Design of a Space Based Internet

Emulation System

by

Sujit Rupert Baliga

B.E. (Electronics and Communications Engineering)

Karnataka Regional Engineering College, 1998

Submitted to the Department of Electrical Engineering and Computer Science
and the Faculty of the Graduate School of the University of Kansas in
partial fulfillment of the requirements for the degree of Master of Science

_____

Professor in charge

_____

_____

Committee Members

_____

Date Thesis Accepted

Abstract

Satellite communication currently involves the use of expensive hardware and proprietary protocols. The Internet Protocol envisages cost-effective solutions and ever-improving mechanisms and provides an ideal medium for communication in space. The Space Based Internet Emulation System seeks to design and evaluate the provisioning of IP in satellite communications in controlled hardware emulation scenarios. The architecture and design of the emulation system are explored. A central controlling and node status monitoring emulation manager is constructed and the design of a centralized routing and scheduling entity are examined. The problem of data routing in an Earth-orbiting satellite system is analyzed and avenues explored to address it. The system is tested and evaluated with various scenarios.

Acknowledgements

I would like to thank my advisor Dr. Gary J. Minden for having given me this opportunity to work on the Space Based Internet project and for his guidance along the way. I would also like to thank Dr. Joseph B. Evans and Dr. Victor Frost for being on my thesis committee.

A huge amount of gratitude goes to Leon S. Searl for his invaluable guidance and input at all stages in this work.

Special thanks go to Sandhya Rallapalli for being a wonderful friend and colleague and for coming up with a whole lot of invaluable suggestions in the course of this work.

I would like to thank Magesh Kannan for being a precious source of information. Every question you asked was an eye-opener.

I would also like to acknowledge Karthik, Pooja and Boon who also collaborated with me on the SBI project. Working with you people was a pleasure.

I would also like to acknowledge Pratibha, Sunitha, Devang, Raghu, Siva and everybody else I might have come in contact with in these two and a half years for making my stay here truly memorable.

Lastly, I would also like to thank my parents and my sister for having encouraged me in every aspect of my life. What I am today is entirely due to them.

Table of Contents

List of Figures

List of Tables

1 Introduction

Ever since the first satellite was launched into space in 1957, satellites have been used for a variety of missions including Earth-observation, military applications and communications. Currently, it is estimated that there are around 8000 active satellites in Earth orbit with plans to launch many more afoot. This promises a vast potentially untapped resource for communication purposes. If each satellite in Earth orbit were made capable of generating, terminating and switching traffic, this would lead to an immense increase in capacity for communications on Earth. A lot of research is currently in progress on using satellites as a low cost, high bandwidth channel for communications.

1.1 Earth Orbiting Satellites

Satellite orbits conducive for communication purposes can be broadly classified into three categories:

- Geosynchronous Satellites (GEO)
- Medium Earth Orbit Satellites (MEO)
- Low Earth Orbit Satellites (LEO)

1.1.1 Geosynchronous Satellites

Geosynchronous Orbit (GEO) satellites are found at an altitude of about 22400 miles (36000 Km). GEO satellites have an orbit period equal to the rotation period of the Earth. Geostationary satellites are GEO satellites with zero eccentricity and zero

inclination, i.e., they orbit the Earth over the equator. GEO satellite systems offer maximum coverage of the Earth area with a minimum number of satellites. Three GEO satellites can provide coverage over the whole Earth. For these reasons, GEO satellites are mainly used for data broadcast purposes. However, the round trip time is about 250 ms and they typically have a very high bit error rate.

1.1.2 Medium Earth Orbit Satellites

Medium Earth Orbit (MEO) Satellites are found at altitudes ranging from a few hundred to a few thousand miles (1000 Km to 10000 Km.). MEO Satellites typically have lower error rates and lower path delay.

1.1.3 Low Earth Orbit Satellites

Low Earth Orbit (LEO) Satellites are found at altitudes ranging from 200 to 600 miles (300 Km to 1000 Km). They have low path delay and low bit error rates. A large number of LEO satellites, however, will be required for total Earth coverage due to their low altitudes. Majority of the Earth Observation Satellites are launched into LEO orbits due to the high resolution possible for the Earth's surface. These satellites orbit the Earth once every 90 minutes approximately. Thus, LEO satellites form the best bet for low cost, low delay and high data rate communication networks.

1.2 Networking in Space

The first use of satellites for communication used the bent-pipe relay method where satellites in Geosynchronous Orbit were used to receive data from a terrestrial source

and then broadcast it over a large area. This is mainly due to the large footprint for GEO satellites that enable them to access a large part of the Earth's surface. Satellites in Low-Earth Orbit are used for Earth-Observation purposes due to the high-resolution capability at these altitudes. This requires the use of high data-rate, high-capacity recorders to record the data input over the observation area and output the data once the satellites come in view of a controlling ground station. This brings an unpleasant latency into picture, leading to delays in analyzing critical data. In recent times, satellite constellations at the LEO level have been envisaged and put into practice to create a satellite network in space to avoid this latency. These networks suffer from the disadvantage of being locked into a proprietary, high-cost communications service. Using a widely preferred protocol such as the Internet Protocol for satellite networking would be highly advantageous in terms of reducing costs and satellite specific communication equipment and creating the potential for evolving with technology [1]. A huge amount of research is currently ongoing on the potential of using IP and related protocols in an Earth orbiting satellite network [2].

Currently, a number of satellite constellations are in LEO and MEO orbits and take part in a communications network, the most notable being Teledesic, GlobalStar, Iridium and many more still being envisioned.

In Geosynchronous orbit, the Tracking and Data Relay Satellite System (TDRSS) [3] employs seven data relay satellites intended for use for relaying data for the Space Shuttle, the International Space Station and numerous Earth Observation Satellites.

1.3 Space Based Internetwork

Currently, Earth Observation satellites perform a variety of tasks ranging from a constant low-bandwidth observation of solar wind intensity to bursty, high capacity image captures of specific areas on the Earth's surface. This necessitates the use of high data rate, high capacity onboard data recorders for data storage. The data is then downloaded when the satellite passes over the intended ground station. The links between satellites and ground stations require complex expensive hardware and the use of custom protocols for downloading the data. A "single mission" observation satellite is thus only concerned with its acquisition schedule and its downlink schedule.

Using a standardized network protocol like the Internet Protocol (IP) can facilitate the use of low-cost hardware used in ground based networking onto satellites. In addition, recent rapid technological advances in IP show great potential for the future of high-speed networking and this bodes well for satellite networking. Satellites that now have to store data until they reach Line-of-Sight with the ground station before they can download it can relay via other satellites to the ground station as the data is being picked up. If a number of satellites in Earth orbit are capable of relaying data, this can develop an internetwork of satellites in space leading to the concept of a Space Based Internet (SBI) [4].

Coordinated data collection among satellites is possible once communication between satellites is accomplished. This is especially useful for data gathering for unique

events and can provide multiple perspectives on events. This can create the potential for “joint missions” in the future across multiple platforms and programs.

The system becomes scalable very quickly as more and more satellites are launched into space. Satellites that can accomplish the tasks of data gathering as well as data relay can be launched at lowered costs with the use of standard SBI modules. In addition, specialized relay satellites can be launched to provide extended bandwidth capacity and capabilities for the internetwork.

The use of satellite specific communication systems can be eliminated and low cost, high performance hardware can be incorporated onboard observation satellites to enable data relay in space.

This can soon evolve into an extension of the terrestrial Internet and provide a new high-bandwidth solution for personal communications. This may be the first step for the start of a new era of Inter-Planetary Internet [5].

1.4 Need for an Emulation System

Experiments in the real world entail the use of expensive hardware. Building satellites and launching them for purposes of testing the feasibility of a satellite network can lead to high costs being involved, in terms of both time and money. Software built for SBI would have to be loaded onto the satellites and tested in space. This leads to a very low repeatability factor and designing a scalable prototype turns out to be very expensive.

A software simulation, on the other hand, though simple, has many disadvantages; it does not work in real time and it does not incorporate the intricacies of the real world system. These detract from the effectiveness of a simulation system, though they are relatively easier and cheaper to build.

An emulation system effectively overcomes the deficiencies of a simulation while avoiding the high costs and overhead involved with the real world. An emulation system incorporates real world software on hardware that emulates the intricacies of the actual system. By emulating the environment of space on computer hardware, the costs of building a prototype system reduce considerably. Software that can be used on the emulation system can be used in the real world with minimum changes. Tests can be executed in real time to gauge the effectiveness and robustness of the system. An emulation system provides a high degree of scalability and repeatability and can be implemented in a relatively short time. Therefore, an emulation system is a very good choice for developing the architecture for a Space Based Internet.

1.5 Thesis Organization

This thesis explores the architecture and design details of the Space Based Internet Emulation System. A prototype emulation system is proposed, implemented and evaluated.

The acronym ‘SBI’ is henceforth used in this document to denote the SBI Emulation System being developed.

In Chapter two, the objectives of the SBI Emulation System are enumerated and a design is proposed for the architecture according to the requirements of the system.

Chapter three explores the design and implementation details of the Emulation Manager, which controls and monitors the entire SBI system.

The architecture and implementation of a centralized Operations Node is described in Chapter four.

Chapter five looks into the problem of routing in a space based internet. The design of a simple centralized topology generation algorithm is explained in this chapter.

Chapter six validates the performance of the SBI Emulation System. Test scenarios are described and the results are evaluated.

Chapter seven concludes the thesis and presents the scope for future work on related topics.

2 Space Based Internet Emulation System

This chapter provides an overview of the SBI Emulation System. The objectives of the system are discussed, the requirements mentioned and the architecture for the SBI emulation system is developed. A brief explanation is given on the components of the architecture.

2.1 SBI Objectives

The Space Based Internet Emulation System seeks to fulfill the following objectives.

- Provide internetworking between satellites for coordinated and continuous data collection: The primary objective for an SBI is to ensure continuous data collection and an uninterrupted data flow through the network. This emulation system seeks to figure out satellite network topologies for maximum uninterrupted coverage of the Earth.
- Emulate the space communication environment characteristics in hardware: This involves implementing innovative algorithms for emulating the latency and bit error rate characteristics of space communication links.
- Prototype the SBI node software required for a SBI: The node software developed for the SBI emulation system should be possible to be used in the real world with minimum changes. This software should include topology generation and data scheduling algorithms.

- Develop innovative topology generation algorithms for routing in a SBI: The space satellite network requires the use of dynamic routing algorithms for the purpose of routing data. A number of constraints are imposed on a satellite network. These constraints need to be addressed and solutions proposed for overcoming them.
- Implement traffic models for space based applications: Earth Observation Satellites perform a variety of data gathering functions. By developing realistic traffic models for the satellite data, the performance of the SBI emulation system can be accurately gauged.

2.2 SBI Requirements

The following requirements are addressed in the architecture of the SBI Emulation System.

1. The SBI Emulation System shall have a central controller managing the emulation.
2. The emulation shall consider satellites in Earth orbit as well as ground stations in the network.
3. Satellite orbits shall be considered up to and including geo-stationary orbits.
4. Satellite link communication characteristics like high delay and high bit error rates shall be emulated in the system. However, effects of interference by any other source including the Van Allen Belts shall not be taken into consideration.

5. Traffic models for satellite instruments shall be generated and used in the emulation.
6. Algorithms for routing data in the network shall be developed.
7. A centralized model shall be developed for the network, wherein a central server generates the routes for the network and the instrument scheduling data and transfer this onto the SBI nodes.
8. Enhancement of the software in the future is a possibility. Therefore, a modular form of programming shall be used for the Operations Node, so that replacing entire portions of algorithms with others shall be easily accomplished.

2.3 Space Based Internet Emulation Features

This section describes the general characteristics associated with the proposed Space Based Internet System. The different types of nodes and the types of networks in the SBI system are discussed.

2.3.1 SBI Nodes

The SBI Nodes are classified into three broad categories:

- Data Source and Relay Satellites
- High Speed Relay Satellites
- Ground Station Facilities

2.3.1.1 Data Source and Relay Satellites

The Data Source and Relay Satellites (DSR) perform the task of observation and data gathering. The Earth Observation Satellites form a major portion of this category. The satellites may pick up data over land, ocean, ice or according to the satellite day and night times. If a satellite has only one antenna, it can only perform the task of data sourcing. In addition, some satellites may possess multiple antennae and in such cases perform the additional task of data switching in the satellite network.

2.3.1.2 High Speed Relay Satellites

Satellites may be put into Earth orbit for the purpose of data switching of external traffic. These satellites provide additional capacity for the SBI Network and are called the High Speed Relay Satellites (HSR). These satellites will most likely be in medium Earth orbit or in geo-synchronous orbit to provide greater area of coverage over the Earth. The high-speed relay satellites possess multiple omni-steerable antennae and can contact any other SBI node in its line of sight.

2.3.1.3 Ground Station Facilities

Specialized ground stations located on the surface of the Earth and used for the primary purpose of data traffic termination are called facilities. The facilities have the capability of contacting any satellite in its line of sight and will possess multiple steerable antennae. A facility may also perform the task of an Operations Node, wherein it computes the routing topology for the SBI network and the data schedules for the data source satellites.

2.3.2 SBI Networking

The real-world SBI system utilizes IP networking over RF or optical links. The SBI Emulation System uses IP over Ethernet for emulating the space communication. The satellites and facilities utilize a high bandwidth communication link for data transfer. The antennae employed in a SBI are assumed to be omni-orientable, i.e., they can be steered in any required direction so as to be able to contact any other antenna in its field of view.

2.4 Space Based Internet Emulation Architecture

This section describes the architecture of the Space Based Internet Emulation System. An outline of all the modules in the emulation system is given. Detailed descriptions of all the modules are explored in the following chapters.

2.4.1 Overview

The SBI Emulation System is comprised of these major modules:

1. Emulation Software: This module controls and monitors the emulation scenario and emulates portions of the scenarios that will be deployed in hardware in the actual SBI system.
2. Node Software: This module is mostly comprised of the actual software that would run aboard the SBI system. It comprises the systems that accomplish the routing, switching, instrument scheduling and data sourcing tasks.

3. Emulation Network: The emulation network forms the part of the emulation system that takes care of the data transfer mechanism.

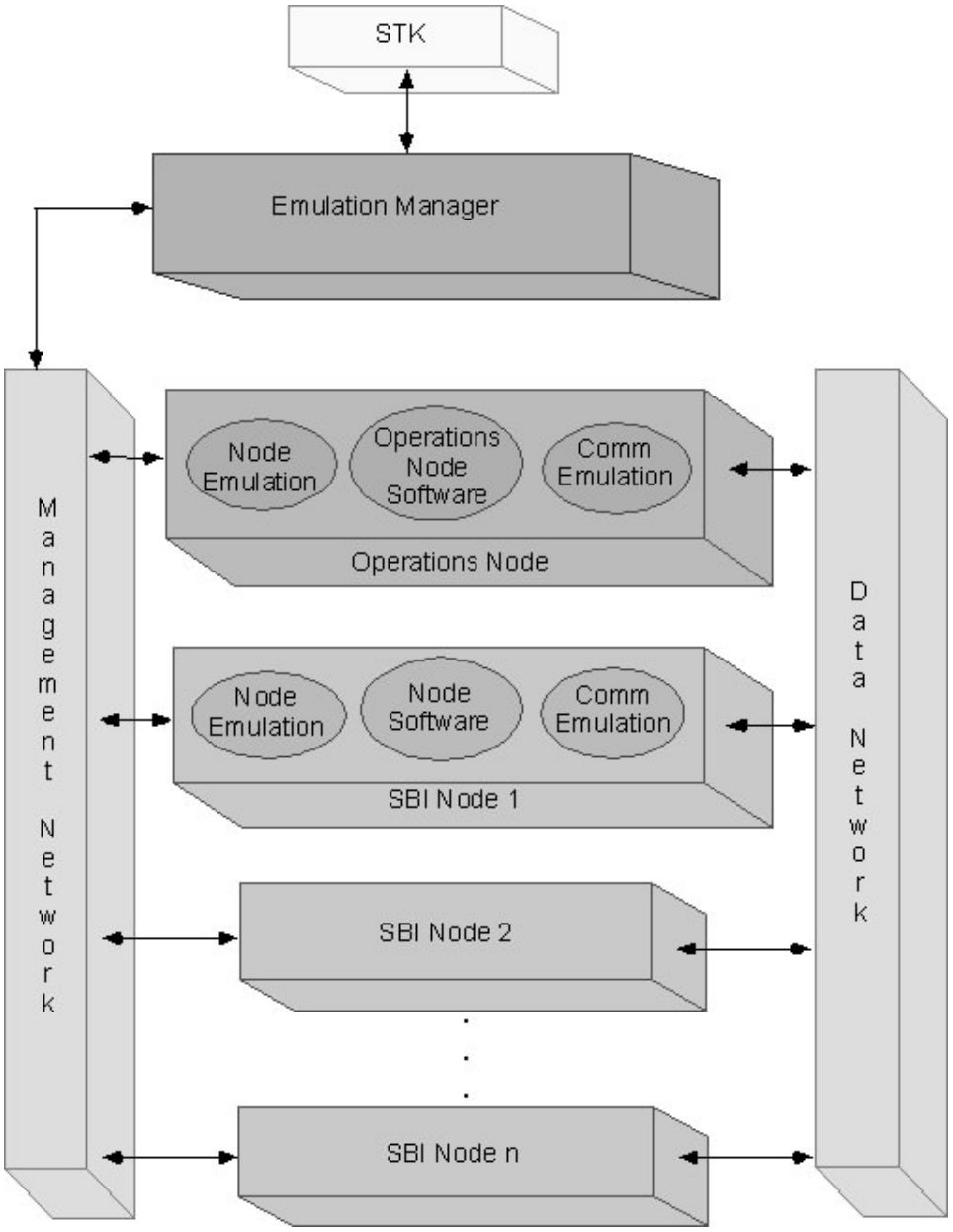

Figure 2-1: SBI Emulation Architecture

2.4.2 Emulation Software

The emulation software component of the SBI Emulation System controls, configures and monitors the emulation scenario. The Emulation Software emulates portions of the actual SBI environment to provide the impression of operating in the real world. It also acts as a user interface to provide a graphical and statistical output to the user.

The Emulation Software system is made up of the following modules.

- Emulation Manager
- Node Emulation Software
- Node Communication Emulation

2.4.2.1 Emulation Manager

The Emulation Manager is a separate process that oversees the running of the system. The Emulation Manager sends system configuration commands to the nodes and controls the emulation scenario. It also provides an interface for starting the emulation scenario and initiates the emulation on the nodes. During the emulation, control commands are sent to the nodes and status reports are received from the nodes periodically. An Operations Channel exists between the Operations Node and the other nodes. This passes through the Emulation Manager that receives the routing table updates that are destined for the nodes and forwards them to the appropriate nodes.

2.4.2.2 Node Emulation Software

The Node Emulation Software resides on the SBI Nodes and prepares the nodes for the emulation. It also controls and monitors the SBI Node attributes. The Emulation Manager maintains a multicast connection with the Nodes, and sends commands and receives status messages from the Nodes via this connection. The main features of the Node Emulation Control module are tabulated here.

