Faculdade de Engenharia da Universidade do Porto

IP Traffic Measurements between IP Service Providers(ISP)

Jorge Filipe Gomes da Silva

Dissertation developed in the framework

of Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major in Telecommunications

Supervisor: José António Ruela Simões Fernandes (PhD)

Co-supervisor: Filipe Miguel Sousa (Eng)

May, 2008

Abstract

The existing Internet evolved from the ARPANET, an experimental data network founded by U.S. Defense Advanced Research Projects Agency (DARPA) in 1960s. The entire technology is supported by a Datagram model, where each packet carries the complete source and destination address and is individually and independently delivered to its destination through the network.

The adoption of the TCP/IP Protocol took place in mid-1980s. This evolution became crucial to inter-connect different networks in the world and to render a universal service to applications.

The current Internet service is often referred to as a best effort. Best effort service is the simplest type of service that a network can offer; this service does not provide full reliability, since the delivery of packets is not guaranteed.

However, best effort service is not the best service to support current applications, such as real-time and multimedia applications, because this service does not guarantee packet delivery to the destination and the packets can have a large delay variation.

Nowadays, the Internet is the world's largest public network. The web also serves as a platform for the deployment of new applications such as streamed multimedia services, Voice over IP (VoIP), peer-to-peer (P2P), and real time applications.

This evolution of the Internet has brought new challenges. Many of the applications need a very good network performance to support their service requirements. So it was defined the following objective: Quality of Service (QoS) for Internet. Internet QoS is defined as a measurable level of service delivered to network users, which can be characterized by a set of metrics (e.g., packet loss probability, delay, jitter or delay variation, available bandwidth, throughput). Such QoS can be provisioned by network service providers in terms of a service agreement (e.g., a Service Level Agreement or SLA) between subscribers and providers. A QoS mechanism is a set of protocols designed to endow network devices to serve the competing applications in Internet differently by following a set of predefined policies. The role of policies in a QoS mechanism is to determine when a network device should serve packets of a particular application in the presence of packets from other contending applications. Traffic measurements give a very good perspective of what is going on all over the network.

In Europe measurement between ISP's are performed by a non-profit center: RIPE NCC. These measurements could be used to supervise and monitor intra and inter-domain traffic.

RIPE NCC (Réseaux IP Européens / NCC Network Coordination Center) offers the service of network management known as Test Traffic Measurements (TTM). It allows to comprehensively and continuously monitoring the connectivity between specific networks and with other parts of the Internet. Measurements carried out with TTM includes one way delays between hosts (latency), packet losses, path information ("traceroute"),

bandwidth, delay variation (jitter). But this system implies Internet Services Providers (ISP) to install and configure their own hardware.

The main purpose of this thesis is to implement and design the NSIS application M-NSLP. The M-NSLP was designed to perform measurements between two different network domains. This application is able to configure an export protocol to export metering reports to a collector from nodes located between end-hosts.

Resumo

A Internet existente hoje em dia evoluiu a partir da ARPANET, uma rede de dados experimentais fundada por E.U. Agência de Projectos de Pesquisa Avançada da Defesa (DARPA) em 1960. A tecnologia é suportada por todo um modelo baseado em datagramas, onde cada pacote carrega o endereço completo de origem e de destino, cada pacote é único, e independentemente entregue até ao seu destino através da rede.

A aprovação do protocolo TCP / IP teve lugar em meados dos anos 80. Esta evolução foi crucial para inter-conectar diferentes redes em todo o mundo e para tornar um serviço de aplicações, universal.

O serviço actual de Internet é muitas vezes referida como um serviço "'best effort"'. Este serviço é o tipo mais simples que uma rede pode o ter, este serviço não oferece segurança, a retransmissão dos pacotes é limitado, e a entrega dos dados não é garantida. No entanto, o serviço "'best effort"' não é o melhor para suportar as actuais aplicações, tais como aplicações em tempo-real e aplicações multimédia, uma vez que este serviço não garante entrega pacotes para o destino e os pacotes podem ter uma variação atraso grande, e poderá inviabilizar a sua utilização.

Actualmente, a Internet é a maior rede pública do mundo. A web também serve como uma plataforma para a implantação de novas aplicações, tais como serviços de streaming multimédia, voz sobre IP (VoIP), peer-to-peer (P2P), e aplicações em tempo real. Esta evolução da Internet trouxe novos desafios. Muitas das aplicações precisam de um bom desempenho da rede para apoiar os serviço requeridos. Portanto, foi definido o seguinte termo: Qualidade de Serviço(QoS) para a Internet. QoS na Internet é definida como um nível mensurável da qualidade de serviço prestado aos utilizadores de uma rede, que pode ser caracterizado por um conjunto de métricas (por exemplo, pacotes perdidos, probabilidade de atraso, jitter(variação do atraso), largura de banda disponível, e capacidade da rede). A Qualidade de serviço, QoS, pode ser assegurado por prestadores de serviços de rede,com acordos de serviço (por exemplo, um acordo de nível de serviço ou SLA) entre os assinantes e os fornecedores.

Um mecanismo de QoS é um conjunto de protocolos concebidos para dotar o dispositivo de uma rede, para que este trate as aplicações concorrentes da Internet de maneira diferente, seguindo um conjunto de predefinido de políticas. O papel das políticas num mecanismo de QoS é a de determinar quando um dispositivo de rede deve processar pacotes de uma aplicação específica, na presença de outros pacotes. As medições de tráfego dão uma boa perspectiva do que se passa em toda a rede.

Na Europa as medição entre ISP'são realizadas por uma entidade sem fim lucrativos chamada RIPE NCC. Essas medidas realizadas pelo RIPE são utilizadas para supervisionar e monitor tráfego intra e inter-domínios.

RIPE NCC (Réseaux IP Européens / NCC Network Coordination Center) oferece o serviço de monitorização de rede, este serviço é conhecido como Medições de tráfego

de teste(Ttm). Este serviço permite perceber de uma melhor forma a performance da Internet. As Medição realizadas por TTM incluem medições one-way delay, atrasos entre hosts (latência), pacotes perdidos, traceroute, largura de banda, e variação atraso (jitter). Contudo este sistema implica que os Internet Services Providers (ISP) sejam possuidores deste hardware, que apenas é instalado e configurado por eles.

