Research Curriculum at the Naval Postgraduate School (NPS) uses a Java-based discrete event simulation package, called Simkit. Simkit allows rapid development of discrete event simulation models (Buss & Stork, 1996). This tool provides the foundation for the movement simulation. [12]

Simkit is kind of discrete event simulation tool, which is composed of independent, interacting, concurrently operating components. The main properties of Simkit are as follows.

- An Application Programmer Interface (API) package for easily creating Discrete Event Simulation (DES) models
- Written in Java, runs on any Java 2 platform and modern web browsers
- Open Source
- Installable from the web
- Small execution footprint

4.1.2.2 Simkit Basic Structure

Simkit is a package, which is used for implementing component-based simulation models. It is written in the Java™ programming language, so it gives hardware and operating system independence to the user. Simkit uses Event Graphs methodology [13]. They are composed of three elements:

- Event node,
- Scheduling edge,
- Canceling edge.

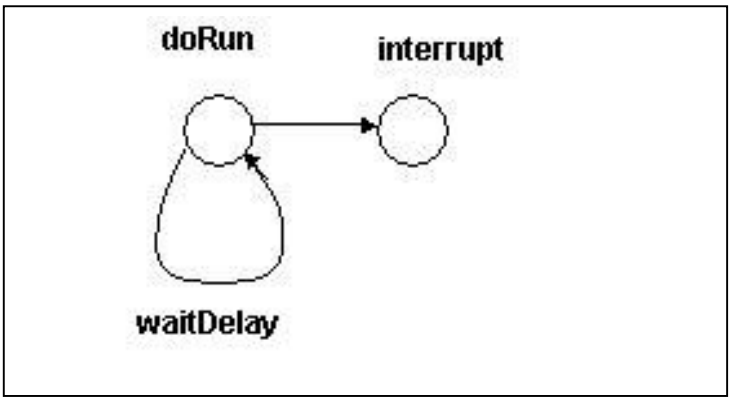

Figure 25: A sample Simkit Event Graph

Simkit Event node corresponds to instance methods having a special prefix (do methods), Scheduling edge is used for processing an instance method called waitDelay(), and the Canceling edge is used for processing an instance method called interrupt(). Simkit's implementation is composed of related interfaces, the most important of these will be explained below.

> *"SimEvents are placed on the **Event** list by a SimEntity object by invoking its instance method waitDelay (String, double). The first argument is the name of the SimEvent, and it is meant to match a corresponding user-written instance method in the SimEntity that gets invoked when the EventList processes the SimEvent. The second argument is the amount of time until the **event** is scheduled to occur, the delay time. An **event** occurs when it becomes the next **event** on the **Event** List. When that happens, the **Event** List removes the **event** and makes a callback to the SimEntity that originally scheduled the SimEvent by invoking its handleSimEvent (SimEvent) method. This approach is a simple and ypical implementation of an **Event** List"* [13].

> *"Simkit uses Java.s reflection to match the name of an event with a method in the SimEntity that has the same name with "do" prefixed. Thus, the SimEvent called "Move" will cause an instance method called doMove() to be invoked on the scheduling SimEntity when that SimEvent occurs."* [13]

4.1.2.3 Main Simkit Classes Used In the Simulation

The main Simkit classes and methods used in the simulation are as follows.

simkit. SimEntityBase: the main class, which schedules all the events of Simkit. As mentioned earlier, it contains waitDelay (), interrupt () and interruptAll () methods.

simkit.Schedule: Allows controlling the simulation. Starting and ending of schedule is done within this class. Events are stored in it and it has a time stamp to schedule the events. It contains startSimulation (), pauseSimulation (), isRunning () and interruptAll () methods.

simkit.SimEvent: The scheduled events on the event list which will be delivered to SimEventListeners.

simkit.SimEventListener: It is an interface, which listens to SimEventSources. It consists of a single method, processSimEvent (SimEvent).

simkit.SimEntity: The interface, which is used for scheduling events on theEvent List.This interface contains the waitDelay() and interrupt() methods.

simkit.smd.Mover: The basic interface for the creation of the components. Each component can be created as mover. Mover class has setName (), getName (), moveTo (Coordinate) and getCurrentLocation () methods, which are frequently used.

simkit.smd.Coordinate: Using for moving the movers. The calculations are done with using the geographic coordinate system positions and the functions give outputs as coordinates. By using the Coordinate class, the coordinates are created to give the positions of the movers.

4.2 Architecture of the Simulation

The basic structure of the simulation can be explained in the following way: Simkit produces non-visual simulation outputs; OpenMap tool displays Geographic Information System (GIS) data as layers; and the NID system simulation program

converts Simkit output data into OpenMap on two layers and provides interactive simulation controls.

4.2.1 Simulation Layer

The developed simulation is an application of OpenMap's event messaging algorithm, which shows animated discrete event layers. The Simulation Layer (SL) developed for this simulation utilizes the a *Model-View-Controller* (MVC) paradigm to separate the simulation and display components. The MVC paradigm is an approach to programming that separates data input, data processing, and data output in such a way that either the input or the output can be modified without having any impact on the processing, see Figure 26.

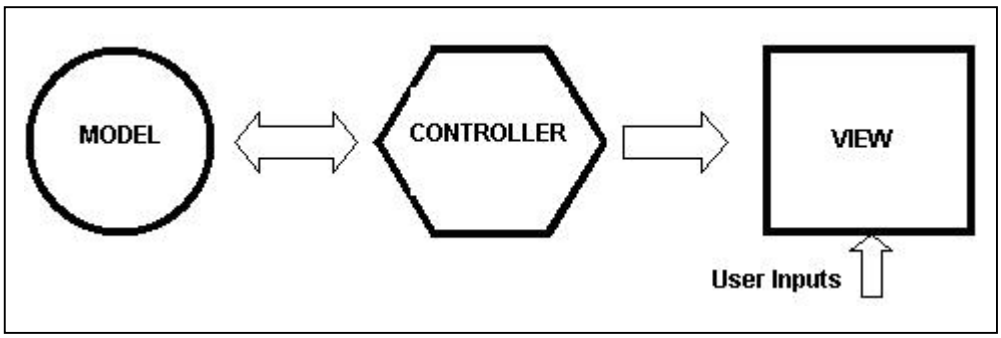

Figure 26: Model View Controller paradigm

The implementation of the MVC paradigm is as follows:

Model: A discrete event simulation utilizing Simkit;

View: It is a geo-referenced icon representing the position of the system simulated in the model;

Controller: A Simkit's mover manager that controls model movement.

The simulation layer is implemented as a Layer Bean. The OpenMap's event passing paradigm is used for communicating with the outputs of the Simkit. All units in the simulation have the Simkit's mover manager characteristics. The positions of the units are referenced via a Java runtime event. This position data is geo-

referenced and displayed in the simulation layer. The simulation is capable of displaying and animating any Simkit entity. The simulation layer is implemented as two similar layers in OpenMap [8].

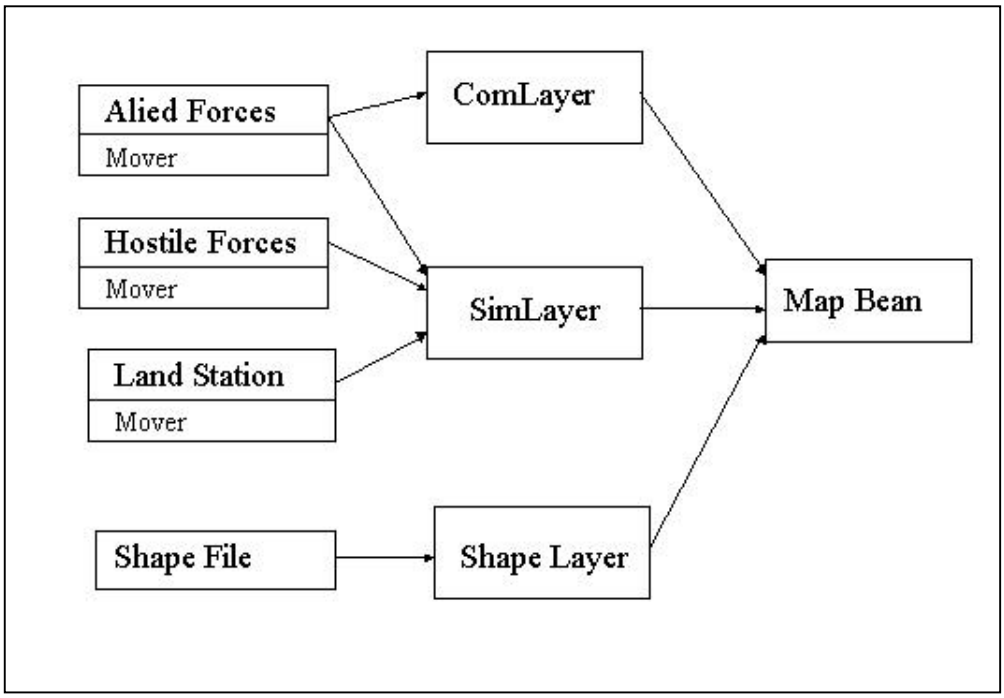

Figure 27: NID system simulation architecture

4.2.1.1 SimLayer

SimLayer is the layer that simulation runs on. Every ship except the commander ship and every land station is shown by SimLayer. When the SimLayer is initiated, the tasks given below are performed:

1. The properties are read from the basic OpenMap properties file and the user properties file.

2. The connections with the SimKit classes are obtained.

3. The ships are created, their numbers and names are given. Their positions and future position are read from the files.

