Design and Implementation of Lightweight SNMP for Micro-Satellites Using Generative Programming

By

Huan Qi, B.Sc.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Master of Science Information and System Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

December 18, 2002

© Copyright 2002, Huan Qi
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
The undersigned hereby recommend to the Faculty of Graduate Studies and Research acceptance of the thesis,

**Design and Implementation of Lightweight SNMP for Micro-Satellites Using Generative Programming**

submitted by

**Huan Qi**

in partial fulfillment of the requirements for the degree of

Master of Science Information and System Science

_________________________

Dr. Frank Dehne
(Director, School of Computer Science)

_________________________

Dr. Michel Barbeau
(Thesis Supervisor)

Carleton University
December 18, 2002
Abstract

With the increased interest in micro-satellite networks, the promise of an economically viable extension of terrestrial network systems has become possible. These systems would provide connectivity to areas where existing terrestrial networks have previously been either infeasible or impractical. Micro-satellite networks impose many constraints that are seldom seen in terrestrial networks. A network management protocol suitable for micro-satellite networks remains largely unexplored.

This thesis presents a solution towards overcoming the difficulties and limitations experienced by the current standardized network management protocol – SNMP – with respect to micro-satellite network management or resource-constrained networks. In the presented implementation, the memory footprint, network overhead, processing time, and round trip time imposed by LW-SNMP has been significantly reduced, with respect to SNMPv1. Moreover, this solution has been implemented through the use of generative programming to achieve high configurability.
Acknowledgements

I wish to take this opportunity to thank all of those who have helped me in many different ways, although some of their names are not mentioned here, due to the limited space.

First and foremost, my grateful thanks go to Dr. Michel Barbeau, the supervisor of my thesis as well as my master program. His wisdom, guidance, advice and support, which could not have been more prompt whenever I needed it, have helped me throughout my Master program, from the choices of courses to the completion of this thesis.

Many thanks also go to the Department of Computer Science, Carleton University. Its well-designed Master program on information and system science studies, as well as its strong support, have allowed me to significantly extend my knowledge, and to well prepare myself for the subsequent pursuit of the subject matter of my thesis. I would especially like to thank Dr. Francis Bordeleau and Dr. Evangelos Kranakis for kindly attending my presentation and for their invaluable comments, and Ms. Linda Pfeiffer for her always warm and patient help.

During the process of my Master program, I have also received assistance and support from many other individuals. I wish to express my deep gratitude to our project group members, especially Song Zhang, Honghui Luo, Zheyin Li and Maoning Wang for their trust, cooperation, generous help, invaluable discussions and kindly encouragement. I also wish to acknowledge Lichun Zhu, Yufang Zhu, Shu Lin, Rui Mao, Tingjun Wen, Qing Huang, and Tao Wan for their in-depth and constructive comments on my thesis. I am also indebted to Miguel Vargas Martin and Vicky Bell for their kind help with the English review of my thesis.

My special thanks go to many of my friends: their understanding and mental support have made my Master journey a much warmer and pleasant one! I would especially like to thank Xiaoqiong Shu, Qing Xia, Jian Zhang, Dongqing Zhang, Chunfang Zheng, Hong Liu, and Vivian Shi.

Above all, exceptional and affectionate thanks go to my family: my husband Fan Zhang, my parents Ruilan Zhang and Tang Qi, my parents-in-law Yourong Wang and Xuezhong Zhang, my brother Wei Qi, my sister Aining Zhang and my brother-in-law Shabai Huang. It was the love, the understanding, the appreciation, the encouragement, and the concrete help from my FAMILY, which enabled me to accomplish this work. I feel indeed lucky that I have such a wonderful family!
Table of Contents

Abstract .................................................. iii
Acknowledgements ........................................ iv

CHAPTER 1: INTRODUCTION ............................................. 1
1.1 Motivation ................................................. 1
1.1.1 Constraints of Micro-satellites ...................... 1
1.1.2 Problems with Using SNMP in Micro-satellite Networks .. 1
1.1.3 Requirements of the Micro-satellite Network Management Protocol .. 5
1.2 Thesis Contribution .................................. 7
1.3 Thesis Outline ......................................... 8

CHAPTER 2: BACKGROUND .......................................... 10
2.1 SNMP ...................................................... 10
2.1.1 ASN.1, SMI, and BER ............................ 11
2.1.2 SNMPv1 .............................................. 15
2.1.2.1 Origin of SNMPv1 .............................. 15
2.1.2.2 SNMPv1 Architecture ......................... 17
2.1.2.3 SNMPv1 Data Representations .................. 18
2.1.2.4 SNMPv1 Operations .......................... 19
2.1.2.5 SNMPv1 Limitations ......................... 21
2.1.3 SNMPv2 .............................................. 23
2.1.3.1 Enhancements of SNMPv2 over SNMPv1 ......... 24
2.1.4 SNMPv3 .............................................. 25
2.1.4.1 Enhancements of SNMPv3 over SNMPv2 ........... 26
2.1.4.2 User-based Security Model .................... 26
2.1.4.3 View-based Access Control Model ............. 28
2.1.4.4 Characteristics of Using SNMPv3 .............. 29
2.2 Data Collection via MIB-II .......................... 30
2.3 CMIP .................................................................................................................. 33
2.4 Generative Programming for PIX .................................................................... 34
  2.4.1 Overview of Generative Programming ............................................. 35
  2.4.2 Protocol Implementation Framework for Linux (PIX) ................. 37
    2.4.2.1 PIX Domain Analysis ............................................................ 38
    2.4.2.2 PIX Domain Design ............................................................. 42
    2.4.2.3 PIX Domain Implementation ............................................. 44
    2.4.2.4 Configuration of a Network Management Protocol ............. 46
2.5 Satellite Network Management ..................................................................... 47
  2.5.1 Network Management in ATM LEO Satellite Networks .......... 47

CHAPTER 3: SPECIFICATIONS AND FEATURES OF LW-SNMP .............. 50
  3.1 LW-SNMP Services ....................................................................................... 50
    3.1.1 General Characteristics of LW-SNMP Services ....................... 50
    3.1.2 LW-SNMP Services ........................................................................ 53
  3.2 LW-SNMP Protocol Functions ................................................................... 54
  3.3 LW-SNMP Domain Feature Modeling ..................................................... 55
  3.4 LW-SNMP Domain Architecture Design .............................................. 61
  3.5 Implementations of LW-SNMP Domain Components ..................... 65
  3.6 LW-SNMP Domain Generator ................................................................. 68

CHAPTER 4: DETAILED DESIGN AND IMPLEMENTATION OF LW-BER,
  MD5-Authentication, AND Centralized-Service ....................................... 73
  4.1 LW-BER Feature .......................................................................................... 73
    4.1.1 Objectives ....................................................................................... 73
    4.1.2 Approach ......................................................................................... 74
    4.1.3 Algorithms Used in Numeric Representations ............................ 79
    4.1.4 Characteristics and Limitations .................................................... 81
  4.2 MD5-Authentication Feature ................................................................. 82
    4.2.1 Approach I: Key-Preappended-and-Appended MD5 ............ 83
    4.2.2 Approach II: HMAC-MD5 .............................................................. 83
CHAPTER 5: PERFORMANCE EVALUATION .......................................................... 93
  5.1 Introduction ......................................................................................... 94
  5.2 Testbed Network ................................................................................ 94
     5.2.1 Hardware Used in the Testbed Network ...................................... 94
     5.2.2 Software Used in the Testbed Network ...................................... 95
  5.3 Experiments, Results, and Discussions ............................................. 95
     5.3.1 How Large Are the Memory Footprints of the LW-SNMP and UCD-SNMP? ................................................................. 96
     5.3.2 How Much Network Capacity Is Consumed by the LW-SNMP and UCD-SNMP Operations? .................................................... 96
     5.3.3 How Much Processing Time Is Consumed by the LW-SNMP and UCD-SNMP Operations? .................................................... 103
     5.3.4 How Much Time Is Consumed by the LW-SNMP and UCD-SNMP Operations under Low Bandwidth Point to Point Links? .......... 106

CHAPTER 6: CONCLUSIONS ....................................................................... 111
  6.1 Summary ............................................................................................ 111
  6.2 Future Work ....................................................................................... 113

BIBLIOGRAPHY ......................................................................................... 115
List of Tables

Table 3-1  Availability of management services in network management protocols........53
Table 5-1  Memory footprint of network management protocols...............................96
Table 5-2  IP packet sizes in bytes for request operations with a string value...........97
Table 5-3  IP packet sizes in bytes for request operations with an integer value.........100
# List of Figures

| Figure 2-1 | Data representation conversion alternatives | 12 |
| Figure 2-2 | ASN.1 example | 13 |
| Figure 2-3 | SNMP architecture | 17 |
| Figure 2-4 | SNMPv1 non-trap message format | 19 |
| Figure 2-5 | SNMPv3 message format | 30 |
| Figure 2-6 | MIB-II group structure | 31 |
| Figure 2-7 | Scope of MIB-II in a protocol stack | 32 |
| Figure 2-8 | Feature diagram of PIX | 41 |
| Figure 2-9 | GenVoca Grammar for PIX | 43 |

| Figure 3-1 | Blocked and non-blocked LW-SNMP service primitives | 52 |
| Figure 3-2 | LW-SNMP error indicator | 53 |
| Figure 3-3 | Activity diagram symbols | 54 |
| Figure 3-4 | Activity diagram showing LW-SNMP protocol functions | 55 |
| Figure 3-5 | Feature diagram of LW-SNMP domain | 57 |
| Figure 3-6 | Component categories for LW-SNMP domain | 62 |
| Figure 3-7 | Component categories dependencies | 63 |
| Figure 3-8 | Component categories layered architecture | 64 |
| Figure 3-9 | GenVoca grammar for LW-SNMP domain | 65 |
| Figure 3-10 | Segment of Encryption class declaration | 66 |
| Figure 3-11 | Segment of MD5 authentication class declaration | 67 |
| Figure 3-12 | Segment of Centralized-Service class declaration | 68 |

| Figure 4-1 | LW-Msg message format | 74 |
| Figure 4-2 | SNMPv1 message in serialized form | 76 |
| Figure 4-3 | Encoding integers | 80 |
| Figure 4-4 | Decoding integers | 81 |
| Figure 4-5 | Structure of using Key-Preappended-and-Appended MD5 | 83 |
| Figure 4-6 | Activity diagram of reception of a message | 88 |
| Figure 4-7 | Activity diagram of delivery of a message | 90 |

| Figure 5-1 | Ratios over various number of string-typed variable bindings | 99 |
| Figure 5-2 | Ratios over various number of integer-typed variable bindings | 101 |
| Figure 5-3 | Processing time used by network management protocols | 104 |
| Figure 5-4 | Response time of a request with one variables over different bandwidths | 108 |
| Figure 5-5 | Response time of a request with 10 variables over different bandwidths | 108 |
| Figure 5-6 | Response time of a request with 50 variables over different bandwidths | 109 |
# List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentX</td>
<td>Agent eXtensibility</td>
</tr>
<tr>
<td>ASN.1</td>
<td>Abstract Syntax Notation One</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>BER</td>
<td>Basic Encoding Rule</td>
</tr>
<tr>
<td>CMIP</td>
<td>Common Management Information Protocol</td>
</tr>
<tr>
<td>DER</td>
<td>Distinguished Encoding Rule</td>
</tr>
<tr>
<td>DES-CBC</td>
<td>Data Encryption Standard Cipher Block Chaining</td>
</tr>
<tr>
<td>DPI</td>
<td>SNMP Distributed Programming Interface</td>
</tr>
<tr>
<td>EPROM</td>
<td>Erasable Programmable Read Only Memory</td>
</tr>
<tr>
<td>GP</td>
<td>Generative Programming</td>
</tr>
<tr>
<td>HEMS</td>
<td>High-level Entity Management Systems</td>
</tr>
<tr>
<td>HMAC</td>
<td>Hash Message Access Control</td>
</tr>
<tr>
<td>HMP</td>
<td>Host Management Protocol</td>
</tr>
<tr>
<td>IAB</td>
<td>Internet Activities Board</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>LEO</td>
<td>Low Earth Orbit</td>
</tr>
<tr>
<td>LWER</td>
<td>Lightweight Encoding Rule</td>
</tr>
<tr>
<td>LW-BER</td>
<td>Lightweight Basic Encoding Rule</td>
</tr>
<tr>
<td>LW-SNMP</td>
<td>Lightweight Simple Network Management Protocol</td>
</tr>
<tr>
<td>MD5</td>
<td>Message Digest Five</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base</td>
</tr>
<tr>
<td>NCC</td>
<td>Network Control Center</td>
</tr>
<tr>
<td>NMP</td>
<td>Network Management Protocol</td>
</tr>
<tr>
<td>NMS</td>
<td>Network Management Station</td>
</tr>
<tr>
<td>OBC</td>
<td>On-Board Computer</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PING</td>
<td>Packet Internet Groper</td>
</tr>
<tr>
<td>PIX</td>
<td>Protocol Implementation Framework for Linux</td>
</tr>
<tr>
<td>PPP</td>
<td>Point-to-Point Protocol</td>
</tr>
<tr>
<td>PVP</td>
<td>Permanent Virtual Path</td>
</tr>
<tr>
<td>SGMP</td>
<td>Simple Gateway Monitoring Protocol</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td>SMI</td>
<td>Structure of Management Information</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLV</td>
<td>Type, Length, and Value</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>UI</td>
<td>Uniform Interface</td>
</tr>
<tr>
<td>USM</td>
<td>User-based Security Model</td>
</tr>
<tr>
<td>VACM</td>
<td>View-based Access Control Model</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter aims at establishing the context in which this thesis has been written through a brief introduction of the motivation and objectives for the scientific investigation that culminated in this thesis. The main contributions of this work are then summarized, followed by an overview of the forthcoming chapters.

1.1 Motivation

1.1.1 Constraints of Micro-satellites

Advances in satellite technology are allowing researchers to develop nano- and micro-scale devices rapidly and inexpensively in space. Micro-satellites [1] are devices with the following characteristics: short round-trip propagation delay (10 – 40 ms) [2]; short development time on smaller budgets; the ease to follow advanced technology; and the capability to fulfill many functions, such as telecommunications, earth observations, scientific researches, and technology demonstrations [3]. The micro-satellite of interest in this research imposes some constraints on size, mass, on-board computer (OBC), power budget, communication data rate, and contact time. The following description of the constraints is based on [4]. First of all, it is 10 cm × 10 cm × 10 cm in size and 1 kg in mass, thus the payload needs to be carefully budgeted with the consideration of volume and mass in mind. The OBC, based on a low-power ARM7 core, is a single board
computer operating at up to 40 MHz with 128 KB erasable programmable read only memory (EPROM); 2 MB static random access memory (SRAM); and 32 MB of external FLASH. With the assumption of 97 minutes orbit with 60 minutes of sunlight, the power budget for the OBC is 0.61 Wh. The modem of the OBC supports data rates of 1200, 2400, and 4800 bits per second (bps). On average, there are four contact periods a day, with a total daily time of between 28 and 29 minutes. The potential operating system of the micro-satellite is an embedded configurable operating system (eCos) [37]. eCos, a real-time operating system from RedHat, is targeted toward dedicated. The operating system of the ground station is x386/LINUX. The designed micro-satellite is at an altitude of 650 km above the earth’s surface, which limits the size of the converge footprint. Therefore, a number of satellites are needed to fulfil the aforementioned functions. These satellites form a network, in which each node could be a source, a destination or a router. Traffic may flow between space-based sources and space-based destinations. Consequently, the space-based networks may experience congestion, corruption and outage. Providing an overall monitor and control of such networks is frequently necessary because it assists operators in managing and planning the satellite communication networks more efficiently, making optimal use of available resources. Otherwise, it would be impossible to collect a piece of simple data about a node, such as the satellite system’s up time, and it would be impossible to change a value about the network information, such as a next hop for an entry of a routing table.