O principal objectivo desta tese é o desenvolvimento de uma aplicação para o NSIS, a aplicação M-NSLP. O M-NSLP foi concebida para efectuar medições entre diferentes domínios e diferentes redes (ISP). Esta aplicação é capaz de configurar um protocolo de exportação, para exportar os relatórios de medições, gerados por qualquer nó que participe num processo de medição.

Acknowledgment

I owe my thanks to my supervisors, Prof. Dr. José António Ruela Simões Fernandes and Eng. Filipe Miguel Sousa, for their support, guidance and availability but also for their patience, in corrections and questions.

I am grateful to all my colleagues for their support, and all my friend's that always gave me the strength that i needed.

Finally, I would like to express my gratefully to my parents, for a lifetime of devotion to me and my education.

Jorge Filipe Gomes da Silva

Contents

List of Figures

List of Tables

ACRONYMS

AF Assured Forwarding

API Application Programming Interface

AS Autonomous System

ASN Autonomous System Number

BGP Border Gateway Protocol

BPF BSD Packet Filter

BSD Berkeley Software Distribution

CAIDA Cooperative Association for Internet Data Analysis

CPOC Configuration Point of Contact

DAC Data Analysis Client

DCCP Datagram Congestion Control Protocol

DiffServ Differentiated Services

EF Expedited Forwarding

ETSI European Telecommunications Standards Institute

GIST General Internet Signaling Protocol

GPS Global Positioning System

HPC High Performance Connection

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

INESC Instituto de Engenharia de Sistemas e Computadores

IntServ Integrated Services

IP Internet Protocol

IPMP IP Measurement Protocol

IPPM IP Protocol Measurement

ISO International Organization for Standardization

ISP Internet Service Provide

ITU International Telecommunications Union

IXP Internet eXchange Point

LAN Local Area Network

MA Messaging Association

MC Measurement Cliene

MGEN Multi-Generator

MIB Management Information Base

MNE Metering NSIS Entity

MNI Metering NSIS Initiator

MNR Metering NSIS Receiver

MRI Message Routing Information

MRS Message Routing State

NAT Network Address Translator

NCC Network Coordination Center

NIMI National Internet Measurement Infrastructure

NLANR National Laboratory for Applied Network Research

NPD Network Probe Daemon

NSF National Science Foundation

NSIS Next Steps in Signaling

NSLP NSIS Signaling Layer Protocol

NTLP NSIS Transport Layer Protocol

P2P Peer to Peer

PHB Per-Hop-Behavior

PSAMP Packet Sampling

QNE QoS NSIS Entity

QNI QoS NSIS Initiator

QNR QoS NSIS Receiver

QoS Quality of Service

QSPEC QoS Specification

RAO Router Alert Option

RII Request Identification Information

RIPE Réseaux IP Européens

RMF Resource Management Function

RMON Remote Network Monitoring

RSN Reservation Sequence Number

RSVP Resource Reservation Protocol

RTFM Real time Traffic Flow Measurement

RTT Round Trip Time

SCTP Stream Control Transmission Protocol

SID Session Identifier

SII Source Identification Information

SLA Service Level Agreement

SNMP Simple Network Management Protocol

TCP Transmission Control Protocol

TLS Transport Layer Security

TTL Time-To-Live

TTM Test Traffic Measurements

UDP User Datagram Protocol

VoIP Voice over IP

Chapter 1

Introduction

The current Internet service is often referred to as a best effort. Best effort service is the simplest type of service that a network can provide, this service does not offer full reliability, the retransmission of packet is limited, and their delivery is not guaranteed. This service, supported by IP Networks, isn't bad for traditional Internet applications like web and mail, but nowadays, Internet has much more advanced applications, such as, audio, and video streaming. This kind of Internet applications demand high data throughput capacity and have low-latency requirements.

To solve such problems, it was necessary to develop new protocols for signalling, to perform measurements for QoS control.

Few years ago, the Next Steps in Signaling working group was formed in the IETF to create a new signaling framework flexible enough to fit both current and future needs.

The NSIS framework consists of two layers, the NSIS Transport Layer Protocol (NTLP) is known as the General Internet Signaling Transport(GIST), and the NSIS Signaling Layer Protocol (NSLP). The NSLP Layer has tree well known Internet applications such as QoS reservation (QoS-NSLP), Metering (M-NSLP), and Network Address Translation and Firewall (NAT/FW NSLP).

The performance of the network can be measured with parameters like bandwidth, delay, jitter, and loss. These parameters can be measured actively by insertion of test traffic or passively by observing user generated traffic.

In this thesis, we will describe the M-NSLP specifications and Implementation. This work was developed to perform IP Network measurements using signaling protocol.

1.1 Goals

The main goal of this thesis is to optimize and expand an existing specification of NSIS, developed at Designed and Analysis of Communication Systems (DACS) group [2], in such

a manner that Internet Services Providers may use it to perform dynamic Measurements between different domains. In particular the objectives for the thesis are:

- Study the current specifications of existing export protocols.
- Overview of current specifications of existing tools for passive and active measurements.
- Characterize the current specifications and implementation of the NSIS protocol suite and in particular the Metering-NSLP protocol.
- Design and implement a Metering application API that can be used between NSIS and applications providing real-time traffic generator(MGEN).
- Define Metering Specification (MSPEC) object, in order to perform MGEN configuration, measurement schedule, and export measurements to a collector.
- Setting up a demonstration of a Measurement using a Testbed Platform.

The NSIS application M-NSLP was designed to perform exactly what was proposed, allowing dynamic end-to-end measurements, in IP Networks.

1.2 Implementation Requirements

To reach the objectives set in the previous section, the implementation should follow a number of requirements:

- The test of Metering-NSLP prototype should follow the specification as strictly as possible.
- Implementation and design should be done to respect the scalability feature. This means that the implementation must be structured in such a way that future changes and extensions of the protocol and even unimplemented functionality can be easily incorporated.
- The implementation should be both portable and configurable; this means that it should be able to run easily on any open source distributions.