- Configure the Ethernet and Virtual Ethernet [6] devices on the SBI Nodes according to the scenario specifications.
- Setup the QoS features on the Linux kernel as required by the emulation scenario.
- Start the Operations Node and Node programs when the emulation is started.
- Update the communication link parameters according to the link emulation.
- Periodically send status messages regarding the number of packets processed on each interface used in the emulation.
- Destroy the QoS setup and Virtual Ethernet devices on termination of the emulation scenario.

The features of the Node Emulation Control Modules will not be discussed in this document and serve as a background for the actual thesis.

2.4.2.3 Node Communication Emulation

The Node Communication Emulation software consists of the instrument emulation and communication emulation systems. The instrument emulation software emulates the data gathering mechanisms of Earth Observation Satellites and initiates data

traffic according to the satellite traffic models. The communication emulation software emulates certain satellite communication characteristics that exist in the real world. The two main features of a real world satellite communication link are high link delays and high bit error rates.

Since the emulation system is a terrestrial network utilizing Ethernet cables for the communication medium, the delay times and bit error rates of a real world satellite system have to be reproduced in software.

The SBI System also makes use of Virtual Ethernet Devices in the Communication Link Emulation. The Virtual Ethernet devices are built over the physical Ethernet devices and serve as the communication interfaces for the emulation.

The design and implementation of the satellite communication link emulation including the link properties and the Virtual Ethernet interface used in SBI is explained in [6].

2.4.3 Node Software

The SBI Node software consists of the actual software that will be used in the SBI system in the real world. There exist two types of nodes based on the functionality performed:

- SBI Node or Common Node
- Operations Node

2.4.3.1 SBI Node

The SBI Node performs data traffic switching according to the entries in the routing table. The routing table is filled by the route updates received from the Operations Node. The Node software also performs data traffic sourcing modeling the actual data generation characteristics of Earth Observation Satellites.

2.4.3.2 Operations Node

There exists a centralized Operations Node that performs some of the major functions in the system. The Operations Node computes the network topology and generates route updates for the network, which are then transmitted to all the nodes for updating their routing tables. The Operations Node also calculates the data scheduling times for the Data Source satellites and transmits this information to the relevant nodes in order for them to begin data traffic generation.

2.4.4 Emulation Network

The emulation network consists of the transmission channels for the SBI traffic. There exist two main types of channels:

- Management Network
- Data Network

2.4.4.1 Management Network

The Management Network is a low bandwidth network connecting the Emulation Manager to the SBI Nodes. The Management Network carries control and

configuration information from the Emulation Manager to the SBI Nodes. Status information from the nodes is also carried to the Emulation Manager in the reverse path.

2.4.4.2 Data Network

The Data Network carries the SBI data traffic between the SBI Nodes. The Data Network comprises high bandwidth channels for transmitting the satellite observation traffic from the originating Data Source satellite to the terminating facility. The Virtual Ethernet devices form an integral part of the Data Network.

2.5 External Software

The Space Based Internet Emulation System makes use of certain external software modules for performing the orbital computations associated with a satellite system. A data traffic generator is also used to validate the design of the emulation system. The SBI system utilizes two software suites for this purpose:

- Satellite ToolKit
- Netspec

2.5.1 Satellite ToolKit

The Satellite ToolKit (STK) [7] program from Analytical Graphics Inc. is a software suite designed for modeling satellite and space vehicle systems. STK provides easy and fast methods of satellite propagation and calculation of satellite coverage times.

The main features of STK utilized in the SBI system are outlined below:

- Creation of scenarios with satellites and ground stations
- Satellite propagation with different types of propagators
- 2-Dimensional and 3-Dimensional views of the scenario
- Ability of a client program to connect to STK via a TCP socket
- Generating reports detailing access times and distances between nodes
- Generating reports of satellite positions and daylight times
- Creation of satellite chains and constellations

2.5.2 NetSpec

NetSpec [8][9][10] is a traffic generator program developed at the University of Kansas. The main feature of NetSpec is that it supports different types of data traffic models including burst type, mpeg, video and voice traffic. It also involves a server daemon that precludes the invocation of an explicit server during the actual data transfer process. This places it apart from other data traffic generators like ttcp that require an explicit server process to be invoked on the server machine. Because of this feature, the client can directly start the data traffic generation process.

This chapter provided a description of the architecture of the SBI Emulation System. The next chapter discusses the architecture and design of the Emulation Manager and related software.

3 SBI Emulation Manager

This chapter explores the architecture and design of the SBI Emulation Manager (EM). The first section describes the overall architecture and the subsequent sections explain the detailed design of the modules in the EM.

3.1 Architecture

This section describes the architecture of the Emulation Manager. The Emulation Manager is made up of the following modules:

- I/O Interface: The Input/Output Interface for the SBI Emulation System is composed of an XML Interface, Status Windows and a graphical display interface.
- Node Database: Information regarding the SBI nodes is stored in the Node Database.
- STK Interface: The STK interface consists of a TCP connection to the Satellite ToolKit application. STK is used for retrieving access and position reports of the nodes and as a graphical system output.
- Operations Node Interface: This interface connects to the Operations Node and is used mainly as a mediator between the Operations Node and STK.
- Operations Channel Interface: The Operations Channel Interface receives messages from the Operations Node destined for the SBI Nodes and appropriately

forwards them. This includes the routing table updates and the instrument scheduling messages.

- Node Manager: The Node Manager configures the nodes and readies them to receive the emulation messages. The network interfaces and the QoS modules on the nodes are configured and updated through commands sent by the Node Manager.

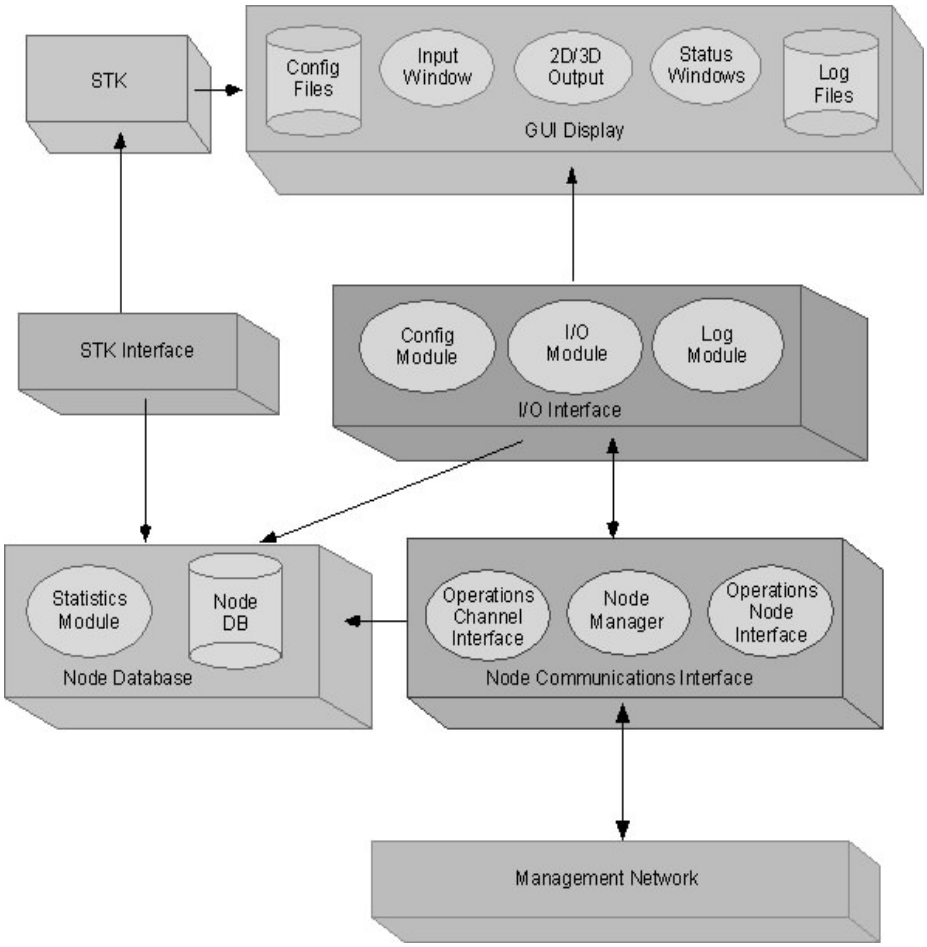

Figure 3-1: Emulation Manager Architecture

The following sections describe the functions and design of the modules in detail.

3.2 I/O Interface

The I/O Interface in the SBI Emulation System is composed of the following modules:

- XML Interface [11]
- Status Windows
- STK Graphical Displays

The following sections detail the working of these modules.

3.2.1 XML Interface

The SBI Emulation System makes use of an XML file as input for the emulation scenario. The XML file is parsed and the scenario information is stored in the Emulation Manager. This section provides an overview of the XML interface used in the SBI Emulation Manager.

The SBI XML Interface uses the Apache Xerces [12] package for XML parsing. This section provides a general overview of XML and then goes on to describe the main components of the SBI XML Interface. The Document Object Model [13] format of parsing XML files and the two main parts of the SBI XML Interface, the Validator and the XML Parser are discussed.

3.2.1.1 Overview of XML

The eXtensible Markup Language (XML) is a markup language along the lines of HTML. XML provides an interface for describing data. The main difference between HTML and XML is that HTML uses a set of pre-defined tags and XML allows the

use of user-defined tags. This allows XML to be used in different types of data processing applications.

XML provides the use of two main types of file definitions.

- Document Type Data [14]
- XML Schema Definition [15][16]

The Document Type Data (DTD) definition for XML is a very simple model for a XML file. A DTD defines the structure of an XML file and provides a definition for the individual elements of the XML file.

The XML Schema Definition (XSD) provides a far more complex and restricting definition for an XML file. The Schema definition provides the basic data types and allows extensibility to higher user-defined data types. User-defined data formats are also allowed in a schema, which can be used for strict syntax checking for data objects. The Schema interface uses a schema definition file that provides a listing of the allowable syntax for the XML file. This definition file is used both for validating the syntax as well as parsing the contents of the XML file.

3.2.1.2 Document Object Model

The data from an XML file can be parsed and manipulated using a Document Object Model (DOM) interface. The DOM interface is a programming interface for XML documents and allows a program to navigate an XML file and access and modify its individual elements. The DOMParser is an interface of the xerces package that allows parsing of a XML file in accordance with the specified schema. It provides methods for specifying a schema definition file and validating a XML file in accordance with

the schema and provides a tree view of the XML file tags. Individual nodes in the tree can then be accessed and their values stored.

3.2.1.3 SBI Validator

The Validator class extends the DOMParser class of the Xerces XML parser package. The Validator checks for correctness of the scenario file according to the schema. The schema used is outlined in Appendix II. This is accomplished by invoking the `parse()` method of the DOMParser class.

3.2.1.4 SBI XML Parser

The XMLParser class is a SBI specific parser that parses all the tags in the XML file. The XMLParser class contains the `parse()` method, which parses the tags in the scenario XML file. The values defined in the tags are then stored in the Node database.

3.2.2 Status Windows

The SBI Emulation System makes use of two status windows to display the working of the system. The status windows are the Network Status and the Instrument Status. This section provides an overview of the features of these windows.

3.2.2.1 Network Status

The Network Status Module keeps track of the network conditions in the SBI Emulation System. The conditions that are monitored and displayed to the user are:

- Network Routes: This module monitors the routes in the SBI network and route updates are displayed in the status window.
- Link Specifications: This module also keeps track of the link specifications on the satellites and ground stations in the SBI network. The parameters that are monitored are the interface bandwidths and the delay and bit error rates for the communication links established in the network.
- Interface Status: This module also tracks the status of the network interfaces on the satellites and ground stations. The number of bytes and packets transmitted and received on each interface are displayed to the user.

3.2.2.2 Instrument Status

The Instrument Status Module keeps track of the satellite instruments in the emulation scenario. This module monitors the following parameters

- Instrument specifications: The instrument specifications that are displayed on the Instrument Status Window include the instrument data traffic rate and the data type.
- Data Traffic Specifications: The Instrument Status Window keeps track of data traffic generation by the SBI nodes and displays a decrementing counter for data transmission time.

3.2.3 STK Graphical Displays

The Satellite ToolKit application is the main output display interface for the SBI Emulation System. STK provides two types of displays, a 2-Dimensional display and a 3-Dimensional display.

3.2.3.1 STK 2-D Window

The STK 2-D Window provides an equidistant cylindrical projection view of the Earth. The nodes comprising of the satellites and ground stations are super-imposed on this view. SBI uses STK's real-time animation capabilities for displaying the SBI network. Network connections in the SBI network are also capable of being displayed in the window.

3.2.3.2 STK 3-D Window

The STK 3-D Window provides a view of the Earth as a globe. Satellites and ground stations can be viewed on and above the Earth at the specified altitudes. Provision for rotating the Earth and zooming in on specific portions of the display exist as well as viewing the display from the point of view of different objects in the scenario. The 3-D window also displays the SBI network connections.

3.3 Node Database

The SBI Emulation Manager stores information regarding the items in the scenario. These items include satellites, facilities, their network interfaces and satellite instruments. The specifications for these objects are read from the input XML file and

stored for retrieval during later processing. This section details the various objects and the data that are stored within them.

3.3.1 Scenario

The Scenario object contains the specification of the emulation scenario being run and uses the following data items:

- Scenario Name: The scenario is identified by a unique name, which is also used to define the scenario in STK.
- Scenario Start Time: The scenario start time defines the start time of the emulation.
- Scenario End Time: The scenario end time defines the end time of the emulation.

3.3.2 Interface

The network interfaces on the SBI nodes are the devices used for data transfer. These interfaces are specified in the scenario and setup on the nodes.

- Interface ID: The identification of the interface
- Interface bandwidth: The bandwidth allocated to the interface

3.3.3 Instrument

The satellite instruments model the data traffic in a SBI scenario. The parameters that specify the instruments are:

- Instrument Name: A unique name for the instruments
- Instrument Data Type: The type of data traffic generated by the instrument

- Data Rate: The data traffic rate generated by the instrument
- Destination Name and Port: The destination for the data traffic

3.3.4 Satellite

The Satellite object stores information regarding the satellite nodes in the scenario.

- Satellite Name: this name is unique in the scenario and defines the satellite object.
- Propagation parameters: SBI uses the two-body propagation method for satellite propagation. These parameters define the orbits of the satellites.
- Instrument Specifications: The specification of the instruments on board the satellites is provided.
- Interface Specifications: The network interfaces on board the satellites are specified here.

3.3.5 Facility

Facilities or ground stations are generally control centers and act as the end-points for data in a SBI network. The facilities are specified by the following parameters.

- Facility Name: This is a unique name in the scenario.
- Positional parameters: The position of a facility on the surface of the Earth is specified by the latitude and longitude.
- Interface Specifications: The network interfaces on board facilities is specified here.

3.4 STK Interface

The STK Interface library interfaces with the Satellite ToolKit application. The SBI scenario is executed in STK with the real-time offset specified. A TCP connection is initiated to STK and command requests are sent through this connection. The STK Interface performs the following functions:

- Starts up STK and initiates a connection to it
- Loads the user-specified scenario and fills it with the satellites and facilities required for the emulation
- Configures 2-D and 3-D graphical outputs for the scenario
- Creates chains between nodes as directed by the Operations Channel according to the routes setup in the scenario

In addition, the STK Interface also forwards requests from the Operations Node to STK and returns the replies. The main requests are for the following:

- Current Emulation Time
- LOS Access Time for access between specified nodes
- Range between specified nodes
- Orbital position information for satellites
- Sunlight exposure times for satellites

3.5 Operations Node Interface

The Operations Node Interface receives request messages from the Operations Node. The Operations Node sends the requests to STK and forwards the responses back to the requesting module. The requests received from the Operations Node are:

- LoS access time reports: Access Time Reports are requested for each pair of nodes in the scenario. Access Time Reports are a set of records containing the times for each access interval for a pair of nodes. Each access record contains two fields: start time and end time. The start and end times signify the access interval for the nodes.
- Node distance reports: Node Distance Reports are also requested for a pair of nodes. A Node Distance Report is a set of records containing the distance between the nodes at an instant of time. Each record consists of two fields, the time and the range. It signifies the range between the nodes at that instant of time.
- Satellite position reports: Satellite position reports are requested for a particular node in the scenario. Satellite position reports enumerate the satellite's footprint on the Earth in latitude and longitude format. The report is composed of records containing the time and the satellite position at that instant of time.
- Emulation Time: The Operations Node requests the emulation time at startup.
- Orbital Period: The Operations Node on startup requests the Orbital Period of satellites. This is used for modeling the data gathering capabilities of satellites.

3.6 Operations Channel Interface

The Operations Channel receives the routing and instrument scheduling updates from the Operations Node and transmits them to all the nodes. The messages are also sent to the other Emulation Manager modules for processing. The Operations Channel Interface listens on the Operations Channel for messages from the Operations Node. Messages received are routing updates destined for the nodes and instrument scheduling messages for the nodes for the purpose of generating data traffic. The received messages are copied and the original message is sent to the destined node via the Management Network. A copy of the message is sent to the Emulation Manager modules where it is used for the display interface and statistical computations.

3.7 Node Manager

The Node Manager works as a Node Controller Interface and sends commands to and receives status reports from the Nodes. The Node Manager configures the nodes and readies them for the emulation. The main functions performed by the Node Manager are:

- Query the nodes and select the requisite number of nodes needed for the emulation
- Configure the physical Ethernet interfaces on the nodes
- Create and configure the Virtual Ethernet interfaces on the nodes
- Create and configure the QoS features on the interfaces

- Start the execution of the Node Software on all the nodes and the Operations Node Software on the Operations Node
- During the emulation run, the Node Manager updates the QoS features periodically
- On shutdown of the emulation scenario, the Node Manager shuts down the QoS and the Virtual Ethernet interfaces.

3.8 Emulation Manager Functionality

This section enumerates the functionality of the Emulation Manager. The Emulation Manager performs the following tasks:

- It parses the XML input file chosen by the user
- Starts up STK and instantiates a connection to it
- Starts the Operations Channel, Node Monitors, Status Modules.
- Sends commands to the Node Manager to ready the nodes for emulation according to the scenario requirements
- Sends commands to the Node Manager to start the emulation on the nodes
- Receives report requests from the Operations Node, forwards requests to STK and returns the reports to the requesting node
- Receives route updates and scheduling messages on the Operations Channel and forwards these messages to the destined nodes
- Sends a copy of these messages to the output interface for display
- Receives status messages from the nodes and sends them to the output interface

This chapter provided a detailed view of the Emulation Manager and its associated modules. The next chapter describes the architecture and functionality of the centralized Operations Node.

4 Operations Node

The SBI Emulation System uses a central specialized node called the Operations Node. The Operations Node computes the routing updates for the SBI network and feeds them to all the nodes in the network. The Operations Node also computes the data scheduling times for the node instruments according to the traffic models incorporated into it. These scheduling times are then sent to the corresponding nodes and the traffic generation is started. This chapter explores the architecture of the Operations Node and the design of the individual modules in the Operations Node.

4.1 Architecture

The Operations Node Software is made up of three main programs:

- Attributes Program
- Routing Program
- Instrument Scheduling Program

4.1.1 Attributes Program

The Attributes Program performs the following functions:

- The Attributes Program receives information regarding the nodes on start-up.
- The Routing and Instrument Scheduling Programs request information from the Attributes Program for their processing needs. The Attributes Program obtains this information from the Emulation Manager and returns it to them.