1.1.2 Problems with Using SNMP in Micro-satellite Networks
Simple Network Management Protocol (SNMP) is a *de facto* management standard for TCP/IP based networks. It is used to monitor and control the behavior of a network at any moment and to guarantee a properly operated network, with respect to performance, scalability and availability. SNMP version 1 (SNMPv1) [5] is the original network management protocol. It is still primarily used for monitoring as it contains security weaknesses. An example of such a weakness is that a plain text community name as a password is transferred between a sender and a receiver. SNMP version 2 (SNMPv2) [6] is an update of protocol operations of SNMPv1 and still uses the weak security mechanism from SNMPv1, while SNMP version 3 (SNMPv3) [7] employs a User-based Security Model (USM) [8] to provide cryptographic services. The USM currently uses either Message Digest Five (MD5) [9] or Secure Hash Algorithm (SHA) [10] keyed message digests to ensure message authentication and integrity. It also uses Data Encryption Standard – Cipher Block Chaining (DES-CBC) to ensure message privacy. These features are used to provide three distinct levels of security: no authentication with no privacy; authentication with no privacy; and authentication with privacy. The USM provides, for remotely configuring users and their keys, operations to manipulate objects in the USM Management Information Base (MIB). Furthermore, SNMPv3 employs a View-based Access Control Model (VACM) [11] to establish user access privileges. When making access control decisions, SNMPv3 cleanly and separately considers the user originating the message, the level of security applied to the message, the MIB view addressed by the message, and the type of operation requested by the message. The information related to VACM can be configured remotely by setting the appropriate objects in the VACM MIB.
SNMPv1, v2, and v3 all use the Basic Encoding Rule (BER) [12] associated with Abstract Syntax Notation (ASN.1) [13] to encode each field of a message into a (tag, length, value) triplet, whose length is value dependent. ASN.1 is a formal language describing structured information without regard for machine-oriented structures and restrictions. It is used to define the MIB and the protocol message format of SNMP.

Adopting SNMP to manage micro-satellite networks is inappropriate for the following reasons:

- The transfer of a plain text password to a receiving party in SNMPv1 and SNMPv2

SNMPv1 and SNMPv2 are mostly deprived of proper security [14,15]. The security mechanism in SNMPv1 and SNMPv2 is used to transfer a clear text password to a receiving party. In this case, by simply capturing information in the sky, a hostile process can learn the password and overcome the space-based network.

- The large memory footprint, large processing overhead and large network overhead of SNMPv3

Implementing the aforementioned SNMPv3's new security features requires a new SNMPv3 message format that is considerably longer than the SNMPv1 message. The network traffic generated by SNMPv3 regarding authentication is approximately 70% more compared with SNMPv1, in terms of retrieving one MIB variable. The processing time of SNMPv3 with regard to authentication is also increased by approximately a factor
of two with respect to SNMPv1. Both are unsuitable for bandwidth constraint networks. Moreover, an SNMPv3 protocol itself takes approximately 1.02 MB of memory footprint, however, an SNMPv3 daemon, a background server process acting as an SNMPv3 agent, takes approximately 1.56 MB of memory footprint. The micro-satellite has only 2 MB RAM, and if we are running an SNMPv3 daemon, there would be little memory remaining for scientific payload.

- **Large encoding overhead imposed by BER**

According to [16,17], BER imposes a significant number of overhead bits, such as the tag, the value, and the length octets. The tag and length octets are frequently not necessary because either both the sending and receiving parties are aware of the original abstract syntax, or they can be inferred from the abstract syntax.

### 1.1.3 Requirements of the Micro-satellite Network Management Protocol

Based on the understanding of the context of micro-satellite networks, and the problems of current SNMP standards applied to micro-satellite networks, a lightweight SNMP (LW-SNMP) should be designed to meet the following requirements:

- **Small footprint**

The LW-SNMP should provide the limited functionality necessary for managing simple and small satellites with a relatively small footprint in memory, due to the fact that only 2 MB RAM is available on the OBC.
- **Low network overhead**

The management operations consume the link bandwidth and the portion of the bandwidth, or the number of bytes used for the network management, which is thus unavailable for the transport of user data. LW-SNMP should by all means reduce the amount of traffic generated by the management and managed applications, since the maximum bandwidth available for ground to space link is 4800 bps.

- **Low processing overhead**

The processing overhead imposed by the management applications should be minimized since the OBC operates on 40 MHz.

- **Authorized management operation and data integrity concern**

Communications between an earth management station and space micro-satellites are through an insecure channel. Therefore, a method is required in which a management message set from the manager can be validated as authentic or unmodified by an agent. It is particularly necessary in the case where the manager alters the value stored in the agent.

- **Configurability**

The micro-satellites are always dedicated to one mission [3], e.g. communication, earth observation, and technology demonstration. In order to reduce the mission-specific cost, the LW-SNMP should be designed as a generic model to manage a family of different
mission-specific micro-satellites. Then users can configure a desired LW-SNMP according to their specifications.

1.2 Thesis Contribution

This research carried out in this thesis concentrates on how to reduce the network overhead and processing time imposed by the network management protocol with regard to security. It is an attempt to overcome the difficulties experienced by the current network management protocols, by using a generative programming approach to achieve software configurability. There are two main contributions of this research:

- Design LW-SNMP for managing micro-satellite networks

To meet the challenges set out in Chapter 1.1, the LW-SNMP has been designed using the lightweight basic encoding rule (LW-BER) to guarantee the uniform data transfer syntax between the different computer architectures of the ground station to micro-satellites and to reduce the ASN.1 encoding overhead. In addition, the LW-SNMP has been designed to eliminate unnecessary fields of the SNMPv1 message so as to reduce management protocol overhead. Low bandwidth is typically available in micro-satellite network environments, and the reduction of unnecessary network overhead is a major concern here. Compared with SNMPv1, the LW-SNMP has reduced network overhead and processing time. Furthermore, two alternative approaches for authentication mechanisms based on MD5 have been applied to LW-SNMP to ensure data origin authentication and data integrity. Finally, the memory footprint of a LW-SNMP with
authentication is 686 KB without including MIB library.

- **Implementation of LW-SNMP using Generative Programming**

  Generative Programming (GP) [18] is a programming discipline that promotes configurability by means of parameterization. By using the GP method, the LW-SNMP makes ordinary software components more configurable. These components constructed for LW-SNMP are able to adjust their behavior according to externally defined parameters. Therefore, the manager application can configure a desired lightweight network management protocol according to a particular set of encoding rules, security protocols, privacy protocols and network management architectures. There are potentially 60 choices for the LW-SNMP to be configured in order to yield the different lightweight management protocols. Currently, three different configurations of the LW-SNMP are available.

### 1.3 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 delves deeper into the background material related to this thesis research, and surveys the related work that provides the foundation for the research presented herein.

Chapter 3 describes the specification of the LW-SNMP in terms of services and protocol operations. The high-level feature modeling, and interfaces of components forming the
LW-SNMP domain, is carried out with the aim of corroborating the GP approach and its configuration tools.

Chapter 4 describes in detail the designs and implementations of LW-BER, MD5-Authentication and Centralized-Service feature values of the LW-SNMP domain.

Chapter 5 presents a comprehensive examination and evaluation of the LW-SNMP. The testbeds and experiment results are discussed in terms of memory footprint, network overhead, processing time, and round trip time.

Chapter 6 concludes this thesis. It first summaries the achievements and then suggests the perspectives for further development and deployment of the ideas proposed in this thesis research.
Chapter 2

Background Literature

In this chapter, SNMPv1, SNMPv2, and SNMPv3, which are deployed by modern terrestrial networks, are introduced and the differences among these three versions are emphasized. MIBII and its application for satellite communications are then presented. Common Management Information Protocol (CMIP), an Open systems Interconnection (OSI)-based network management standard, is described in brief. Subsequently, the software design method, generative programming, which is able to guide the construction of a configurable lightweight network management protocol, is illustrated with an example. Finally, issues concerning satellite network management, particularly Low Earth Orbit (LEO) satellite network management, are addressed in an attempt to establish the current state-of-the-art in this field.

2.1 SNMP

SNMP is today's dominant network management protocol on the Internet. However, as deployed today, the original version of SNMP (SNMPv1), which transfers a community name as plain text, still falls short of security mechanisms. The community-based SNMP version 2c (SNMPv2) is an experimental SNMP framework that supplements SNMPv1, while SNMP version 3 (SNMPv3) is an extensive SNMP framework that supplements the SNMPv2 framework by supporting a new SNMP message format, a security mechanism for message and access control. Before we introduce network management protocol, two
related concepts, ASN.1 and BER, must be addressed.

2.1.1 ASN.1 and BER

ASN.1, an International Organization for Standardization (ISO)-originated standard, provides a standardized method of defining the abstract syntax used by applications in a way that was independent of local data representation. This is important since different computer architectures use different internal representation, e.g. big endian for Sun SPARC and Motorola processors, and little endian for Intel and DEC Alpha processors. If an SNMP agent is about to send a response message containing an integer count of the number of received UDP datagrams, how should the representation of the integer value be sent to the SNMP manager entity – in big endian or little endian order? Generally, two options are available: explicit format conversion and common format conversion [31]. These two options are characterized in Figure 2-1.
Figure 2-1 Data representation conversion alternatives.

The explicit format conversion, the advantage of which is that it converts data only once, requires each end system to support conversion to every different machine format. The common format conversion uses a standardized transfer format and performs conversion to and from the local machine formats. The advantage in this is that it requires only one type of conversion per end system and the end system does not need to have the knowledge of the data representation used by the remote system. However, a disadvantage of the common format conversion is that two conversions are always applied to the data, even if two end systems use the same local data representation [31].

ASN.1 that deploys the common format conversion is a formal language. The grammars of this language are defined in [13]. The notation of ASN.1 includes provisions for specifying various primitive data types, e.g. integer values, boolean values, bit strings,
octet strings, real values, and enumerated types. It also provides the syntax for defining constructed types that are useful type definitions derived from the primitive data types. An example of ASN.1 is shown in Figure 2-2. Here, "::=" is an assignment operator in ASN.1. The example defines three types: Object is a structure that contains two members, namely an ObjectClass (an ASN.1 type defined below) and an ObjectInstance. The ObjectClass is a union with two members, namely globalForm of type OBJECT IDENTIFIER and nonSpecificForm of type INTEGER. The ObjectInstance is also a union with three members, namely, distinguishedName, nonSpecificForm, and enumerateForm.

```
Object ::= SEQUENCE {
  objectClass ObjectClass
  objectInstance ObjectInstance }

ObjectClass ::= CHOICE {
  globalForm [0] IMPLICIT OBJECT IDENTIFIER,
  nonSpecificForm [1] IMPLICIT INTEGER }

ObjectInstance ::= CHOICE {
  distinguishedName [2] IMPLICIT DistinguishedName,
  nonSpecificForm [3] IMPLICIT OCTET STRING,
  enumerateForm [4] IMPLICIT INTEGER }
```

**Figure 2-2 ASN.1 example.**

Once data structures can be described in a machine-independent fashion, there must be some way of transmitting those data structures unambiguously over a network. ASN.1 also provides BER that specifies how instances of objects defined using the ASN.1 data description language are to be transferred over the network. BER adopts a so-called TLV (Type or Tag, Length, Value) mechanism to encode and decode primitive and constructed
data for transmission. For each data in the abstract syntax to be transferred, a data type, a
length of an encoded data, and an actual value in a serialized form are transferred. With
this convention, the received data is essentially self-identifying.

There are three ways to encode an ASN.1 value under BER, the choice of which depends
on the type of the value and whether the length of the value is known. The three choices
are primitive definite-length encoding, constructed definite-length encoding, and
constructed indefinite-length encoding [12]. Simple non-string types employ the
primitive definite-length encoding method; structured types employ either of the
constructed encoding methods; and simple string types employ any of the encoding
methods, depending on whether the length of the string is known. BER has three or four
parts:

**Identifier octets:** these identify the class and tag number of the ASN.1 value, and
indicate whether the method is primitive or constructed.

**Length octets:** for the definite-length encoding methods, these give the number of
content octets. For the constructed indefinite-length encoding method, these indicate that
the length is indefinite.

**Content octets:** for the primitive definite-length encoding method, these give a concrete
representation of the value. For the constructed encoding methods, these give the
concatenation of the BER encoding of the components of the value.

**End-of-contents octets:** for the constructed indefinite-length encoding method, these
denote the end of the contents. For the other methods, these are absent.
Based on the understanding of ASN.1 and BER, three versions of SNMP will be illustrated below.

2.1.2 SNMPv1

Simple Network Management Protocol (SNMP) is designed to provide management capability for TCP/IP-based networks. It defines a format for representing management information and a framework for organizing distributed systems into managed systems and managed agents. A number of specific database structures, known as management information bases (MIBs), have been defined as part of the SNMP suite. However, the details of MIB are outside the scope of this thesis, and they are not discussed here. The origin, architecture, data representations, operations, and limitations of SNMPv1 will be addressed in detail.

2.1.2.1 Origin of SNMPv1

Through the late 1970s, no management protocols such as SNMP existed. The one tool that was effectively used for management was the Internet Control Message Protocol (ICMP). ICMP, which is available on all devices that support Internet Protocol (IP), provides a means for transferring control messages from routers and other hosts to a host, then it can provide feedback about problems in the environment [14]. Packet Internet Groper (PING) is the most notable tool which uses ICMP, along with IP header options, such as the interval between a request and the number of times required to send a request,
to perform simple but powerful management functions [15]. However, in the late 1980s, when the growth of the Internet became exponential, accompanied by a growth in complexity, it was no longer possible to rely on the aforementioned simple tools to solve the management problems. The starting point for providing specific network management tools was the Simple Gateway Monitoring Protocol (SGMP) issued in November 1987. The SNMP derived from the SGMP that was used to monitor and configure the gateway remotely. Being a successor of SGMP, SNMP extended its scope and included the management of end systems [15]. SNMP was developed by the Internet Engineering Task Force (IETF) in 1988. IETF is a subsidiary group of the Internet Activities Board (IAB) and provides standard and simplified management of TCP/IP-based internetworking. At the same time two other approaches emerged:

- **High-Level Entity Management Systems (HEMS):** This was a generalization of Host Management Protocol (HMP), which was the first network management protocol used on the Internet.

- **CMIP over TCP/IP:** This was the Common Management Information Protocol under development of the ISO.

In early 1988, the IAB approved the further development of SNMP as a short-term solution and CMOT as the long-term solution [15]. In 1990, the original Internet-standard Network Management Framework was defined in [5], which is called "A Simple Network Management Protocol".
2.1.2.2 SNMPv1 Architecture

SNMPv1 is a simple message-based request and response protocol that typically runs as an application above the User Datagram Protocol (UDP), which is a connectionless protocol. SNMPv1 limits itself to the UDP, which gives the implementers the ability and responsibility to manage lost packets and to perform necessary retransmissions.

![SNMP Architecture Diagram](image)

**Figure 2-3 SNMP architecture.**

Figure 2-3 presents the protocol context of the SNMPv1 architecture. The data path between the manager application process and the agent application process passes through four layers: UDP, IP, Data Link, and Physical Link on the manager side, and passes through the same layers in reverse on the agent side on the agent side. The SNMPv1 architecture includes:
• **Network Management Station (NMS):** this is where the network management applications reside.

• **SNMP Management Agent:** provides information contained in a database to manage applications and accepts control information.

• **Management Information Base (MIB):** defines the information that can be collected and controlled by the management application.

• **Network Management Protocol:** used by the network management station and the agents to exchange management information.