1.3 Definitions

In this section we shall describe the basics definitions as Autonomous System (AS), and Internet Service Provider (ISP) to provide an adequate understanding of the problem.

Autonomous System is defined as a group of IP networks and router under control of one or more ISP's. An AS provides connections through itself to its neighboring networks,

using BGP (Border Gateway Protocol). Every AS is associated with a unique AS number (ASN) which is assigned by the Internet Assigned Numbers Authority (IANA). The ASN is also used for the exchange of exterior routing information between neighboring ASs. AS numbers are divided into two ranges. The first range, from 1 to 64511, contains public AS numbers that may be used on the Internet. The second range, from 64512 to 65535, is reserved for private use and should not be advertised.

Internet service providers (ISPs) [3] provide Internet access and related services to their customers who generally have to pay a monthly access fee.

An ISP houses a collection of routers in a so called Point of Presence (POP). Usually different ISPs tend to have routers in the same building which is called an Internet eXchange Point (IXP). IXPs allow Internet service providers to exchange traffic with each other. There can be distinguished three types of peering agreements among ISPs:

- Customer-to-Provider
- Provider-to-Customer
- Peer-to-Peer

In this thesis we will use definitions provided by ITU-T [4], to characterize measurements methodologies and architectures. We will define all components that take part in the measurement operations.

- A **host** is a system that communicates using any adequate Internet protocol; normally a host can be called a end-system.
- A **router** is a node that enables communication between other nodes by forwarding IP packets based on the content of their IP destination address field.
- A **source host** is a host and a complete IP address where end-to-end IP packets originate. In general a host may have more than one IP address.
- A **destination host** is a host and a complete IP address where end-to-end IP packets are terminated. In general a host may have more than one IP address.
- A **link** is a point-to-point (physical or virtual) connection used for transporting IP packets between a pair of hosts or routers between two adjacent IP systems.

All definitions provided by IETF (IPPM working group) are equivalent to ITU ones. The IPPM definition is provided by [5].

1.4 Methodology of Dissertation Work

All work done in this thesis, was based on a previous research work to provide a large overview over existing technologies and methodologies to perform measurements.

The whole work implied the research and analysis of scientific texts that provided a great background for the state of the art analysis and a good knowledge of existing measurement tools, and their metrics. In order to avoid that the dispersiveness of the research work, a plan was outlined and regular meetings with the supervisors were realized. The work was previously planned and designed to respect deadlines. The purpose of this thesis is to provide a helpful overview of the architecture of the developed software.

1.5 Thesis Structure

This thesis proceeds as follows. Sec. 2 will provide an overview of the main objectives for all subsequent work, such as methodologies, tools selections, and analysis as well of the NSIS protocol. In Sec. 3, the state of the art it is discussed. In Sec. 4, it is described the Metering-NSLP Signaling Layer Protocol (NSLP) implementation, the upper layer metering functionality of NSIS. Sec. 5 provides a description of the experiments performed to evaluate the functionality of the developed software. The last sections provide the conclusions of this thesis.

Chapter 2

Requirement Analysis

2.1 Characterization

In Telecommunication, to characterize good Internet performance, the word QoS - Quality of Service is used. Besides QoS, others applications related with traffic measurement will be considered, such as Traffic Engineering, Traffic Profiling, Usage-based Accounting, and Attack/Intrusion detection, but for now we will focus on QoS measurements. QoS already forms an important part of the studies being carried out by the International Standards Organization (ISO), European Telecommunications Standards Institute (ETSI), Internet Engineering Task Force (IETF), and the International Telecommunications Union (ITU). ITU-T E.800 [6] specification defines quality of service as "the collective effort of the service performance which determines the degree of satisfaction of the end-user".

This specification describes the relationship between two main classes: User oriented and Network oriented QoS. The main components that determine the degree of satisfaction of the end-user are the operability and serviceability of service. Another important aspect is security. As far as Network QoS, the performance aspects are related with resources and facilities, and integrity. The integrity of user information is a crucial aspect of QoS. The network should provide a security system.

The Internet architecture has evolved as a result of work carried out by working groups from ITU-T and IETF. IPPM Working group, from IETF, defined most of all measurement parameters. In [5] is specified the general framework for QoS measurement within IP networks. This important RFC defines the fundamental concepts of metric and measurement methodology and its entire framework. Metrics can be characterized by the following types: singleton, sample and statistical.

A singleton metric is defined as an atomic metric resulting from measures made on one or several non-sampled datagram's.

Sample metrics derived from a set of singleton metrics by selecting a number of instances of these metrics.

Finally, statistical metrics derived from a sample metric by computing some statistics on the singleton metrics located in the sample. For ITU-T X.140 [7] the main aspects of QoS are access delay, incorrect access probability, access denial probability, user information transfer delay, user information transfer rate, user information error probability, extra user information delivery probability, user information misdelivery probability, user information loss probability, disengagement delay, disengagement denial probability, service availability, user information transfer denial probability and, finally, service outage duration in case of data network; all X.140 is user-oriented definition.

2.2 Measurements

Measurements can be required for different applications to measure IP Flow information export. This kind of export information can be characterized as Usage-based accounting, traffic profiling, traffic engineering, attack/intrusion detection and finally QoS monitoring [8].

- **Usage-based Accounting** is a business model for selling IP-services, this task of accounting is done by whoever provides the IP service, in general ISPs.
- **Traffic Profiling** is a process of characterization of IP flow, that provide better understanding of network in analysis.
- **Traffic Engineering** is a concept of modeling, and optimization of network utilization.
- **Attack/Intrusion detection** is a concept, characterized for capturing, and analysis flow information. The analysis of data network, and its statistical processing, can give some information about attack/intrusion detection.
- **QoS Monitoring** is the passive measurement of quality for IP flows.

2.3 Measurements Techniques

Measurements can be characterized as active and passive, active Measurements are characterized by the insertion of artificial traffic into the current network traffic to test the network response. Common methods are the measurement of Round Trip Times (RTT) and one-way delay measurements. Using RTT method allows the measurement of the total propagation delay to and from an observation point. A one-way delay measurement provides an estimation of the time it takes to propagate a signal between two points in a

network. To have some accuracy and valid results, a synchronous clock system should be implemented. This can be made using a NTP server or a GPS system.