- The Attributes Program provides the data regarding the nodes to the Routing Program that processes the information and generates an efficient topology for the network.
- The Attributes Program passes on the node information to the Instrument Scheduling Module that computes the schedule for the nodes.

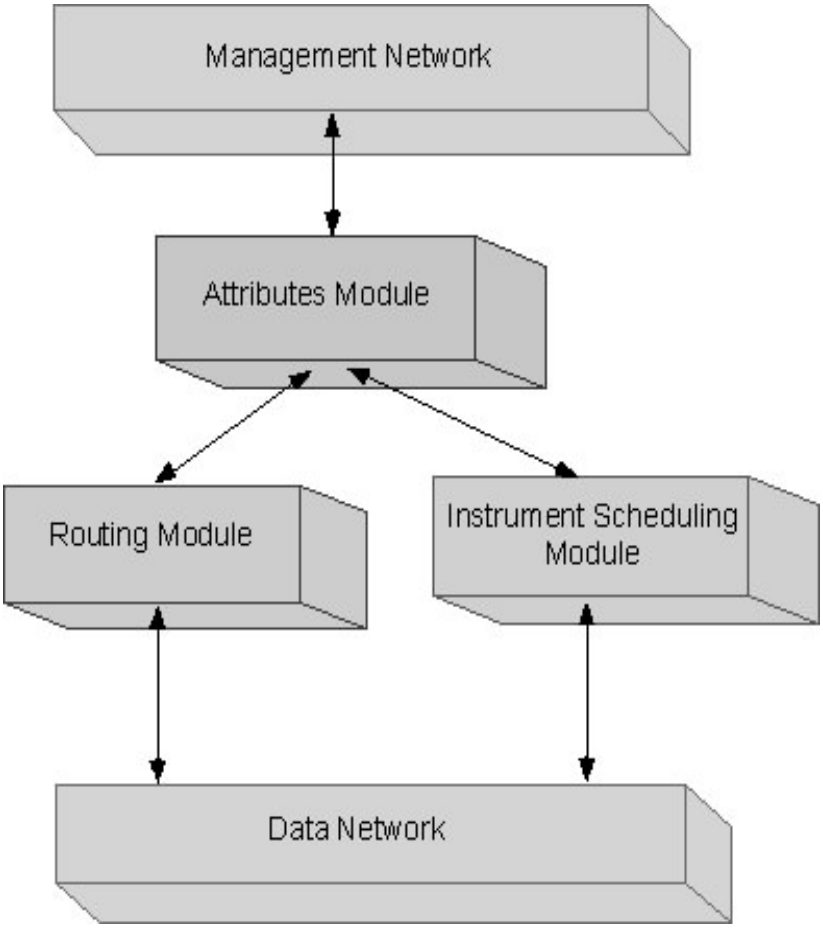

Figure 4-1: Operations Node Architecture

4.1.2 Routing Program

The Routing Program develops efficient routes for data flow in the SBI network. The Routing Program performs the following tasks:

- It requests LOS times and access distances between each pair of nodes from the Attributes Program as required.
- The Routing Program computes the topology of the network using this information.
- The new set of routes are then computed and transmitted to all the nodes in the network.
- It then waits for the next LOS event to occur for the next topology generation.

4.1.3 Instrument Scheduling Module

Satellites provide Earth surface observation data over a variety of conditions depending on the Earth's topology. Satellites can scan and generate data over land, ocean, ice or a combination of these. The SBI Emulation System contains an Instrument Scheduling Module [12] that performs the task of scheduling the data traffic generators according to the actual satellite types used in the emulation. The Instrument Scheduling Module performs the following functions:

- It receives the node information on startup from the Attributes Program.
- It requests and receives satellite position information and satellite sunlight time information from the Attributes Module.

- The scheduling times for the satellite instruments are computed and transmitted to the corresponding nodes to generate data traffic.

This thesis concentrates on the design of the Attributes Program and the Routing Program. The next section describes some of the design strategies for the SBI Nodes. The later sections describe the detailed design of the Attributes Program and the Routing Program. The design of the Instrument Scheduling Program is out of the scope of this work. Considerable work on satellite data traffic models and instrument scheduling emulation has been described in [12].

4.2 Design Strategies

This section details some of the design strategies employed in the SBI Emulation Operations Node. These include the emulation time handling mechanism, the data termination interface strategy and the route update time handling problem.

4.2.1 Modular Design

The architecture of the Operations Node has been designed keeping modularity in mind. The main rationale for the architecture has been the thought that any module should be replaceable with an equivalent module at some point. The centralized architecture presented herein is a stepping-stone towards evolving into a more distributed structure, where each SBI node performs its own routing and scheduling computations.

4.2.2 Emulation Time

The Emulation Time handling mechanism is one of the base features of the Operations Node. The SBI emulation system runs with a real time offset, i.e., in real time starting from the date and time specified. The emulation start time is received from the Emulation Manager on startup. The emulation time library then calculates the offset of the current system time from the emulation time and maintains this difference. When the current emulation time is required, the offset in relation to the current system time is used to calculate the current emulation time.

4.2.3 Data Termination Interfaces

A dynamic routing environment leads to a host of complex issues. One such is the problem of data termination interfaces, which are the end-point interfaces for an application generating data traffic. The data termination interfaces need to be fixed for data transfer at the application layer. However, in an ever-changing topology such as a satellite network, the routes change with every change in topology and the routes to the termination points also change. In order to keep the termination points constant in the emulation system, experiments were carried out regarding the use of the "dummy interfaces". Linux provides the use of "dummy interfaces" which can be assigned IP addresses and can be used as data source and destination points in a socket application. The dummy interfaces are used in the SBI emulation system for this purpose. Under this system, each node has one dummy interface and it acts as both a source and terminating point for data traffic at the application layer. This

requires the use of two routing commands for a node, one for routing data to the incoming interface of the destination node and one to the dummy interface of the destination node. This is illustrated by the following representation.

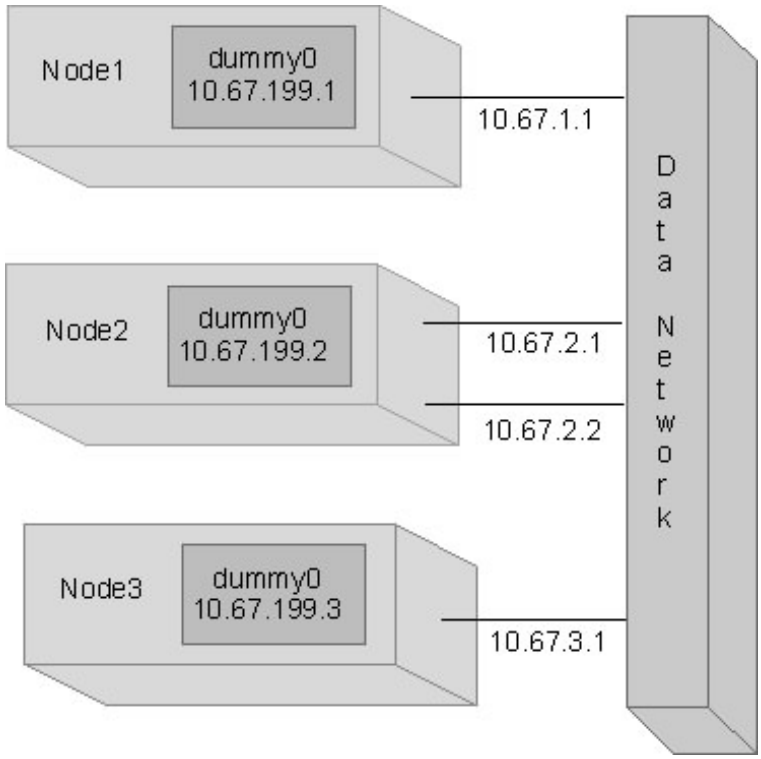

Figure 4-2: Illustration of Dummy Interface

The following tables detail the route entries present on Node1, Node2 and Node3 respectively.

Destination	Gateway
10.67.2.1	10.67.1.1
10.67.199.2	10.67.2.1
10.67.3.1	10.67.2.1
10.67.199.3	10.67.2.1

Table 4-1: Route entries for Node1

Destination	Gateway
10.67.1.1	10.67.2.1
10.67.199.1	10.67.1.1
10.67.3.1	10.67.2.2
10.67.199.3	10.67.3.1

Table 4-2: Route entries for Node2

Destination	Gateway
10.67.2.2	10.67.3.1
10.67.199.2	10.67.2.2
10.67.1.1	10.67.2.2
10.67.199.1	10.67.2.2

Table 4-3: Route entries for Node3

The extra route entries make it possible for the other nodes to access the destination node's dummy interface for data transfer purposes. Thus, an application on Node1 is able to initiate data transfer with the source IP address as 10.67.199.1 and the destination IP address as 10.67.199.3.

4.3 Attributes Program

The Attributes Program acts as a mediating entity between the Operations Node computational modules and the Emulation Manager. The Routing and the Scheduling Modules request node information updates from the Attributes Program. This information is in turn requested from the Emulation Manager and the replies are returned to the requesting program.

The Attributes Program stores the initial node information relayed from the Emulation Manager at the beginning of the emulation. This information is stored in data structures and is relayed to the other Operations Node programs at the start of the scenario. The following section mentions the data structures present in the Attributes Program and the next section discusses the sequence of execution.

4.3.1 Data Structures

The Attributes Program maintains a node data table with information about all the nodes in the scenario. The node data table contains the following information:

- Node identification number
- Type of Node: Satellite or Facility
- Orbital parameters of the satellite node or position of the ground station facility
- Communication characteristics: Bit Rates, Signal Strength
- The number of antennae on the node
- The IP address for each node antenna

4.3.2 Functionality

The Attributes Program acts as a link between the Routing and Scheduling Programs and the Emulation Manager. As such, the Attributes Program effectively isolates these programs from the Emulation Manager. The Routing and Instrument Scheduling Programs request for access and position reports from the Attributes Program. The Attributes Program contacts the Emulation Manager and receives this

information, which is then relayed back to the requesting program. The sequence of events in the Attributes Program is as follows:

- On startup, the Attributes Program receives information from the Emulation Manager regarding the scenario. The information received is pertaining to the details of the interfaces on all the SBI nodes and the instruments on the Data Source and Relay Satellites.
- The Routing and Scheduling Programs initiate a connection to the Attributes Program on startup. The Attributes Program then transfers the interface data to the Routing Program and the instrument data to the Scheduling Program.
- It then waits for requests from the two programs. Upon reception, it contacts the Emulation Manager and relays the replies to the requesting program.

4.4 Routing Program

The Routing Program computes the topology of the network and generates the routing table updates for all the nodes in the SBI network. The routing updates are then transmitted to the nodes through the Operations Channel. The following modules are used in the Routing Program:

- Route Interface: The Route Interface module performs the necessary configurations prior to computing the network topology.
- Access Module: The Access Module retrieves, stores and processes the access times for LOS access between each pair of nodes in the SBI network.

- Topology Generator: This module performs the computations involved in generating a network topology for the SBI system.
- Route Generator: The Route Generator Module takes the generated topology and computes the routing table updates for all the nodes in the SBI network.

A pictorial view of the Routing Program modules is shown in Figure 4-3.

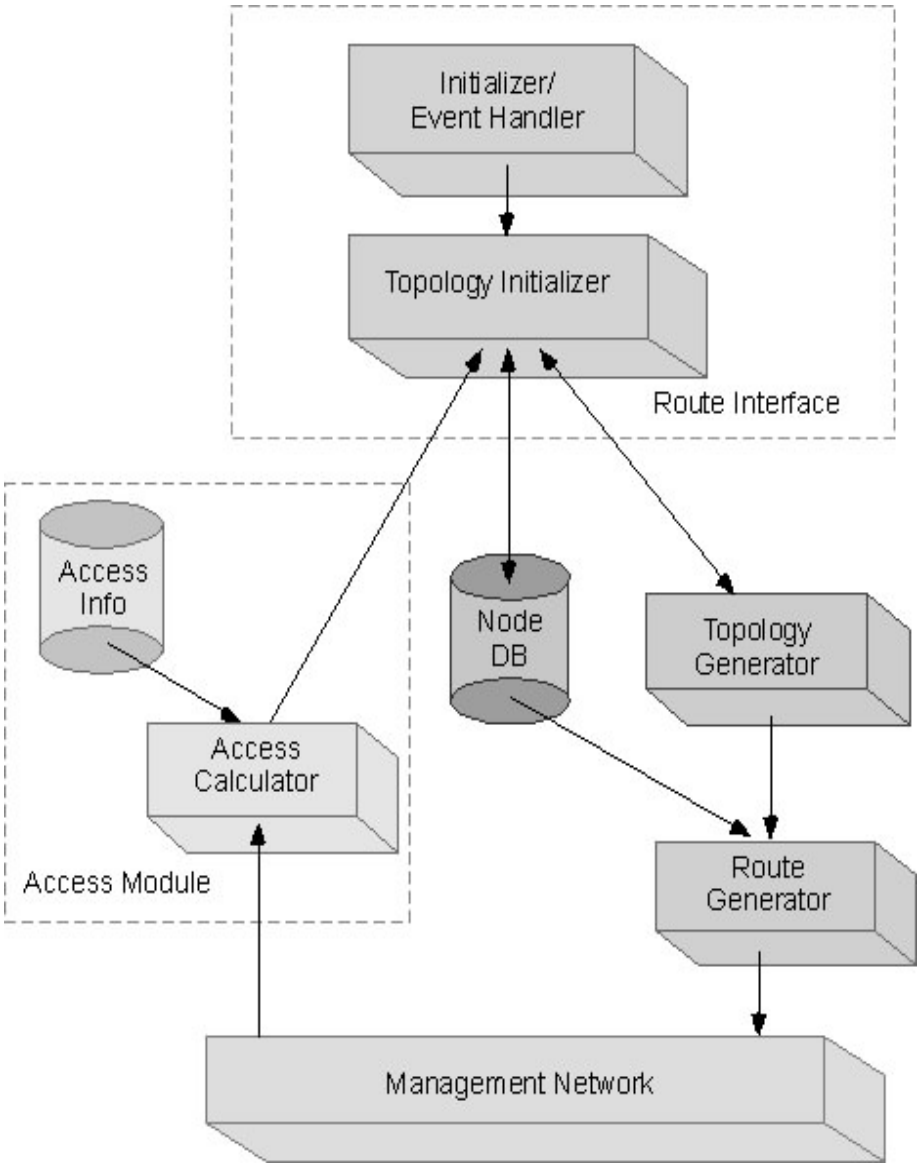

Figure 4-3: Routing Program Architecture

The flow of events in the Routing Program is given in Figure 4-4.

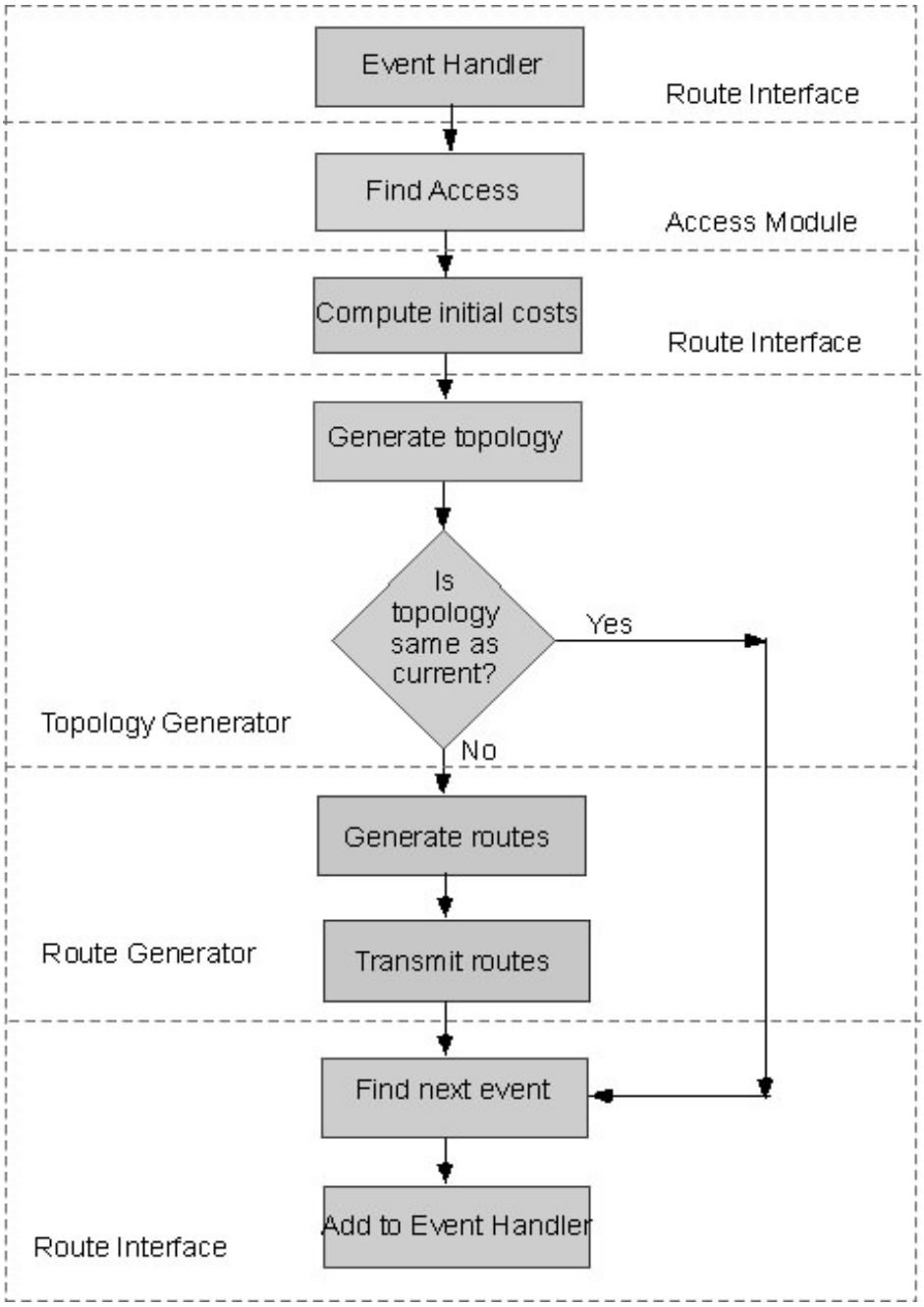

Figure 4-4: Routing Program Flow of Events

The following sections describe each of the modules in detail.

4.4.1 Route Interface

The Route Interface Module performs some of the preliminary actions in computing the network topology. The main functions of the Route Interface Module are:

- Topology Initialization: The Topology Initialization Module performs the initialization functions before computing the network topology.
- Event Queue: The Event Queue Module is used to set events in the Routing Program.