4. The land stations are created, their names and numbers are given.

5. Listens to the mouse events; starting the simulation or clicking on any component

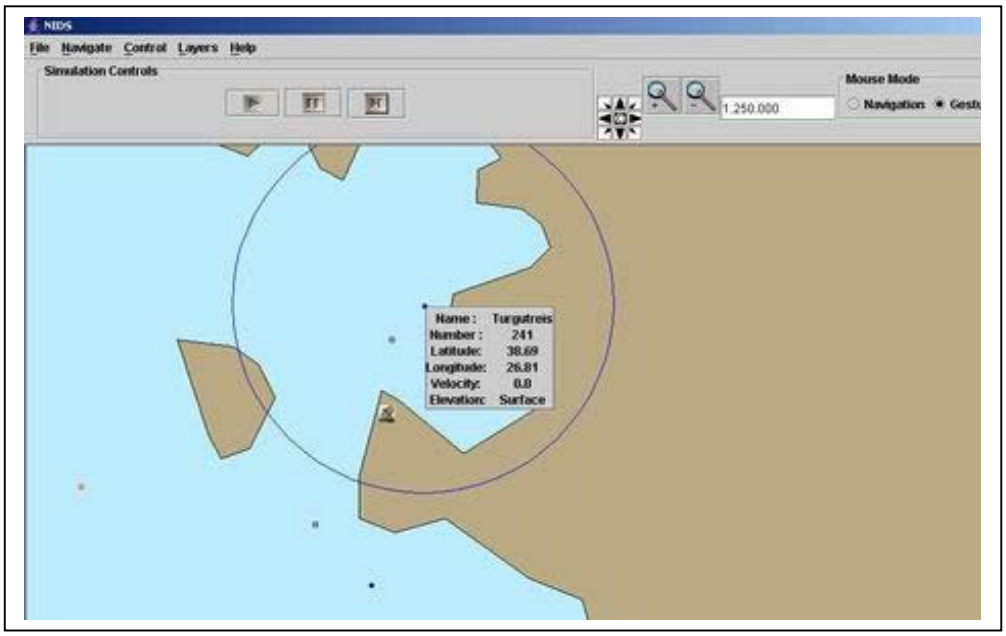

Figure 28 : SimLayer information view

Also this layer allows information to be obtained about components. By clicking the mouse on any component, following information will be displayed:

1. Name
2. Identification number
3. Latitude
4. Longitude
5. Simulation speed (ships only)
6. Elevation (“surface” or elevation of a plane)
7. UHF Range (by a circle on the map)

First six items can be seen in a pop-up window, the seventh one is visualized by a circle, which has the radius of UHF Range. This circle is centered at the selected unit, showing the transmission covarage area of the unit on the map. In an OpenMap application, the MouseListeners are active only when the mouse mode is set to *"gestures"* mode.

4.2.1.2 ComLayer

This layer is similar to the SimLayer, it again accomplishes the reading of the properties tasks. But this layer creates only one ship. This ship can be considered as the commander ship. In the simulation, all units are acting independently and their positions can be observed from the SimLayer. All the ships except the commander's ship move to their positions defined earlier. In a real situation consider a ship having a commander. Normally in a TIDS, the commander can see only his own ship if the ship is not using the NID system. If any packet is received from another ship, then it can be recorded and seen by its TIDS. The ComLayer simulates this situation. When the SimLayer is made inactive, only commander's ship and the ships, which are identified from the received packets will be shown by ComLayer. While the simulation is running the commander ship should be receiving some INFPACs and NTPACs. The information in these received packets are plotted on the map as there are real platforms on those positions. The ComLayer creates artificial icons to show the received packet information. These artificial icons are removed after changing the projection.

As the mouse events are treated also by the ComLayer, this layer implements different functions for the mouse events. By clicking any free point (any position that has no ships on it) on the ComLayer, the commander ship moves to that position while the simulation is running. This gives great flexibility to the user for implementing different movements beyond the arising situations.

After arrival of some INFPACs to the commander's ship, the routing table of the ship is updated. By clicking on the commander ship, a pop-up menu is shown which includes information about the commander's ship (name,id number) and the routing table information. The table shows how to reach the other allied ships and the land

stations, using which platform and the number of hop counts on that route, needed to reach that destination.

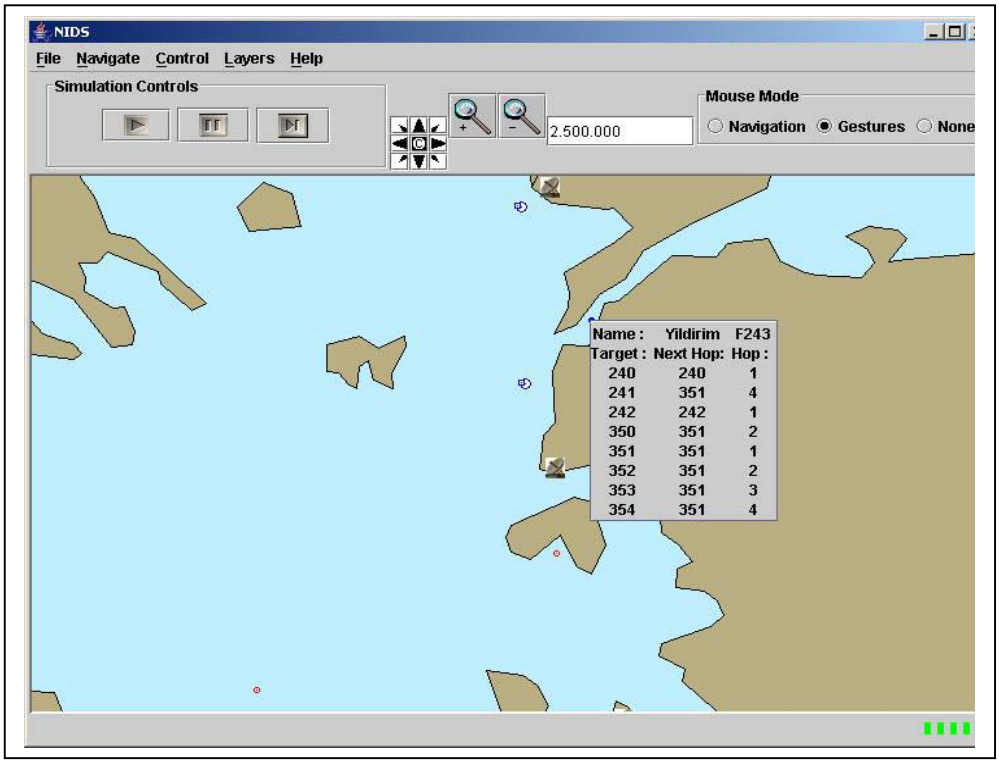

Figure 29: ComLayer information and the routing table

4.2.2 Scenario Of The Simulation

The scenario of the simulation is developed in a plain manner. There are two battalion size forces. With the commander there is a VIP person who has to be protected from the enemies. The main task of the commander's ship, is cruising safely from a given point to another. While cruising, the commander's ship will be receiving the information packets of the allied ships and the new track packets of the hostile ships. So the commander will decide which course to turn for a safe sailing. The schedule of the simulation is given in Table 6.

Table 6: Tasks of the simulation

Task No	Task
1	Create the Commander's Ship Layer
2	Create the Allied Forces Layer
3	Create the Land Station Layer
4	Create the Hostile Forces Layer
5	Read all the ships next coordinates, store them
6	Locate all the platforms on the map
7	Start the schedule
8	Make each ship prepare the information packet
9	Make the ships search their areas
10	If any new track is detected, make the ship prepare the NTPAC.
11	Make the packets to be distributed to every ship which are in-range of NID system
12	Control the position and movements of the other ships
13	Turn off the Hostile Forces Layer
14	Turn off the Allied Forces Layer
15	Check that any information is being received from NID system
16	Move the commander's ship to the safe position
17	After arrival check for the packets on the commander's ship

To achieve the tasks on the schedule, some objects are created and their functions are run. These objects will be explained next, in detail.

4.2.3 Implementation Of The Simulation

The simulation starts with reading the properties file of OpenMap. After the general properties, the layers are created. While creating the layers, some java classes depending on that layers are initialized. The simulation implements the functions in these classes.

4.2.3.1 Reading The Properties File of OpenMap

The OpenMap properties file controls how components are loaded into the OpenMap application. It also specifies initial properties of OpenMap, such as the initial projection of the map, the layers that should be available, the layers that will be active on startup and lets user adjust all the layers' attributes for their startup configuration. Most importantly, this file also lets user add and remove different components from the application itself. User can modify it with any text editor.

4.2.3.1.1 Reading The General Properties

When the simulation is started, first, the properties of OpenMap are read. The original *"openmap.properties"* file is read first, then the *"openmap.properties"* file which is created by the user is read later. The second properties file includes the data which the user created. A portion of an example properties file is given in Figure 30. [9]

```
openmap.Title=NIDS
openmap.Version= 1.0
openmap.BuildDate=June 25, 2004

openmap.Latitude=38.921385
openmap.Longitude=25.9574
openmap.Scale=2500000
openmap.Projection=mercator
openmap.Width=900
openmap.Height=700
openmap.x=-1
openmap.y=-1
openmap.BackgroundColor=FF89C5F9
```

Figure 30: OpenMap general properties ("openmap.properties")

The "Title" field is written as the label of the frame; the "Version" and "BuildDate" forms the main information of the created applet. This information is visualized by clicking the "About" submenu of the "File" menu on OpenMap frame.