Passive Measurements uses the existing traffic in the network, monitoring through fixed observation points. Since no test traffic is sent, passive measurements can only be applied in situations where the traffic of interest is already present in the network. Passive measurement is also referred to as non-intrusive or as measurement of observed traffic. It cannot provide the kind of controllable experiments that can be achieved with active measurements.

In this thesis we will describe in more detail signaling protocol to perform measurements in the following chapters. These measurements will be complemented with export protocol as well.

The Next Steps in Signaling (NSIS) suite of protocols is envisioned to support various signaling applications that need to install and/or manipulate such state in the network. NSIS can be describe as a collection of protocols that enable end-to-end signaling pertaining to a flow or a collection of flows across heterogeneous IP based networks. NSIS is a modular concept, decouples the signaling application from the signaling services, examples of those services are QoS reservation, NAT and firewall traversal and Metering. QoS is the most developed application for NSIS, and considered the most important, but in this thesis we will discuss Metering application and tool and technologies to support it.

Chapter 3

State of The Art

In this section we will present existing technologies and architectures for traffic measurement in an IP network. These technologies are implemented with well known tools and techniques. We will discuss about all technologies and platforms needed for this implementation. First we will present an overview over Measurement Organizations, as RIPE, NIMI, and CPOC. After we present and discuss the Export protocol that was implemented in order to export measurement records to a flow. Finally we will describe the application functionality and a small overview about the two others NSIS applications, QoS-NSLP, and NAT/firewall-NSLP.

3.1 Measurement Organizations and Platforms

In Europe the Network monitoring process is carried out by RIPE NCC (Réseaux IP Européens / NCC Network Coordination Center). RIPE is a collaborative organization open to all European Internet Service Providers. The objective of RIPE is to ensure the necessary administrative and technical coordination to allow the operation of a European IP network.

The goal of the organization is to provide services for the benefit of the IP-based network operators' in Europe and surrounding areas.

To perform this service a project was created called TTM (Test Traffic Measurements); the main goal of this project is to carry out performance measurements according to well-defined and scientifically defendable standards set forth by standards bodies such as the IETF.

The control of network, by TTM, is made by this general setup 3.1, this architecture designed for export data flows is very similar to our implementation . The protocol to be specified and developed must execute this task.

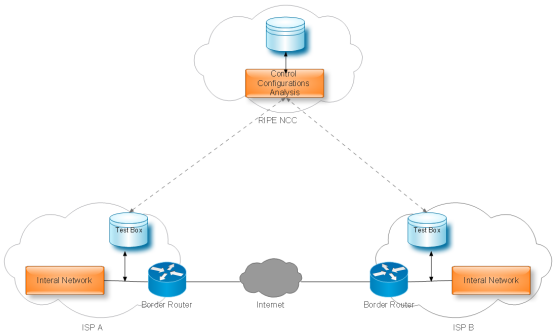

Figure 3.1: Ripe Measurement model

Besides RIPE NCC, others platforms can be considered for traffic measurements, such as NIMI (National Internet Measurement Infrastructure) [9]. This was the first measurements platform used for research. This platform was inspired by another one called Network Probe Daemon (NPD), developed by Paxson [10]. The NIMI platform is composed of four components: NIMI probe, CPOC (Configuration Point of Contact), MC (Measurement Client), and DAC (Data Analysis Client). The architecture of NIMI is described in 3.1. The client (MC) sends a solicitation to execute a traceroute (step 1), to a host; for each measurement the probe NIMI consults the CPOC to get the authorization for the measurement (step 2). In step 3 the probe NIMI executes the traceroute to the hosts, and sends the result to DAC.. DAC is the data base of all measurements sent by NIMI (step 4).

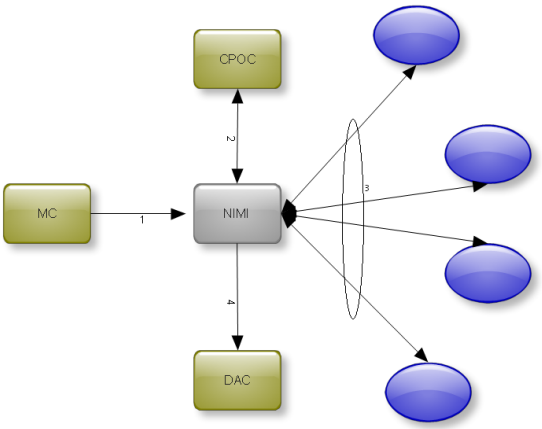

Figure 3.2: Architecture of NIMI

This architecture gave a good support to a progressive implementation of measurements using probes.

Other architecture to be considered is the Planet lab platform [11]. Planet Lab is the most prominent Internet testbed today. It is a canonical example of an openly

available network testbed that is designed to support many different types of network-related experiments at the same time.

Other measurement projects are in constant activity examples are, NLANR, IPMON, and CAIDA.

NLANR (National Laboratory for Applied Network Research) is a laboratory responsible for an extensive infrastructure for analysis of networks [12]. The main task of this laboratory is monitoring the American HPC (High Performance Connection) network community, funded by NSF (National Science Foundation). The platform of NLANR maintains projects in active and passive measurements. People from the NLANR proposed IPMP (IP Measurement Protocol) [13] as a substitute for the use of protocols such as ICMP, TCP and UDP for active measurements, since these have limitations for use in measurements, because they were not designed for this purpose.

The IPMON (IP) [14] is a passive monitoring system capable of capturing samples on high-speed links in a network backbone. The IPMON provided a system highly innovative, since it is capable of simultaneously capturing information with a high level of accuracy on multiple links geographically dispersed. Another innovative feature was the timestamps of synchronized clocks from GPS. These features enabled the detailed review of the behavior of Internet traffic.

CAIDA (Cooperative Association for Internet Data Analysis) [15] is a collaborative association with participants from the commercial sector as well as from the government and academic world.