### 2.1.2.3 SNMPv1 Data Representations

The formats of the information exchanged between a manager and an agent must be the same for any implementation. It must be independent on the hardware and strong-typed notations [14]. In order to achieve a uniform and unambiguous representation of the information delivered over the network, SNMP uses a subset of the ASN.1 known as SMI (Structure of Management Information) [14] as its data presentations. The SMI is not only used to define the formats of the messages exchanged by the management protocol but also used to define the managed objects.

The SNMPv1 message consists of a version identifier, a community name and a Protocol Data Unit (PDU). The community name determines a set of the MIB information available to the manager, thus enabling different administrative views of the agent's information, e.g. read-only or read-write access.
There are two distinct types of PDUs in SNMPv1, a trap and a non-trap PDU. The trap PDU contains a type, an enterprise, an agent-addr, a generic-trap, a specific-trap, a time stamp, and variable bindings. The type indicates that this is a trap PDU. The enterprise is a type of object-generating trap. The agent-addr is an address of the object-generating type. The generic-trap is an integer value representing one of the seven generic traps defined in [5]. The specific-trap represents a trap defined for a specific enterprise. The time stamp indicates the time that the trap was generated. The variable bindings provide additional information relating to the trap. The non-trap PDU shown in Figure 2-4 contains a type, a request-id, an error-status, an error-index, and variable bindings. The type indicates the PUD type. The request-id is a number used to identify a request sent from a manager, which enables the agent to match the Get-Response PDU to one of the Get-Request, Get-Next-Request or Set-Request PDU. The error-status is an integer that represents the status of the message. The error-index is an integer indicating which variable binding has an error. The variable bindings represent the instances of the managed objects that are operated on by the management commands.
2.1.2.4 SNMPv1 Operations

The SNMP operations are based on a trap-directed polling technique. Polling is a request-response interaction between a manager and an agent [15], while trap-directed polling places emphasis on the manager and agent communications between an asynchronous interrupt driven and polling based approach. The manager can query any agent and request some information values, and the agent responds with the information from its MIB. There are three operations performed by the manager: Get-Request, Get-Next-Request, and Set-Request. There are two operations performed by the agent: Trap and Get-Response.

The Get-Request operation is the simplest one to use in retrieving a primitive value of a managed object. It contains one or more requests for the MIB variables, which are named in the variable bindings. The result of a Get-Request operation usually causes values according to the object identifiers; however, when this is not possible, an error is generated and filled into a field of the SNMPv1 message by the agent.

The Get-Next-Request operation is very similar to the Get-Request operation, except that the requested object is the one next to the object identifier specified in the variable bindings. The next managed object is the first instance in the lexicographical order. The Get-Next-Request operation permits the viewing of managed objects without prior knowledge. If an object that is being searched is clear, the Get-Request will return it. The common use of a Get-Next-Request is to retrieve a table object. The simplest use is
to start at the beginning of a table, await the response and then issue another Get-Next-Request with the object identifier returned. As in the Get-Request operation, either the desired values or an error are returned by the Get-Next-Request operation.

The Set-Request is the only operation which is allowed to change the value of a managed object. The operation performs all-or-nothing instead of best effort, which means all the instances in the variable-bindings are all set, or nothing is set. The atomic feature of the Set-Request operation results in a complex two-phase operation on the agent side. The first phase is a validation phase, in which all variables in the variable bindings are checked for accessibility and writeability. If all variables are writable, the second phase will then start and all the variables will be set. If during any phase the actions fail, all the original values must be recovered and the error needs to be set [15].

Trap is quite different from other SNMPv1 operations in that it is issued by an agent with an asynchronous notification of some significant event. There are seven different types of trap defined in [5]: coldStart trap, warmStart trap, linkDown trap, authenticationFailure trap, egpNeighborLoss trap, and enterpriseSpecific trap.

2.1.2.5 SNMPv1 Limitations

The increasing popularity of SNMP in the late 1980s and early 1990s, combined with the more complex worldwide networks, resulted in an awareness of the deficiencies in SNMPv1. According to [15] and [19] the major limitations of SNMPv1 are as follows:
- SNMPv1 uses trivial security. The security mechanism in SNMPv1 is community-based, which is known as trivial authentication. The community name itself is not encrypted so a tool can easily ascertain the community name. Once the correct community name has been discovered, an unauthorized party could perform all the protocol operations. This might be the true reason why some SNMPv1 vendors do not want to implement a Set-Request operation. Thus, SNMPv1 is better suited for monitoring rather than control.

- BER in SNMPv1 imposes large network overhead.

- SNMPv1 is not well suited for retrieving large amount of data, such as an entire routing table. In SNMPv1, access to a table entry is no different from accessing an object identifier.

- SNMPv1 traps are unacknowledged. In the typical case where UDP/IP is used to deliver trap messages, an agent is not ensured that a critical message has been received by the management station.

- SNMPv1 does not provide manager-to-manager communications. In other words, there is no mechanism that permits a management system to be aware of the devices and networks managed by another management system.

- SNMPv1 does not define enough error codes. This may cause ambiguous understanding of the cause of an error on the part of the manager. In many cases the manager has to successively apply for parts of the original request, in order to find the problem.

- SNMPv1, using a minimal set of protocol operations, complies with the objective of simplicity.
- SNMPv1 is not suitable for managing really large networks because of the performance limitations of polling. Based on the polling mechanism, one packet must be sent out in order to get back one packet of information. This type of polling results in large volumes of routine messages and generates problem response times that may not be acceptable.

- SNMPv1 has inadequate functions for retrieving bulk information, which results in performance problems. Accessing MIB tables containing repeating variables requires successive Get-Next-Request operations to an agent. If the MIB tables are very large, it takes a considerable amount of time to complete all of the necessary transactions. This is extremely resource-intensive in real time, network bandwidth, and the agent’s CPU time.

SNMPv1 facilitates the addition of management capabilities to a device without impacting on the operation of the device or its performance, since it is indeed lightweight. Moreover, the message size of SNMPv1 is small, which allows for low network overhead to be achieved. The above collection of these limitations is the primary reason for requesting proposals for a successor to SNMPv1.

### 2.1.3 SNMPv2

SNMPv2 is a revised protocol, which includes enhancements to SNMPv1, but still uses the existing community-based security and the same message format of SNMPv1. The following describes the enhancements to SNMPv1 provided by SNMPv2.
2.1.3.1 Enhancements of SNMPv2 over SNMPv1

SNMPv2 provides several enhancements to SNMPv1, which fall into three basic categories: enhancement to SMI, enhancement to manager-to-manager capability and enhancement to protocol operations. The SNMPv2 SMI extends the SNMPv1 SMI into macros that define object types to include several new data types. Another enhancement in this category is that a new convention has been provided for creating and deleting conceptual rows in a table. Since the details of the SNMPv2 SMI do not directly affect this thesis, they are not examined here. Instead, the focus is on the enhancements to protocol operations.

- SNMPv2 supports improved efficiency and performance by introducing a Get-Bulk-Request operation to allow the manager to retrieve a large amount of data. In particular, it is well suited to retrieving multiple rows in an MIB table [15].

- SNMPv2 provides manager-to-manager communications by introducing an Inform-Request operation that is an acknowledged trap type, to facilitate the building of a hierarchical network management system. The Inform-Request operation enables one manager to send a trap type of information to another manager [15].

- SNMPv2 changes the atomic get-response operation to a non-atomic one, to permit partial responses to a request. For example, a Get-Request operation carries more than one variable binding, in one of which an error occurs then none is returned in SNMPv1. However, in SNMPv2, the valid ones are returned and the error index is set to the location of the invalid one in the variable bindings. The non-atomic get-
response reduces the overall management traffic.

- SNMPv2 offers more error handling by errors and exceptions to better inform users about the cause of a failed operation. The improved error handling translates to fewer message exchanges needed to resolve problems.

Although SNMPv2 implements the above enhancements, it does not incur any additional overhead as it uses the same message format of SNMPv1. However, it could be argued that the above enhancements to protocol operations, e.g. the Get-Bulk-Request operation, cause an additional load on the agent. Still, SNMPv2 uses the simple and unsecured password-based authentication, known as the community feature, provided in SNMPv1. To fix the lack of security, a number of independent groups began to work on a security enhancement to SNMPv2, namely SNMPv2u and SNMPv2*. Ultimately, these two approaches served as inputs to a new IETF SNMPv3 group. In April 1999, this group had produced a set of proposed standards for SNMPv3 published as [7, 8, 11, 20, 21, 22]. In December 2002, these SNMPv3 specifications and documentation were standardized.

2.1.4 SNMPv3

SNMPv3 is not a stand-alone replacement for SNMPv1 or SNMPv2, but an enhancement that adds security capacity to be used in conjunction with SNMPv1 or SNMPv2 [15]. SNMPv3 is significantly more complex than either SNMPv1 or SNMPv2, thus an overview is provided below.
2.1.4.1 Enhancements of SNMPv3 over SNMPv2

One primary goal of SNMPv3 is to address a serious deficiency in the earlier versions, i.e. the lack of security. The other primary goal of SNMPv3 is to allow evolution with more clearly defined components while still keeping the fundamental architecture consistent. The enhancements provided by SNMPv3 fall into three categories [15]:

- A new architectural framework that defines the SNMPv3 engine contains several subsystems according to different functionality. This enhancement effectively promotes extensibility for future development.
- A User-based Security Model (USM) that ensures the data authenticity, integrity, and privacy. This enhancement provides security features at the message level.
- A View-based Access Control Model (VACM) that facilitates remote configuration of the SNMPv3 entities, so that the remote administration of the SNMPv3 entities becomes a much simpler task. VACM determines whether a given principal is allowed access to particular MIB objects to perform particular functions, and operates at the PDU level.

Of these three categories, the security and access control capabilities and their associated costs are most relevant to this thesis. The following will discuss them in detail.

2.1.4.2 User-based Security Model

The main idea of USM is to use the traditional concept of a user identified by a userName
in association with some security information. USM identifies SNMPv3 entities by their
snmpEngineID. SNMPv3 entities could be traditional agents running on managed
deVICES, traditional network management stations, and mid-level managers, etc. The
username is needed for auditing and authorization purposes, while the snmpEngineID is
needed to identify the target of an SNMP operation. USM illustrates the use of Hash
Message Authentication codes HMAC-MD5-96 [23] and HMAC-SHA-96 [23] as the
authentication protocols and the use of Cipher Block Chain mode of Data Encryption
Standard (CBC-DES) [24] as the privacy protocol. It also demonstrates the procedures
for key generation, update, and use. Particularly, according to [33] USM provides
protection against the following principal threats:

- **Modification threat**: an unauthorized entity could alter an in-transit SNMP message
generated on behalf of an authorized user in such a way as to effect unauthorized
management operations, including the setting of object values. The impact of this
threat is that an unauthorized entity could change any management parameter,
including those related to configuration, operations, and accounting.

- **Masquerade threat**: management operations not authorized for some users may be
attempted by assuming the identity of another user who has the appropriate
authorization.

- **Disclosure threat**: this refers to the risk of eavesdropping on the exchanges between
a managed agent and a management station. For example, the observation of a set
command that changes a password would enable an attacker to discover the new
password.
• **Message stream modification threat:** the risk that messages may be maliciously reordered, replayed or delayed to an extent which is greater than can occur through the natural operation of a sub-network service, so as to effect unauthorized management operations.

At the time of message transmissions, the USM uses a private key stored in a local and remote user, and the CBC-DES for encryption. It then uses an authentication key that is stored in the local and remote users, and HMAC-MD5-96 to produce a MAC that contains 12 octets. Upon receiving the message, the USM first checks the incoming MAC against a MAC that is calculated using HMAC-MD5-96 and the authentication key; if the two values match, then the message is assumed to be authentic (comes from the alleged source and has not been altered in transmission). Then, the USM checks whether the message is within a valid time window. The timeliness mechanism provided by the USM guards against message delay and message replay. If the message is not timely, it is abandoned as not authentic. Finally, the USM performs a decryption using CBC-DES and the private key and returns the plain text [33].

### 2.1.4.3 View-based Access Control Model

The main idea of VACM is to provide a model to determine whether a given principal is permitted to access a particular MIB object. VACM makes the use of a MIB that defines the access control policy for an agent and makes it possible for remote configuration to be used. When a SNMP request PDU is being processed, VACM is called to ensure the
MIB objects specified within the variable bindings are allowed to be accessed. When a notification that could be a SNMP trap or inform request is being generated, VACM is called to ensure the MIB objects specified for the variable bindings are allowed to be accessed.

2.1.4.4 Characteristics of Using SNMPv3

Although SNMPv1, SNMPv2, and SNMPv3 more or less implement security features at the application level, ensuring that the communication channel is protected from end to end, SNMPv3 provides a truly end-to-end authentication, privacy, and access control protection.

Implementing the SNMPv3's new security and access control features require a new message format. The new SNMPv3 message is shown in Figure 2-5. The SNMPv3 message is significantly larger than the SNMPv1 message. Although the imposed network overhead according to the header of the SNMPv3 message could be amortized over a large number of variable bindings in one request message, with a small number of variable bindings in one request message, the SNMPv3 consumes much more link capacity than either SNMPv1 or SNMPv2. The network overhead imposed by SNMPv3 will be further explored by performance evaluations, discussed in Chapter 5.
msgVersion
msgID
msgMaxSize
msgFlags
msgSecurityModel
msgAuthoritativeEngineID
msgAuthoritativeEngineBoots
msgAuthoritativeEngineTime
msgUserName
msgAuthenticationParameters
msgPrivacyParameters
contextEngineID
customName
PDU

Figure 2-5 SNMPv3 message format.

In addition, adding the aforementioned complex security and access control model into a SNMPv3 daemon increases the memory footprint of the SNMPv3 daemon, and places additional computational load on the SNMPv3 daemon, possibly degrading the performance of devices. The memory footprint of SNMPv3 and the processing time of a SNMPv3 operation will be further explored by performance evaluations, discussed in Chapter 5.

2.2 Data collection via MIB-II

The management information to be exchanged between the ground management station and the micro-satellites is defined in terms of managed objects, collected in management information base (MIB) modules [14]. ASN.1 is used to specify the data in a MIB.
There are three kinds of MIB modules: standard, experimental, and enterprise-specific [14]. In our case, we define an experimental MIB module specific to micro-satellite networks, i.e. micro-satellite MIB (MSMIB).

When defining the micro-satellite MIB, the management protocol stack, the requirements of the management station and the micro-satellite network interfaces, and the relevant management objects already defined in the existing MIB standards should be considered. In this case the Internet standard MIB-II [34] developed by IETF has been investigated. Figure 2-6 illustrates the MIB-II tree which contains the system group, interface group, address translation group, IP group, TCP group, UDP group, EGP group, transmission group, and SNMP group.

![MIB-II diagram]

Figure 2-6 MIB-II group structure.

In fact, the MIB-II defines the variables for managing a TCP/IP protocol stack via application management, transport management, network management, and transmission management, as shown in Figure 2-7. The shaded portions are addressed by MIB-II.

New MIB modules that extend the MIB-II are continuously being created by various IETF working groups. It is also common for enterprises or individuals to define or
extend enterprise-specific or experimental MIBs [35]. As a result, managed devices are frequently complex collections of manageable components, independently installed on a managed node. In this case, the need to dynamically extend the management objects within a node has given rise to a variety of "extensible" agents. A standard protocol for agent extensibility called AgentX [35] provides the technical foundation required in solving the aforementioned problem. The goals of extensible agents are to separate an SNMP protocol engine from MIB instrumentation and to facilitate the dynamic extension with new MIB modules.

Figure 2-7 Scope of MIB-II in a protocol stack.