The starting position of OpenMap (center of the map) can be set to a required position by declaring the Latitude and Longitude fields. Scale of the map is selected with the Scale field as 1/Scale value. The "Projection" field is used for setting the projection type to start the map with.

The “Width” and “Height” fields determine the dimensions of the frame in pixels. The “x” and “y” fields are used for the position of the top-left corner of the frame on the screen. If they are selected as smaller than 0, the frame is located at the center of the screen. The “BackgroundColor” sets the background color of the map, in hex AARRGGBB values (AA is transparancy, RR, GG and BB are red, green and blue in hex values between 00-FF (0-255)) [9].

4.2.3.1.2 Creating the Layers

After the main properties of OpenMap are read, the layers, which will be used, are created. This layers cause initializing of some classes. As an example, some layers are created and the “allied” layer’s properties are declared in Figure31.

```
openmap.layers= allied hostile landStations
                shapePolitical daynight date

openmap.startUpLayers= allied hostile landStations
                shapePolitical


allied.class=SimLayer
allied.prettyName=Allied Forces Layer
allied.mover=simkit.smd.BasicMover
allied.moverMan=AlliedForce
allied.icon=allied.gif
```

Figure 31: Definition of a layer in “openmap.properties”

The “layers” field lists the available layers. Also the layers, which will be needed at the start of the OpenMap, are listed in the “startUpLayers” field. The user can select the marker names of the layers and each layer’s properties should be given by declaring the marker name of the layer. The user can remove a layer from the application by removing the layer’s marker name from the openmap.layers property list. He doesn’t have to delete all of the layer’s properties as well. If the user wants to add a layer to the application, he should add its marker name to the “openmap.layers” property list, and then add its properties to this file. As a minimum, Layers all need a “marker name”.class property, and a “marker name”.prettyName property (for the GUI components) [9].

Table 7: Layers created in the simulation

OpenMap Layer	Layer Definition Class	Initialized Class
Commander Layer	ComLayer	AlliedForce
Allied Force Layer	SimLayer	AlliedForce
Land Station Layer	SimLayer	LandStation
Hostile Force	SimLayer	HostileForce

In the example, besides the mandatory properties for OpenMap, the required fields of the Allied Forces Layer are also described.

4.2.3.2 Creating the Objects

After reading the properties file the simulation creates these layers:

1. Allied Forces Layer
2. Commander Ship Layer
3. Land Stations Layer
4. Hostile Forces Layer

All the layers except the Commander Layer are actually visualized by the same class: SimLayer. This layer creates the units: Ships and the Land stations. Number of the units is set by the "numMovers" property. It moves the ships and controls the units to do different tasks periodically by using the BasicMover class of the SimKit. This function is set by the mover property. The "moverMan" property allows for each type of units to initialize different classes. The SimLayer creates the movers, and then lets each layer to shape the movers according to the classes of "moverMan" propery. The icons of all layers are set by the icon propery.

4.2.3.2.1 Allied Forces Layer

The AlliedForce class creates the ships, which belong to the Allied Forces Layer. Before calling the AlliedForce class the ships are only movers (SimKit.smd.mover class).

By initializing AlliedForce, the names and numbers of the ships are set. The initial positions and the next positions are loaded from the specified files. All the ships are added to the SimKit Event Listener to do their actions. The velocity of the mover is set by the “vel” propery. The range of communication is set by the UHF propery. After creating the Allied Forces layer all the Allied Forces ships are in the same state: Wait for starting the simulation.

4.2.3.2.2 Commander Layer

It is one of the layers, which includes only one ship. This layer allows the stored INFPACs and NTPACs to be visualized. Also this layer shows the routing table of the commander’s ship.

Like Allied Forces Layer, this layer also initializes the AlliedForce class. It constructs only the commander ship. Locates the icon of the commander ship at the given position within the “latMover” and “lonMover” properties. The UHF and the “vel” properties are set as in the Allied Forces Layer.

4.2.3.2.3 Land Station Layer

This layer is created like the other layers but initializes a different class: LandStation. LandStation class gives each land station a number and a name. While initializing the LandStation class, the positions and the routing information are loaded. Since the land stations are static, the connections are wired with the fiber optical lines. They have a land propery, declaring that it is a land station. This property shows that the station is not allowed to move. After initialiazing the Land Station Layer, a sink tree is created and every station would know how to distribute incoming packets to the other land stations, so that even ships that are not within the UHF range of a transmitting ship will be able to receive packets via a land station that is within its own UHF range. The numbers of the land stations simulates this task in a simple manner: The land stations are created in order of their locations from North to South. The first one is given number 350, second 351 and so on. The sink tree has linear structure and every land station is directly connected to two neighbours and its messages are transported to them in one hop. For example land station number 352 is connected to 351 and 353 directly and if the hop count of the packet is 3 while sending from 352, the neighbours will broadcast it with hop count 4.

4.2.3.2.4 Hostile Forces Layer

Hostile Forces Layer is used to model the enemy ships. It initializes the HostileForce class. The names and the numbers are given, starting positions and the next positions of the enemy ships are loaded from the files while initializing its class.

The enemy ships are also visualized on the SimLayer like the allied ships. The difference from the allied ships is having the "hostile" property to be true. This propery makes the enemy ship different from the friend.

After all these layers are created the icons are located on the map at the starting positions of each unit. Since all the units are added to the SimKit event listener, all of them wait in silence for the simulation start.

4.2.3.3 Starting the Simulation & Maintaining the NID system

Before starting the simulation, all the ships and the land stations are located at their starting positions on the map. As mentioned earlier Allied Forces Layer and Commander Ship Layer implements the AlliedForce class methods, Hostile Forces Layer implements the HostileForce class methods and the Land Station Layer implements the LandStation class methods. After starting the simulation by pressing the "play" button, the simulation becomes active. The units except the land stations move to their next positions with pre-determined velocities. Since each class has different tasks, the class functions are explained in detail.

4.2.3.3.1 AlliedForce Class

This is the main class doing the tasks of the allied ships. All allied ships and the commander ship can perform the functions on AlliedForce. Any object of AlliedForce performs these tasks during a simulation cycle.

1. Prepares the information packets containing the ship's identification number, position, course and speed.
2. Sends this packet by broadcasting to all other allied ships.
3. Controls the area around it to check if there is a new track.

4. If it finds a new track, calculates its course and speed, prepares a NTPAC including the position of the new track.

5. If any packet is received, controls it if the packet is duplicate. If not, processes it:

6. Saves the position of the track to the TIDS.

7. Updates the routing table

8. Broadcasts the NTPAC to its neighbours.

9. Starts to move to the next position

The received information is stored in the received packets array. The AlliedForce unit checks the packet from this array. If the packet is coming from a new ship, the array in the new ship's id index is filled with the information on the packet. If the packet indicates the new position of a known ship, the information in the received packet array, known ship id index, is updated.

4.2.3.3.2 LandStation Class

After the simulation becomes active the land stations start to listen their environments in a passive manner. They do not periodically send packets. While cycling if they received any packet from the ships they perform below tasks.

1. Check if the information packet has been processed before. If yes, terminate processing of that packet

2. Broadcast the packet to the ships within the land station's UHF range.

3. If the packet has been received for the first time or has a hop count less than the peviously processed one, distribute the packet to the other land stations using the sink tree algorithm.

4.2.3.3.3 HostileForce Class

This is the basic class and actually runs no important function. The hostile ships are created within this class and these ships only move to their next positions. They have

a single task, which is searching for the commander ship after arriving at their next locations.

4.2.4 Procedures for Routing the Packets and Updating the Routing Tables

As explained earlier, there are 3 types of classes, which perform the main functions of the simulation. HostileForce class is responsible for moving the enemy ships to their assigned positions. It has no other function. AlliedForce class not only has the task of moving the allied ships, but also building the network between the allied ships. LandStation class is responsible for distributing the packets coming from ships.

All allied ships prepare INFPACs periodically. If an allied ship detects any hostile ship, then an NTPAC is prepared. The process starts with preparing one of these packets.

4.2.4.1 Preparing the Packets

All allied ships have a number of predefined positions loaded on their next position arrays. When a ship arrives at its next position, it prepares an INFPAC, which includes id, name, course, speed and position of the ship. The PrepareInfPac function accomplishes this task. The pseudo-code of the function is as follows:

```
 Procedure prepareInfPacket(){
//Returns the prepared INFPAC
      next ← nextCoord of the ship;
      current ← current location of the ship;
      course ← getCourse(current,next);
      myPacket ← new Packet(ship id, sender id,
      course, speed of the ship,current);
      return myPacket;
}
```

The getCourse function gives the course of the ship in degrees from the North Pole in clockwise direction. It takes two coordinates as input and returns the course as an integer.