4.4.1.1 Topology Initialization

The Topology Initialization Module performs the preliminary functions related to the topology generation for the SBI network. This includes the initialization process and the cost calculation for the routes. The Topology Initialization Module performs the following functions:

- For each link in the network, it is found if the links have line-of-sight to each other.
- If line-of-sight is present, the distance between the nodes comprising the link under consideration is found. The initial cost is calculated based on the distance.
- If the two nodes do not have line-of-sight to each other, the maximum cost (of 1000) is attached to the link.

4.4.1.2 Event Queue

The Event Queue is set according to the line-of-sight events that occur in the SBI network. The Access Module provides information regarding the event times in the

network. These event times are provided to the Event Queue Module and an alarm is set for the event time. The event alarm runs the event handler method, which calls the methods for computing the network topology.

4.4.2 Access Module

The Access Module handles the LOS access reports in the Routing Program. The Access Reports are used to determine the access times and range between nodes in the network.

4.4.2.1 Data Structures

The Access Module handles a data structure for each of the nodes in the scenario. This structure contains the access information regarding each of the other nodes in the network relative to the current node. The Access Module uses the following data items for it's processing:

- Destination ID: This signifies the destination node for the current source node.
- Current Record Pointer: This points to the current position in the access record.
- Start Time: The start time refers to the access start time for the current access record.
- Stop Time: The start time refers to the access stop time for the current access record.
- Current Range Record Pointer: This points to the record under processing in the range report.

Two kinds of reports are processed for calculating the access between the nodes. The first one is the access report, which is a series of records detailing the times of access for the nodes. A record in the access report is as follows:

2002/01/01 02:47:03.30,2002/01/01 03:28:52.67

The first term represents the access start time and the second term represents the access stop time.

The second type of report is the range report. The range report also contains two fields, the time and the range between the nodes at that point in time. A range report record is as follows:

2001/10/03 00:02:00.00 13571.037274

4.4.2.2 Functionality

The following is the sequence of events in the Access Module:

1. The Route Interface Module requests access reports for a pair of nodes when calculating the costs of the communication links in the network.
2. If no report exists, an access report is requested for the next 24 hours from the Attributes Program. Else, step 6 is executed.
3. If no access report is received, then there is no access for the period specified. The startTime and stopTime fields are updated to the end time of the report request.
4. The access report is analyzed and the current record pointer is updated to point to the next field. The record is processed and the start and stop times are stored in the startTime and stopTime fields respectively.
5. If the current time is between start and stop times, then the nodes have access.

6. If a report already exists, then the startTime and stopTime fields are compared with the current time.
7. If the current time is greater than stop time, the next record in the report is accessed. If there are no more records, then steps 2 to 5 are executed. Else, steps 4 and 5 are executed.

4.4.3 Topology Generator

The Topology Module performs the task of generating a network topology for the SBI network. The Topology Module takes the communication link costs for each of the links as input and computes a new optimized topology. The algorithm used for the computation is discussed in the next chapter.

4.4.4 Route Generator

The Routing Module uses the computed network topology as input from the Topology Module and generates the routing table updates for all the nodes in the SBI network.

4.4.4.1 Functionality

The Routing Module determines interfaces for the source, hop and destination for each route. The Routing Module works as follows:

1. The computed topology for the network is received from the Topology Module.

2. For each of the nodes, the routes are determined to the rest of the nodes in the new topology. Each route is signified by source, the next hop and destination interfaces.
3. The routes are computed afresh as outlined below for each new topology. This reduces the computational complexity rather than utilizing the previous set of routes to configure the new set.
4. A recursive algorithm is employed to determine interfaces for all the routes.

The method of determining the new route configuration is described here. A route consists of a source, nexthop and destination interfaces denoted by *src*, *hop* and *dst* respectively.

1. First, a check is made whether a route already exists between *src* and *dst*. If yes, the same route is used.
2. Else, if the *src* and *hop* nodes are different, it is checked whether a route exists between *src* and *hop*.
3. If yes, it is checked if there exists a route between *hop* and *dst* nodes. If so, the same routes are used and then step 6 is executed. Otherwise, a recursive computation is attempted to find a route between the *hop* and *dst* nodes by executing from step 1 again.
4. Else, it is checked if there is a route between the *hop* and *dst* nodes. If so, any of the free interfaces on the *src* and *hop* nodes and the *dst* interface from the existing route is used for the route. The additional related routes are then added in step 6.

5. If none of the above routes exist, a free interface on the *src* and *hop* nodes is used and a new route is found between the *hop* and *dst* nodes by executing from steps 1 again for the *hop* and *dst* nodes.
6. The additional routes added for each new route are:
   - The route from *src* to *hop* with *hop* as the next-hop.
   - The reverse route from *hop* to *src*.
   - The reverse route from *dst* to *src*.

This chapter described the design of the Operations Node. The workings of the Attributes Program and the Routing Program have been explained. The next chapter discusses a simple topology generation algorithm for the SBI Emulation System.

5 SBI Network Topology

Routing is an integral part of any communication network. An efficient routing algorithm helps in the continuous movement of data in an ever-changing network. Routing updates allow a node to perform switching of data traffic. In a network whose nodes are in constant motion relative to one another, the problem of routing is multi-fold and has to be updated every time there is a change in the network topology. In the SBI network, the satellites are in constant motion and gain and lose line-of-sight of one another from time to time [18][19]. Thus, a good routing algorithm is required for generating the network topology for the SBI network. This chapter discusses the problems of routing in space and looks at the accepted methods of route computation before outlining a simple and efficient algorithm to generate the network topologies in the SBI emulation system.

5.1 Routing In Space

The space environment is very different from the terrestrial internet environment. High delays and high bit error rates characterize the data transmission in space. Satellites move at high speeds and gain and lose line-of-sight of one another in minutes. Most of the Earth-Observation Satellites reside in LEO orbit and therefore have a very small footprint on the ground and consequently a small window of access with a ground station. Due to these factors, the routing tables for the satellites need to be updated very often. Terrestrial routing techniques like OSPF and BGP are not suited for the satellite network as they involve transfer of information between the

nodes. This is not suited for the satellite environment as they necessitate unnecessary overhead, the routes take a while to stabilize and the information quickly becomes obsolete due to the rapidly changing satellite topology. However, the space segment has a number of advantages, the main ones being:

- Predictability: Satellites move in an orbit around the Earth. The position of the satellites at any point of time can be accurately pinpointed with the help of orbit prediction software.
- Periodicity: Each satellite has a definite period of revolution around the Earth. Extending this to multiple satellites, the satellite network topology will also be periodic in nature.

Two measures have been proposed [19] for routing in the satellite network.

- Discrete Time-Dynamic Virtual Topology Routing (DT-DVTR)
- Virtual Node

These work on the principle of hiding the mobility of satellites from the routing protocols running over the network.

5.1.1 DT-DVTR

Discrete Time-Dynamic Virtual Topology Routing is a method of routing wherein the satellite network is assumed to be a static network for certain periods of time and the routes computed for the network as for a static network. The link activation and deactivation is assumed to occur at discrete times and the topology is computed for

the intervals between these times as for a static network. The routes can be computed offline and broadcast to the satellites in this method.

5.1.2 Virtual Node

In the Virtual Node method of routing, the routes are computed over a static virtual network and the satellites are assumed to move over this virtual network taking the place of each other as they move. For each connection handover, the satellites transfer all their routing information from one to another as they switch places in the virtual network. This method of routing is suited to constellation topologies where the next satellite takes the place vacated by the previous one in the constellation. However, Earth Observation Satellites are not located in constellation orbits and this form of routing is not suited for their topology.

5.2 Routing in the SBI Emulation System

This section describes the characteristics of the routing strategies employed in the SBI Emulation System. The methodologies explained in the previous section are refined and applied in this scenario.

5.2.1 General Characteristics

The SBI Emulation System incorporates the DT-DVTR version of routing in its satellite environment. Gain or loss of line-of-sight between the SBI nodes is characterized as an event. The events are assumed to occur at discrete instants of time and the SBI network is taken to be static in between events. Thus, the routing

protocol does not see the mobility in the satellite system and works on static network topologies. The routes are computed for the static network using the inter-node distance as the metric. The time between events is taken to be small enough so that the metrics are assumed to be constant in this interval.

5.2.2 Route Update Time Handling

The Space Based Internetwork is a good example of a highly dynamic topology. Satellites gain and lose line-of-sight to each other as they move in their orbits. A larger scenario leads to a higher number of connections that are added and removed in a certain time interval. If every connection handover were taken care of as a separate event, the route computations would overload the node's computing powers. For this reason, route computations are done only at discrete intervals. Route update times are rounded off to facilitate better handling of events. Connection gain events should come into picture after the event occurrence. Similarly, connection loss events should come into picture before the actual event time. By this line of reasoning, connection gain events are rounded up and connection loss events are rounded down. A better illustration of this concept can be seen in the timeline in Figure 5-1.

In the figure, the routes are computed at *r* and *r+1*. Events occur at *n-2*, *n-1*, *n*, *n+1* and *n+2*. Events *n-2* and *n* refer to new connections being made due to a pair of nodes coming into line-of-sight of each other. Events *n-1*, *n+1* and *n+2* refer to connections being broken due to a pair of nodes losing line-of-sight to each other.

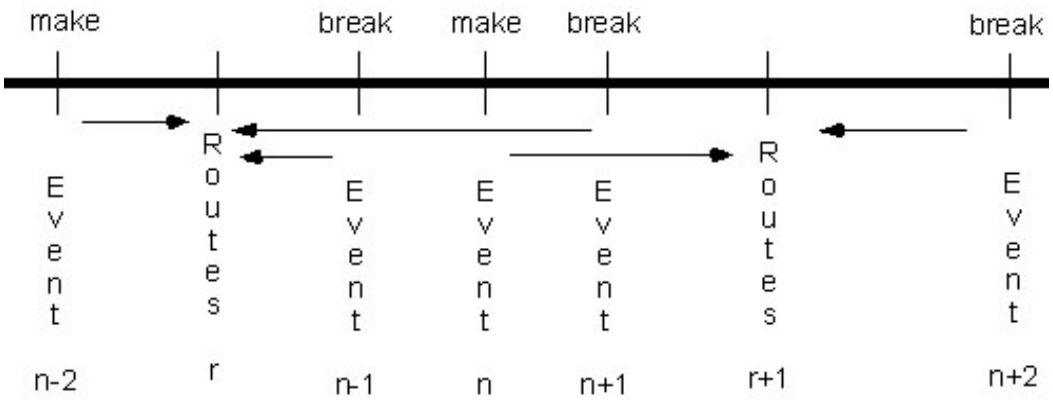

Figure 5-1: TimeLine

Event *n-1* has to be handled in the route update at *r*, since the route update *r+1* is too far ahead in the future and the connection would have already broken by then. Similarly, Event *n* has to be handled in route update *r+1*, as the nodes would not have come into sight of each other at the time of the route update *r*. Thus, Events *n-2*, *n-1* and *n+1* are handled in the route update *r* and Events *n* and *n+2* are handled in the route update *r+1*.

5.3 Algorithm Overview

The SBI Emulation System utilizes an extended minimum spanning tree algorithm [20][21][22] to determine the topology of the SBI network. This section provides an overview of the algorithm employed. The main features of the algorithm are:

- Minimum spanning tree generation
- Link limiting

5.3.1 Minimum Spanning Tree Generation

A minimum spanning tree represents a connected, acyclic minimum weight sub-graph containing all the vertices of a graph. A minimum spanning tree assures that all the nodes in a network will be included in the tree at a minimum cost. This make it perfectly suited for generating the route topology for a network. The SBI Emulation System makes use of a minimum spanning tree algorithm to build the network topology. The minimum spanning trees are computed for each node to generate the least-cost paths for accessing every pair of nodes in the network.

5.3.2 Link Limiting

A minimum spanning tree ensures that a node is able to contact all other nodes in the network with minimum cost. However, the number of interfaces on a node is fixed and may be less than the number of interfaces required for contacting all the nodes using the MST generated routes. Under such circumstances, the number of links in the MST generated routes has to be limited to generate a physically feasible topology. Links can be limited based on a number of factors, such as amount of usage, cost of the link etc. An algorithm is presented that limits links to generate a feasible MST network topology.

5.4 Algorithm Description

This section describes the algorithm used in the SBI Emulation System for generating the topology of the SBI network. An example topology composed of six nodes is used

to illustrate the working of the algorithm. The links formed in this topology are presented in Figure 5-1, which also shows the costs for the links.

5.4.1 Initialization

Two matrices are used in computing the topology, a cost matrix and a hop matrix. The cost matrix is used for determining the Link State between two nodes. The hop matrix is used for determining the nexthop for every communication link. The cost used to compute the topology is mainly a factor of the distance between the nodes under consideration.

The matrices are initialized as follows. For every pair of connected nodes, the respective entries in the cost matrix are set to the cost of the links between the nodes. In the hop matrix, the entry for each of these nodes is set to the adjacent node. If there is no access between a pair of nodes, the corresponding entry in the cost matrix is set to MAX_COST (1000 here) and in the hop matrix to –1.

The initial cost and hop matrices are shown here:

Destination Source	**1**	**2**	**3**	**4**	**5**	**6**
1	-	3	5	5	8	9
2	3	-	2	1000	4	7
3	5	2	-	9	3	5
4	5	1000	9	-	1000	4
5	8	4	3	1000	-	2
6	9	7	5	4	2	-

Table 5-1: Initial Cost Matrix

Destination Source	1	2	3	4	5	6
1	-	2	3	4	5	6
2	1	-	3	-1	5	6
3	1	2	-	4	5	6
4	1	-1	3	-	-1	6
5	1	2	3	-1	-	6
6	1	2	3	4	5	-

Table 5-2: Initial Hop Matrix

Figure 5-2 shows the initial topology. The numbers near the 'x' marks denote the nodes and the numbers at the mid-point of the link lines refer to the link costs.

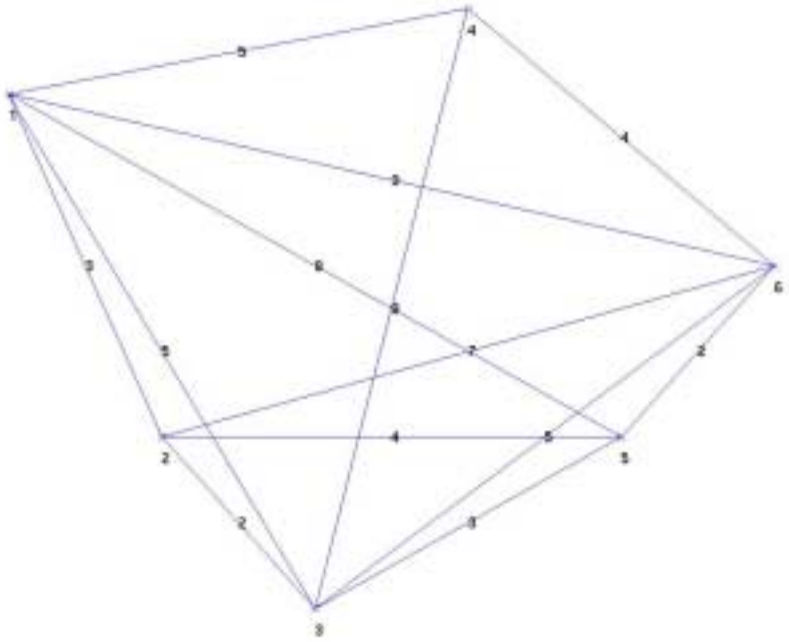

Figure 5-2: Initial Topology

5.4.2 Spanning Tree Computation

The spanning trees for each node are computed after the initialization. Computing the minimum spanning trees from each node provides the best way to reach all the other nodes in that tree from the root node. The spanning trees are computed in the network as follows: For every node in the network, it is attempted to insert another node in between that and every other node, the attempt succeeding if the new cost is lesser than the earlier cost to network between those nodes. In such cases, the hop matrix entry is updated to reflect the new nexthop for the communication between these pair of nodes and the cost matrix entry is updated to reflect the new cost for the link. At the end of this exercise, the matrices will contain the best way to reach any pair of nodes in the network. The cost and hop matrices after the spanning tree computations are shown below:

Destination Source	**1**	**2**	**3**	**4**	**5**	**6**
1	-	3	5	5	7	9
2	3	-	2	8	4	6
3	5	2	-	9	3	5
4	5	8	9	-	6	4
5	7	4	3	6	-	2
6	9	6	5	4	2	-

Table 5-3: Cost Matrix after First Spanning Tree Computation

Destination Source	1	2	3	4	5	6
1	-	2	3	4	2	6
2	1	-	3	1	5	5
3	1	2	-	4	5	6
4	1	1	3	-	6	6
5	2	2	3	6	-	6
6	1	5	3	4	5	-

Table 5-4: Hop Matrix after Spanning Tree Computation

The Figures 5-3 to 5-8 show the results after the spanning tree computations.