Besides AgentX, SNMP Distributed Program Interface (DPI) [36] is another subagent protocol proposed by IETF. It permits end-users to dynamically add, delete or replace management variables in the local MIB without requiring recompilation of the SNMP
2.3 CMIP

CMIP is an OSI-based network management protocol. It was funded by governments and large corporations to replace and make up for the deficiencies in SNMP, thus improving the capabilities of network management systems. It is a well-designed protocol that defines how network management information is exchanged between network management application and management agents. It uses an ISO reliable connection-oriented transport service, and has a built-in security that supports access control, authentication, and security logs. The network management application can initiate transactions with management agents using the following operations [32]:

- **ACTION**: To request an action to occur as defined by the managed object.
- **CANCEL_GET**: To cancel an outstanding GET request.
- **CREATE**: To create an instance of a managed object.
- **DELETE**: To delete an instance of a managed object.
- **GET**: To request the value of a managed object instance.
- **SET**: To set the value of a managed object instance.

A management agent can initiate a transaction with the network management application using the **EVENT-REPORT** operation. This operation can be used to send notifications or alarms to the network management application based upon predetermined conditions.
set by the network management application using the ACTION operation.

The biggest feature of the CMIP is that its variables not only relay information to and from the terminal (as in SNMP), but they can also be used to perform tasks that would be impossible under SNMP. Therefore, CMIP results in a more efficient network management system. Less work is required by a user to remain updated on the status of their network. Furthermore, CMIP is a safer system as it has built-in security.

However, CMIP requires a large amount of system resources, which results in very few implementations. Additionally, CMIP is very complex, thus making it difficult to program. Therefore, skilled personnel with specialized training may be required to deploy, maintain, and operate a CMIP based network management system.

2.4 Generative Programming for PIX

PIX (Protocol Implementation for Linux) is a protocol stack development tool using the GP approach to solve problems of reprogramming similar behavior and configuration of crosscutting concerns in communication systems [25]. It is a model consisting of software components that can be assembled to build a communication system. PIX models a family of telecommunications protocol stacks using the following abstractions: Protocol, Session, Participant, Map, Message, Event, and Uniform Interface. In this section, GP is overviewed in brief. PIX is illustrated in detail, as it is the foundation of the LW-SNMP implementation.
2.4.1 Overview of Generative Programming

According to [18], GP is a software engineering paradigm based on modeling software system families so that, given a particular requirement specification, a highly customized and optimized intermediate or end-product can be automatically manufactured on demand from elementary, reusable implementation components by means of configuration knowledge. It is aimed at program families in order to achieve configurability, and represents a particular method of domain engineering. Moreover, it requires a means of specifying family members, implementing components as a generation base, and a configuration knowledge that maps from a specification to a product. Typically, GP relies on code generators. In C++, template metaprogramming has been used to generate tailored configurations of classes and functions.

The insight of a program family is that a designer carefully considers the basic abstractions shared by all members of a family, namely, commonalities and differences of the family members, particularly with regard to variability, so that the cost of evolvement and maintenance of the programs will be reduced [18]. The main reason for this is that the members of a program family share basic design decisions and can use identical or similar resources. Program families provide a solution that satisfies the individual requirements and avoids performance deficiencies caused by services that are not necessary for a less demanding user.

According to [18], the goals of GP are:
- To decrease the conceptual gap between program code and domain concept
- To achieve high configurability and adaptability
- To simplify managing many variants of a component
- To increase efficiency both in space and execution time

In order to achieve these goals, the GP approach deploys the following principles:

- **Separating problem space and solution space**

  Problem space includes the concepts and features on the part of the programmer, whereas solution space includes the implementation components with all the possible combinations. The mapping between the problem space and solution space is the configuration knowledge. The advantage of this principle is that it allows for the evolution of both spaces in a relatively independent way.

- **Separating main concerns**

  To avoid program code that deals with many issues simultaneously, GP aims at separating each issue into a distinct set of code. These sets of code are combined to generate a needed component [18]. Separating main concerns facilitates the modeling of a complex system.

- **Parameterization of differences**

  Parameterization adopted from generic programming allows us to compactly represent families of components.
• **Analysis and modeling of dependencies and interactions**

Not all parameter value combinations are valid, and the values of some parameters may imply the values of some other parameters. These dependencies are referred to as horizontal configuration knowledge, since they occur between parameters at one level of abstraction [18]. This allows us to capture the invalid combinations at an early stage.

• **Minimizing overhead and performing domain-specific optimization**

By using static metaprogramming, the components are generated at compile time. Therefore, much of the overhead resulting from unused code, run time checks and unnecessary levels of indirection may be minimized.

There are three main steps for approaching GP. Inside each step there are some sub-steps to be followed. These steps, as well as the sub-steps, will be addressed in the next section via a concrete example – PIX.

### 2.4.2 Protocol Implementation Framework for Linux (PIX)

PIX is a framework for modeling a family of telecommunication protocols. It provides a collection of abstract classes that encapsulates the common operations of a family of protocols. As such, it is regarded as a “code template” while all common code is present in it but the key methods and other details still need to be supplied. PIX facilitates the separation of responsibilities of the different layers and their final integration.
This section is based on the knowledge of GP to demonstrate the use of a domain engineering method applied for GP in support of PIX.

2.4.2.1 PIX Domain Analysis

Domain analysis is the first step of the domain engineering method for GP. It consists of the analysis of domain scope and context, the analysis of commonalties, variability, and dependencies within a family of systems, and the development of reusable and configurable requirements [18].

To start with the domain analysis for PIX, we first need to study the available telecommunication protocols. Derived from the x-kernel that is a user-level implementation of protocols in C++, our PIX provides protocol, session, uniform interface, message, map, participant, and event abstractions.

Protocol is an abstraction that, on one hand, defines the operations by which other local protocols invoke its services. On the other hand, it provides the format of messages exchanged by the remote peers. One specific protocol consumes the services applied by its lower protocol and it is requested to provide services to its higher protocol. Moreover, a protocol provides basic management to associated sessions.

A session is an abstraction of a local end-point of a communication channel. It is associated with a particular protocol. Therefore a session, loosely speaking, is a local
representative of a protocol, which interprets messages and maintains the state variables associated with the channel. For example, the local end-point of a manager-to-agent channel is abstracted as an SNMP session and is embedded and managed by an SNMP protocol.

A Message is an abstraction of a byte string which is manipulated for transmission and reception in a protocol stack.

A Map is an abstraction of a function mapping external keys to internal values. It provides operations for maintaining a set of bindings. Both the ActiveMap and PassiveMap are derived from Map. An ActiveMap maps the keys found in incoming messages to the sessions that process the messages. Such a key is called an ActiveId, which is used to distinguish sessions that are parts of a particular protocol. The ActiveId is a fixed length octet string, which typically is constructed from various fields in a message header. Typically the value associated with an ActiveId is a pointer to a Session object. A PassiveMap is used to bind keys found in incoming messages into protocol objects, thereby allowing a protocol to create a session when a message related to a new connection arrives.

A Participant is an abstraction of an identity of an application, a protocol or a session where, as a matter of convenience, the high level protocol identifies itself in first elements of a participant.
An Event allows a protocol to schedule a procedure that is to be called after a period of time.

A Uniform Interface (UI) provides operations common to all protocols and sessions. The main operations of the UI are as follows:

- **Operations of Protocol**
  
  XOpen (): Actively opens a session. The parameters of this operation are a local participant, a remote participant, and a reference of the higher level protocol.
  
  XOpenEnable (): Passively opens a session. The parameters of this operation are a local participant and a reference of the higher level protocol.
  
  XControl (): Accesses or modifies the state variables of a protocol.
  
  XDemux (): Demultiplexes an incoming message to a session. The parameters of this operation are the incoming message and the lower level session.

- **Operations of Session**
  
  Xpush (): Sends a message
  
  Xpop (): Receives a message
  
  XControl (): Accesses or modifies state variables

The domain analysis, an identification procedure of the commonalities and variability among the members of a family of software, results in a feature diagram that contains four kinds of features: mandatory features, optional features, alternative features, and or-
features. The notation of a mandatory feature is an edge with a filled-in circle. The notation of an optional feature is an edge with an empty circle. The notation of an alternative feature is an edge connected by an arc. The notation of an or-feature is an edge connected by a filled arc.

The diagram in Figure 2-8 is a tree-like feature diagram of PIX, in which the root represents the concept of a protocol and the remaining nodes denote features. The relationship among them is consist-of. For example, the description of a message consists of the description of a single-buffer message and a buffer-tree message.

![Feature diagram of PIX](image)

**Figure 2-8 Feature diagram of PIX.**

The Message and Participant features are common to all protocols. A possible choice for the Message feature could be either a single buffer message, as in Linux, or a directed
acyclic graph of buffers, as in x-kernel.

The Session, EventManager, ActiveOpen, and PassiveOpen features are the variability of the protocol concept. The Session feature is optional because a protocol need not consist of a session if the protocol is a high-level protocol that abstracts an application role or a low-level protocol that abstracts hardware devices. Session consists of the Message mandatory feature and Eventmanager optional feature. The ActiveOpen feature is optional for a protocol because certain protocols, e.g. the high-level or low-level protocols mentioned above, do not provide communication channels and certainly do not need the ActiveOpen feature. PassiveOpen is an optional feature because enabling addresses is needed only for nodes running servers, which may not be required, for instance, by handheld computers.

2.4.2.2 PIX Domain Design

The purpose of a domain design is to design the target architecture and software components. PIX adopted GenVoca architecture for this particular purpose. According to [18] there are five steps in deploying the domain design:

Identify the main functionality in the feature diagrams from the domain analysis

Enumerate component categories and components per category

Identify “use” dependencies between component categories

Sort the categories into a layered architecture
Write down the GenVoca grammar

The GenVoca grammar finally formalizes the elements of the feature diagram and the dependencies between categories of components, where each layer is defined as a refinement of the layer below. The GenVoca grammar for PIX is listed in Figure 2-9.

<table>
<thead>
<tr>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWithOptAO:</td>
</tr>
<tr>
<td>ActiveOpen[Config]</td>
</tr>
<tr>
<td>Config:</td>
</tr>
<tr>
<td>ProtocolUI</td>
</tr>
<tr>
<td>EventManager</td>
</tr>
<tr>
<td>ActiveId</td>
</tr>
<tr>
<td>ActiveMap</td>
</tr>
<tr>
<td>PassiveId</td>
</tr>
<tr>
<td>PassiveMap</td>
</tr>
<tr>
<td>ProtocolUI:</td>
</tr>
<tr>
<td>Protocol[ProtocolUI</td>
</tr>
<tr>
<td>ProtocolUI</td>
</tr>
<tr>
<td>Message</td>
</tr>
<tr>
<td>Session</td>
</tr>
<tr>
<td>Participant</td>
</tr>
<tr>
<td>SessionUI:</td>
</tr>
<tr>
<td>Session[SessionConfig]</td>
</tr>
<tr>
<td>SessionConfig:</td>
</tr>
<tr>
<td>Message</td>
</tr>
<tr>
<td>EventManager</td>
</tr>
</tbody>
</table>

Figure 2-9 GenVoca grammar for PIX.

The configuration repository, namely SessionConfig, of the Session feature consists of Message and EventManager. The SessionConfig is used to define the SessionUI layer, which reflects the Session concept.
The Protocol concept has been distinguished between the essential features that are required to interact with another protocol and the additional features that are needed to define the internal of a protocol [25]. This distinction clearly separates the protocol interface from the protocol implementation. The essential features of the Protocol concept are collected in ProtocolUIConfig, which is a Protocol configuration repository. The ProtocolUIConfig is used to define the layer above, that is ProtocolUI which defines the aforementioned protocol interface. The layer right above ProtocolUI is the configuration repository of protocol implementation, namely Config. Config is used to define the ActiveOpen feature of the Protocol concept. The PassiveOpen feature depends on the ActiveOpen feature with the refinement of ActiveOpen. The uppermost layer in Figure 2-4 is the Protocol Layer, which groups three different alternatives separated with a vertical bar. The PassiveOpen alternative, parameterized with POWithOptAO, provides a basis for protocols that have sessions with client/server behavior [25]. The ActiveOpen alternative, parameterized with Config, provides a basis for protocols with sessions with client behavior only. The ProtocolUI alternative, parameterized with ProtocolUIConfig, provides a protocol without embedded sessions, such as the top-level protocols and device drivers.

2.4.2.3 PIX Domain Implementation

Domain implementation usually contains an implementation of the components in the GenVoca architecture using C++ templates, and a configuration generator using configuration knowledge and template metaprogramming.
Based on the GenVoca architecture, we can implement the components. As we have seen in the architecture, the component in the upper layer uses the component in the adjacent lower layer. Therefore, the component of a layer must take the component of its lower layer as a parameter as so to be automatically notified of the variations. In C++ the class templates can be used to implement the parameterized components. The outline of class templates representing the protocol, ActiveOpen, PassiveOpen, and Session can be found in [25].

A generator is another class template that encapsulates the rules defining how components can be assembled together. It takes a specification of what needs to be assembled and verifies its consistency and provides default values. There are three generators in PIX, named SESSION_UI_GENERATOR, PROTOCOL_UI_GENERATOR, and PROTOCOL_GENERATOR.

The generator of the uniform interface (UI) of sessions takes a session description and returns a finished session UI base class. During this generation, the actual classes of the Message and EventManager features are selected. The result of the assembly is returned as the RET type member. Similarly, the UI of the protocols can be generated by specifying the Message, Participant, and Session features. The ProtocolUI obtained is a base class for all protocols. The PROTOCOL_GENERATOR verifies the input specifications, such as the ProtocolUI, the activeOpenFlag, and passiveOpenFlag, to generate a customized protocol with the inclusion of the ActiveOpen or PassiveOpen features. The result is a base class that can be used through an extension relationship to
define a specific protocol [25].

2.4.2.4 Configuration of a Network Management Protocol

A user level Network Management Protocol (NMP) is used as an example to show how to configure a desired protocol using PIX.

The NMP activeId is a defined structure of a key installed in the active map. The activeMap is used to keep track of network management sessions managed by NMP. First, the activeId and activeMap need to be configured. The activeId of NMP is the UDP session. The activeMap is defined as an actualization of the Map template. The configuration passed to the template defines the size of the key, the type of values pointed to by a pointer paired with the key, and the hash size [25]. Second, the passiveId and passiveMap need to be configured. The passiveId of the NMP, namely a local port installed in the passiveMap, is paired with an instance of an NMPEnable structure. The NMPEnable includes an integer which counts the number of invocations to the open enable operation that have been performed using a local port and a pointer to the high level protocol which was the first to invoke the open enable operation on this port.

The NMP can be generated by specifying the protocolUI, the activeId, the activeMap, the enable object, the passiveId, the passiveMap, and with activeOpen features, since the NMP needs the mechanism in order to carry out client and server communications. The NMP should be configured with the passiveOpen feature, since the NMP needs the
mechanism to realize a daemon operation.

The NMP session is derived from the SessionUI, specifying the single buffer message and simple event manager features.

2.5 Satellite Network Management

It is widely accepted that satellite network management is a useful tool in assisting operators to manage and plan the satellite communications network more effectively and achieve the optimal use of available resources. Most adopted tools for monitoring and controlling satellite networks are related to SNMP, because SNMP is a freely available off-the-shelf product, and a robust as well as widely distributed protocol in the terrestrial communication environment. Trails of network management protocols, such as SNMP and CMIP at Adastral Park, indicate that there are no major problems when operated on satellite [26]. In this thesis, network management in the ATM LEO satellite networks will be introduced in order to establish the state-of-the-art in satellite network management.

2.5.1 Network Management in ATM LEO Satellite Networks

A typical ATM LEO satellite network comprises a ground-based segment, a satellite-based segment, and a Network Control Center (NCC). Network management is performed by the NCC with the help of the on-board satellite ATM switches, and
provides the overall control of the ATM LEO satellite network [27].

The Network Management System involves the NCC, on which the network management applications reside; a set of managed satellite ATM nodes (LEO satellites); a set of management channels; and a network management protocol (SNMP), used to exchange management information.