If a ship detects any track it calls prepareNTPacket function. This function is very similar to prepareInfPacketFunction.

```
Procedure prepareNTPacket(New ship){

 //Returns the prepared NTPAC
 next ← next coordination of the new ship
 current ← current location of the new ship;
 course ← getCourse(current,next);
 myNTPacket ← new Packet(ship id, track id,track course,
track speed, current);

 return myNTPacket;
}
```

Each time a new track is detected by an allied ship, it will send an NTPAC containing its id and a new track id, along with course, speed and location information for the track.

4.2.4.2 Sending the Packet

After a ship prepares a packet it calls sendPacket function. This function will simulate the delivery of the packet to all units within the UHF range.

```
Procedure sendPacket(packet){
 distance ← 0.0;

 For s=0 to maximum_ship_number do{
 //the sender ship only saves the information on its routing
table
       if(s=ship's own number){
             savePacket(packet);
             continue; }

       distance ← distance(sender,ship[s]);

       if(distance<UHFrange) ship[s].receivePacket(packet);
 }

 For s1=0 to maximum_land_station_number do{
       distance ← distance(sender,landStation[s1]);
       if(distance<UHFrange)
 landStation[s1].processPacket(packet);
}
```

The sendPacket function first checks all ships and then all land stations. Sender ship only saves the packet to its routing table. If any other ship is within the UHF range of the sender, that ship's "receivePacket(packet)" function is called. If any land station is

within the UHF range of the sender, than the land station's "processPacket" function is called.

4.2.4.3 Receiving and Re-Transmitting the Packet

The ships implement "receivePacket" function. This function controls the incoming packet, if it is old or processed before it will be ignored. If the packet contains new or fresh information it will be saved to the routing table.

```
Procedure receivePacket(packet){
 //First the packet is controlled

       if(ship number=packet's target id) return;
// ship's own message,ignoring..


 For f=routeCount to 0 do {

//If the new message number is larger than the old one //or
the message numbers are equal but arrived one has //less hop,
meaning a shorter path has been found

if (routetable[f].target = packet.target){
 if((routetable[f].number < packet.number) ||
((routetable[f].number =packet.number)&&(packet.hop <
routetable[f].hop))){

 updateTable (routetable[f],packet);
 forwardPacket(packet);
 return;
             }
       }
 }//End For

//Packet's target is not in the routing table

 routetable[routeCount] ← new RouteTable(packet);
 routeCount++;
 forwardPacket(packet);
 return;
}


Procedure forwardPacket(packet){
 int tempsender ←packet.sender();
 packet.sender ← ship id;
 packet.hop++;
 sendPacket(packet);
 packet.sender ← tempsender;
 packet.hop--;
}
```

4.2.4.4 Land Station Packet Processing

The land stations implement "processPacket(packet)" method. This method first checks the packet. The land station distributes the packet, if the packet has not been processed before or the packet number is in the processed packet list but has a smaller hop count than the route recorded in the routing table.

```
Procedure processPacket (packet){

 For f=received_Message_Counter-1 to 0 do{
       if(receivedMessage[f].number=packet.number)
       if(packet.hop()< hopcount[f]) {
 //Same packet but less hop count
             hopcount[f] ← packet.hop();
             distributePacket(packet);
             return;}
        else return; //The packet has been processed
 }//End For

//The packet has not been processed before
 receivedMessage[receivedMessageCounter[ship]] ←
packet.number;
 hopcount[receivedMessageCounter[ship]] ← packet.hop;
 receivedMessageCounter[ship]++;
 distributePacket(packet);
}
```

The distributePacket method simply broadcasts the packet to the ships around and sends it to the neighbours on sink tree.

```
Procedure distributePacket(packet){

 broadcastPacket(p);

 if((packet.sender < land station id)&&(next ≠ null)) {
 //coming from the north or a ship and having a south
neighbour
       tempsender ← packet.sender;
       packet.sender ← land_station_id;
       packet.hop++;
       next.processPacket(packet); //next land station
       packet.hop--;
       packet.sender ← tempsender;
       }

       if(((sender is ship)||(sender > land station
id))&&( previous ≠ null)){
```

```
      //Coming from the south or a ship and having a //north
      neighbour
      tempsender ← packet.sender;
       packet.sender ← land_station_id;
       packet.hop++;
       previous.processPacket(packet);
       packet.hop--;
       packet.sender ← tempsender;
       }
 }


Procedure broadcastPacket(packet){
 //Sends the packet to the ships around

 packet.sender ← land_station_id;
 packet.hop++;

 For c=0 to max_ship_number do{

       distance ← distance(land_station,ship[c]);
       if(distance<UHFrange) ship[c].receivePacket(packet);
 }//End For

 packet.hop--;
}
```

4.2.4.5 Updating the Routing Tables

The ships periodically update their routing tables.

```
Procedure updateRouteTable(){

 if(routeCount<1) return; //No information on routing
 //table

 For h=routeCount-1 to 0 do {

       if(distance(ship,routetable[h].sender)>UHFrange){
      //Coming from a platform, which is currently
unreachable

 if(h ≠ routeCount-1) routetable[h] ←
routetable[routeCount-1];
       routeCount--;
       }

if(sender is a land station)   updateForLS(ship_id);
      }//End For
}
```

The updateForLS function looks up the information on all ships' routing tables for a target ship and land station. If any ship uses the land station to reach the target ship, the information is old and must be deleted.

```
Procedure updateForLS(target,land_station){

 For y=0 to max_ship_numbers do {

 if(y=target) continue;

       For h=ship[y].routeCount-1 to 0 do{

 if((ship[y].routetable[h].target=target)&( ship[y].routetab
le[h].hop=
abs(ship[y].routetable[h].sender-land_station)+2)){
 //Any ship using land_station to reach to target

if(h ≠ ship[y].routeCount-1)
ship[y].routetable[h]=ship[y].routetable[ship[y].routeCount-
1];

 ship[y].routeCount--;

 }End if

 }//End For h

 }//End For y
                              }
```

CHAPTER 5

IMPLEMENTATION RESULTS AND OBSERVATIONS

This chapter gives the results of the application. The performance of the NID system will increase depending on the positions of the ships and land stations. After calculating the average distance between them, it has been verified that by 5 land stations, an effective listening area covering the Turkey's entire Aegean coast can be created if their positions are selected accurately. In the last part of the chapter the execution of the simulation will be explained by giving some example scenarios.

5.1 The Implementation Results

5.1.1 Locating the Land Station and the Ships

Having the main purpose of maintaining all the ships' position information in a single Tactical Information Display System, Naval Information Distribution System has been tested many times by different navy officers. The ships have been positioned at different locations, sometimes closer to each other, sometimes far away. The simulation has been run with one or more land stations. After all these trials some conclusions are drawn.

1. Since many people could be using this system, the common approach to the tool was ease of use and simple implementation.

2. As it is possible to add new layers to the simulation such as planes and helicopters, the simulation is easily extensible.

3. The map information can be added to the simulation as a different layer, as the simulation supports different kinds of map formats it will be possible to work on a small area by using very detailed small-scale maps.

4. The only way to giving paramater values to the simulation and interacting with the system is using the buttons and clicking the mouse on the TIDS. This prevents the insertion of false values and allows the simulation to be protected from human errors.

5. The countries, which have long and curled coasts like Turkey, by locating the land stations to the appropriate positions, can execute the search and control mission on their territorial waters with the assistance of a minimum number of war ships. Also their searching area can easily cover the backsides of the islands (normally invisible from coasts) by locating ships at positions that can observe those locations.

6. If the ships' positions are selected to be too close to each other, the performance of the simulation seems to decrease and the number of duplicate packets increases. As it is shown in Figure 32, if the number of "close-in" ships increases, the number of received duplicate INFPACs increases dramatically. If the number of the ships within the same UHF range is n, then each one of the ships should normally receive n-1 required INFPACs while $(n-1)^2$ duplicate INFPACs are actually received. For examle if 4 ships were close to each other and all 4 ships send INFPACs, each ship should receive 3 INFPACs and 9 duplicate INFPACs. Thus there would be a total of 36 transmissions. This can cause congestions and decreases the performance of the ship.