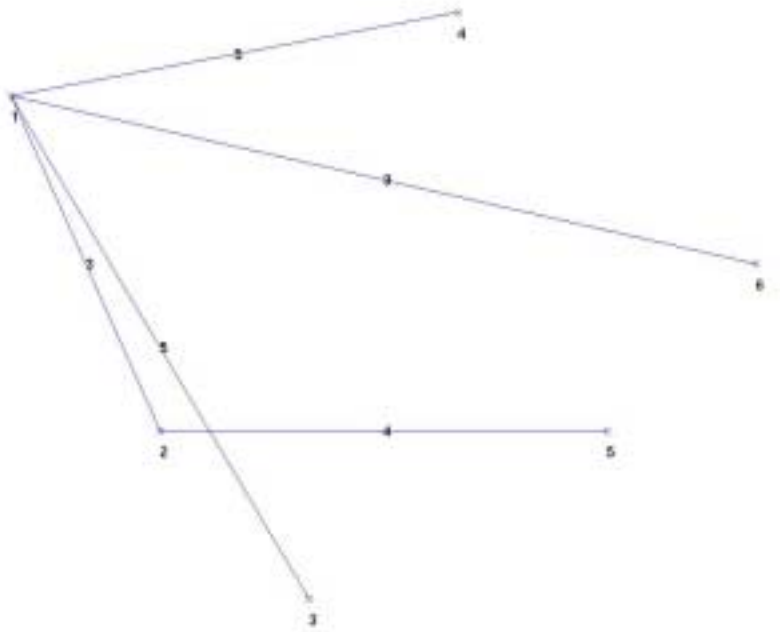

Figure 5-3: First Iteration for Node 1

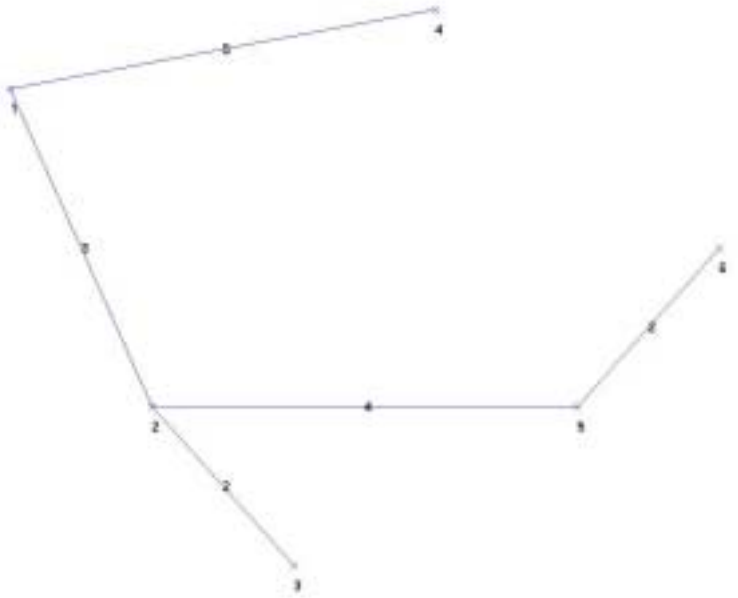

Figure 5-4: First Iteration for Node 2

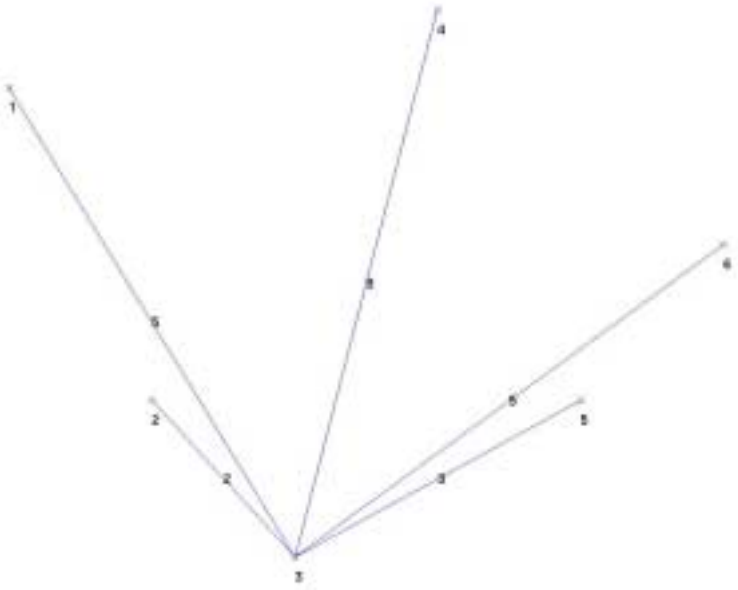

Figure 5-5: First Iteration for Node 3

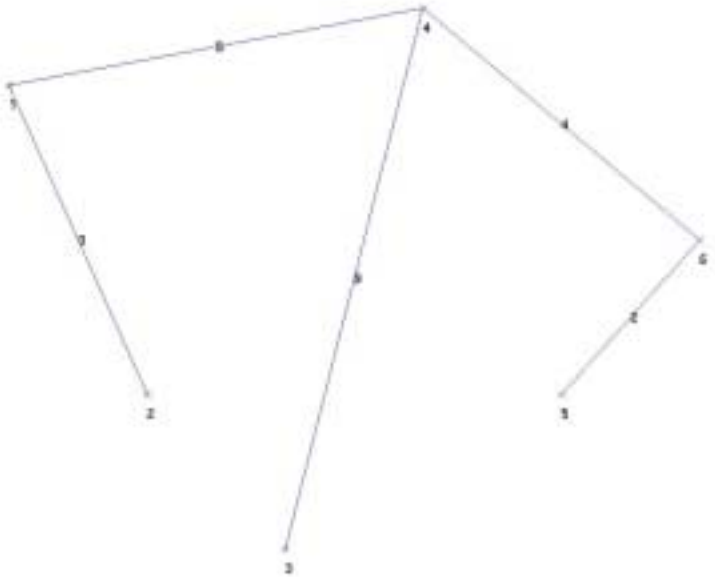

Figure 5-6: First Iteration for Node 4

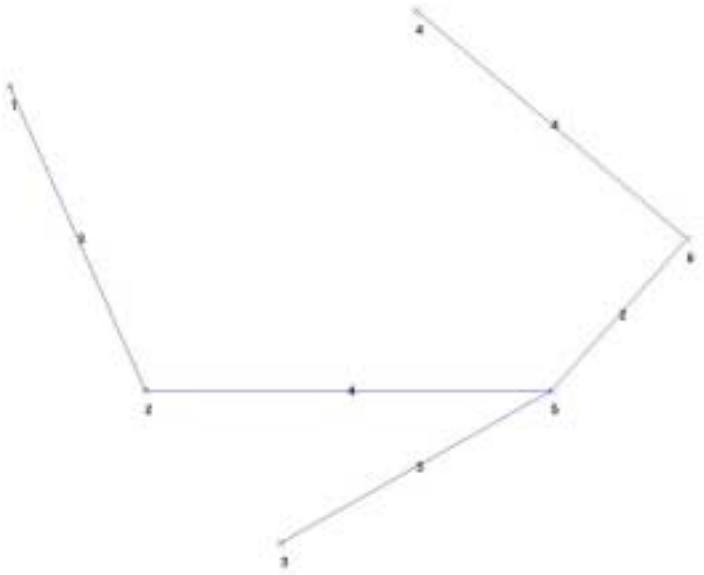

Figure 5-7: First Iteration for Node 5

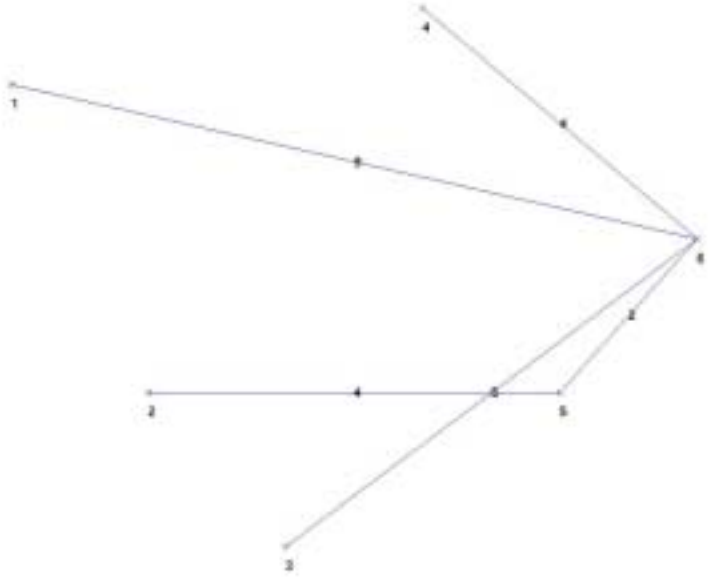

Figure 5-8: First Iteration for Node 6

5.4.3 Link Limiting

The next task is to identify if any of the nodes are using more links than the number of interfaces they possess. Link limiting is to be considered for those nodes that have more links than interfaces. The number of links versus the number of available interfaces per node is tabulated in Table 5-5.

Nodes	**1**	**2**	**3**	**4**	**5**	**6**
Interfaces	3	3	3	3	3	3
Links	4	3	5	3	3	4
Removal	1	0	2	0	0	1

Table 5-5: Interface and Link Comparison

From the table, it is seen that nodes 1, 3 and 6 need more links than there are interfaces available. Furthermore, nodes 1 and 6 use only one extra link and node 3 uses two extra links. Therefore, link limiting is to be considered for nodes 1, 3 and 6.

The criteria for link limiting are:

- Least-used links: The links used the least number of times are removed first. The reason for removing these links is that the least-used links should cause the least amount of change in the topology after removal. Alternate routes should be fairly easy to find for the removed routes.
- Maximum-cost links: The links having the highest costs are removed. This is done so that the most expensive links are removed from the picture.

Destination Source	**1**	**2**	**3**	**4**	**5**	**6**
1	-	3	1	2	-	1
3	1	1	-	1	1	1
6	1	-	1	2	3	-

Table 5-6: Finding Least Used Links – Hop Matrix

The corresponding original costs for these links are reproduced in Table 5-7.

Destination Source	**1**	**2**	**3**	**4**	**5**	**6**
1	-1	3	5	5	8	9
3	5	2	-1	9	3	5
6	9	7	5	4	2	-1

Table 5-7: Original Cost Comparison for Link Removal –Cost Matrix

For node 1, the links 1-3 and 1-6 are the least used. Looking at Table 5-7, it is seen that the link 1-6 has the maximum cost of the two. Thus 1-6 is a candidate for removal. For node 3, all the links are used only once. In addition, two links have to be removed for node 3. One link that can be removed is 3-4, which carries a cost of 9. There exists a choice for the second removal, since both links 3-1 and 3-6 have a cost of 5. Here, a simple first found rule is used and the link 3-1 is removed. Similarly, for node 6, since the link 1-6 has already been targeted for removal, no more links need to be removed. Therefore, the three links removed are: 1-3, 3-4, 1-6. These links are removed from the original topography of the network. The new network topography is shown in Figure 5-9.

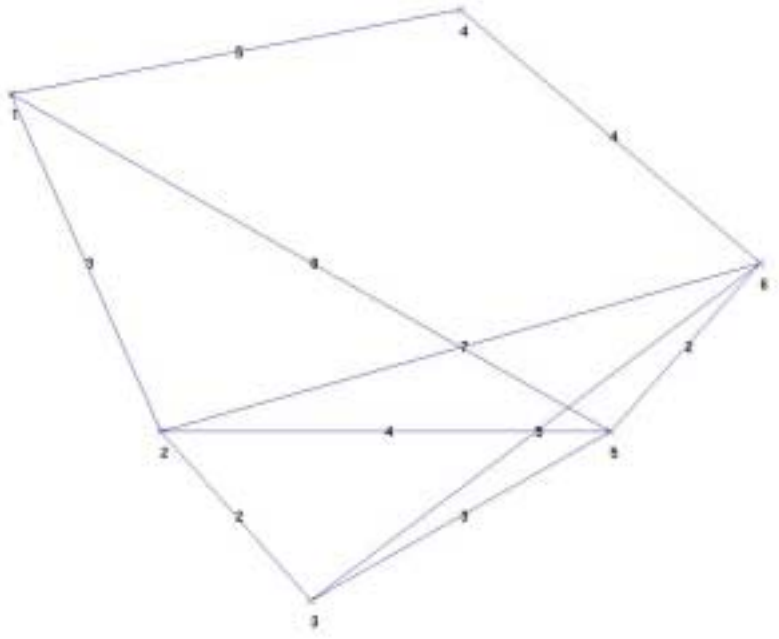

Figure 5-9: Topology after Link Limiting

5.4.4 Spanning Tree Re-computation

The spanning tree is recomputed for the network for the following reasons:

- Removing the least-used links could render certain nodes inaccessible. Re-computing the spanning trees will provide alternate routes in the network for such nodes.
- Removal of links results in a differing topology from the original. The re-computation of the network topology has to be done to change all the routes that were affected by the removal.

The spanning tree is recomputed and the procedure outlined in the previous sections is repeated until a satisfactory network topology is configured, where all the nodes are accessible. Tables 5-8 and 5-9 show the cost and hop matrices used for the second MST computation.

Destination Source	**1**	**2**	**3**	**4**	**5**	**6**
1	-1	3	1000	5	8	1000
2	3	-1	2	1000	4	7
3	1000	2	-1	1000	3	5
4	5	1000	9	-1	1000	4
5	8	4	3	1000	-1	2
6	1000	7	5	4	2	-1

Table 5-8: Cost Matrix for Second Iteration

Destination Source	1	2	3	4	5	6
1	-1	2	-1	4	5	-1
2	1	-1	3	-1	5	6
3	-1	2	-1	-1	5	6
4	1	-1	3	-1	-1	6
5	1	2	3	-1	-1	6
6	-1	2	3	4	5	-1

Table 5-9: Hop Matrix for Second Iteration

The computation of the spanning trees with the modified topology is done again using the original matrices with the removed links. The new cost and hop matrices are shown in Tables 5-10 and 5-11.

Destination Source	1	2	3	4	5	6
1	-1	3	5	5	7	9
2	3	-1	2	8	4	6
3	5	2	-1	9	3	5
4	5	8	9	-1	6	4
5	7	4	3	6	-1	2
6	9	6	5	4	2	-1

Table 5-10: Cost Matrix after Second Iteration of Spanning Trees

Destination Source	1	2	3	4	5	6
1	-1	2	2	4	2	4
2	1	-1	3	1	5	5
3	2	2	-1	6	5	6
4	1	1	6	-1	6	6
5	2	2	3	6	-1	6
6	4	5	3	4	5	-1

Table 5-11: Hop Matrix after Second Iteration of Spanning Trees

Figures 5-10 to 5-15 show the new spanning trees for each of the nodes after the second iteration.

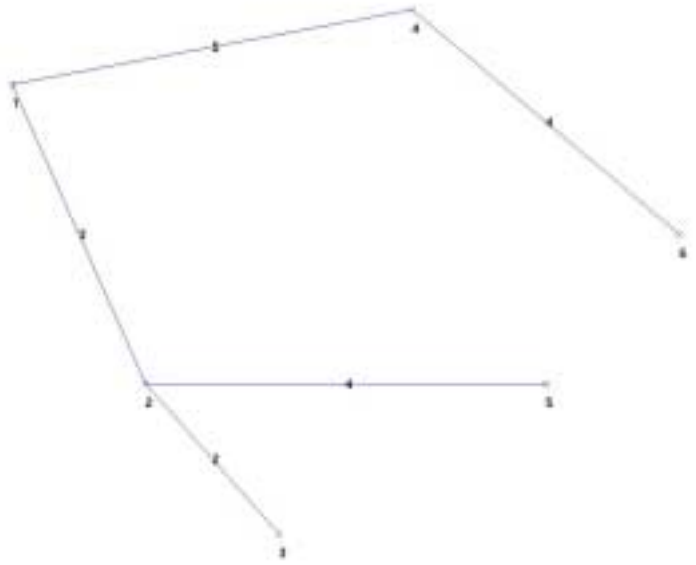

Figure 5-10: Second Iteration for Node 1

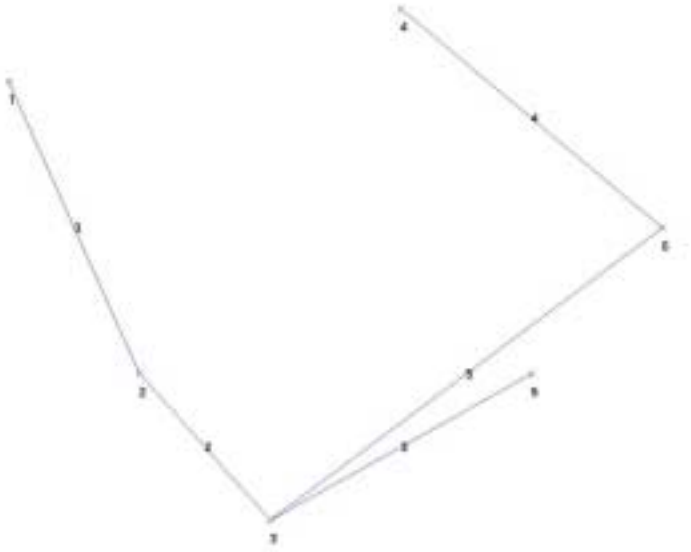

Figure 5-11: Second Iteration for Node 2

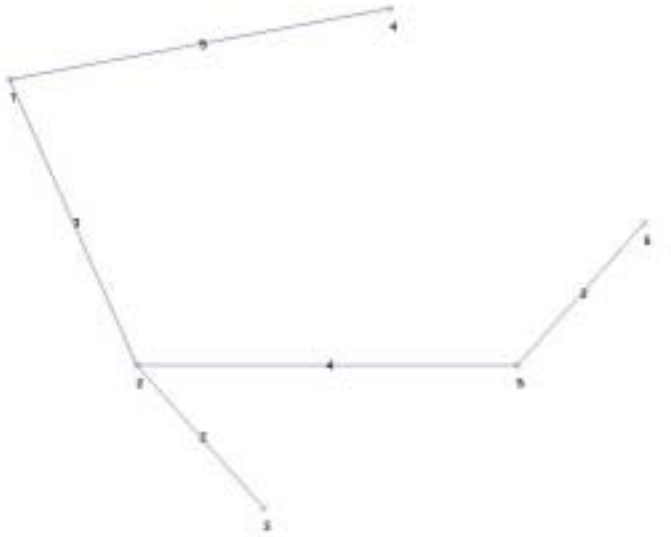

Figure 5-12: Second Iteration for Node 3

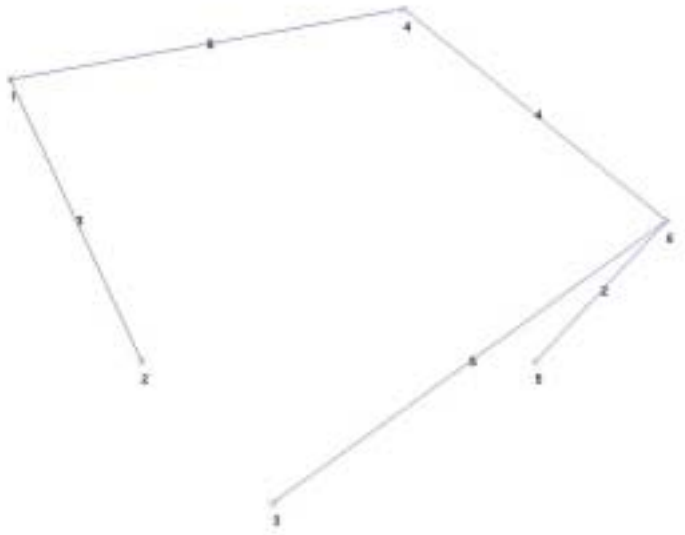

Figure 5-13: Second Iteration for Node 4

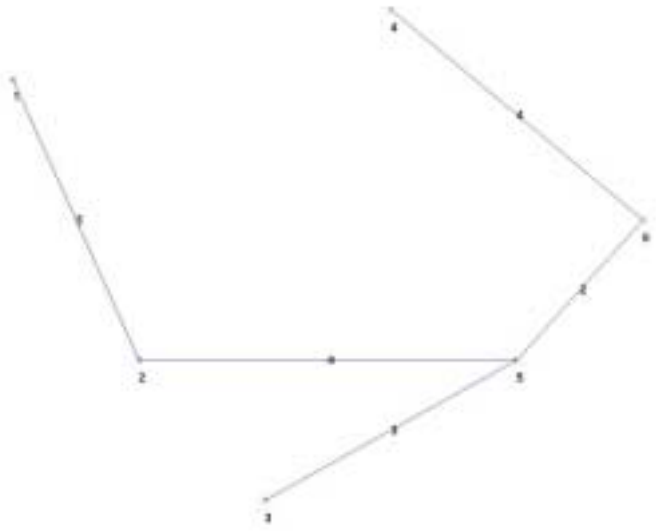

Figure 5-14: Second Iteration for Node 5

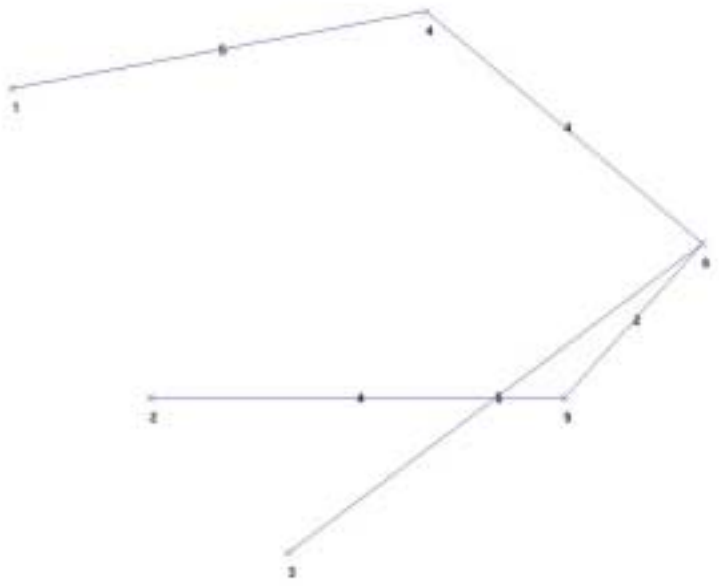

Figure 5-15: Second Iteration for Node 6

5.4.5 New Topology

The topology resulting from the re-computation of the spanning trees is checked and it is ensured that none of the nodes use more links than they can accommodate. It is seen in Table 5-12 that the resulting topography is physically feasible.