The main purpose of this association is the investigation of the structure and behavior of the Internet through measurements for a better understanding of the network for its extension to global levels.

All of these projects and measurements platforms have the purpose of giving a better understanding of the network; the knowledge of the Internet behavior will be a great support for extension of the Internet. In the following sub-section we will present existing measurement techniques, and architectures.

3.2 Measurements Architectures

In this section it will be described how the existing measurement architectures implement traffic measurement. The existing standardized measurements architectures are RTFM, IPPM, IPMP, IPFIX, RMON, and PSAMP.

RTFM Traffic Flow Measurement seeks to provide a well-defined method for gathering traffic flow information from networks and internetwork. The Real time Traffic Flow Measurement (RTFM) Working Group has produced a measurement architecture to achieve this. The architecture defines three entities Meters, Meter Readers, and Managers. The Meter entity was defined as an observer of network traffic flows. With these observations

it is possible to build up a table of flow data and records of them. A Meter Readers entity collects traffic flow data from meters. Finally, the Managers entity oversees the operation of meters and meter readers.

The RTFM architecture [16] is supported and used by IETF to characterize traffic measurement and reporting for data network. RTFM Traffic Measurement Architecture provides a general framework for describing and measuring network traffic flows. Flows are defined in terms of their Address Attribute values and are measured by a 'Traffic Meter'.

Additional components that may participate in the RTFM architecture have also been defined, such as:

- **An RTFM MIB.** [17] Management Information Base (MIB) is a data base used by RTFM Traffic Meter to specify the flows to be measured. It also provides an efficient mechanism for retrieving flow data from the meter using SNMP.

IPPM Another standard supported by IETF was the IP Protocol Measurement (IPPM) [5]. This architecture defines a Management Information Base (MIB) designed for use with network management protocols in TCP/IP-based networks. In particular, this MIB specifies the objects used for managing the results of the IPPM metrics, for pushing alarms, and for reporting the results.

IPMP In [18] the IP Measurement Protocol (IPMP) is characterized as a protocol that addresses several of the limitations of using existing protocols to encapsulate packet probes. This IP measurement protocol considers both the packet delay and the path of a packet takes in a single packet exchange between the measurement host and the echo host. This protocol is based on an echo request and reply packet exchange for measuring packet delay and associated path metrics, and is similar to the technique that ping/traceroute use with the ICMP echo capabilities. IPMP is carried directly inside an IP packet in order to make an echo packet obvious to the routers connecting the hosts involved in the measurement. The echo reply packet has been designed so that an echo host can construct an echo reply packet with very few modifications to the echo request packet. This Measurement protocol has a problem of priority given to ICMP packets by the routers.

IPFIX Internet Protocol Flow Information eXport (IPFIX) [19] is a protocol developed by the IETF IPFIX working group, which aims to standardize the format used for the export of network flow data toward data collection devices and network management systems.

The simplified version of IPFIX architecture 3.3 is compounded with two devices called Exporter (Observation Point), and Collector. Exporter can usually be a central node in a Network (router or a switch), or an external probe connected to this device.

The exporter needs to have a metering process running at all times, capturing the traffic on a network. The metering process is also responsible for starting the streaming and constructing flow records. The exporter needs also to have an exporting process running. This last process does not run all times, but is called when either the flow buffer is full, or after a certain time period. When the exporting process is called, it walks the flow cache, and starts sending the flows to a designated collector.

The collector has a collecting process running at all times responsible for receiving flows. After the collector receives data flows, it might be saved in a database, but this process is out of IPFIX scope [20].

Based on the requirements defined in [21] different existing protocols (NetFlow v9, Diameter, CRANE, IPDR) were evaluated as candidates to provide the basis for a future IPFIX protocol [22]. All of these protocols have the same properties as IPFIX. But they can be applied for different kind of measurements.

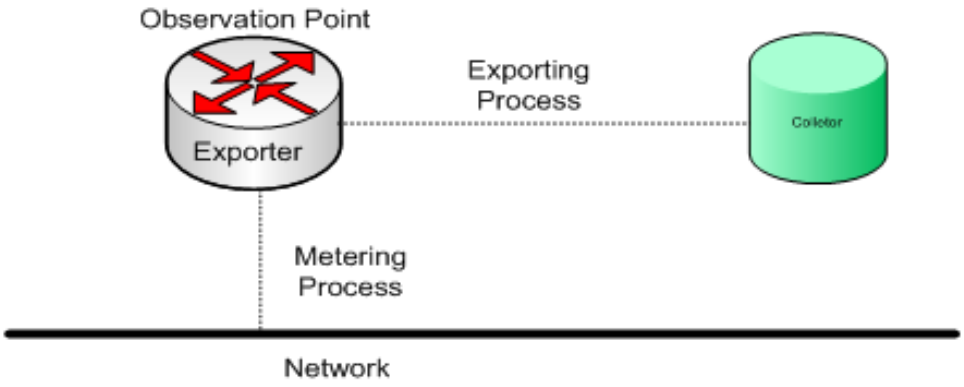

Figure 3.3: IPFIX Architecture

RMON Remote network monitoring is a monitoring standard defined by IETF. Remote network monitoring devices, often called monitors or probes, are instruments that exist for the purpose of managing a network. Often these remote probes are stand-alone devices and devote significant internal resources for the sole purpose of managing a network. RMON assumes the availability of network monitoring devices called monitors or probes. By monitoring packet traffic and analyzing the headers, probes provide information about links, connections among stations, traffic patterns, and status of network nodes. Hence, RMON can be regarded as following a traffic-oriented approach because the status of the network is determined by direct inspection of the packets flowing in it, rather than inspection of the status of each device.

A probe in RMON can detect failures, misbehaviors, and identify complex relevant events even when not in contact with the management station, which is likely to happen when the network is overloaded or in critical conditions. In addition, the agent on the probe

can also do periodic checking and semantic compression, which further increases decentralization. Characteristics of RMON monitoring system are offline operation, proactive monitoring, problem detection and reporting, value added data, and multiple managers.