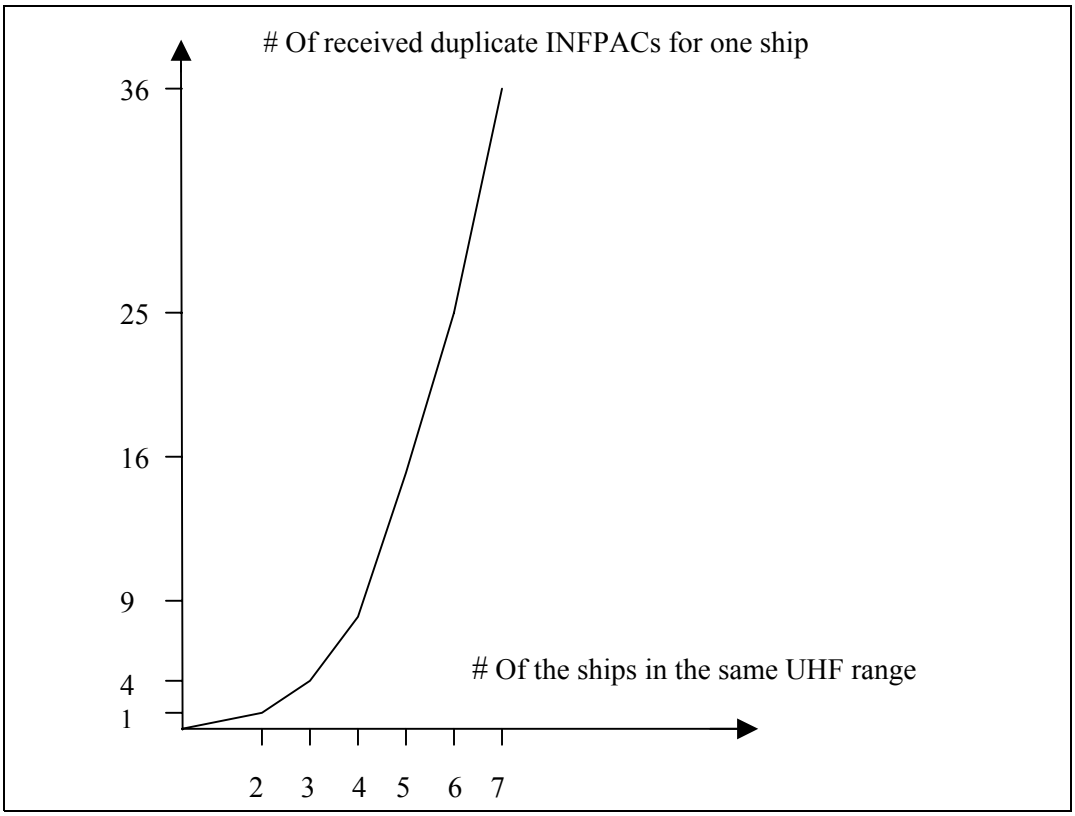

Figure 32: Icrementing of the duplicate packet number

As a solution to the duplicate packet problem, the ships can be located with a distance of approximately UHF range from each other as shown in Figure 33. With this solution each ship receives only one duplicate packet (receiving its own INFPAC from the neighbour ship). Also this solution will maximize the range of area under allied ships' control. But this approach would not be useful everytime. Although this makes preparing the routes an easy task, in an emergency situation such as demage to a ship, the connection will be lost. Also this approach may not be reliable enough for military concerns.

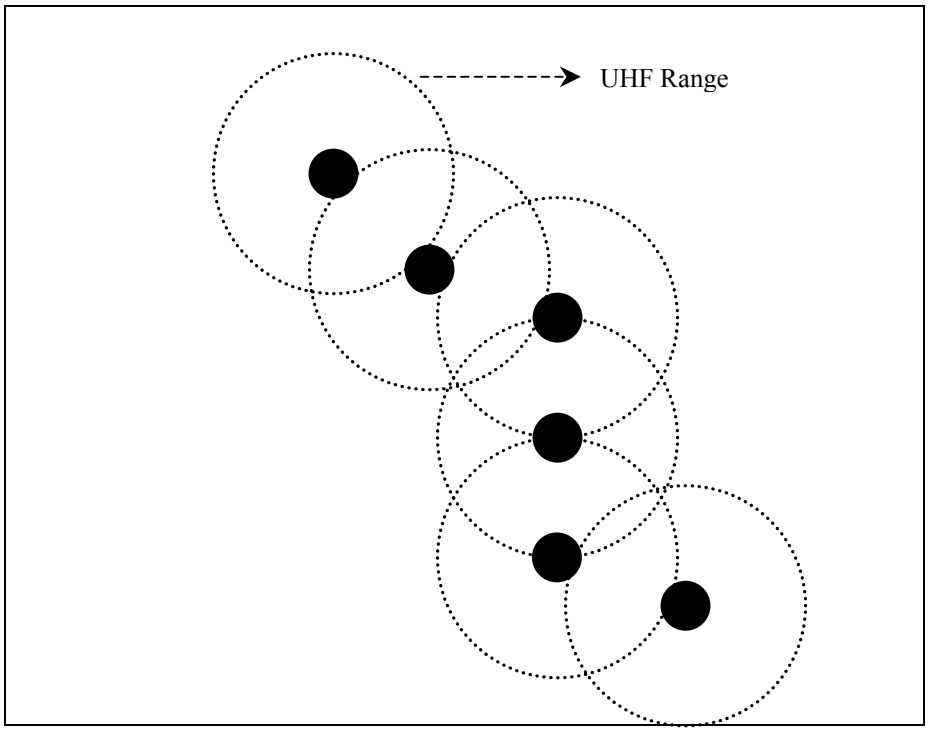

Figure 33: The best arrangement of the ships

7. If the land stations were located close to each other such as smaller than the 2*UHFRange, there could still be duplicate packet problems. If the prepared packet of one ship were received from two land stations, both land stations would try to distribute the same packet to the other stations; also they both would receive the same packet from the other and try to broadcast it to the environment.

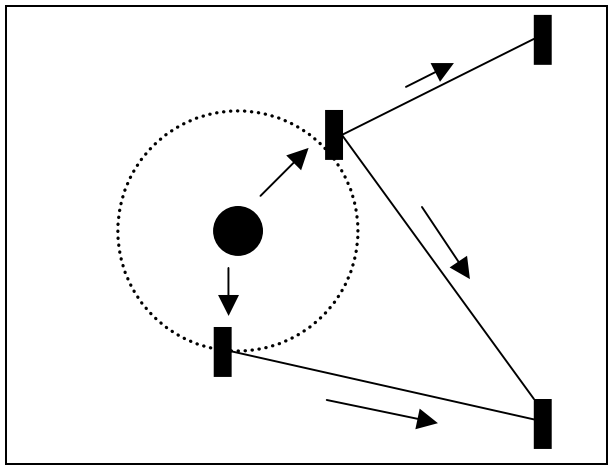

Figure 34: Arrival of a packet to 2 land stations

The solution to this problem is locating the land stations with the distance of 2*UHFRange. This would allow receiving one packet from at most one land station. Selecting the distance more than the 2*UHFRange would result in having some blind (uncontrolled) areas.

8. After placing the land stations at the appropriate positions, no duplicate packet is received from the land stations. Because of the sink tree algorithm, the packet is transported along the tree and since there would be no loops in a tree, all land stations would receive a packet only once.

9. By using the NID system, since the ships can track the positions of each other, preparation and maintance of the routing tables would be an easier task. There would be no need to discover a path. By the arriving INFPACs, the hop counts for a destination and the id of the intermediate ship should be stored in the routing table automatically.

5.1.2 Controlling the Aegean Sea Coast Area

Countries that have a long coastline like Turkey have to create an efficient watching area on their own waters. On a smooth coast like Black Sea coast or Mediterrian coast building this watching area is not so difficult. Having some radar stations on these coasts make it possible to search huge areas. But coasts like Aegean Sea have a wavy structure and contain a lot of islands. This stuation makes the searching function harder. Because a radar station won't be able to detect a ship hidden behind an island or a cape. So, in Aegean Sea a lot of radar stations are required. This is not a good solution because of the cost of building so many radar stations. Using war ships to search the area can be thought as a solution. Then, transmitting the information to the other platforms arises as another problem.

In the simulation, many experiences showed that by 5 Land Stations, which have been placed at carefully chosen appropriate positions, it is possible to build an effective watching area on Aegean Sea coasts of Turkey. This effective watching area maintains receiving the information from the allied ships, which are located closer than the accepted minimum UHF range of 40 NM. So any allied ship, if it is within range, can transmit to the other platforms information; the information is about itself or hostile ships, which cruise close to the coasts of Turkey. Also, all ships will have information about the positions of all other allied platforms, which are within the range of any one of

the allied ships. The coverage of this extended watching area is shown in Figure 35. Informing the positions of hostile ships, which are close to our coasts would bring an important improvement to our national security.

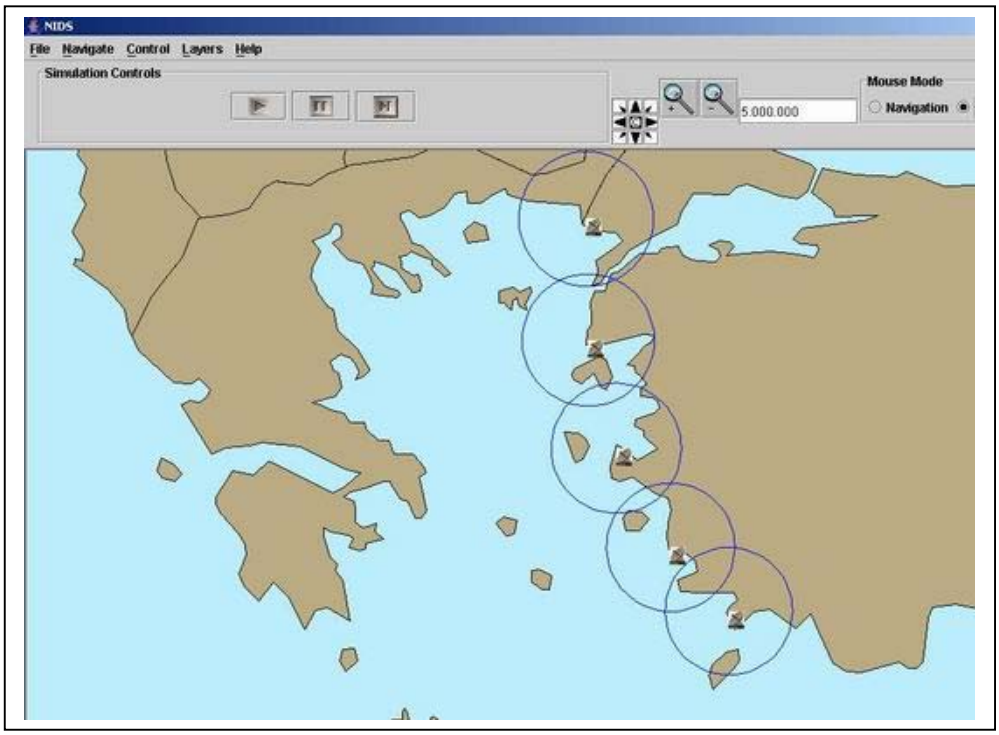

Figure 35: Extended watching area of the NID system

5.2 Observations

The simulation can be tested by using different scenarios and placing the ships at different locations and observing the progress of the simulation would give a better understanding and experience. However since the number of the ships and land stations can be increased, some scenarios can be merged and the activation of the main tasks can be observed in a single scenario.