In [27], a decentralized network management approach has been proposed, in which the control functions are distributed between the on-board satellite ATM switch and the NCC. The management Virtual Channels (mVCs) which carry the management information allows the NCC to reach the satellites at all times. Signaling establishes the mVC and without handoff, the management channel will remain unchanged. To add robustness, a stand-by solution using a Permanent Virtual Path (PVP) has been proposed in [27]. The PVP is a reconfigured virtual path for management information exchanges between the NCC and each satellite ATM node in case of signaling failure. It will be activated when the satellite is seen from the NCC. This definitely makes it possible for the NCC to reach the satellites at all times.

The major areas of ATM satellite network management include configuration, performance, and resource management. Configuration management consists of ATM network element configuration, configuration of ATM user network interface, and configuration of PVPs. Performance management consists of Radio layer performance monitoring. Global resource management includes ATM network entity congestion
monitoring for real-time performance analysis.
Chapter 3

Specifications and Features of LW-SNMP

This chapter will first describe the external behavior of LW-SNMP services and how these services can be realized in terms of protocol functions. Subsequently, the LW-SNMP, which models the protocol functions as software components with commonalties and variability, is illustrated through feature analysis. The dependencies of these features, expressed by a GenVoca grammar, are defined, followed by the presentation of how the aforementioned components are automatically assembled to a LW-SNMP suitable for a particular mission-specific micro-satellite through the use of a LW-SNMP generator.

3.1 LW-SNMP Services

LW-SNMP services are used to exchange management information between network management applications. These services hide the details related to the communications of the management information, such as encode, decode, authentication, encryption, and decryption.

3.1.1 General Characteristics of LW-SNMP Services

LW-SNMP provides Get-Request, Get-Next-Request, Get-Bulk-Request, Set-Request,
and Get-Response services. Before describing the service elements of LW-SNMP, the general characteristics of the LW-SNMP services can be listed as follows:

- **Connectionless**

  All the LW-SNMP services are connectionless services. Applications use the services to exchange management information without setting up a connection. There are three justifications for using an LW-SNMP over a connectionless transport protocol. First, the LW-SNMP should be able to operate in an unreliable environment, in which connections may be lost. Second, the LW-SNMP is designed to have low resource utilization; a connectionless service consumes fewer resources than a connection-oriented service. Third, if the network is experiencing packet loss, the likelihood of a three-way handshake completion such as (Transmission Control Protocol) TCP, is much smaller than that of a single, self-contained packet such as (User Datagram Protocol) UDP making it from the management station to the managed node [14].

- **Blocked and Non-blocked Request**

  The LW-SNMP services provide both blocked (synchronous) and non-blocked (asynchronous) modes of request services. A blocked request service will return as soon as a corresponding response from an agent is received by a manager, or a timeout or an error condition occurs. A non-blocked request service will return as soon as the request has been sent, and does not wait for the response from an agent. Figure 3-1 shows the scenario in a blocked and non-blocked request service, respectively.
Figure 3-1 Blocked and non-blocked LW-SNMP service primitives.

- Error indication from the LW-SNMP

The LW-SNMP itself may generate an error indication and imply the reason for the failure before a request is transmitted, which reduces network traffic. An error indication is shown in Figure 3-2.
3.1.2 LW-SNMP Services

The services provided by the LW-SNMP are used to retrieve or modify a set of specific management information. Table 3-1 shows these services directly deduced from [20], and it also shows the availability in each version of the network management protocols.

<table>
<thead>
<tr>
<th>Service</th>
<th>Support in SNMPv1</th>
<th>Support in SNMPv2</th>
<th>Support in SNMPv3</th>
<th>Support in LW-SNMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get-Request Service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Get-Next-Request Service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Get-Bulk-Request Service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Set-Request Service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Trap Service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Inform-Request Service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3-1 Availability of management services in network management protocols.

The LW-SNMP not only provides the Get-Request, Get-Next-Request, and Set-Request
services, but also provides the Get-Bulk-Request service. The idea behind the Get-Bulk-Request service is to minimize network interactions by allowing an agent to return a large packet [14].

The LW-SNMP implementations currently available do not provide Trap and Inform-Request services. The Trap service could provide immediate notification of an extraordinary event, such as a link going down. The Inform-Request service, an acknowledged trap, facilitates the interactions between managers. The support of Trap and Inform-Request services in the LW-SNMP is reserved for future work.

3.2 LW-SNMP Protocol Functions

This section describes the protocol functions that the LW-SNMP has to perform in order to provide the services identified in the previous section. These protocol functions performed by the LW-SNMP are presented using activity diagrams. Figure 3-3 displays the graphical notation of the activity diagram.

![Activity diagram symbols.](image)

Figure 3-3 Activity diagram symbols.
Figure 3-4 shows the activity of the LW-SNMP functions on delivery of a message. An LW-SNMP message is constructed according to a received service primitive and its associated parameters. An encoding rule is then used to serialize the LW-SNMP message into a byte stream. Subsequently the encryption protocol is used to encrypt the byte stream into a private data. Finally, an authentication protocol is used to calculate a digest for the private data with a shared secret key. The digest is concatenated together with the private data as an authenticated private byte stream, transmitted over a transport layer. As such, a reversed sequence is applied if an LW-SNMP message is received.

Figure 3-4 Activity diagram showing LW-SNMP protocol functions.

3.3 LW-SNMP Domain Feature Modeling

From Section 3.2, the major functions of the LW-SNMP include encode, decode,
authentication, encryption, and decryption. Among these functions, there exist many variations. For example, an encoding rule can be a Distinguished Encoding Rule (DER), a Lightweight Encoding Rule (LWER), or LW-BER, and an authentication protocol can be a Key-Preappended-and-Appended MD5-based or HMAC-MD5 based approach. In order to design the LW-SNMP to operate on all of these variations, the GP approach is chosen because it allows us to specify information at compile time. An example of this would be the choice in selecting which encoding rule to use, and whether to make use of encryption or no encryption. Using the GP approach, we do not lose efficiency by providing generality.

A family of LW-SNMP can be modeled using the following features: LW-SNMP, EncodingRule, Authentication, Encryption, and Service. To interact with an LW-SNMP, only the EncodingRule and Service features are required. The Encryption feature is optional because the LW-SNMP is mainly used to transfer management messages, and neither password nor confidential information, and therefore encryption may not be the first priority. The Authentication feature is also optional because, in some cases, LW-SNMP may be used for monitoring rather than modifying. Under this circumstance, authentication may be neglected. For each identified feature of the LW-SNMP, some variations exist. First, the EncodingRule feature has as its feature values the DER, the LW-BER, and the LWER. Second, the Authentication feature has the Key-Preappended-and-Appended MD5 and HMAC-MD5 as its feature values. Finally, the Service feature has the Centralized-Service and the Distributed-Service as its feature values. The EncodingRule and the Authentication variation points are marked as open where new
direct sub-features are expected. An open feature is presented by enclosing the feature name in brackets. The EncodingRule has been marked open to indicate that, if needed, other encoding rules may be added. The above stated results in a feature diagram where each feature is a property with importance, and is a point where choices can be made to define a particular instance of a concept that is the root of the feature diagram. The feature diagram of the LW-SNMP concept is shown in Figure 3-5. A brief description of each feature, as well as its values, follow this diagram.

![Feature Diagram of LW-SNMP Domain]

Figure 3-5 Feature diagram of LW-SNMP domain.

- **EncodingRule**

  LW-BER derived from BER is proposed for the bandwidth and on-board computer constraint of micro-satellite networks. LW-BER aims to reduce the encoding overhead imposed by BER. Since the management message exchanged between a manager and an
agent is a fixed data structure, the type value can be eliminated without any ambiguity being introduced. The constructed type encoding overhead, e.g. the type and length, can also be eliminated in any circumstance, because both the manager and the agent are aware of the data structure. All object identifiers manipulated by the manager and the agent begin with the same prefix, so LW-BER removes the prefix on encoding. In summary, tag omission, elimination of redundant nested tag and length, and removing the common prefix of object identifiers are the ways in which LW-BER reduces encoding overhead. In addition, LW-BER pursues the balance between encoding overhead reduction and time computing.

[12] defines the Distinguished Encoding Rule (DER) that uses a TLV format which is similar to BER, but eliminates the many alternative encodings that are possible with BER. DER is useful in cases such as cryptography, where independently generated encoding must be identical.

LWER is designed for the purpose of enabling high-speed ASN.1 encoding/decoding [30]. LWER encodes data types in a similar way to the local storage of many computer systems. Most primitive types, such as INTEGER, BOOLEAN, REAL and ENUMERATED, are encoded as fixed-length fields of four bytes. BIT STRING and OCTET STRING types are encoded as a length code and a pointer. The pointer value indicates the position of the string data later in the encoding sequence. Tag values are never included in the LWER. In general, the LWER increases the encoding overhead significantly, e.g. it always uses four octets to encode an integer value. LWER is suitable
for high-speed applications where bandwidth is adequate, but processing resources are extremely limited.

- **Authentication**

An authentication protocol supports data integrity, which guarantees that the data is not altered or destroyed in an unauthorized manner. It also supports data origin authentication, which guarantees that the claimed origin of the received data is correct. Both of these are accomplished by computing a digest over a message and a secret key. However, the message itself is not encrypted, and the third party can still view the message. For satellite communications, the management messages sometimes need to be exchanged in a secure mode, e.g. authenticated and encrypted, which depends on a specific situation. If the management information is all read-only, the network management can only monitor the network instead of modifying it. In this case, the authentication is not necessary. As a result, the Authentication feature is optional.

With regard to Figure 3-5, two choices for performing authentication are all based on MD5. MD5 message-digest algorithm [9], developed by Professor Ronald L. Rivest of MIT, is an algorithm that takes a message with an arbitrary length as input and produces an output with a 128-bit "fingerprint" or "message digest" of the input. It is supposed to be computationally infeasible to produce two messages which have the same message digest. This uniqueness enables the message digest to act as a 'fingerprint' of a message. Essentially, MD5 is a way of verifying data integrity, and is much more accountable than a checksum.
Key-Preappended-and-Appended MD5 is an approach to using keyed MD5 to perform authentication. It adds a shared secret key before and after the data payload, which results in an input message to the MD5 algorithm, while a 16-byte output is considered as a message digest.

HMAC-MD5 [23], a standard approach, combines an MD5 cryptographic hash function with a secret key for calculation and verification of the message authentication values.

- **Encryption**
  The encryption feature is optional in our design because in some radio frequencies, such as amateur radio frequencies, encryption is not allowed, whereas it will be needed when the LW-SNMP becomes a commercial application.

  The most widely used encryption scheme is based on the Data Encryption Standard (DES). For DES, data is encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms the 64-bit input in a series of steps into a 64-bit output. To reverse the encryption, the same steps, using the same key, are required.

- **Service**
  The Service feature plays an essential role in the LW-SNMP since it is responsible for packing, unpacking, processing, encoding, decoding, and authenticating an LW-SNMP message. It also provides the interfaces of the external services for the management applications.
The Centralized-Service assumes that there is one centralized manager exchanging management messages among all the agents. Therefore, it can only provide Get-Request, Get-Next-Request, Get-Bulk-Request, Set-Request, and Get-Response services.

The Centralized-Service-With-Trap provides a trap-directed polling service. It not only supports all services provided by Centralized-Service, but also supports the unsolicited Trap service initiated from an agent.

The Distributed-Service provides the centralized services, and supports a manager-to-manager service. In this case, there may exist more than one manager in the network, and the management tasks are distributed to several managers.

The Distributed-Service-With-Trap extends Distributed-Service to support the trap service.

Please note that among the features in the proposed feature diagram, we have implemented the LW-BER, key-Preappended-and-Appended MD5, HMAC-MD5, and Centralized-Service without Trap features.

3.4 LW-SNMP domain architecture design

A feature diagram models the commonalities and variability of the LW-SNMP family members. The GenVoca architecture uses these features to build a set of configurable
layers containing one or more classes. The legend five steps discussed in Section 2.4 are followed in designing a GenVoca architecture for the LW-SNMP domain.

Identify the main responsibilities in the feature diagram

The main responsibility of the LW-SNMP is to exchange management messages between a manager and an agent, which includes encoding, decoding messages, encrypting and decrypting messages, authenticating messages and providing external services to the management applications.

- Enumerate component categories and components per category

Based on the first step, the component categories for the LW-SNMP and the components per category are shown in Figure 3.6.

<table>
<thead>
<tr>
<th>EncodingRule</th>
<th>Authentication</th>
<th>Encryption</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>DER</td>
<td>Key-Preappended-and-Appended MD5</td>
<td>DES</td>
<td>Centralized-Service</td>
</tr>
<tr>
<td>LW-BER</td>
<td>HMAC-MD5</td>
<td></td>
<td>Distributed-Service</td>
</tr>
<tr>
<td>LWER</td>
<td></td>
<td></td>
<td>Centralized-Service -With-Trap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Distributed-Service  -With-Trap</td>
</tr>
</tbody>
</table>

Figure 3-6 Component categories for LW-SNMP domain.

- Identify “uses” dependencies between component categories
As shown in Figure 3-7, the Encryption uses the EncodingRule, since the encoded message needs to first be encrypted. Authentication uses encryption and finally Service uses Authentication. Encryption and Authentication are optional; the Service can directly manipulate messages after authentication or encoding.

![Diagram](image)

**Figure 3-7 Component categories dependencies.**

- Sort the categories into a layered architecture

The component categories can be further arranged into a hierarchy of layers in such a manner that the categories, which are dependent on other categories, are moved so that they are above the categories which require them. In Figure 3-8 the bottom components are formed as a ConfigurationRepository that allows all layers to obtain configuration information directly. The dashed box around Authentication and Privacy indicates that these two layers are optional.
Figure 3-8 Component categories layered architecture.

- Write down the GenVoca grammar

The layered architecture allows a component from one layer to take another component from the layer right below it as its parameter. A GenVoca grammar is a set of rules for presenting this layered architecture. As shown in Figure 3-9, the configuration repository includes different encoding rules. The Service takes Authentication as its parameter and it may also take Privacy as its parameter, since Authentication is optional. Privacy is also optional, and therefore the Service may take EncodingRule as its parameter. If the Privacy feature is selected, then Authentication must be selected.
3.5 Implementations of LW-SNMP Domain Components

The architecture design of the LW-SNMP allows us to implement its components. Since a component from a given layer takes a component from the layer below it as its parameter, we need to implement the components as parameterized components in a bottom-up order. At this point, we will focus only on the interface of these components. The detailed implementation will be addressed in the next chapter.

An encoding rule can be a DER, a LW-BER, and a LWER; however, all of them need to provide the common interfaces for serializing a management message and converting the serialized stream to a management message in order to be used by the layer above it. The interfaces of all the encoding rules are shown below.
Privacy is an optional layer, so it could be implemented as inheritance-based wrappers. The specific implementation is to override those two interfaces provided by EncodingRule, which is to add encryption functionality to the writeLwmsg function, and to add decryption functionality to the readLwmsg function. Figure 3-10 shows the declaration of this class.

```cpp
template <class Config> class Encryption: public Config::EncodingRule
{
public:
    // make Config available as a member type
    typedef Config Config;

public:
    // override the functions inherited from its super class.
    int writeLwmsg (unsigned char *bp, struct LWMsg *msg);

    int readLwmsg (unsigned char *bp, struct LWMsg *msg);
};
```

Figure 3-10 Segment of Encryption class declaration.