Nodes	1	2	3	4	5	6
Interfaces	3	3	3	3	3	3
Links	2	3	3	2	3	3
Removal	0	0	0	0	0	0

Table 5-12: Interface and Link Comparison after Second Iteration

The final network topography is shown in Figure 5-16.

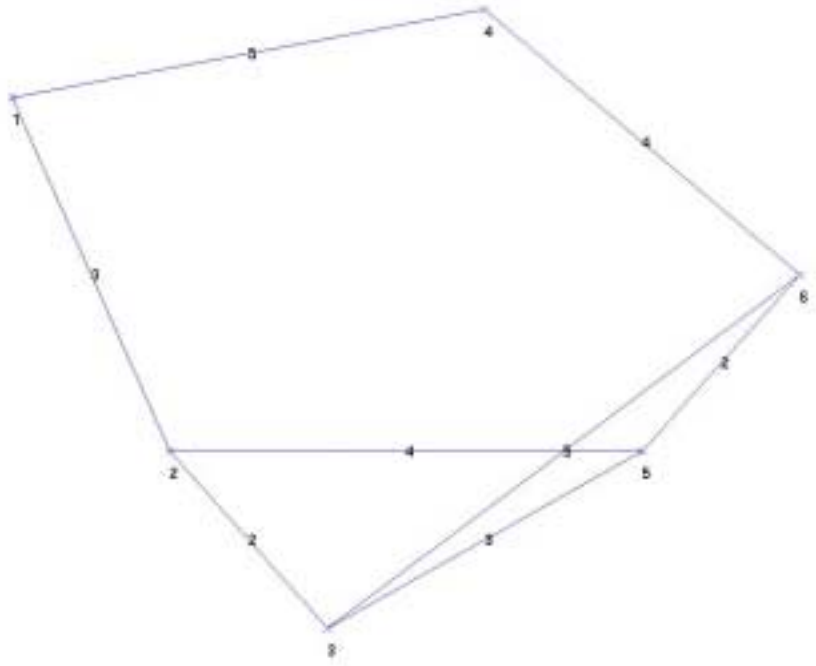

Figure 5-16: Final Topology

The proposed algorithm is thus seen to have resulted in a feasible solution to the problem of limited interface networks.

5.5 Algorithm

This section provides a more concise description of the algorithm detailed in the previous few sections.

1. A cost matrix is used for determining the Link State between two nodes. A hop matrix is used for determining the nexthop for every communication link required.

2. Initially, all the entries in the cost matrix are set to infinity (or a very high value).
3. For every pair of connected nodes, the respective entries in the cost matrix are set to the cost of the links between the nodes. In the hop matrix, the entry for each of these nodes is set to the connected node.
4. Copies of the original cost and hop matrices are maintained.
5. In each iteration of the cost matrix computation, it is attempted to insert another adjacent node in between every pair of nodes, the attempt succeeding if the new cost is lesser than the earlier cost to network between those nodes. In such cases, the hop matrix entry is updated to reflect the new nexthop for this communication link.
6. At the end of this exercise, the matrices will contain the best way to reach any pair of nodes in the network.
7. A copy of the cost matrix is made at this point.
8. 'Ln' denotes the maximum number of links possible for each node 'n'. In order to have at most 'L_n' connected neighbors for each node, the spanning tree is checked for the number of links to each node. If there are greater than 'L_n' links originating from node 'n', the least used links are removed from the original cost and hop matrices.
9. The computation of the spanning trees with the modified topology is done again using the original matrices with the removed links.

10. The new cost matrix generated is compared with the copy made in step 7, generated from the previous iteration. If the matrices are different, steps 7, 8 and 9 are run again.

This chapter provided a description of the routing algorithms used in the SBI Emulation System. The chapters up to this point have discussed the architecture and design of the emulation system. The next chapter provides test cases and results for validating the system.

6 Testing and Results

The SBI Emulation System design and implementation needs to be validated for accuracy and correct behavior. Various scenarios can be created and fed into the Emulation System and the behavior analyzed and verified for correctness. A number of scenarios have been created and used to test the emulation system and the routing algorithm. A few of the test cases are tabulated here and the behavior of the emulation system described.

6.1 Test Cases

Two separate test scenarios have been developed to test the SBI Emulation System.

- Nine Node Polar Satellite Scenario
- Twenty Five Node Constellation Scenario

The following sections describe the test cases in detail.

6.1.1 Nine Node Scenario

The Nine Node Scenario uses eight equally spaced satellites in polar orbit and a ground station facility residing at the North Pole. The satellites each have four interfaces and the facility has two interfaces. There are instruments on board two of the satellites. The scenario start time is set to 06:30:00 05/01/2002. This scenario was designed so that every satellite node has access to two of its nearest neighbor satellites on either side at all times. The following table provides details regarding the satellite propagation:

	Semi-major Axis (km)	Eccentricity	Inclination (degrees)	Argument of Perigee (degrees)	Right Ascension of Ascending Node (degrees)	Epoch Time	Orbital Period (minutes)
Rly1	9578	0	90	0	0	09/01/2001 01:00:00	90.32
Rly2	9578	0	90	0	0	09/01/2001 01:19:26	90.32
Rly3	9578	0	90	0	0	09/01/2001 01:38:52	90.32
Rly4	9578	0	90	0	0	09/01/2001 01:58:18	90.32
Rly5	9578	0	90	0	0	09/01/2001 02:17:44	90.32
Rly6	9578	0	90	0	0	09/01/2001 02:37:10	90.32
Rly7	9578	0	90	0	0	09/01/2001 02:56:36	90.32
Rly8	9578	0	90	0	0	09/01/2001 03:16:02	90.32

Table 6-1: Satellite Propagation – Nine Node Scenario

6.1.2 Twenty Five Node Scenario

The Twenty Five Node Scenario uses eight equally spaced satellites in three orbits and a facility residing at Goddard. The three orbits used for the satellites are:

- Eight satellites propagate equally spaced in an equatorial orbit
- Eight satellites propagate in an orbit at 60 degrees inclination with the equator.

- Eight satellites propagate in a second orbit also at 60 degrees inclination with the equator.

This particular topology was chosen to have maximum coverage over the Earth as possible with less than 30 satellites. Table 6-2 provides details of the satellite orbits. The RAAN and the Epoch Time vary from the base values with increments as shown in the table.

	Semi-major Axis (km)	**Eccentricity**	**Inclination (degrees)**	**Argument of Perigee (degrees)**	**Right Ascension of Ascending Node (degrees)**	**Epoch Time**	**Orbital Period (minutes)**
Orbit 1	7878	0	0	0	0	09/01/2001 00:00:00 + 00:14:30	115.98
Orbit 2	7878	0	60	0	162.46 + 3.63	09/01/2001 00:10:00 + 00:14:30	115.98
Orbit 3	7878	0	60	0	341.20 + 3.63	09/01/2001 00:05:00 + 00:14:30	115.98

Table 6-2: Satellite Propagation – Twenty Five Node Scenario

6.2 Results

This section presents the results for the tests conducted on the SBI Emulation System. A screenshot of the execution of the Emulation System is shown.

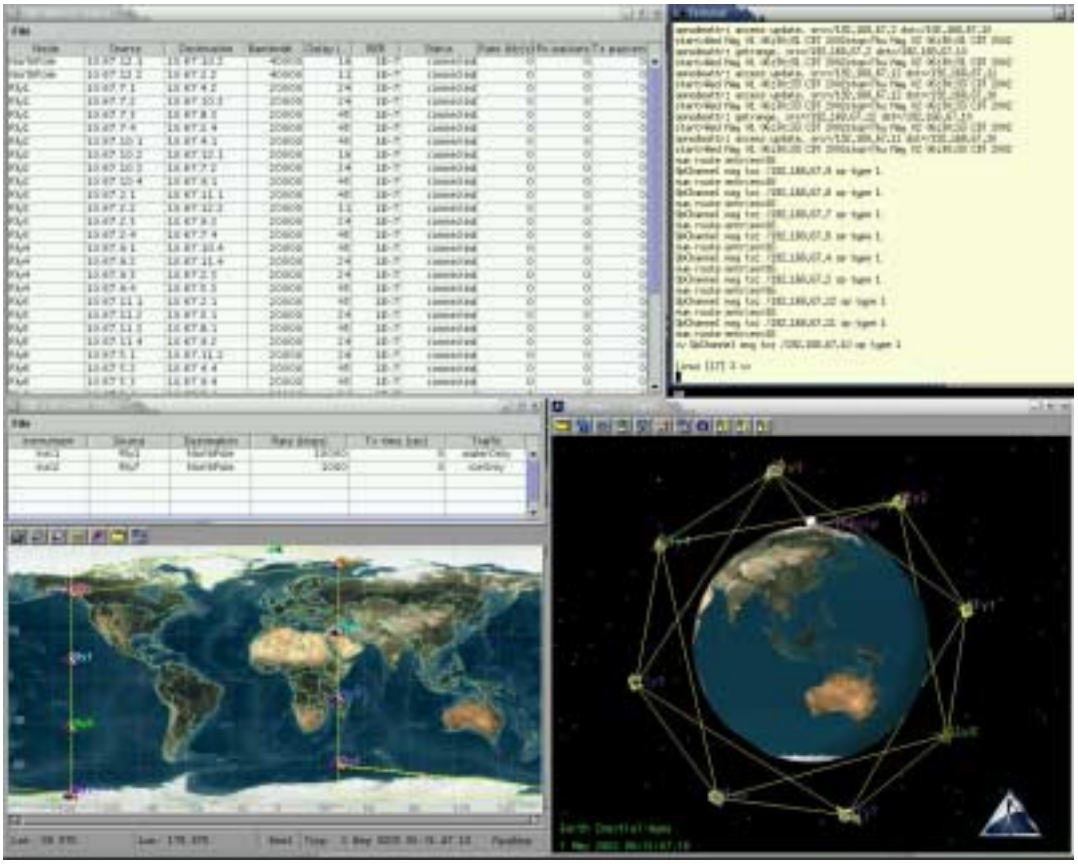

Figure 6-1: SBI Screen Shot

In anti-clockwise order from the top left-hand corner are the Network Status Window, the Instrument Status Window and the 2-Dimensional and 3-Dimensional STK Windows.

6.2.1 Nine-Node Scenario

This section details the results obtained from the Nine Node Scenario. A comparison is presented of the routing event times with different route update times used. This serves to illustrate the number of events that occur in a scenario. Routing Tables are provided from one of the nodes as an illustration of the route changes in the emulation. Images from the emulation are also presented to illustrate the route changes.

6.2.1.1 Route Update Time Comparison

The following tables show a comparison between a high and a low event round off time. The scenario described in Table 6-3 has a lowest time difference of 200 seconds between events and the scenario in Table 6-4 has a lowest time difference of 10 seconds between events. The times of the first few connections to the NorthPole facility are tabulated.

It is seen from the tables that with a 10-second round off, events occur as often as 100 seconds apart in this scenario. This problem becomes more acute as the number of nodes in the scenario increases.

In the Nine-Node Scenario, each satellite has a 41:49 minute window to the ground station facility. Figures 6-2, 6-3 and 6-4 showcase the route changes during the first few minutes of the emulation. Satellites Rly2 and Rly3 have access to the facility at the beginning. When the first event occurs, satellite Rly4 gets access in place of Rly2 and when the second event occurs, satellite Rly5 gets access in place of Rly3.

	Start Time	Event Time	Stop Time	Event Time
Rly1	08:13:01	08:13:10	08:54:50	08:54:40
Rly2	06:30:00 08:32:27	Initial 08:32:20	06:38:47 09:14:16	06:38:40 09:14:10
Rly3	06:30:00 08:51:53	Initial 08:52:00	06:58:13 09:33:42	06:58:00 09:33:30
Rly4	06:35:50	06:36:00	07:17:39	07:17:30
Rly5	06:55:16	06:55:20	07:37:05	07:37:00
Rly6	07:14:42	07:14:50	07:56:31	07:56:20
Rly7	07:34:08	07:34:10	08:15:57	08:14:50
Rly8	07:53:34	07:53:40	08:35:23	08:35:10

Table 6-3: Event Times with 10 second Minimum Interval

	Start Time	Event Time	Stop Time	Event Time
Rly1	08:13:01	08:13:20	08:54:50	08:53:20
Rly2	06:30:00 08:32:27	Initial 08:33:20	06:38:47 09:14:16	06:36:40 09:13:20
Rly3	06:30:00 08:51:53	Initial 08:53:20	06:58:13 09:33:42	06:56:40 09:33:20
Rly4	06:35:50	06:36:40	07:17:39	07:16:40
Rly5	06:55:16	06:56:40	07:37:05	07:36:40
Rly6	07:14:42	07:16:40	07:56:31	07:53:20
Rly7	07:34:08	07:36:40	08:15:57	08:13:20
Rly8	07:53:34	07:56:40	08:35:23	08:33:20

Table 6-4: Event Times with 200 second Minimum Interval

The route update time needs to be tweaked according to the scenario complexity to be able to generate the longest possible access times while reducing the prospect of excessive event occurrences.

Table 6-5 presents a comparison of access times between a facility and a satellite at various altitudes.

Altitude (km)	Access Time (min)
500	11
800	15
1000	17
1500	23
2000	28
3000	39
5000	62
10000	208

Table 6-5: Comparison of access times with satellite altitude

This shows that a satellite in LEO orbit (usually approx. 800 km) has about 15 min. of access time to a ground station facility. Choosing a very high route update time can cause small access intervals to get totally blacked out, whereas choosing very low route update times can result in excessive number of route changes in this interval. Therefore, a balance needs to be achieved where a few route changes occur and every satellite gets the maximum possible time of access to a ground station.

The results presented in the next two sections have been generated with tests using a 200-second route update interval.

6.2.1.2 Routes at NorthPole Facility

The node to IP address translation for the Nine Node Scenario tested is given in Table 6-5. Tables 6-6, 6-7 and 6-8 provide the routes for the NorthPole facility at the initial stage and after the first and second route updates respectively.

	Data Network IP Addresses	**Data Termination Interface**
NorthPole	10.67.12.1-2	10.67.199.12
Rly1	10.67.9.1-4	10.67.199.9
Rly2	10.67.8.1-4	10.67.199.8
Rly3	10.67.6.1-4	10.67.199.6
Rly4	10.67.10.1-4	10.67.199.10
Rly5	10.67.5.1-4	10.67.199.5
Rly6	10.67.11.1-4	10.67.199.11
Rly7	10.67.3.1-4	10.67.199.3
Rly8	10.67.7.1-4	10.67.199.7

Table 6-6: Node to Address Translation - Nine Node Scenario

Route Snapshot taken at 06:32:00

Destination	**Node**	**Gateway**	**Node**
10.67.6.2	Rly3	10.67.12.1	NorthPole
10.67.199.11	Rly6	10.67.6.2	Rly3
10.67.199.10	Rly4	10.67.6.2	Rly3
10.67.8.3	Rly2	10.67.12.2	NorthPole
10.67.199.9	Rly1	10.67.8.3	Rly2
10.67.10.2	Rly4	10.67.6.2	Rly3
10.67.199.8	Rly2	10.67.8.3	Rly2
10.67.199.7	Rly8	10.67.8.3	Rly2

10.67.9.4	Rly1	10.67.8.3	Rly2
10.67.199.6	Rly3	10.67.6.2	Rly3
10.67.5.4	Rly5	10.67.6.2	Rly3
10.67.199.5	Rly5	10.67.6.2	Rly3
10.67.3.4	Rly7	10.67.8.3	Rly2
10.67.7.2	Rly8	10.67.8.3	Rly2
10.67.199.3	Rly7	10.67.8.3	Rly2
10.67.11.3	Rly6	10.67.6.2	Rly3

Table 6-7: Routes at NorthPole facility - 1

Route Snapshot taken at 06:48:00

Destination	Node	Gateway	Node
10.67.8.1	Rly2	10.67.6.2	Rly3
10.67.10.1	Rly4	10.67.12.1	NorthPole
10.67.6.2	Rly3	10.67.12.2	NorthPole
10.67.199.11	Rly6	10.67.10.1	Rly4
10.67.199.10	Rly4	10.67.10.1	Rly4
10.67.199.9	Rly1	10.67.6.2	Rly3
10.67.199.8	Rly2	10.67.6.2	Rly3
10.67.199.7	Rly8	10.67.6.2	Rly3
10.67.199.6	Rly3	10.67.6.2	Rly3
10.67.5.4	Rly5	10.67.6.2	Rly3
10.67.7.4	Rly3	10.67.6.2	Rly3
10.67.199.5	Rly5	10.67.6.2	Rly3
10.67.3.4	Rly7	10.67.6.2	Rly3
10.67.11.1	Rly6	10.67.10.1	Rly4
10.67.199.3	Rly7	10.67.6.2	Rly3
10.67.9.2	Rly1	10.67.6.2	Rly3

Table 6-8: Routes at NorthPole facility - 2

Route Snapshot taken at 07:00:00

Destination	Node	Gateway	Node
10.67.10.4	Rly4	10.67.12.2	NorthPole
10.67.6.2	Rly3	10.67.10.4	Rly4
10.67.199.11	Rly6	10.67.5.2	Rly5
10.67.199.10	Rly4	10.67.10.4	Rly4
10.67.8.3	Rly2	10.67.10.4	Rly4
10.67.199.9	Rly1	10.67.10.4	Rly4
10.67.199.8	Rly2	10.67.10.4	Rly4
10.67.199.7	Rly8	10.67.5.2	Rly5
10.67.9.4	Rly1	10.67.10.4	Rly4
10.67.199.6	Rly3	10.67.10.4	Rly4
10.67.199.5	Rly5	10.67.5.2	Rly5
10.67.3.4	Rly7	10.67.5.2	Rly5
10.67.5.2	Rly5	10.67.12.1	NorthPole
10.67.199.3	Rly7	10.67.5.2	Rly5
10.67.11.3	Rly6	10.67.5.2	Rly5
10.67.7.1	Rly8	10.67.5.2	Rly5

Table 6-9: Routes at NorthPole facility – 3

From the tables, it is seen that the nodes Rly2 and Rly3 have access to the facility at the initial stage, and Rly4 replaces Rly2 after the first route update and Rly5 replaces Rly3 after the second route update. This is also seen in the Figures 6-2, 6-3 and 6-4, which are snapshots of the 3-D and 2-D windows of the SBI output.