In the created scenario the initial positions of the ships and land stations are shown in Figure 36. There are 4 allied war ships and 2 hostile ships. The allied war ships are colored blue; one of them is the commander ship. Clicking the mouse to a free location on the map will move the commander ship to that position. The hostile ships are colored

in red and just as the allied war ships (except the commander's) move to their predefined positions.

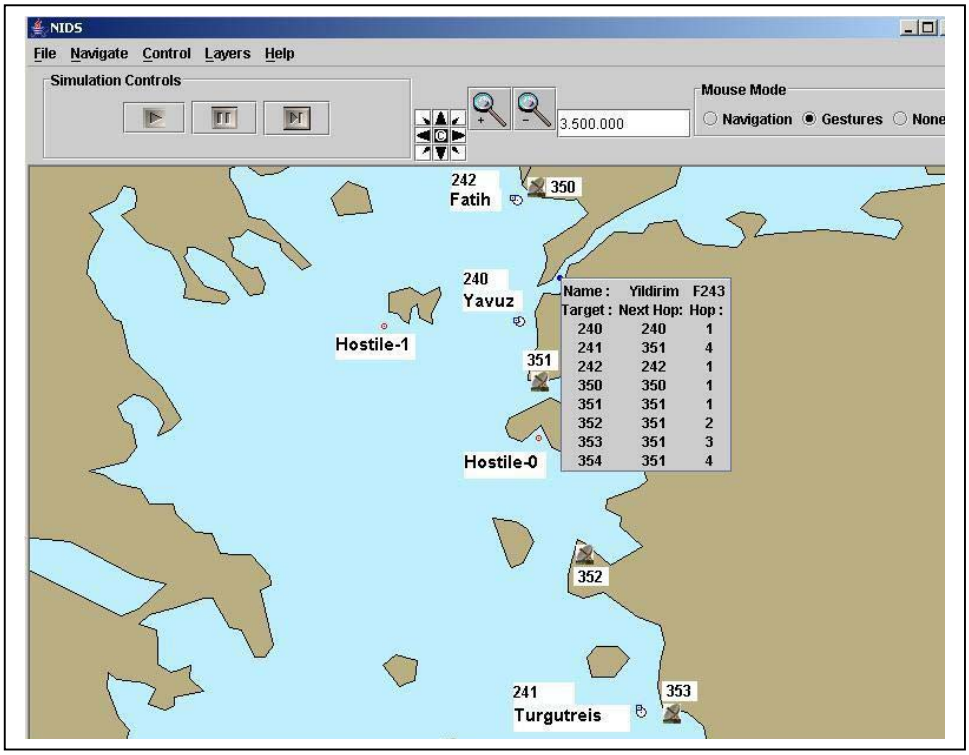

Figure 36: Initial configuration of the simulation

5.2.1 Enlarging the Scope of the Tactical Picture

The commander ship (Yıldırım in this scenario) is passing through the Çanakkale Strait. There is another ship (Yavuz) 30 miles away from the Yıldırım on its southwest side. The mission of Yavuz is searching the South coast of Mytillini Island. There is a hostile war ship (Hostile-0), which is hidden on south of the Mytillini Island. Of course Yıldırım cannot detect Hostile-0 because of the range and prevention of radar beams by the hills on Mytillini Island. The initial contents of the TIDS of Yildirim are shown in Figure 37.

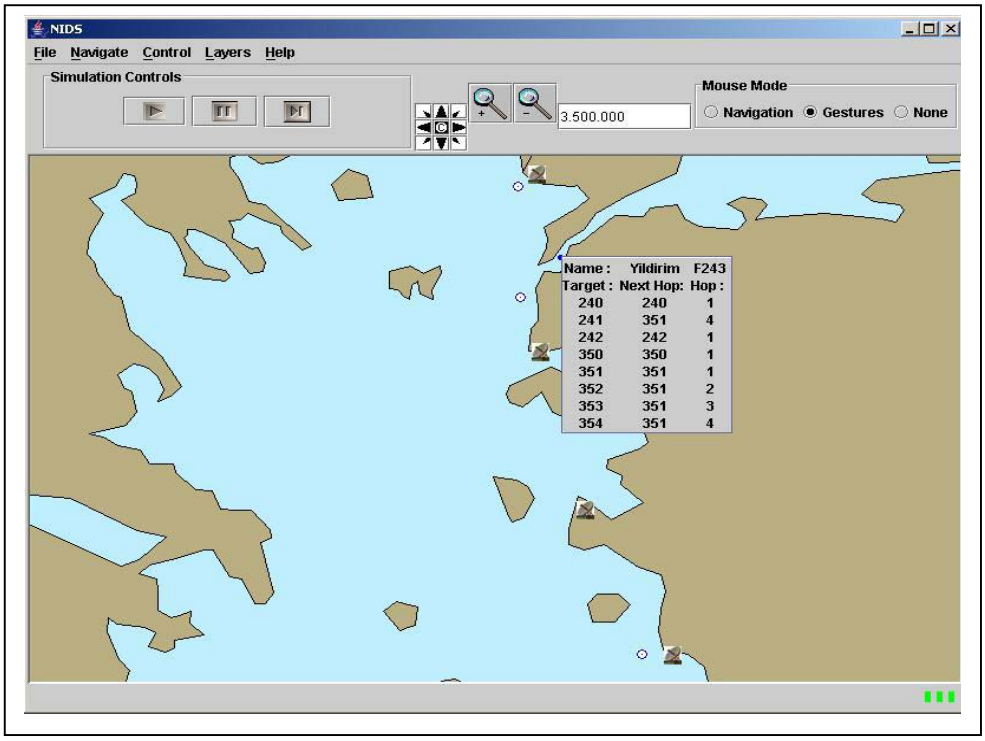

Figure 37: Initial contents of the commander ship

After Yavuz starts moving, the distance from the Yıldırım increases and after a while Yıldırım can get the Yavuz's INFPACs not directly but from the land station on Baba Cape (351). When Yavuz approaches to the Hostile-0 closer than the detection range (20 NM), it prepares an NTPAC and broadcasts the packet. The NTPAC is received by the Baba land station and the packet is broadcast over sink tree. Yıldırım is in the range of the Baba land station, so the commander ship will get the information about Hostile-0 and the TIDS of Yıldırım is updated, the hostile war ship is plotted as the information came from Yavuz. The new tactical picture is shown in Figure 38.

The scope of the Yıldırım's TIDS has become larger at this time. In the initial configuration the commander could not see the south of Mytillini Island but in the current configuration the scope of the picture has been extended 20 miles to southwest side. If there weren't a land station on Baba Cape, the information coming from Yavuz would be lost and the commander ship wouldn't know that there were a hostile war ship on south of Mytillini.

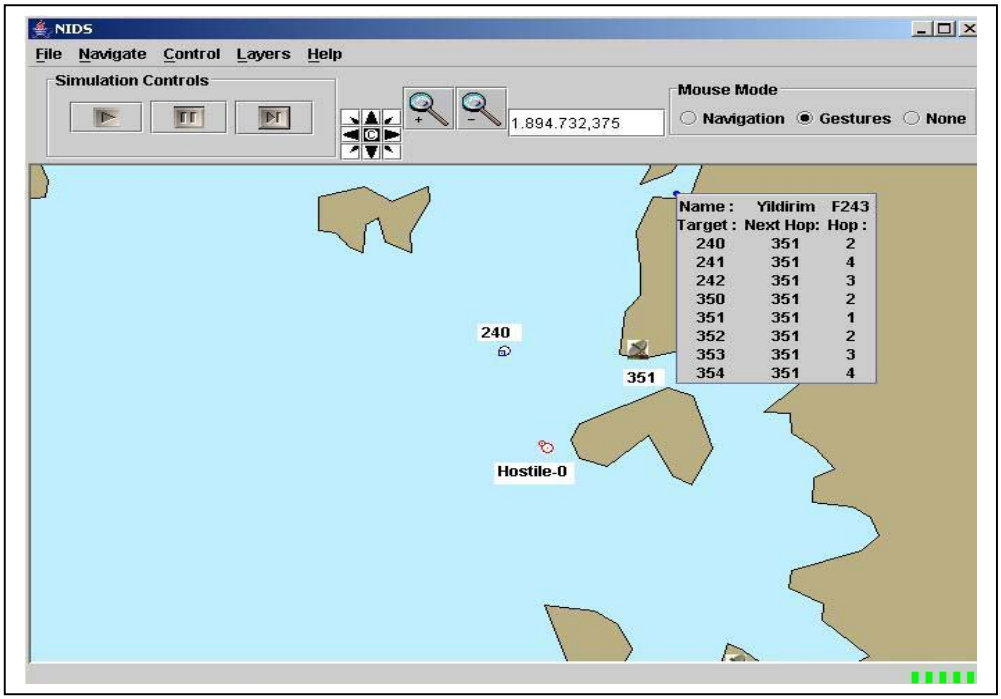

Figure 38: Enlargement of theTIDS of the commander ship

5.2.2 Getting Out of the UHF Range

TCG Fatih is located in Saros Gulf close to Enez land station. Its job is searching the Saros Gulf and if any track is identified, broadcasting the information to the other allied units. There is a hostile war ship (Hostile-1), which is 50 miles away from Fatih on its southwest side. Before starting of the simulation Fatih cannot identify Hostile-1. After the simulation starts, Fatih cruises to its southwest and Hostile-1 moves to its northeast side. They become closer and after they are closer to each other than the identification range, Fatih detects Hostile-1 and sends an NTPAC. The packet is transported to commander ship (TCG Yıldırım) via Enez land station and it is plotted on TIDS of Yıldırım as shown in Figure 39.