The Authentication component takes the Encryption as its parameter. Since it is an optional layer, it could be implemented as inheritance-based wrappers, which means that the templates derive from their parameters. In this case it is the Encryption that is to be refined. The specific implementation is to override those two functions provided by Encryption, which is to add authentication to the writeLwmsg function, and to add authentication checking to the readLwmsg function. Figure 3-11 shows the declaration of
this class.

```
template <class Encryption> class MD5-Authentication : public Encryption {

public:
    // make Config available as a member type
    typedef typename Encryption::Config Config;

private:
    // retrieve needed types from the configuration repository
    typedef typename Config::EncodingRule EncodingRule;

public:
    // override the functions inherited from its super class.
    int writeLwmsg (unsigned char *bp, struct LWMsg *msg);
    int readLwmsg (unsigned char *bp, struct LWMsg *msg);
};
```

Figure 3-11 Segment of MD5 authentication class declaration.

The Service component uses its lower layer to process management messages. It also
provides end-point communication channel services. In fact, it is a session
implementation in PIX. It not only provides a refined implementation of the uniform
interface for sessions, e.g. pushing and popping data through the data channel, but also
provides the service interfaces for an application, e.g. Get-Request, Set-Request, and Get-
Response services. The declaration of this class is shown in Figure 3-12. The Service
class takes Authentication as a parameter. The Authentication class is instantiated in the
Service class.

67
template <class Authentication, class SNMPSession, class ServiceInterface>
class CentralizedService: public SNMPSession, public ServiceInterface {

private:
    Encryption e;

public:
    // refine the methods inheritance from SNMPSession
    Outcome xPush(Message *msg);
    Outcome xPop(SessionUI *IIs, Message *msg, unsigned char *hdr);

    // provide the external service for management applications.
    Outcome synGetRequest (int udpsd, int num_of_bindings, binding
                        *bindings, LWMMsg *rmsg);

    Outcome asynGetRequest (int udpsd, int num_of_bindings, binding
                        *bindings, LWMMsg *rmsg);

    Outcome receiveRequest (int udpsd, LWMMsg *rmsg);
};

Figure 3-12 Segment of Centralized-Service class declaration.

3.6 LW-SNMP domain generator

Based on the feature diagram of the LW-SNMP domain in Figure 3-5, a total of $3 \times 3 \times 4 + 3 \times 2 \times 4$ or 60 different LW-SNMP configurations could be derived from it. The detailed explanations are as follows. The LW-SNMP has two mandatory and two optional direct features. If the optional encryption feature is not chosen, there are three EncodingRule alternative subfeatures, three Authentication alternative subfeatures, and four Service alternative subfeatures. Therefore, in this case, we have $3 \times 3 \times 4$ different
combinations. If the optional encryption feature is chosen, then the authentication feature must be chosen. In this case we have three choices for the EncodingRule feature, two choices for the Authentication feature, one choice for the Encryption feature, and four choices for Service. Thus in this case, we have $3 \times 2 \times 4$ different choices. These two cases cover all the possibilities, for a total of 60 different configurations. A generator implemented as a template takes the description of a LW-SNMP, such as a specific encoding rule, an authentication protocol, as its parameters and returns a finished LW-SNMP class.

There are four steps for implementing the LW-SNMP generator that is deemed to be a template metafunction, using the techniques presented in Section 2.4.
• To introduce a short name for the fully parameterized generator template and to retrieve all the needed parameters from the LW-SNMP specification

```c
enum EncodingRule {der, lwber, lwer};
enum Encryption {with_encryption, no_encryption};
enum Authentication {with_keyed_md5, with_hmac_md5, no_md5};
enum Service {distributedservice, distributedservicewithtrap, centralizedservice, distributedservice, centralizedservicewithtrap};
template <EncodingRule encodingRule = lwber,
  Encryption encryption = no_encryption,
  Authentication authentication = no_md5,
  Service service = centralizedservice>
class LW-SNMP_GENERATOR
{
  public:
    typedef SNMP_GENERATOR <encodingRule, encryption, authentication, service> Generator;
  private:
    enum {
    // parse domain specific language
    isLWBER = encodingRule == lwber, hasEncryption = encryption == with_encryption,
    withAuthentication = authentication == no_md5,
    isCentralizedservice = service == centralizedservice
    };
}
```

• To use the metacode structures `IF<>` and `SWITCH<>` to select the appropriate implementation components based on the specification parameters [19].

```c
// assemble encoding rule
typedef SWITCH<encodingRule, CASE<der, class DER, CASE<lwber, class LW-BER, CASE<per, class PER >> >>>::RET EncodingRule_;
// assemble encryption
typedef IF<hasEncryption, Encryption<Generator>, EncodingRule_>::RET Encryption_;
// assemble authentication
typedef SWITCH<authentication,
  CASE<with_keyed_md5, Key-Preappended-and-Appended MD5<Encryption_>,
  CASE<with_hmac_md5, HMAC-MD5<Encryption_>,
  CASE<no_md5, Encryption_>>,::RET Authentication_;
// assemble service
typedef SWITCH<service,
  CASE<centralizedservice, Centralized-Service<Authentication_, SNMPSession, SNMPInterface>,
  CASE<distributedservice, Distributed-Service<Authentication_, SNMPSession, SNMPInterface>>,
  CASE<distributedservicewithtrap, Centralized-Service-Trap<Authentication_, SNMPSession, SNMPInterface>>,
  CASE<distributedservicewithtrap, Distributed-Service-Trap<Authentication_, SNMPSession, SNMPInterface>>,::RET Service_;
public: typedef Service_ RET;
```

• To create the configuration repository in the form of a member structure

```c
struct Config{
  typedef EncodingRule_ EncodingRule;
};
```
The detailed explanation of the above code is as follows:

Lines 1 to 4 define the vocabulary for the different encoding rules, encryption protocols, authentication protocols, and service types.

Lines 5 to 9 define the complete LW-SNMP generator class with several features as actual parameters of a template class.

Line 12 defines a short name for the complete generator.

Lines 13 to 19 parse the domain-specific language.

Lines 20 to 21 assemble the encoding rule components, according to the template’s actual parameters.

Lines 22 to 23 assemble the encryption components, according to the template’s actual parameters.

Lines 24 to 28 assemble the authentication components, according to the template’s actual parameters.

Lines 29 to 34 assemble the service components, according to the template’s actual parameters.

Line 36 returns the assembled service base class.

Lines 37 to 41 create the configuration repository in the form of a member type.

LW-SNMP_GENERATOR<> evaluates the input parameters, and computes the types for the configuration repository. To generate a desired LW-SNMP, the management application needs to indicate a specific encoding rule, whether there is an authentication protocol or no authentication, with encryption or no encryption, and centralized service or distributed service. In the research carried out in this thesis, a LW-SNMP with LW-BER,
with Key-Preappended-and-Appended MD5 authentication, without encryption, and
centralized service is desired. The following code demonstrates how to generate the LW-
SNMP class based on LW-SNMP_GENERATOR.

typedef LW-SNMP_GENERATOR<lwber, no_privacy, with_keyed_md5,
centralizedservice>:: RET LW-SNMP;

If the default configurations of a LW-SNMP are not satisfied, DER, HMAC-MD5,
encryption, and distributed service are needed. The following code generates an instance
of a LW-SNMP class with the above specification.

typedef LW-SNMP_GENERATOR<per, with_privacy, with_hmac_md5,
distributedservice>:: RET LW-SNMP;
Chapter 4

Detailed Design and Implementation of LW-BER, MD5-Authentication, and Centralized-Service Features

This chapter concentrates on the detailed design and implementation of LW-BER, MD5-Authentication, and Centralized-Service features.

4.1 LW-BER Feature

LW-BER is derived from BER but with the elimination of the unnecessary encoding overhead imposed by BER. Due to the adoption of LW-BER, LW-SNMP has significantly reduced the network overhead for access and modification of agent MIB variables. This section will explain the objectives and approaches of LW-BER in detail.

4.1.1 Objectives

Micro-satellite networks tend to be more bandwidth-constrained than terrestrial networks. The combined constraints of power and weight make bandwidth a scarce commodity. Therefore the efficient use of limited bandwidth is an ultimate objective of LW-SNMP. However, at the time BER was chosen for SNMP, network overhead was not a real issue. The reason behind the choosing of BER was due to its availability and simplicity of implementation [30]. Nowadays, BER is considered as being fairly inefficient in terms
of network overhead. [16,17] describe this issue in detail. [31] describes several
techniques for reducing the ASN.1 encoding rule. The objective of LW-BER is to
eliminate the unnecessary encoding overhead octets caused by BER, without introducing
any ambiguity and extra computation delay.

4.1.2 Approach

A new data structure of a LW-SNMP message called LW-Msg has been designed. It is a
simplified SNMPv1 message with no community name field. The version field has been
kept in LW-Msg because not every network runs the same version of a network
management protocol. The version is important since it allows several versions of a
network management protocol to run simultaneously. The community name, a plain text
password, is transmitted between management applications, which is known as a trivial
authentication. LW-SNMP provides keyed MD5 authentication to realize true
authentication, so that the community name is no longer needed in the LW-Msg. Figure
4-1 shows the format of LW-Msg.

![LW-Msg message format](image)

As we know from Section 2.1.2, a significant portion of the ASN.1 encoded message is
occupied by the overhead octets, such as tag value and length octets. Another cause of
this overhead is the redundant transmission of an object identifier prefix, as we can see that the object identifier of all MIB variables starts at an identical prefix that is iso.org.dod.internet. In summary, the LW-BER, which is designed with the goal of reducing the network overhead, is based on the following three mechanisms:

- **Omission of encoding nested structures**

  BER transfers syntax and always has the format of a triplet TLV. All the fields – T, L, and V – are series of octets. The value V can, itself, be a triplet TLV as it is constructed. In some case, the nesting is necessary to adequately describe the data to be transferred. However, often it is possible to omit all levels of nesting during encoding in our approach, because both the sender and the receiver are aware of the structure and syntax of the transferred data. Therefore LW-BER only concerns the encoding and decoding of primitive types.

- **Omission of type**

  The type value in BER may seem to be a vital component, but it is often unnecessary since both the sender and the receiver are aware of the original data structure. Therefore LW-BER always omits type octets

- **Omission of constant object identifier prefix**

  All the potential object identifiers of our MIB begin with the same prefix (1.3.6.1); as a consequence, LW-BER removes it on encoding and adds it on the decoding.
Based on the mechanisms used to reduce network overhead, LW-BER is implemented.

In order to clearly demonstrate how LW-BER achieves its goal and to quantify the achievement in a formula, an example of how BER and LW-BER encode the same request message is illustrated in Figure 4.2 above.
In Figure 4-2 the cross-lined areas are the unnecessary octets in LW-BER; however, they are mandatory in BER. First, all the primitive type octets are eliminated. Second, all the constructed type octets and, most of the time, the constructed length octets are eliminated. Third, the community name TLV octets are all eliminated, since there are no such fields in the LW-Msg. Fourth, the object identifier type octet and part of the object identifier value octets are eliminated. Finally, if the PDU is a Get-Request PDU, then the octets of the variable value TLV are all eliminated, since the variable value is always null.

Assume that the community name is "demopublic"; the message is a Get-Request message with N variable bindings; the length of each object identifier in each variable binding is nine. The total reduced network overhead of LW-BER in bytes can then be calculated by a formula, which is, however, an approximate value because the constructed length octets of BER are value-dependent. The average value of the encoding length of a constructed type is two, since two bytes could be used to encode a length value through 128 to 32767. It is not expected that the length of a management message be beyond that range. The following factors in terms of a request and a corresponding response contribute to this achievement:

- **Reduced bytes of a Get-Request message**

  \[
  \text{<message nested encoding tag>} + \text{<message nested encoding length>} + \text{<version encoding tag>} + \text{<version encoding length>} + \text{<community name encoding tag>} + \text{<community encoding length>} + \text{<community name value>} + \text{<PDU nested encoding length>} + \text{<request id encoding tag>} + \text{<partial request id value>} + \text{<error status encoding}}
\]
tag> + <error status encoding length> + <error index encoding tag> + <error index encoding length> + <variable bindings nested encoding tag> + <partial variable binding nested encoding length> + N × (<variable binding nested encoding tag> + <variable binding nested encoding length> + <object id encoding tag> + <partial object id value> + <value encoding tag> + <value encoding length>)

=1+2+2+1+1+1+1+10+2+1+1+1+1+1+1+1+1+1+N×(1+1+1+3+1+1)

=(29+8×N) bytes

- Reduced bytes of a corresponding Get-Response message

<message nested encoding tag> + <message nested encoding length> + <version encoding tag> + <version encoding length> + <community name encoding tag> + <community encoding length> + <community name value> + <PDU nested encoding length> + <request id encoding tag> + <partial request id value> + <error status encoding tag> + <error status encoding length> + <error index encoding tag> + <error index encoding length> + <variable bindings nested encoding tag> + <partial variable binding nested encoding length> + N × (<variable binding nested encoding tag> + <variable binding nested encoding length> + <object id encoding tag> + <partial object id value>)

=1+2+2+1+1+1+1+10+2+1+1+1+1+1+1+1+1+1+N×(1+1+1+3)

=(29+6×N) bytes

Therefore the reduced network overhead of LW-BER with respect to BER is approximately: 29+8×N+29+6×N=(58+14×N) bytes (N is the number of variable bindings in a request message).
4.1.3 Algorithms Used for Numeric Representations

This sub-section provides detailed implementation information on how to encode and decode an ASN.1 integer. The algorithms and representations used in this section are derived directly from [12,15]. These algorithms are described here to support that the implementation works are self-identified.

Encoding an INTEGER value should be in a primitive form. The content octets represent a binary form of the integer, when it is positive, or its two's complement, when it is negative. The first byte is set to be zero when a positive integer is encoded on a whole number of octets whose most significant bit is set to one. To avoid redundancy, LW-BER does not permit the first nine bits to be zero- or one-filled. Therefore, to encode an integer, one must find the number of significant (nonzero) bytes in it and then copy these bytes into the encoding. The representation in LW-BER is octet-aligned; that means that if the number of bits is not a multiple of eight, then the significant bits are filled with zeroes, before two's-complementing it if negative. Figure 4-3 shows the encoding of positive and negative integers. An integer value from -128 to 127 can be encoded into a single octet, while an integer value from -32768 to 32767 can be encoded into two octets. In general, if there are b bits in the content string, integer numbers from $-2^{b-1}$ to $2^{b-1} - 1$ can be encoded [15].
positive integer \( N \)
most significant bit = 0
\((N < 128 \text{ and } N = 88)\)
\[ \begin{array}{cccccc}
0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\end{array} \]

most significant bit = 1
\((N > 128 \text{ and } N = 218)\)
\[ \begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
\end{array} \]

negative integer \( N \)
\((N = -236)\)

\(| N | \text{ in binary} \)
\[ \begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
\end{array} \]

1's-complement
\[ \begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
\end{array} \]

2's-complement
\[ \begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{array} \]

**Figure 4-3 Encoding integers.**

To decode an integer, we first need to know the value of the most significant bit. If it is one, then the decoded integer is a negative integer; otherwise, it is a positive integer. As shown in Figure 4-4, decoding a positive integer is straightforward, since all the bits except the most significant bit are the binary forms of the magnitude of the integer. The decoded value of a positive integer can be obtained through the first formula in Figure 4-4. A negative integer whose magnitude is \( X \) and for which \( X \) has a value of less than or equal to \( 2^{N-1} \) is represented by calculating \( 2^N - X \) [15]. If an \( N \)-bit sequence of binary digits \( b_{N-1} \ b_{N-2} \ b_{N-3} \ldots \ b_1 \ b_0 \) is interpreted as a two's complement representation of a negative integer, then the value can be calculated by the second formula in Figure 4-4.
4.1.4 Characteristics and Limitations

As illustrated in Section 4.1.2, the LW-BER's ability to encode a management request and a corresponding response has reduced approximately \((58 + 14 \times N)\) bytes with respect to BER. Furthermore, LW-BER has been programmed to support both big-endian and little-endian host byte orders, since there is no standard between the big-endian and little-endian byte orderings and we encounter systems that use both formats.