6.2.1.3 Route Snapshots

The following figures provide the three-dimensional and a projection view of the routes created during the emulation run. Figure 6-2 shows the initial routes.

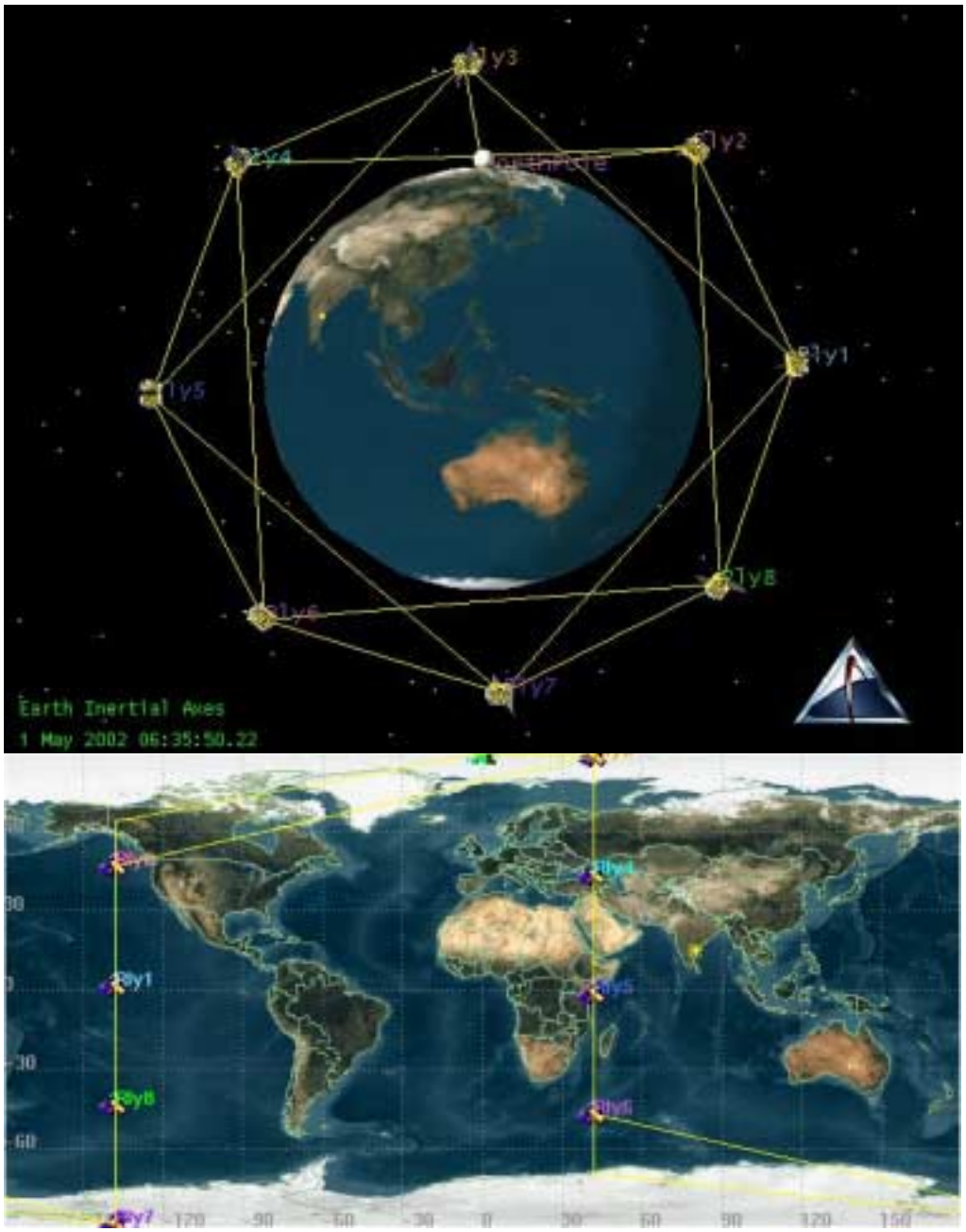

Figure 6-2: Initial Topology – Nine Node Scenario

The topology after the first route updates is shown in Figure 6-3. This event occurred at 06:36:40AM.

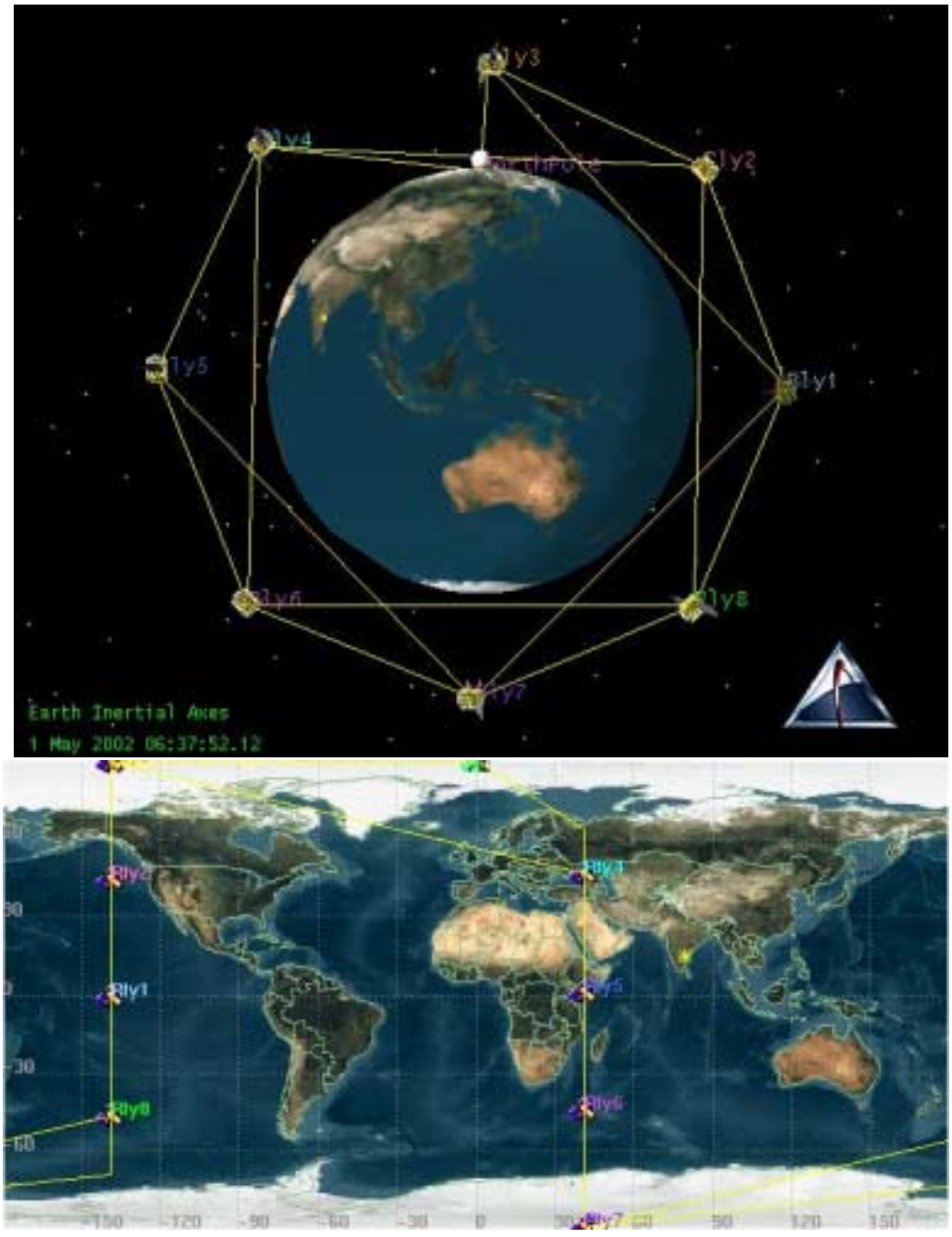

Figure 6-3: Topology after First Update

The topology after the second update is shown in Figure 6-4. This event occurred at 06:56:40AM.

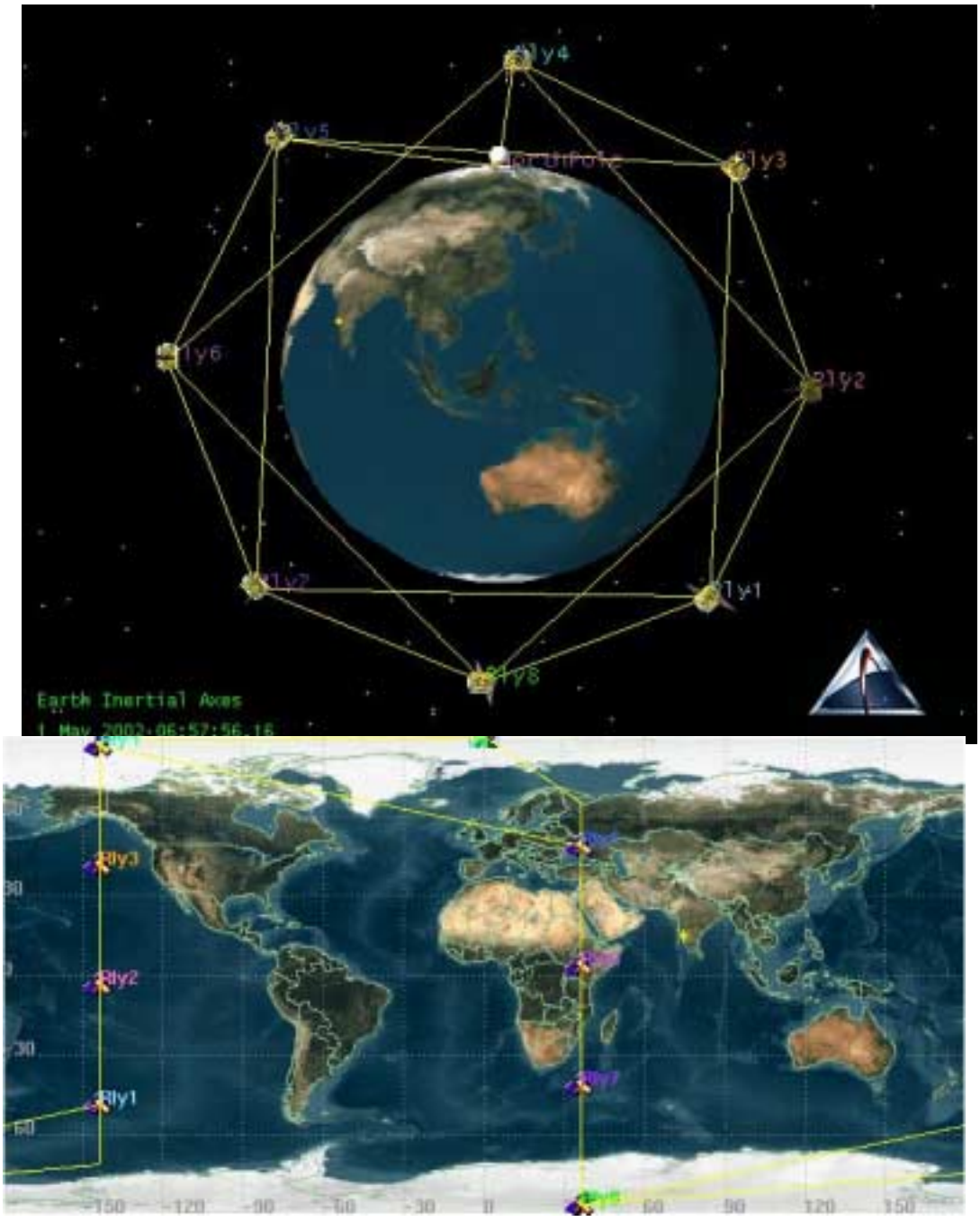

Figure 6-4: Topology after Second Update

6.2.2 Twenty Five-Node Scenario

The initial routing configuration is shown in Figure 6-5.

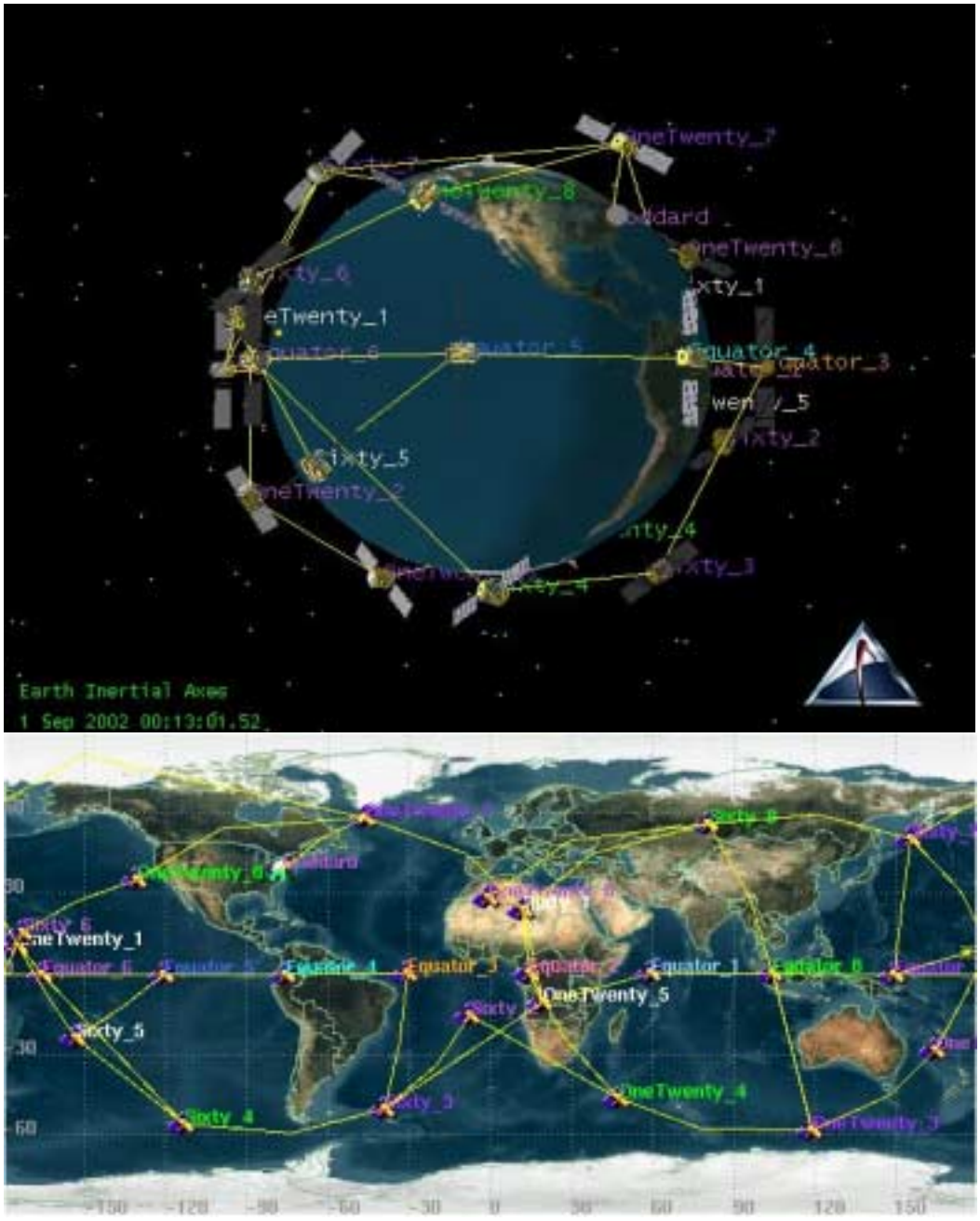

Figure 6-5: Initial Topology - Twenty Five Node Scenario

The topology after the first route updates is shown in Figure 6-6. This event occurred at 00:13:20AM.

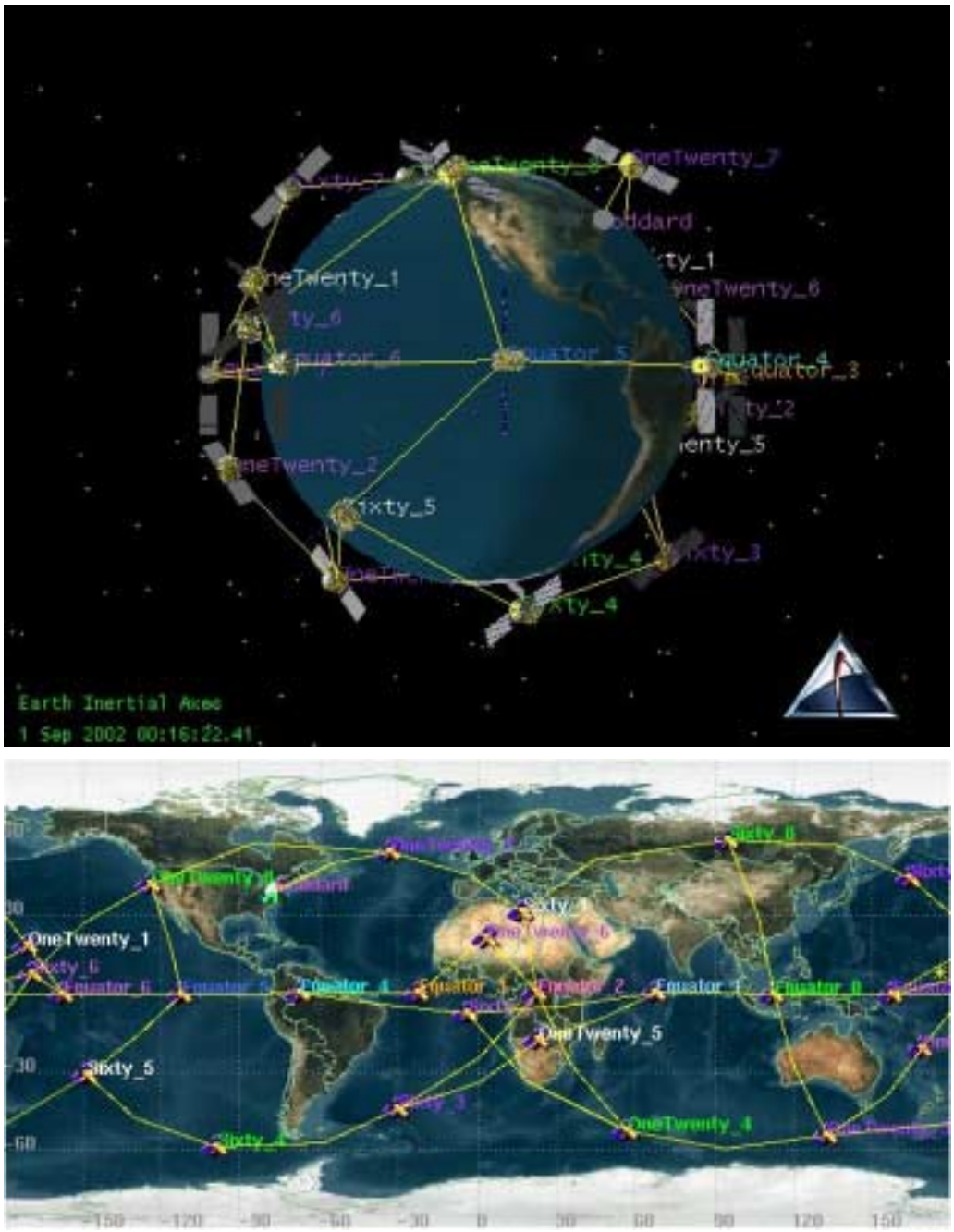

Figure 6-6: Topology after First Update

The topology after the third route updates is shown in Figure 6-7. This event occurred at 00:20:00AM.