Offline operations are useful in some conditions when a management station may not be in constant contact with its remote monitoring devices. In these occasions this MIB allows a probe to be configured to perform diagnostics and to collect statistics continuously, even when communication with the management station may not be possible or efficient.

When a diagnostic need to be performed without any interruption, it is useful to use a proactive monitoring, where all resources are available on the monitor, and it is potentially helpful for it continuously to run diagnostics and to log network performance. The monitor is always available at the onset of any failure. It can notify the management station of the failure and can store historical statistical information about the failure. In case of some failure or bad behavior from a specific group of probes, it's possible to highlight those hosts on the network that generate the most traffic or errors. The probe can give the management station the precise information it needs to solve a class of problems. This can be called, value added data.

PSAMP Packet sampling (PSAMP) is widely implemented in today's routers, where the sampling method usually keeps one out of N sequential packets. Some line card engines can also sample consecutive packets. The framework definition of Packet Sampling is provided by [23]. This document describes a standard set of capabilities for network elements to sample packets and report on them. One motivation to standardize these capabilities comes from the requirement for measurement-based support for network management and control across multi vendor domains. This requires wide consistency in the types of sampling schemes available, the manner in which the resulting measurements are presented, and consequently, consistency of the interpretation that can be put on them.

The framework for passive measurement has three main parts: the selection of packets for measurement, the creation and export of measurement reports, and the content and format of the measurement records.

Compared with similar work the PSAMP measurement capabilities are positioned as suppliers of packet samples to higher level consumers, including both remote collectors and applications, and on board measurement-based applications. Indeed, the development of the standards within the framework described here should take into account the measurement requirements of standards in other IETF WGs, including IPPM and TEWG. Conversely, we expect that aspects of this framework not specifically concerned with the central issue of packet sampling may be able to leverage work in other WGs. The prime example is the format and export of measurement reports, which may leverage the work of IPFIX.

Packet sampling is very useful when one wants to get traffic information at a higher resolution than that obtained from the Simple Network Management Protocol (SNMP) coarse-grained counters or active probe data. For instance, packet sampling will perform

very well to estimate the average packet rate [24]. Packet sampling is also well suited for collecting other basic statistics, such as source and destination IP addresses, route prefixes and autonomous system numbers. However its performance when recovering higher level statistics, such as the spectral density of the packet arrival process or distribution of flow length, is very poor [25].

Despite these standard architectures implement what we want them to do in one way or another, with intrusive processes (active measurements) or passive measurements or even with example of architecture RMON, which is used for networks management. The architecture that most closely matches than we propose to do is without doubt the IPFIX, which uses a process of export of information, which can be used in the event of traffic between AS's.

3.3 Technology and Tools

In this section of the thesis is presented and overview, measurement tools, and traffic generators with their analysis and qualification. Many software tools have been developed to gain information on network performance. All of them incorporate various techniques and focus on different network characteristics.

3.3.1 Measurement tools

In this report we will consider metrics to qualify each measurement tool in order to select one or many, which can be capable to support our protocol simulation. These metrics to be considered are: Latency, throughput, RTT, Bandwidth, one-way delay, Packet loss, and jitter. In this section we will consider also Sniffer Packet tools, as tcpdump, and Wireshark. Tcpdump will be configured in our implementation to monitor each node in end-to-end measurements, this implementation will be discussed in following chapters. Wireshark was also an important tool in all process.

TcpDump

Tcpdum will be configured to capture traffic and to monitor each nodes like MNE in this case. It was considered a bottleneck when dealing with very high-speed inputs. tcpdump is an open-source packet capture program that runs on UNIX platforms [26]. tcpdump use the pcap library to capture network traffic.

The packet capture library (libpcap) provides a high level interface to the packet capture systems [26]. This library is capable of handling BSD Packet Filter (BPF) commands [26].

These commands allow for specific types of packets to be filtered out before they are handed over to the high-level program. tcpdump supports several different network protocols. Processing decisions are based on the link layer, network layer, and transport

layer information. tcpdump is a good choice for small to medium-sized networks, since each packet is immediately written to disk.

WireShark

WireShark works much like tcpdump, because it is also a packet capture program [27].

WireShark is a free application with a graphical user interface. This protocol analyzer also uses libpcap to provide the packet capturing functionality. This enables the use of BPF commands while capturing data. Like tcpdump, Wireshark can either collect data live, or parse a previously captured network data file. WireShark is able to understand data from several different other captures programs.

The actual analysis that takes place in WireShark also supports several different protocols. This analysis is even customizable as new protocol modules can be added to the program. Filters enable the display of just the data that a user is interested in viewing.

cprobe and bprob

Each measurement tool implements a different technique to perform its measurement. One of those is packet dispersion technique. This technique has been used to design cprobe and bprob [28].

Bprob sends a series of packets across a network path with as little space as possible between them, thus creating a "packet train" (a packet train of length two is called "packet pair"). But these packets have some problems because their type is ICMP, ECHO, or UDP, and as we know some routers may not process them as quickly as others.

The analysis of packet dispersion technique is made by measuring the gap between two packets. It is performed by sending a packet train to a target host, which echoes them back. When passing though a link with smaller capacity, packets will be queued. After that link, the packet have the gap between them, this technique is useful to estimate available bandwidth 3.4 [28], [29].

In [29], it is argued that only packet pairs, not packet trains should be used. This new view of packet-trains was used to develop the Pathrate tool. Another measurement tool to be compared with cprobe is Pathload [30].

Pathload does not calculate its available bandwidth directly. Its algorithm is an adaptive process based on increasing one-way delays of packet pairs when the probing rate of the sender increases. The packet pair technique may become inaccurate if the amount of cross-traffic increases. This problem is discusses in detail in [31].