The current implementation of LW-BER, though it achieves its primary goal – reducing network overhead – has limited itself to encode and decode some ASN.1 primitive types. The specific primitive types supported in LW-BER are INTEGER, OBJECT IDENTIFIER, OCTET STRING, BOOLEAN, NULL, COUNTER. LW-BER is only suitable for using in a LW-SNMP environment, which is another limitation of LW-BER.
4.2 MD5-Authentication Feature

Verifying the integrity and authenticity of management information is a prime necessity in the network management protocol. In particular, a manager and an agent communicating over an insecure channel require a method by which a management message sent from the manager can be validated as authentic (or unmodified) by the agent, since the manager may alter the value of the agent. The most commonly used authentication mechanisms are based on a secret key shared between the sender and receiver, and an authentication algorithm that is a hash function to compute a digest. When the manager transmits a message to the agent, it appends to the message the digest. As a receiving party, the agent recomputes the digest, according to the received message, using the same algorithm and the shared key, then uses the digest to verify whether it equals the digest attached to the received message.

MD5 is an algorithm for carrying out the aforementioned computation task and its implementation is readily and freely available. However, the question is how best to do it, because the MD5-based hash function was not originally designed to be used for message authentication as it did not naturally accommodate the notion of a secret key [15]. Therefore special attention must be paid when using MD5 and a shared key to this end. The remainder of this section will be devoted to discussing two approaches implemented in this thesis to realize MD5-based authentication. Note that both approaches are using the underlying MD5 hash function as a "black box". The MD5 implementation used in both approaches is taken from [9], which describes the MD5
algorithm and gives a reference implementation.

4.2.1 Approach I: key-Preappended-and-Appended MD5

The first approach is to append the shared secret key before and after the data payload and to apply the MD5 hash function to the concatenation of data payload and the key. This approach is simple and does not impose any computation overhead to the MD5 algorithm. The Key-Preappended-and-Appended MD5 is depicted in Figure 4-5.

![Diagram of Key-Preappended-and-Appended MD5](image)

Figure 4-5 Structure of using Key-Preappended-and-Appended MD5.

4.2.2 Approach II: HMAC-MD5

Hashed message authentication codes for the MD5 (HMAC-MD5) authentication protocol [23] have been used in the USM of SNMPv3 as a standard approach. A 128-bit
MD5 digest is calculated in a special HMAC manner over the designated portion of a message, and the first 96 bits are included as part of the message sent to the recipient. The size of the digest carried in a message is 12 octets. The algorithm of HMAC is as follows:

\[
\text{HMAC}_k = H[(K^* \oplus \text{opad}) || H[(K^* \oplus \text{ipad}) || M]] [33]
\]

H = MD5 hash function
K = secret key
M = message input to HMAC
B = number of bits in a block
\(K^*\) = K padded with zeros on the left so that the result is b bits in length
opad = the byte 0x36 repeated b/8 times
ipad = the byte of 0x5C repeated b/8 times

Namely,

1. Append zeros to the left end of K to create a b-bit string \(K^*\)
2. Bit wise exclusive-OR \(K^*\) with ipad to produce the b-bit block \(S_i\)
3. Append M to \(S_i\)
4. Apply H to the stream generated in step3
5. Bit wise exclusive-OR \(K^*\) with opad to produce the b-bit block \(S_0\)
6. Append the hash result from step 4 to \(S_0\)
7. Apply H to the stream generated in step 6 and output the result

The above algorithm of HMAC actually pseudo-randomly generates two keys from K
through $H(K^\oplus \text{opad})$ and $H(K^\oplus \text{ipad})$. The HMAC-MD5 approach results in a slower function than the Key-Preappended-and-Appended MD5 approach, as the former requires two extra computations of the compression function (on the blocks $(K^\oplus \text{opad})$ and $(K^\oplus \text{ipad})$). In particular, the computation overhead imposed by HMAC-MD5 has a significant effect for a short data.

The above-mentioned authentication approaches have been designed and implemented in this thesis. One is simpler and faster with 16 bytes of overhead, while the other is a standard approach, which is potentially slower and with 12 bytes of overhead.

### 4.3 Centralized-Service Feature

The most important feature in the LW-SNMP is the Centralized-Service feature, as it uses the other features to provide application-programming interfaces for the management applications. It is an encapsulation of a LW-SNMP session. An LW-SNMP session is an abstraction of a logical channel, namely a connection from a management station to a managed agent. Centralized-Service is responsible for the construction, delivery, and reception of a management message. It provides a reliable service via handling timeouts and retransmissions, and it further manages all the necessary LW-SNMP resources. The Centralized-Service class is easy and safe to use, since it not only provides five basic methods for the management applications, e.g. Get-Request, Get-Next-Request, Get-Bulk-Request, Set-Request, and Get-Response services, but also provides constructors and destructors to automatically allocate and de-allocated all
needed resources. The Resources management provided by Centralized-Service minimizes the likelihood of the corruption of a leaked memory.

Moreover, the Centralized-Service provides both non-blocked and blocked modes of protocol services to the management applications. In the Non-blocked mode, namely asynchronous mode, the management applications have to handle timeouts and retransmissions. While in the blocked mode, the timeouts and retransmissions are already programmed.

4.3.1 Major Activities

In order to access or modify an agent MIB, a request must be made by invoking the GetRequest, Get-Next-Request, Get-Bulk-Request, or Set-request functions. All these functions encapsulate the activities of the protocol, which includes the processing of both an incoming message and an outgoing message. In this sub-section, we focus on these two activities to demonstrate how the LW-SNMP session processes an incoming and an outgoing message. The activity diagram is used to represent the internal behaviors of the aforementioned activities.

- Reception of a management message

Figure 4-6 shows the processing of an incoming message. When the Centralized-Service receives an incoming message, it first verifies whether the message is authentic, using the MD5-Authentication function. Upon ascertaining that it is an authentic
message, it then restores the message from a serialized stream to an LW-Msg format using the LW-BER function. In the case that the message is not authentic, the state variable, namely the number of the unauthentic message, is incremented and the message is discarded. If decoding fails, the message is discarded. Upon successful decoding, the Centralized-Service then checks the data type. If the type is a request, the message is sent from a manager to an agent. This is accomplished by the Centralized-Service first incrementing the number of snmp incoming packets and then forwarding the message to the application. There is a chance that a response message will be sent, i.e. the message is sent from an agent to a manager. Upon receiving a response message, Centralized-Service finds the request id of the response message in the outstanding queue. If a corresponding request can be found in the queue, then its corresponding request is deleted from the queue; otherwise the replayed message can be deduced and discarded. The response message is further checked with regard to its error status and error index. If the error status and error index are not equal to zero, then the entry with the index in the binding list is deleted and the message is sent to the application. If either the error status or error index is not equal to zero, the response message is discarded. If both error status and error index are equal to zero, the message is sent out.
Figure 4-6 Activity diagram of reception of a message.
• Delivery of a management message

Figure 4-7 shows the process of sending a management message. The Centralized-Service first constructs a LW-Msg based on the incoming parameters. Upon receiving a response message, the Centralized-Service encodes and authenticates the message. Finally the message is sent to the transport layer. Upon receiving a request message, the message is validated as to whether the request id is equal to zero, and the error status and error index are equal to zero. If they are not, then the message is regarded as an invalid message and discarded. Upon receiving a valid set request message, the number of bindings in this message is checked. If there is more than one binding then the message is discarded, since LW-SNMP only allows for the setting of one MIB variable at a time. If it has one binding, then the message is encoded, authenticated and finally sent to the transport layer. If the message is a get-request message, it is first assigned a unique request id and then added to the outstanding request queue. The message is finally encoded and authenticated and sent to the transport layer.
Figure 4-7 Activity diagram of delivery of a message.
4.3.2 Synchronous and Asynchronous Service Primitives

The Centralized-Service is an LW-SNMP session. It provides a method of initiating all the different types of management requests, in which each individual request performs a specific management operation. The Centralized-Service supports both synchronous and asynchronous service primitives. The remainder of this sub-section will introduce these two approaches respectively.

- **Synchronous Service Primitive**

  A synchronous service primitive allows a manager to get objects from an agent. The service primitives will block and only return when a corresponding response is received, or a timeout or error condition occurs.

  The synchronous approach allows us to map an incoming process and its associated outgoing process upon one procedure call, which results in a blocking call. In this case, the management application needs to fill the variable bindings with the request object identifiers. Returned will be the variable bindings with their values set, or explicitly what errors or exceptions have occurred. This approach is simple to implement and does not entail the extra overhead required for handling concurrency. However, if the management application wants to send another request at the same time, it needs to start another thread using the same session.

- **Asynchronous Service Primitives**
Asynchronous service primitives allow a manager to acquire objects from an agent. The service primitive returns as soon as the request message is sent. It does not wait for the response message.

The asynchronous approach, on the other hand, is more difficult to program. Since the manager process does not wait for the delivery of the message, it cannot be informed about errors or exceptions. Therefore we need to provide a callback function when initiating a LW-SNMP session in order to enable the manager to be later notified when the response message arrives. In the Centralized-Service a callback function starts in another thread and it is responsible for printing out the values of the requests.

No timeout and retransmission are supported in asynchronous mode; these have to be programmed by the management application.
Chapter 5

Performance Evaluation

This chapter investigates the performance of the LW-SNMP in comparison with UCD-SNMP-4.2.3 [28]. It covers an overview of the testbed layouts, software and hardware considerations, the data collection procedures, performance evaluation results, and analyses. The different behavior evaluations are carried out through a presentation of the variations on the metrics of protocol memory footprint, network overhead, processing time, and round trip time of UCD-SNMP and LW-SNMP.

5.1 Introduction

The experimental evaluations presented in this chapter answer the following questions:

- How large are the memory footprints of the LW-SNMP and UCD-SNMPv1 and v3?
- How much network capacity is consumed by the LW-SNMP and UCD-SNMPv1 and v3 operations?
- How much processing time is consumed by the LW-SNMP and UCD-SNMPv1 and v3 operations?
- How much time is consumed by the LW-SNMP and UCD-SNMPv1 operations under low bandwidth point-to-point links?
The remainder of this chapter is devoted to describing the testbed network used to conduct these experiments. Subsequently, the experiments designed to investigate these questions are illustrated. Finally the results and the analyses provided by deploying the experiments are discussed.

5.2 Testbed Network

The LW-SNMP is designed for managing micro-satellite networks, however, conducting experiments on real satellites is not possible at this moment. Therefore, a 100Mbps Ethernet testbed network is used to evaluate the network capacity consumed by the LW-SNMP, UCD-SNMPv1, and UCD-SNMPv3 operations, and to evaluate the computation overhead imposed by the LW-SNMP, UCD-SNMPv1, and UCD-SNMPv3 operations. A point-to-point connection is used to evaluate the round trip time of the LW-SNMP, UCD-SNMPv1, and UCD-SNMPv3 operations under low bandwidth. In addition, the LW-SNMP is based on the PIX protocol stack framework. In order to be able to compare with the standard SNMP, a test of the port of the LW-SNMP to the Linux UDP socket API has been conducted.

5.2.1 Hardware Used in the Testbed Network

All computers constructed for the testbed are Intel-based personal computers with modest performance. They are Pentium processors with clock frequencies from 300 MHz to 566 MHz. Each computer has a main memory ranging from 64 MB to 128 MB.
5.2.2 Software Used in the Testbed Network

All the computers in the testbed are installed with a Linux operating system. The LW-SNMP agent and the UCD-SNMP agent run on some hosts and the LW-SNMP manager and UCD-SNMP manager run on a different host.

UCD-SNMP-4.2.3 supports several versions of SNMP. Specifically, it supports SNMPv1, SNMPv2, SNMPv2u, and SNMPv3. In this research, only the SNMPv1 and SNMPv3 with authentication are used in the experiments to perform comparisons.

To trace messages at the manager and agent sides, tcpdump and tcptrace [29] have been used. Tcpdump precisely monitors all the network traffic and generates a single log file that is used by tcptrace to produce several different types of useful output containing information on each connection seen, such as elapsed time, bytes, and segments sent and received, retransmission, round trip times, and throughput.

5.3 Experiments, Results, and Discussions

The primary purpose of these experiments is to compare the LW-SNMP with authentication to UCD-SNMPv1 and UCD-SNMPv3 with authentication. Several other scenarios have been included to show perspective. The following sub-sections examine each of the aforementioned questions and provide discussions on the results obtained.
5.3.1 How large are the memory footprints of the LW-SNMP and UCD-SNMP?

The memory footprint of a current running protocol could be obtained in Linux through firstly using the "ps" command in order to obtain the desired process id and then using the "top -p PID" command to read the memory footprint. "top" provides an ongoing look at processor activity in real time. Table 5-1 shows the memory footprint of an LW-SNMP, a UCD-SNMPv1, and a UCD-SNMPv3.

<table>
<thead>
<tr>
<th>Management Protocol</th>
<th>Memory Footprint (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW-SNMP with authentication</td>
<td>686</td>
</tr>
<tr>
<td>UCD-SNMPv1</td>
<td>1020</td>
</tr>
<tr>
<td>UCD-SNMPv3 with authentication</td>
<td>1020</td>
</tr>
</tbody>
</table>

Table 5-1 Memory footprint of network management protocols.

5.3.2 How much network capacity is consumed by the LW-SNMP and UCD-SNMP operations?

The network capacity consumed by a LW-SNMP operation is examined by having a manager poll an agent on the same subnet. Scenarios consisting of retrieving string and counter MIB variables are designed. The IP packets generated by the Get-Request and Get-Response operations are captured by the tcpdump application running on the manager station.
The first experiment consists of issuing a Get-Request operation to retrieve the sysDescr MIB variable in the system group. The sysDescr variable is of the display string type and provides a description of the entity, such as the hardware and the operating system. The agent obtains this MIB variable through reading from a file. The exact sysDescr in this experiment is "Linux localhost.localdomain 2.4.8-26mdk #1 Sun Sep 23 17:06:39 CEST 2001 i686" with the length of 77 bytes. The result of this experiment is shown in Table 5-2. This table presents the sizes (in bytes) of IP packets carrying Get-Request messages, the sizes of the associated Get-Response messages, and the total sizes of the request and response messages. The network capacity consumed by the LW-SNMP with authentication is less than that of UCD-SNMPv3 with authentication by approximately 45%, and less than that of UCD-SNMPv1 by approximately 14%. Note also that the LW-SNMP with authentication adds a total of 32 bytes of authentication overhead to the request and response messages, while UCD-SNMPv3 with authentication typically adds a total of 150 bytes of overhead for a request and response operation.

<table>
<thead>
<tr>
<th>Network Management Protocols</th>
<th>Get (bytes)</th>
<th>Response (bytes)</th>
<th>Total (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCD-SNMPv1</td>
<td>75</td>
<td>152</td>
<td>227</td>
</tr>
<tr>
<td>LW-SNMP with authentication</td>
<td>58</td>
<td>137</td>
<td>195</td>
</tr>
<tr>
<td>UCD-SNMPv3 with authentication</td>
<td>140</td>
<td>217</td>
<td>357</td>
</tr>
</tbody>
</table>

Table 5-2 IP packet sizes in bytes for request operations with a string value.

To accomplish the network management tasks, the manager may issue a Get-Request operation with more than one variable binding. In order to thoroughly evaluate the achievements of the LW-SNMP in terms of network overhead, eight other experiments
are conducted. These eight experiments have the same scenario as the first experiment, but with 10, 20, 30, 40, 50, 60, 70, and 80 variable bindings. The variable bindings are all octet strings. The average length of the values in the binding list is 77 bytes. To better indicate how these scenarios compare, Figure 5-1 shows the ratio of the total number of bytes transmitted for each scenario to the total number of bytes transmitted using the SNMPv1 approach.