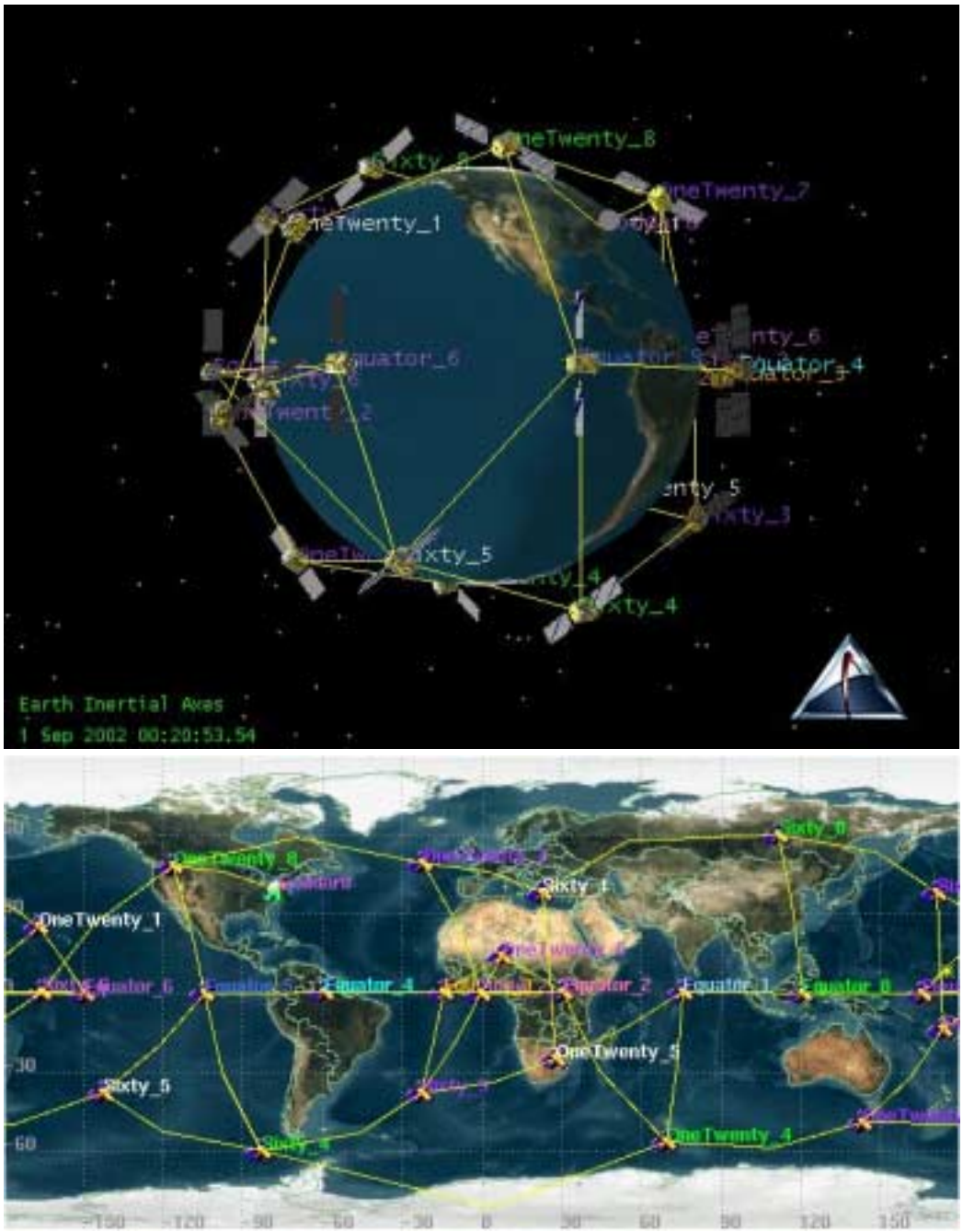

Figure 6-7: Topology after Second Update

The Figures 6-5, 6-6 and 6-7 provided a glimpse of the route changes that occur in the Twenty Five Node Scenario. In Figure 6-5 and 6-6, the node OneTwenty_7 has access to the ground station facility at Goddard. In the next route change in Figure 6-7, the node OneTwenty_8 has gained access to the facility. Various other changes in the topology can also be viewed in the figures.

This chapter provided the results obtained from testing the SBI Emulation System. Snapshots were presented of the emulation and route changes were illustrated. The next chapter offers concluding remarks for this thesis and presents suggestions for future work.

7 Conclusion

The design and implementation of the architecture for the Space Based Internet Emulation System has been described and evaluated in this document. This chapter presents the conclusions reached from the design and evaluation and the scope for future work.

7.1 Concluding Remarks

The architecture and design for a Space Based Internet Emulation System has been successfully implemented and evaluated. The design has been found to be physically feasible. The Emulation Manager forms a controlling and monitoring entity for the system. The centralized architecture for the routing and scheduling tasks has been presented, designed and implemented. Various testing scenarios have been proposed and employed to evaluate the working of the system. The emulation system has been found to be an effective means of evaluating the workings of a Space Based Internet in the real world.

7.2 Future Work

There are a number of methods for enhancing the working of the SBI Emulation System. A few of the improvements that can be envisaged for the emulation system are outlined here.

- Provision for higher number of nodes: In an emulation system, scalability is often a constraint due to the requirements of large amount of hardware. Provisioning

multiple SBI nodes on single hardware units can greatly increase the scalability of the system.

- Advanced routing algorithms: The topology algorithm presented in this thesis only serves to validate the system. Advanced algorithms taking into account bandwidth, multiple routes and other considerations need to be experimented with.
- Route Distribution: Route distribution in the current system is through the Management Network. A more efficient and realistic method of distribution would involve the Data Network. Steps need to be taken in order to implement this feature.
- Complex Scenarios: Complex scenarios need to be created and tested in order to fully validate the SBI Emulation System.

The work on the architecture for the Space Based Internet Emulation System is by no means done. What has been covered in this thesis is merely a beginning.

Appendix I Satellite Orbital Propagation

This Chapter provides an overview of the different terms and definitions involved in satellite orbital propagation [23][24][25]. The parameters required for propagating a satellite in its orbit are explained in some detail.

Satellite propagators are mathematical models of satellite orbits. A number of orbit propagators are currently in use like the two-body propagation model, the J2 perturbation and the MSGP4 models. The SBI Emulation System uses the simple Two-Body propagation model for SBI satellite propagation.

The two-body propagator takes into account only the gravitational pull on the satellite due to the Earth. This propagator defines the following parameters:

- Apogee: The point in the satellite's orbit where it is farthest from the Earth.
- Perigee: The point in the satellite's orbit where it is nearest to the Earth.
- Semi-major axis: This represents the altitude of the satellite from the center of the Earth. In the case of non-circular orbits, the semi-major axis is defined to be:

$$a = \frac{(R_{apogee} + R_{perigee})}{2}$$

  where R_{apogee} and $R_{perigee}$ represent the distance from the center of the Earth to the apogee and perigee points of the satellite respectively.
- Eccentricity: This is a value between 0 and 1, which represents the elliptical shape of the orbit. If the eccentricity is 0, the orbit is a circle.
- Inclination: The inclination of a satellite is given by the angle that the satellite orbit makes with the equator. The inclination varies between 0 and 180 degrees.

- Line of Nodes: The intersecting line between the orbital plane and the equatorial plane is called the line of nodes.
- Argument of Perigee: The angle between the line of nodes and the major axis of the orbital ellipse is called the Argument of Perigee.
- Ascending Node: The Line of Nodes intersects the Earth at two points. The point where the satellite crosses the Earth from south to north is called the Ascending Node.
- Vernal Equinox: The Vernal Equinox is the Ascending Node of the Sun caused by the Earth's orbit, i.e., the point on the Earth's equator where the Sun crosses from south to north. The Right Ascension angle at this point is defined to be zero.
- Right Ascension of Ascending Node (RAAN): The RAAN is used to orient the orbital plane in space along with the inclination parameter. The RAAN represents the angle measured at the center of the Earth between the vernal equinox and the ascending node for the satellite. In other words, the RAAN is the angle at the center of the Earth where the Sun crosses the Equator from south to north and where the satellite crosses the Equator from south to north.
- Epoch Time: The Epoch time is the time when the orbit of a satellite is specified.

Appendix II XML Schema

The SBI Emulation System uses the XML Schema [15][16] interface for validating the XML input file. The schema defines the valid entries in the input file. The schema used in the SBI Emulation System is reproduced here:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
<xs:element name="seqEntries">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Entry" minOccurs='1'
maxOccurs='unbounded'/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="Entry">
 <!-- Sequence of entries for describing the scenario and the
node types -->
 <xs:complexType>
  <xs:choice>
   <xs:element name="Scenario" type="ScenarioType"/>
   <xs:element name="Satellite" type="SatelliteType"
minOccurs='1' maxOccurs='unbounded'/>
   <xs:element name="Facility" type="FacilityType"
minOccurs='1' maxOccurs='unbounded'/>
  </xs:choice>
 </xs:complexType>
</xs:element>
 <xs:complexType name="ScenarioType">
 <!-- Scenario configuration -->
  <xs:sequence>
```

```
     <xs:element name="StartTime" type="xs:dateTime"/>
     <xs:element name="StopTime" type="xs:dateTime"/>
  </xs:sequence>
  <xs:attribute       name="Id"       type="xs:positiveInteger"
use="required"/>
  <xs:attribute name="Name" type="xs:string" use="required"/>
 </xs:complexType>

 <xs:simpleType name="IPAddressType">
 <!-- Format of an IP address for syntax checking -->
  <xs:restriction base="xs:string">
   <xs:pattern value="((([0-9]){1,3})\.){3})([0-9]){1,3}"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="DataTypeType">
 <!-- The types of data scheduling possible -->
   <xs:restriction base="xs:string">
    <xs:enumeration value="iceOnly"/>
    <xs:enumeration value="waterOnly"/>
    <xs:enumeration value="vegLandOnly"/>
    <xs:enumeration value="aridLandOnly"/>
    <xs:enumeration value="landWater"/>
    <xs:enumeration value="landIce"/>
    <xs:enumeration value="waterIce"/>
    <xs:enumeration value="landOnly"/>
    <xs:enumeration value="iceDay"/>
    <xs:enumeration value="waterDay"/>
    <xs:enumeration value="vegLandDay"/>
    <xs:enumeration value="aridLandDay"/>
    <xs:enumeration value="landWaterDay"/>
    <xs:enumeration value="landDay"/>
    <xs:enumeration value="SunriseSunset"/>
```

```
    <xs:enumeration value="allSurfaceDay"/>
    <xs:enumeration value="allSurface"/>
   </xs:restriction>
 </xs:simpleType>

 <xs:complexType name="InstrumentType">
 <!-- The syntax for an Instrument -->
  <xs:sequence>
   <!-- "Dest" can be an IP address or the name of a node. -->
   <xs:element name="Dest" type="xs:string"/>
   <xs:element name="DestPort" type="xs:positiveInteger"/>
   <xs:element name="DataType" type="DataTypeType"/>
   <xs:element name="DataRate" type="xs:unsignedInt"/>
  </xs:sequence>
  <xs:attribute       name="Id"       type="xs:positiveInteger"
use="required"/>
  <xs:attribute name="Name" type="xs:string" use="required"/>
 </xs:complexType>

 <xs:complexType name="InterfaceType">
 <!-- The syntax for an Interface on a node -->
  <xs:sequence>
   <xs:element name="DataRate" type="xs:unsignedInt"/>
  </xs:sequence>
  <xs:attribute       name="Id"       type="xs:positiveInteger"
use="required"/>
 </xs:complexType>
 <xs:complexType name="SatelliteType">
 <!-- The syntax for a Satellite node -->
 <!-- Contains the positional info, interface and instrument
info -->
  <xs:sequence>
   <xs:element name="OpNode" type="xs:boolean" minOccurs='0'/>
```

```
   <xs:element name="SemiMajorAxis" type="xs:float"/>
   <xs:element name="Eccentricity" type="xs:float"/>
   <xs:element name="Inclination" type="xs:float"/>
   <xs:element name="ArgOfPerigee" type="xs:float"/>
   <xs:element name="RAAN" type="xs:float"/>
   <xs:element name="Epoch" type="xs:dateTime"/>
   <xs:element name="OrbPeriod" type="xs:float"/>
   <xs:element      name="Interface"      type="InterfaceType"
minOccurs='1' maxOccurs='unbounded'/>
   <xs:element     name="Instrument"     type="InstrumentType"
minOccurs='0' maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute       name="Id"       type="xs:positiveInteger"
use="required"/>
  <xs:attribute name="Name" type="xs:string" use="required"/>
 </xs:complexType>

 <xs:complexType name="FacilityType">
 <!-- The syntax for a Facility node -->
 <!-- Contains the positional info and interface info -->
  <xs:sequence>
   <xs:element name="OpNode" type="xs:boolean" minOccurs='0'/>
   <xs:element name="Latitude" type="xs:float"/>
   <xs:element name="Longitude" type="xs:float"/>
   <xs:element      name="Interface"      type="InterfaceType"
minOccurs='1' maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute       name="Id"       type="xs:positiveInteger"
use="required"/>
  <xs:attribute name="Name" type="xs:string" use="required"/>
 </xs:complexType>
</xs:schema>
```

Bibliography

1. Paula Shaki Trimble, “NASA demos Space Net”, *Federal Computer Week*, May 1, 2000 http://www.fcw.com/fcw/articles/2000/0501/tec-nasanet-05-01-00.asp
2. Yurong Hu and Victor O.K. Li, “Satellite-Based Internet: A Tutorial”, *IEEE Communications Magazine*, vol. 39 no.3 pp. 154-162, March 2001.
3. NASA TDRSS Page, http://msl.jpl.nasa.gov/QuickLooks/tdrssQL.html
4. NASA Statement of Work, “Architecture for Space Based Internets”.
5. InterPlanetary Internet Special Interest Group HomePage, http://www.ipnsig.org/
6. Sandhya Rallapalli, “Emulation of a Space Based Internet Communication Link: Design and Implementation”, MS Thesis, University of Kansas, Lawrence, 2002.
7. Analytical Graphics Inc. Satellite ToolKit Online Help Pages, http://www.stk.com/resources/help/help/stk43/overview.htm
8. ITTC, University of Kansas, NetSpec HomePage, http://www.ittc.ku.edu/netspec/
9. Roelof Jonkman, “NetSpec: Philosophy, Design and Implementation”, MS Thesis, University of Kansas, Lawrence, 1998.
10. Beng Ong Lee, Victor S. Frost, Roelof Jonkman, “NetSpec 3.0 Source models for telnet, ftp, voice, video and WWW traffic”, Information and Telecommunication Technology Center, University of Kansas, ITTC-TR-10980-19 1997.
11. W3Schools XML Tutorial, http://www.w3schools.com/xml/
12. Apache Xerces XML Java Parser API Documentation, http://xml.apache.org/xerces-j/apiDocs/index.html
13. W3Schools XML DOM Tutorial, http://www.w3schools.com/dom/

14. W3Schools XML DTD Tutorial, http://www.w3schools.com/dtd/

15. W3Schools XML Schema Tutorial, http://www.w3schools.com/schema/

16. World Wide Web Consortium (W3C) Recommendation, “XML Schema Part 0: Primer”, May 2, 2001 http://www.w3.org/TR/xmlschema-0/

17. Karthik Thyagarajan, “Design and Implementation of Data Models and Instrument Scheduling of Satellites in a Space Based Internet Emulation System”, MS Thesis, University of Kansas, Lawrence, 2001.

18. Y. Hashimoto and B. Sarikaya, "Design of IP-based Routing in a LEO Satellite Network," *Proc. of Third ACM/IEEE International Workshop on Satellite-Based Information Services (WOSBIS '98)*, pp. 81--88, October 1998.

19. L. Wood, A. Clerget, I. Andrikopoulos, G. Pavlou, and W. Dabbous, "IP Routing Issues in Satellite Constellation Networks," *International Journal of Satellite Communications*, vol. 19, no. 1, pp. 69--92, January/February 2001.

20. M. Schwartz, “Computer-Communication Network Design and Analysis”, Prentice-Hall, Englewood Cliffs, NJ, 1977.

21. J.L. Kennington and R.V. Helgason, “Algorithms for Network Programming”, John Wiley and Sons, New York, NY, 1980.

22. N. Deo, “Graph Theory with Applications to Engineering and Computer Science”, Prentice-Hall, Englewood Cliffs, NJ, 1974.

23. Miguel Menendez’s article on Keplerian elements, http://www.telecable.es/personales/ea1bcu/kepsen.htm

24. Radio Amateur Satellite Corporation (AMSAT) Keplerian Elements Tutorial, http://www.amsat.org/amsat/keps/kepmodel.html
25. Dr. T. S. Kelso, “Computers and Satellites Columns”, *Satellite Times* http://www.celestrak.com/columns/
26. T. Henderson, Networking over Next-Generation Satellite Systems, PhD thesis, University of California, Berkeley, 1999.
27. Lloyd Wood, Internetworking with satellite constellations, PhD Thesis, University of Surrey, June 2001.
28. Lloyd Wood, George Pavlou and Barry Evans, “Effects on TCP of Routing Strategies in Satellite Constellations”, *IEEE Communications Magazine*, vol. 39 no.3 pp. 172-181, March 2001.
29. H. Uzunalioglu and W. Yen, "Managing Connection Handover in Satellite Networks", *Proc. of IEEE GLOBECOM '97*, November 1997.
30. E. Ekici, I.F. Akyildiz, and M.D. Bender, "A Distributed Routing Algorithm for Datagram Traffic in LEO Satellite Networks", *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 137--147, April 2001.
31. V.V. Gounder, R. Prakash and H. Abu-Amara, “Routing in LEO-Based Satellite Networks”, *Proceedings of IEEE Emerging Technologies Symposium on Wireless Communications and Systems*, Richardson, April 1999.
32. P. Wright, “Counting and constructing minimal spanning trees”, *Bulletin of the Institute of Combinatorics and its Applications*, vol.21, pp 65-76, 1997.

33. Java 2 Platform, Standard Edition, v 1.4.0 API Specification, http://java.sun.com/j2se/1.4/docs/api/

34. Bert Hubert, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, Paul B Schroeder, Jasper Spaans, "Linux Advanced Routing & Traffic Control", http://www.ds9a.nl/2.4Networking/

35. Glenn Herrin, "Linux IP Networking", Version 1 for Linux-2.2.14, May 31, 2000, http://www.kernelnewbies.org/documents/ipnetworking/linuxipnetworking.html

36. Y. Langsam, M. Augustein, A. Tenenbaum, "Data Structures Using C and C++", Prentice Hall Inc. Second Edition 1996.