Pathload

Pathload is based on the self-loading periodic streams technique [30] and measures available bandwidth. These measurements are made by sending various series of packet

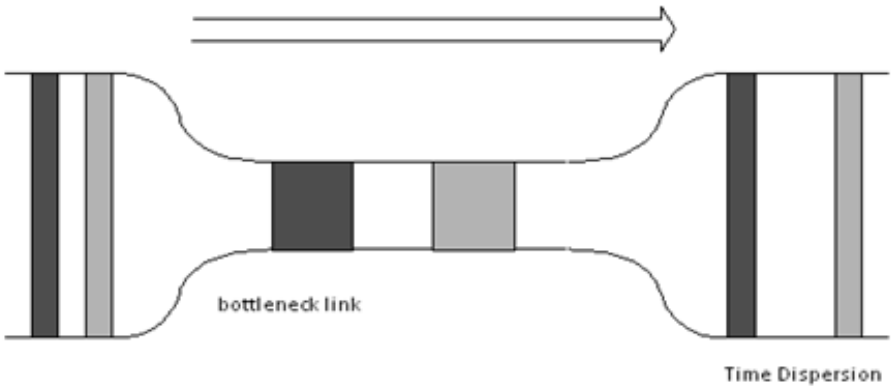

Figure 3.4: Packet Pair Model

streams over the path from source to destination. The rate at which packets are being sent is supposed to be larger than the available bandwidth of the link with the lowest available bandwidth. Packets will be queued on that link, which process them at a constant rate. Before being sent, the packets get a timestamps. Upon receiving two successive packets, the difference in timestamps is compared with the difference in their arrival times.

Pathchar/Pchar, Bing, Clink, and Nettimer

The measurement tools pathchar/pchar, Bing, clink, and Nettimer use the technique Variable Packet Size (VPS) [1]. This technique was first used in the pathchar tool to estimate per-link capacity. The basic idea is that the tool will send a packet in which the TTL (time to live) field in the IP header is set to a specific value. After the decrease of TTL until zero the device will send an ICMP TTL-exceeded packet back to the source. Upon receiving this ICMP packet, the tool can estimate the RTT (round trip time) [1].

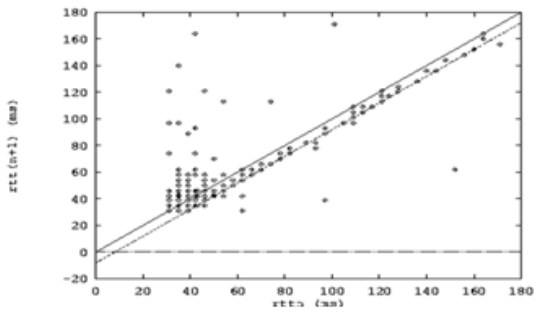

Figure 3.5: Variable Packet Size (VPS) from [1]

In 3.5 illustrates the Variable Packet Size technique for the first hop of a path. The slope of the linear interpolation of the minimum RTT measurements is the inverse of the capacity estimate at that hop.

Unfortunately, Variable Packet Size probing may yield significant capacity underestimation errors if the measured path includes store-and-forward layer 2 switches [1].

Nettimer also uses the packet tailgating technique for measuring bandwidth capacity. All Process is explained in [32].

Finally Treno Tool uses TCP simulation. This methodology is used to measure the bulk-transfer [21] capacity. Treno simulates TCP and uses a slow-start algorithm [33]. Besides all of these active measurement tools, we will consider NetFlow, Cflowd, and Netram. These passive measurement tools need to be installed in routers.

In the following table other tools are presented, but all have the same characteristics as the ones discussed before. All information about these tools is below [34], [35].

Table 3.1: Measurement Tools

Tool Name	Active/Passive	Metrics	Methodology	Protocol	Path/Pre-link
Bprobe	Active	Bandwidth	Packet pair	ICMP	Path
Cprobe	Active	Bandwidth	Packet pair	ICMP	Path
Clink	Active	Bandwidth	VPS	UDP	Per-link
Bing	Active	Bandwidth, loss, latency	VPS	ICMP	Path
Iperf	Active	Bandwidth, loss	Path flooding	TCP, UDP	Path
Netest	Active	Bandwidth	Packet pair	UDP	Path
Netperf	Active	Throughput, Request/Response	Path flooding	TCP, UDP	Path
Ttcp	Active	Bandwidth	Path flooding	TCP, UDP	Path
Pipechar	Active	Bandwidth	Packet train	UDP	Per-link
Pathchar	Active	Bandwidth, loss, latency	VPS	UDP, TCP	Per-link
Nettimer	Active/ Passive	Bandwidth	Packet pair, VPS/tailgating	TCP	Path
Pathload	Active	Bandwidth	SLOPS	UDP	Path
Pathrate	Active	Bandwidth	Packet pair & packet train	UDP	Path
Sprobe	Active	Bandwidth	Packet pair	TCP	Path
Treno	Active	Bandwidth	TCP simulation	UDP, TCP	Path
Pchar	Active	Bandwidth, loss, latency	VPS	UDP, TCP	Per-link
Netflow	Passive	/	/		
Traceroute	Active	Topology, latency, loss	varied TTL	TCP,UDP	Per-link

3.3.2 Traffic Generators

To perform our measurements we need tools capable of generating network traffic. Table 3.2 shows a comparison between the tools discussed below. From the table, it can be observed that the tools create TCP and UDP traffic.

Table 3.2: Traffic Generators

Tool	**Capability**	**Input**
NetSpec	TCP, UDP	CLI
RUDE/CRUDE	UDP	CLI
MGEN	UDP, IP	GUI, CLI

NetSpec

NetSpec is a tool designed to make the process of network testing simpler. It provides sophisticated support for testing the functions and performances of networks. NetSpec comprises of a number of daemons types that are started and controlled by the central NetSpec daemon. A user can then control numerous processes on numerous hosts from a central point of control. NetSpec supports TCP and UDP traffic and it uses command-line interface for input.

MGEN

The Multi-Generator (MGEN) is a tool for generating and measuring UDP, TCP and IP traffic. MGEN generates real-time traffic patterns so that the network can be loaded in various ways. The generated traffic can also be received and logged for analysis purposes. MGEN log data can be used to calculate performance statistics on throughput, packet loss rates, communication delay, and more.

RUDE and CRUDE

RUDE is a tool for generating traffic to the network and CRUDE is a tool for receiving a RUDE packet. RUDE and CRUDE generate and measure UDP traffic. The operation and configuration is similar to MGEN, as they do not share any codes. RUDE and CRUDE do not suffer from the limitations in accuracy that applies to MGEN. It uses command-line interface for input.