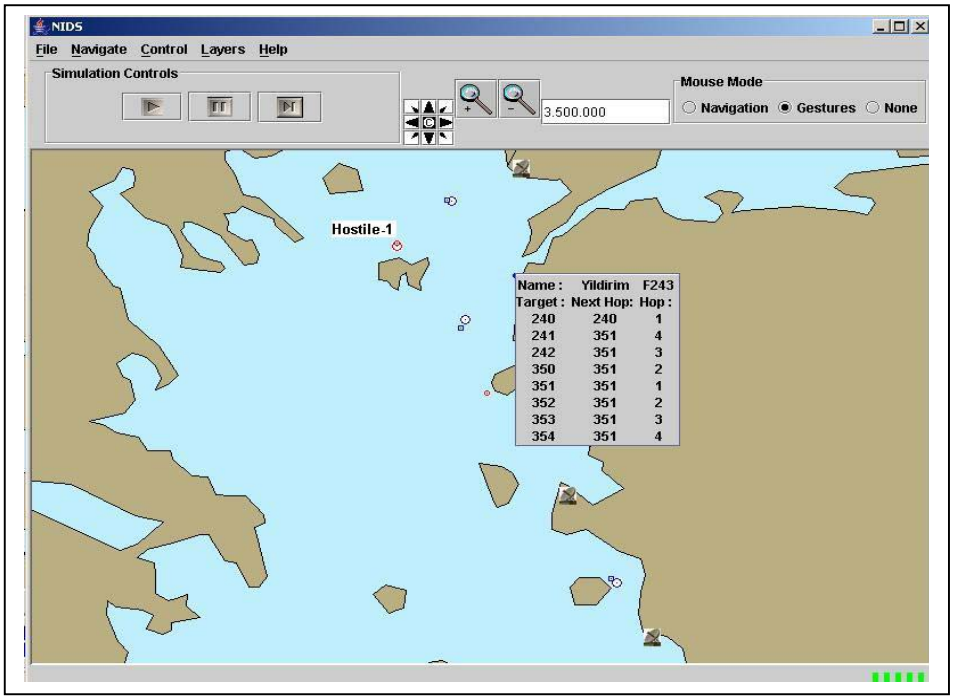

Figure 39: The hostile information on the commander ship

After detection of hostile war ship, Hostile-1 turns to west and moves away to hide from the allied ships. Fatih follows Hostile-1 to inform about its new positions. This process continues until Fatih moves out of the UHF range of Enez land station, which receives and broadcasts the packets of Fatih. After Fatih moves out of the UHF range of Enez, any INFPACs and NTPACs sent by Fatih won't be received by any other allied units. So Yıldırım loses the information about Fatih and Hostile-1. The information loss is shown in Figure 40.

As the NIDS system is dependent on the UHF Range, the ships should not violate this range. If searching of longer distances is needed then other ship(s) should be employed forwarding packets until a land station or the commander ship is reached.

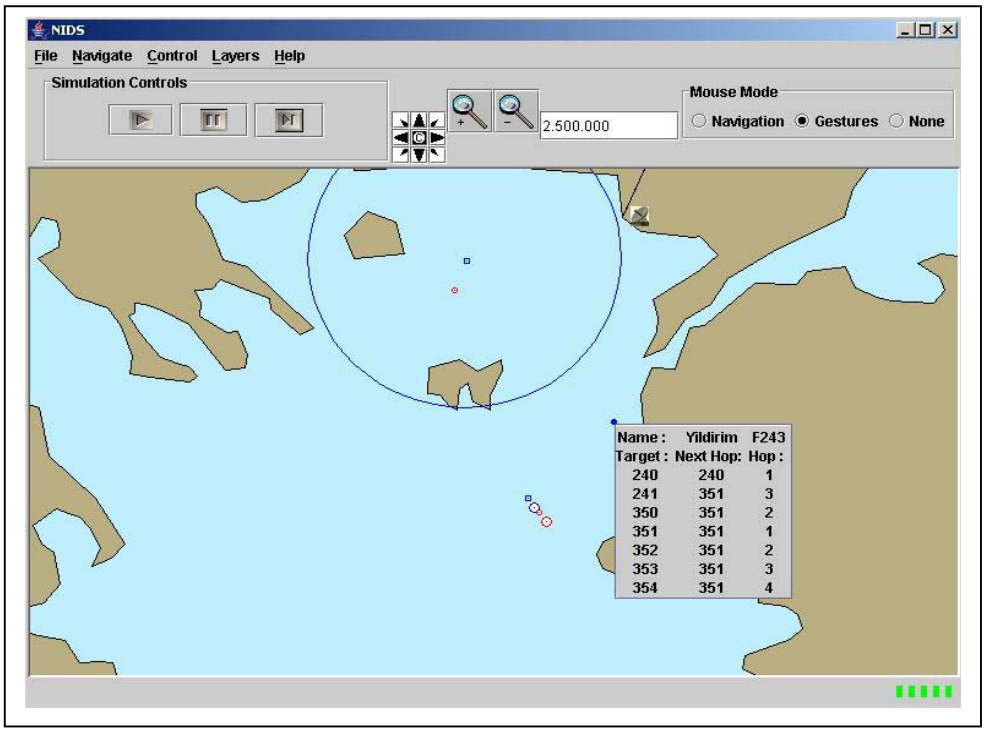

Figure 40: Range violation and the information lost

5.2.3 Finding the Best Route for Ships

There is more than one example of finding the optimal route in the scenario. The routing table of Yıldırım at the starting configuration is shown in Figure 36. Since it can copy the packets coming from the Fatih and Yavuz, its routing table shows that it is possible to reach both of them in one hop. Depending on the algorithm explained earlier, first Yıldırım copies the position of Yavuz via Baba land station, copies it with hop 2, then receives the same message from Enez, but since that packet has hop count 3, Yıldırım ignores it. When Yıldırım copies the INFPAC of Yavuz directly, the packet is copied with hop 1. Like Yavuz, Fatih's INFPAC is transported via Enez and copied at Yıldırım with hop count 2, the packet is transmitted from the Baba land station with hop count 3, so Yıldırım ignores it. But, after Yıldırım receives the INFPAC of Yavuz directly, it updates its routing table with hop count 1.

After Yavuz and Fatih moves far away, their INFPACs will arrive to Yıldırım via the land stations. So, after a while, routing table of Yıldırım is updated and it can reach

Yavuz and Fatih via Baba and Enez land stations, with hop count 2. This new configuration is shown in Figure 41.

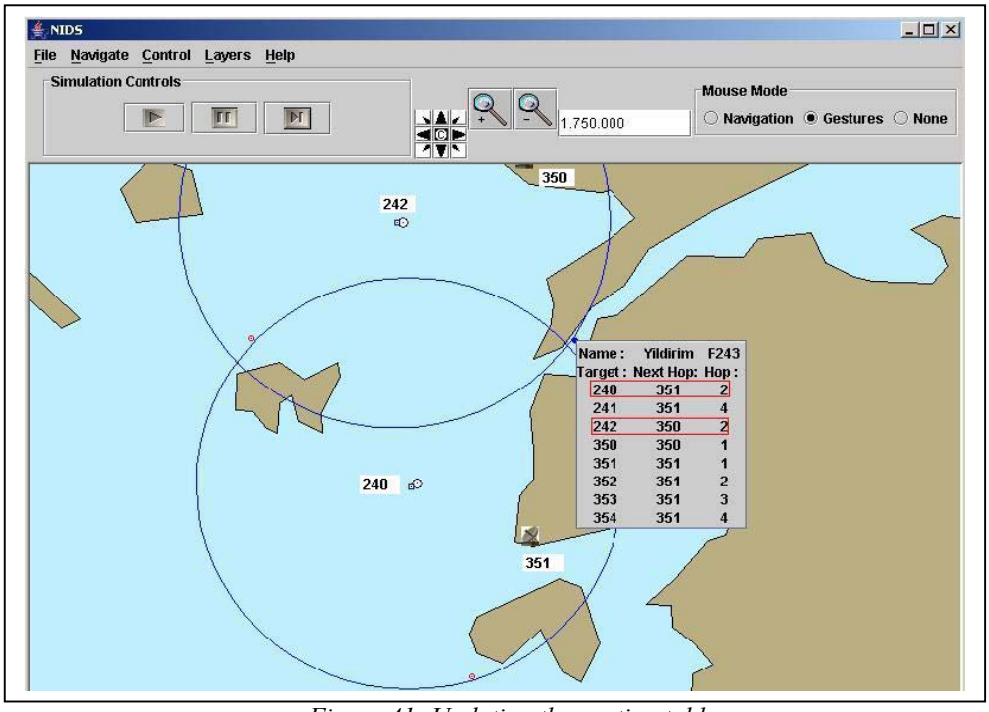

Figure 41: Updating the routing table

5.2.4 Transporting Packets Between Land Stations

The land stations broadcast the packets by using sink tree algorithm. If a land station receives the same packet with less hop count it re-processes the message, otherwise the land station ignores it.