Figure 5-1 illustrates how the number of variable bindings affects the ratios. Observe how for small values of number of bindings, UCD-SNMPv3 with authentication consumes considerably more network capacity than UCD-SNMPv1. Note also that as the number of bindings or the application-layer payload increases, the two versions of UCD-SNMP perform in a more similar manner. For instance, when retrieving 20 variable bindings, UCD-SNMPv3 with authentication only requires 6% more network capacity than UCD-SNMPv1. The reason for this is that UCD-SNMPv3 with authentication adds constant bytes of overhead for security and access control. This portion of the overhead is amortized by the large size of the messages.
Figure 5-1 also shows that the reduced network overhead of the LW-SNMP with authentication is not significantly affected by the number of variable bindings. In the case of one variable binding, the LW-SNMP with authentication consumes 86% of the network capacity that would be consumed by UCD-SNMPv1. As the number of variable bindings increases, the ratio increases very slowly. The reason for this is that the LW-SNMP with authentication not only reduces a constant number of bytes for each management message, but also reduces the number of bytes in each variable binding.

Besides the octet string type, Counter32 is another commonly used type in MIB. The Counter32 uses at most 4 bytes to represent an integer value. Another set of experiments clarifies how the number of variable bindings with relatively short value affects the ratios. The first experiment is designed to enable the manager to retrieve one variable binding with an integer type. The variable binding is the number of UDP received
packets. Table 5-3 shows the sizes (in bytes) of IP packets carrying SNMP-Get messages, the sizes of the associated SNMP-Response messages, and the total sizes of the request and response messages.

<table>
<thead>
<tr>
<th>Management Protocol Type</th>
<th>Get (bytes)</th>
<th>Response (bytes)</th>
<th>Total (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCD-SNMPv1</td>
<td>75</td>
<td>77</td>
<td>152</td>
</tr>
<tr>
<td>LW-SNMP with authentication</td>
<td>58</td>
<td>62</td>
<td>120</td>
</tr>
<tr>
<td>UCD-SNMPv3 with authentication</td>
<td>140</td>
<td>142</td>
<td>282</td>
</tr>
</tbody>
</table>

Table 5-3 IP packet sizes in bytes for request operations with an integer value.

In the above table, the network capacity consumed by the LW-SNMP with authentication is less than that of UCD-SNMPv3 with authentication by approximately 57%, and less than that of UCD-SNMPv1 by approximately 21%. UCD-SNMPv3 with authentication consumes as much as 86% more network capacity than UCD-SNMPv1.

The results of the other eight experiments with 10, 20, 30, 40, 50, 60, 70, and 80 variable bindings performed through retrieving integer variables are shown in Figure 5-2.
Figure 5-2 Ratios over various numbers of integer-typed variable bindings.

Figure 5-2 illustrates how the number of variable bindings affects the ratios. Observe that for small values regarding the number of bindings, UCD-SNMPv3 with authentication requires significantly more network capacity with respect to UCD-SNMPv1. Note also that as the number of variable bindings increases, the two versions of UCD-SNMP perform in a more similar manner. The reason for this is that the network overhead imposed by UCD-SNMP with authentication due to security and access control is amortized by the large size of the message payload. Note also that in Figure 5-2, as the number of variable bindings increases, the LW-SNMP with authentication saves more network overhead with respect to UCD-SNMPv1.

Figure 5-1 and Figure 5-2 show that the LW-SNMP with authentication significantly reduces the network overhead with no regard for the length of the message and the type of the message. However, as the number of variable bindings increases, the saved
network overhead may increase or decrease depending on the average size of the values in the variable bindings. Assume that the average length of the values in the variable bindings is \( L \) and the number of variable bindings is \( N \); then, according to the formula obtained in Section 4.1.2, the saved network overhead in bytes of LW-SNMP with authentication with respect to UCD-SNMPv1 is \( 58 + 14 \times N - 32 \). The network overhead in bytes imposed by SNMPv1 is \( 72 + 28 \times N + L \times N \), inferred from Section 4.1.2. On top of that, we need to add 56 bytes that correspond to the UDP and IP headers. The resulting number of bytes is the IP packet size of UCD-SNMPv1. Therefore, the ratio of the reduced network overhead of LW-SNMP over UCD-SNMPv1 with \( N \) variable bindings is:

\[
\frac{58 + 14 \times N - 32}{72 + 28 \times N + L \times N + 56} \quad (f_N)
\]

The ratio of LW-SNMP reduced network overhead over UCD-SNMPv1 with \( N+1 \) variable bindings is:

\[
\frac{58 + 14 \times (N+1) - 32}{72 + 28 \times (N+1) + L \times (N+1) + 56} \quad (f_{N+1})
\]

We would like to find the average length of the value \( L \), so that given \( L \), no matter what the value of \( N \) is, \( f_N > f_{N+1} \) holds.

\[
\frac{58 + 14 \times N - 32}{72 + 28 \times N + L \times N + 56} > \frac{58 + 14 \times (N+1) - 32}{72 + 28 \times (N+1) + L \times (N+1) + 56}
\]

After reducing the inequality, we obtain:
13 \times L > 532

L > 41

Now, we find that the average length of the variable bindings is greater than 41 bytes, and the reduced network overhead decreases as the number of variable bindings increases. Conversely, we find that if the average length of the variable bindings is less than 41 bytes, then the reduced network overhead increases as the number of bindings increases.

5.3.3 How much processing time is consumed by LW-SNMP and UCD-SNMP operations?

The processing time of a network management operation depends on the following four factors:

i. the processing time to generate a request at the management station

ii. the processing time at the agent to interpret the message

iii. the processing time at the agent to generate the response

iv. the processing time at the manager to receive and interpret the response.

Typically an agent generates a response either through files, memory or system calls. In order to fully evaluate the processing time consumed by the network management operations, three test cases have been designed. The first one is to choose a request whose response needs to be fetched from a file, like sysContact. The second one is to choose a request whose response needs to be fetched from memory, like the number of
SNMP received packets. The last one is to choose a request whose response needs to be fetched from system calls, like sysUpTime.

In this experiment, the manager polls the agent with one variable binding. The elapsed time from the time when the manager sends a request until it gets a response is called the round trip time. We use tcpdump to capture the management traffic. In order to measure the processing time used by the network management protocol, first we use tcpdump to measure the number of bytes transmitted by the network management protocol. Secondly, we develop a simple application right above UDP, which sends the same number of bytes as the network management protocol. With this simple application, we are able to obtain the network transmission delay. Finally, the processing time can be obtained by subtracting the round trip time from the network transmission delay. The results are depicted in Figure 5-3.

Figure 5-3 Processing time used by network management protocols.
The above figure demonstrates that the processing time required by network management protocols depends on the level of security and the mechanism used to implement the security, as well as the location in which the agent obtains the data. Typically, reading from a file is the most time-consuming, compared with reading from a system call or reading from memory. The time required in reading from a file involves the time required in invoking system calls and reading from memory. Reading from a system call requires more time than reading from memory because the context switch time between the kernel space and the user space needs to be counted. Of all the groups in the figure, the processing time required by the LW-SNMP without authentication is the shortest, whereas UCD-SNMPv3 with authentication requires the longest time to process. The LW-SNMP without authentication saves about 26% to 50% processing time with respect to UCD-SNMPv1. This is mainly due to the relatively simple encoding rule and the simplified agent application. Actually, the agent should be designed to trace the request variables into a MIB tree, and after finding where to get the value, the agent goes to fetch the value. However, the development of a desired agent is in the process of being written up by another student; the agent used in this thesis is very simple and for testing only. The processing time of the LW-SNMP with Key-Preappended-and-Appended MD5 authentication requires almost the same time as UCD-SNMPv1, and requires only half the time of UCD-SNMPv3 with authentication. However, the LW-SNMP with HMAC-MD5 authentication requires more time than UCD-SNMPv1 due to the complex computation required by the HMAC-MD5 algorithm. The processing time of the LW-SNMP with HMAC-MD5 authentication is still 20% shorter than that of UCD-SNMPv3 with authentication.
In the further estimation, the processing time required by the LW-SNMP and UCD-SNMP ranges from approximately 100 μs to 550 μs, which is significantly less than the latency in low data rate networks. A critical factor affecting the latency is the transmission delay. Transmission delay is measured by sending a request that is returned to the sender and the round trip time (RTT) is considered to be the transmission delay. In order to measure the transmission delay of the LW-SNMP, a set of experiments have been deployed in the following sub-section.

5.3.4 How much time is consumed by the LW-SNMP and UCD-SNMP operations under low bandwidth point to point links?

The experiment designed to answer this question is based on a serial Point to Point Protocol (PPP) connection. PPP is a mechanism for creating and running IP and other network protocols over a serial link. A standard Ethernet connection operates at 100 Mbs (million bits per second) maximum theoretical throughput, whereas a PPP connection can operate at wide-ranging data rates. The directed PPP connection uses a null-modem cable to establish the testbed, by which the simulation of the direct link between an earth station and a space satellite is carried out. The PPP daemons, on both peers, realize the physical link by setting the same data rate. The specific setup procedure is as follows:

- start pppd on the manager side: pppd defaultroute /dev/cua0/ 57600
  134.117.5.198:134.117.5.248 nobsdcomp nodeflate
• start pppd on the agent side: pppd defaultroute /dev/cua1/ 57600
   134.117.5.248:134.117.5.198 nobsdcomp nodeflate

Please note that if not nobsdcomp and nodeflate are not explicitly specified, the PPP daemon deploys the default compression mechanism.

The response time that is regarded approximately as transmission delay over different bandwidths has been achieved through setting different baud rates of a PPP connection. The baud rate can be used interchangeably with bit per second only if the modulation techniques used transmit one bit with each baud, which is the case in this experiment. Moreover, the default time out of UCD-SNMP has been changed from 1000000 µs to 20000000 µs.

The bandwidths of 57.6 Kbps, 38.4 Kbps, 19.2 Kbps, 9.6 Kbps, and 4.8 Kbps have been chosen to measure the response time of a request. In order to show the perspective three scenarios have been conducted and the results are depicted in Figures 5-4, 5-5, and 5-6 for the response time of retrieving one variable, 10 variables, and 50 variables in one Get-Request operation, respectively.
Figure 5-4 Response time of a request with one variable over different bandwidths.

Figure 5-5 Response time of a request with 10 variables over different bandwidths.
Figure 5-6 Response time of a request with 50 variables over different bandwidths.

Figures 5-4, 5-5, and 5-6 show that as the data rate decreases, the response time of each network management protocol increases. In all of our experiments, the LW-SNMP without authentication requires the shortest time to get a response, followed by the LW-SNMP with authentication, UCD-SNMPv1, and finally by UCD-SNMPv3 with authentication. This behavior is due to the network capacity consumed by each network management protocol. Note also that as the number of variable bindings increases, the response time difference between the LW-SNMP without authentication and LW-SNMP with authentication are less significant. Observe from Figure 5-6 that the response times of the LW-SNMP without authentication and the LW-SNMP with authentication are almost the same. The reason for this is that the LW-SNMP with authentication always transmits 32 bytes more than the LW-SNMP without authentication. This number of bytes is amortized by large variable bindings. The same reasoning can be applied to UCD-SNMPv1 and UCD-SNMPv3 with authentication.
As stated before, the processing time required by the LW-SNMP with key-Preappended-and-Appended MD5 authentication and UCD-SNMPv1 are almost the same. However, from Figure 5-4, we observe that the LW-SNMP with key-Preappended-and-Appended authentication requires up to 60 ms less than UCD-SNMPv1 for retrieving one MIB variable. The LW-SNMP without authentication requires 120 ms less than UCD-SNMPv1 under a 4.8 Kbps connection. The LW-SNMP with Key-Preappended-and-Appended MD5 authentication requires 320 ms less than UCD-SNMPv3 with authentication for retrieving one MIB variable under a 4.8 Kbps connection. The reason for these improvements is that the LW-SNMP retrieves the same MIB variable as UCD-SNMP; however, the former requires fewer bytes than UCD-SNMP.
Chapter 6

Conclusions

This chapter summaries the research and identifies the highlights and limitations of this thesis research. It closes with a discussion on the perspectives of further development and deployment of the ideas proposed in this thesis.

6.1 Summary

In this thesis, network management protocols were investigated for their use as tools for managing micro-satellite networks and it was found that the LW-SNMP is well suited for this task. The LW-SNMP is especially useful in the case where a low communication data rate, power constraints, and a resource-constrained onboard computer are present.

In this research, efforts have been made in multiple aspects around the LW-SNMP design, protocol improvement, implementation, and performance analysis. The research can be summarized as follows:

- Design the LW-SNMP to improve the limitations and shortcomings of the current SNMP to make the LW-SNMP suitable for the micro-satellite environment.
- Implementation of the LW-SNMP using a GP approach under the PIX environment.
• Port the LW-SNMP from a PIX environment to an environment with standard UDP sockets.

• Construction of several testbeds for the performance evaluations.

• Performance comparisons and analyses against the standard SNMPv1, SNMPv3, and LW-SNMP.

The extensive qualitative analysis revealed that the employment of the LW-SNMP provides several advantages over the employment of the standard SNMPv1 and SNMPv3. The specific advantages are as follows:

• The LW-SNMP offers 60 potential configurations in terms of encoding rules, authentication models, encryption models, and management architectures. With the specifications of what is need to be assembled, e.g. a particular encoding rule, a particular authentication model or no authentication, with encryption or no encryption to determine the inclusion of authentication/encryption models, a centralized or distributed network management architecture, the LW-SNMP generator returns a desired LW-SNMP class with all the selected features implemented. My research has provided three different configurations, namely: the LW-SNMP with a lightweight basic encoding rule, no encryption, no authentication, and with a centralized architecture; the LW-SNMP with a lightweight basic encoding rule, no encryption, with Key-Preappended-and-Appended MD5 authentication, and with a centralized architecture; the LW-SNMP with a lightweight basic encoding rule, no encryption, with HMAC-MD5 authentication, and with a centralized architecture.
- The LW-SNMP with a lightweight basic encoding rule, with authentication, without encryption, and with a centralized architecture is suitable for managing micro-satellite networks because it has reduced around 49% of the memory footprint with respect to UCD-SNMPv3 with authentication. The MIB variable bindings in one request range from one variable binding to 80 variable bindings, and the network traffic reduced by the LW-SNMP with the specified features is between 14% and 43% with respect to UCD-SNMPv1, without any extra overhead on computation being introduced. In fact, the LW-SNMP has reduced computation overhead with Key-Preappended-and-Appended MD5 authentication mechanism with respect to UCD-SNMPv1. Finally, the LW-SNMP has reduced response time by approximately 17% with respect to UCD-SNMPv1, and 23% with respect to UCD-SNMPv3 with authentication for retrieving a large number of bindings.

6.2 Future Work

Future work in this thesis research could focus on at least three aspects. First, extend LW-BER to support more ASN.1 types, such as BIT STRING, IPAddress, NsapAddress, Counter64, and UIInteger32, as well as to provide a timeout and a retransmission mechanism for the asynchronous mode of services.

The current LW-SNMP, consisting of eight components, provides 60 valid configurations; however, not all of the components have been implemented. Therefore, the second aspect would be to implement some or all of the unimplemented components.
These components include DER, LWER, Encryption, Centralized-Service-With-Trap, Distributed-Service, and Distributed-Service-With-Trap.

Finally, the LW-SNMP only provides configurability in a static manner (at compile time), and an extension would be to implement the LW-SNMP using a dynamic manner. The ability to configure the LW-SNMP dynamically would not only allow the LW-SNMP to be initially configured at compile time but also then reconfigured at run time without having to shutdown and restart the system, and thereby having no user impact.
Bibliography


[28] Net-SNMP project download site

[29] Tcptrace download site and Tutorial
   http://irg.cs.ohiou.edu/software/tcptrace/tcptrace.html


[37] Information concerning the embedded configurable operating system (eCos)
   http://sources.redhat.com/ecos/