In initial configuration, the commander ship (Yıldırım) is located on a special position. Its packets can be copied by both Enez and Baba land stations. Due to the algorithm, first Enez receives the INFPAC of Yıldırım and distributes it to the other land stations. The land stations Baba, Karaburun, Didim and Aksaz broadcast the packet by incrementing hop count in the packet. Turgutreis receives the packet coming from Didim with hop count 5. But after Baba receives the INFPAC of Yıldırım directly, it re-transmits the packet with one less hop count. Enez ignores the message because it has already broadcast it with the same hop count. But the land stations, which are located at south of Baba, re-transmit the message with one less hop count. So Turgutreis updates its

routing table and changes the hop count to reach Yıldırım to 4. This is the shortest way for Turgutreis to reach Yıldırım. In the other direction Yıldırım coppies to its routing table, the hop count of 4 to reach Turgutreis, as shown in Figure 37.

5.2.5 Information Flow Continuity

The allied war ship TCG Turgutreis is located 15 miles away from Didim's west side. Its mission is searching the Kuşadası Gulf and patrol there. In the starting configuration, it is reachable only from the Didim land station. After starting the schedule, Turgutreis starts to sail north, so its distance from Didim land station increases continuously. As Turgutreis gets closer to Karaburun land station, it enters the Karaburun's UHF listening and transmission area. At this time the packets of Turgutreis will also be delivered by Karaburun with Didim. Karaburun would start to re-transmit the Turgeutreis' packets with hop count 1. Didim will ignore the incoming packets (and so Aksaz wouldn't even get those packets) but the land stations, which are located north of Karaburun, will re-transmit those packets. Then, the ships receiving these packets, update their tables to one less hop count. As one of these ships, Yıldırım will also update its table for Turgutreis to hop count 3. After this updating the routing table of Yıldırım is shown in Figure 42.

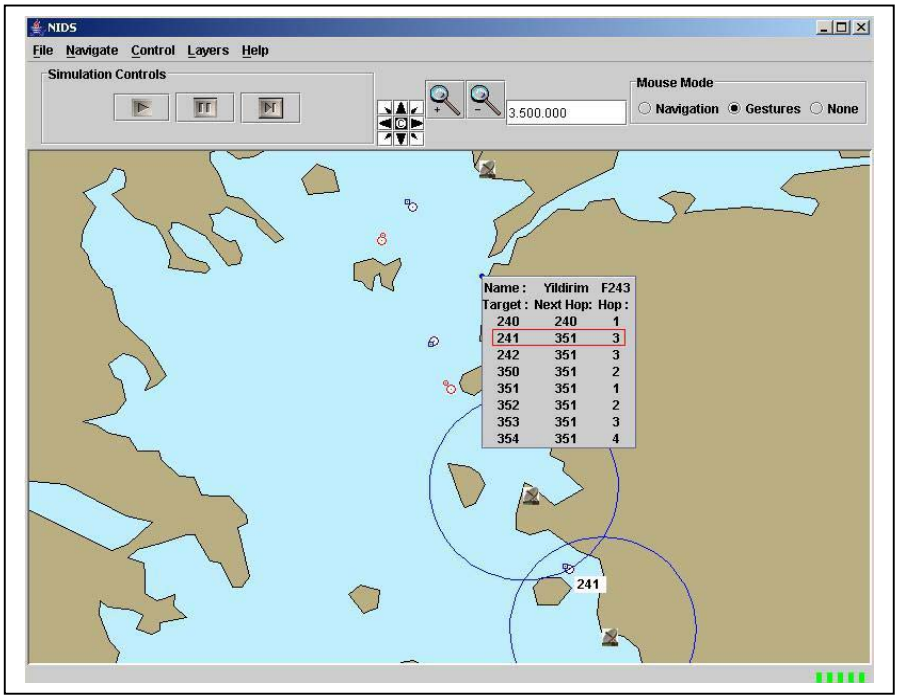

Figure 42: Information flow continuity

After a while, Turgutreis would be out of the UHF range of Didim. Then the packets coming from Karaburun would be processed by Didim and sent to Aksaz after its hop count is increased by one.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

As a necessity of information and communication century, timely delivery of information has become very important. This would be even vital in military applications. As a part of the military, the navies continuously work on developing the information flow systems.

The Naval Information Distribution system brings a major change in this respect. Combining the static network model with mobile network technologies increases control capability on the territorial waters. This approach would be very useful for those countries, which have a navy mostly organized for defensive purposes.

NID system enhancements as given below;

- It would be possible to guard a specific area. This area could even be all of the territorial waters of a country.

- By selecting the suitable distance between the land stations and placing the ships at the most appropriate positions, the information flows without interruption. This approach prevents arrival of the same information more than once and increases the effectiveness of the units.

- In Tactical Information Display System by controlling the allied and enemy, the decision about what to do and where to go, must be made in a short period of time.

- NID system brings an easy solution to routing in a mobile environment. Because of the continuous information flow and routing table maintenance, the message transporting paths are always fresh and active.

- Having a properly working NID system gives country safety and discourages, dissuades the enemy, or the ship, which has the intention of tresspassing.

6.1 Future work

Naval Information Distribution System has a great flexibility. By using a detailed map (many map formats are supported by OpenMap), the geographic information could be much more detailed (and even three dimensional) and the scale of the map could be increased. This would affect the performance of the mission and the decisions made.

The handicap of NID system is being restricted by the UHF range. This is because the voice communication systems and the crypting systems are based on the UHF and HF radio sets on war ships in most of the countries. The satellite voice communication is at the starting phase in Turkish Navy. Also the satellite communication is not yet safe enough to build military applications on them. After developing the crypting technology for satellites, the NID system could work efficiently. Also there will be no need for the land stations. This could be used not only for the defending the country, but also attacking the enemy.

REFERENCES

[1] Andrew S. Tanenbaum. "Computer Networks", Fourth Edition. Vrije Universiteit, Amsterdam, Netherlands. Prentice Hall PTR.

[2] Word IQ Dictionary & Amp, "Definition of Geographic Coordinate System Encyclopedia", "*www.worlddraft.com/encyclopedia/geographic coordinate system*". 12 February 2004. The article originates from Jason Harris' Astroinfo, and taken from the "*http://edu.kde.org/kstars/index.phtml*"

[3] Geospatial Training and Analysis Cooperative,"Introduction to Topographic Maps"JimReisterer. *http://geology.isu.edu/geostac/Field_Exercise/topomaps/grid_assign. 23 October 2003.*

[4] Hans Zuuring, **"**Geographic Information System Methods and Applications I Notes - Part 4", Missoula, Montana, USA, September 2003.

[5] Radio Waves and Radio Wave Propagation. Electrical Engineering Training, Integrated Publishing. "*http://tpub.com/neets/book10/40a.htm*" 12 February 2004.

[6] AN/WSC-3 (V) 11 UHF AM/FM Radio Set Technical Documentation.

[7] Charles E. Perkins, "Ad-Hoc On Demand Distance Vector Routing", Sun Microsystem Laboratories, Advanced Development Group, California.

[8] Patrick Mack. "A Study In Designing A Usable Interface For A Geo-Referenced Discrete Event Simulation", September 2000, Naval Postgraduate School, Monterey, California.

[9] OpenMap Architecture, BBN solutions LLC, Novamber 2001. An Internet draft *"http:// openmap.bbn.com/openmap-arch.html"*

[10] Alec Sharp, "Model-View-Contoller", European Smalltalk Users Group, the University of Berne, 1997.

[11] Arnold H. Buss, "Discrete Event Programming with Simkit", March 2002, Operations Research Department, Naval Postgraduate School, Monterey, California.

[12] Dr. Don Brutzman, Dr. Arnie Buss, Curt Blais and CAPT S. Starr King USN, "Connecting Simkit Discrete Event Simulation (DES) and theNaval Simulation System (NSS) via Web Services for Extensible Modeling & Simulation (XMSF)-Capable Analysis" , Modeling Virtual Environments & Simulation (MOVES) Institute, Naval Postgraduate School, Monterey California, 2004.

[13] Arnold H. Buss, "Component-Based Simulation Modeling", Proceedings of the 2000 Winter Simulation Conference, Operations Research Department, Naval Postgraduate School, Monterey, California.