3.4 QoS Architectures

A lot of solutions have emerged to implement QoS in IP networks in the past years. IETF specifies three different architectures for QoS support in IP Network, these architectures are Integrated Services (IntServ), Differentiated Services (DiffServ), and the Next Steps In Signaling architecture. Each QoS architecture will be defined in the next chapter.

The last architecture mentioned was born because the others two IntServ, and DiffServ suffer from a number of issues and limitations. For this reason, the Internet Engineering Task Force (IETF) is proposing a suite of protocols under the name of Next Steps In Signaling [36]. The goal of NSIS even goes beyond providing QoS signaling, it defines a generalized end-to-end signaling protocol for flows and several signaling applications, one of them being QoS. Other signaling applications such as metering and Network Address Translation (NAT) traversal are also to be defined.

The QoS signaling application in particular is designed in such a way that it is flexible in the types of QoS it can support, not restricting itself to any particular QoS model. In this way, NSIS should be ideally suited to perform dynamic end-to-end soft-state reservations for multimedia flows.

NSIS is able to support end-to-end measurements in IP network; these measurements can have extremely importance because they can certify the network performance, and identify some kind of bottleneck.

In this report the discussion will focus in this signaling protocol.

3.4.1 Integrated Services (IntServ)

IETF has defined two models for QoS IP: Integrated Services (IntServ) and Differentiated Services (DiffServ). In IntServ [37], traffic is serviced in a per-flow basis in accordance with each flow's absolute service request. In this service routers must reserve resources for each flow and isolate flows from one another. The signaling protocol designed for the IntServ model is the Resource ReServation Protocol (RSVP).

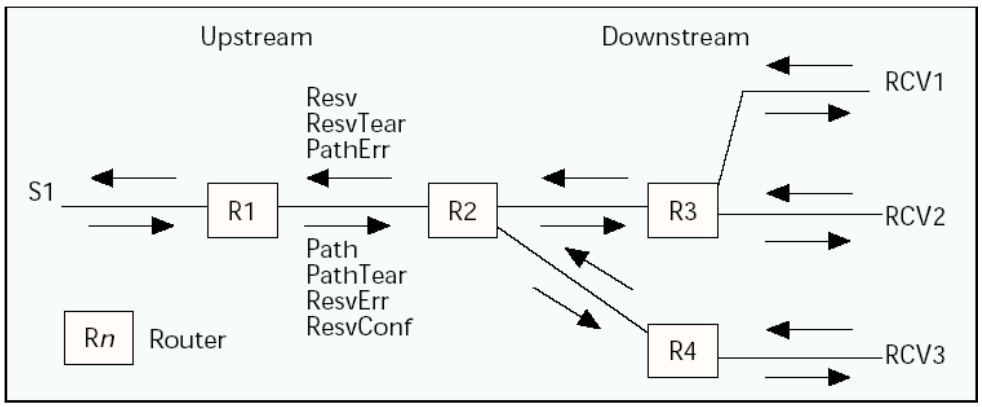

Figure 3.6: RSVP Multicast Reservation

This protocol is used by a host to request specific QoS from the network, and thus some guarantees are provided to applications. IntServ has some drawbacks that limit its application. The necessary isolation of flows may be very expensive when the number of flows is very large, and another limitation is for the signaling protocol (RSVP) that is not appropriate to deal with short flows. The scalability problems of IntServ led to the proposal of another QoS architecture, known as differentiated service (DiffServ).The DiffServ framework is based on the aggregation of service differentiation among these classes [38].

3.4.2 Differentiated Service (DiffServ)

In DiffServ all the complex processing, such as flow classification and traffic dimensioning, is made on the edge routers.

DiffServ intends to meet the need for simplify methods of providing differentiated classes of services over Internet. The DiffServ architecture is characterized by a relative-priority scheme, which marks packets into predefined service classes with different code-points. Packets in different classes receive different services. By aggregating packets

with the same codepoint, the flow receives a particular forwarding treatment, or Per-Hop-Behavior (PHB), at each network node. To enable QoS, preferential treatments on, for example, buffer management and scheduling mechanisms, are given to flows marked with Expedited Forwarding (EF) [39] or Assured Forwarding (AF) [40] classes. These two QoS models were compared by [41], in result of their analysis, IntServ architecture was more suitable for strict QoS guarantees than DiffServ architecture because the former has per-flow-base traffic handling mechanisms but the latter does not.

IntServ and DiffServ could be also distinguished by their differentiated treatment of packet streams, IntServ and DiffServ could be also distinguished by their differentiated treatment of packet streams, IntServ guarantee QoS on a per-flow basis. It requires explicit signaling to reserve network resources along the path to the other end. The biggest drawback of IntServ is the difficulty in monitoring and processing so many flows.

DiffServ give priority to the flows on an aggregated basis, i.e., a set of many flows with similar service requirements are treated the same based on its codepoint. By different treatment of a limited number of classes of services, the performance of the aggregated flow is guaranteed based on its PHB. However, DiffServ applies only to large scale networks and thus cannot provide end-to-end QoS. Another QoS architecture aimed at supporting end-to-end services is the hybrid of IntServ and DiffServ [10]. In this framework, IntServ and DiffServ are used together to meet the needs of large ISPs who manage the backbones on the Internet and the need of the QoS to end users.

For IntServ, DiffServ, or a hybrid of both architectures to work, they need to be deployed within all ISP domains. This implementation makes possible high performance networks and enables to support streaming or high quality TV.

3.4.3 NSIS

NSIS was proposed to suppress shortcomings of RSVP. These shortcomings are:

- Mobility of nodes is not supported.
- Complexity, and IP multicast is not supported
- Limited data length, as fragmentation is not allowed
- Security problems.

In the following chapters, the IETF NSIS protocol will be described in more detail.

3.4.3.1 NSIS Framework

The IETF NSIS protocol suite [42] is conceptually divided into two layers. The lower layer is known as the NSIS Transport Layer Protocol (NTLP), the upper as the NSLP.

The NSIS working group specifies GIST [43] as the protocol that operates as NTLP. For Protocol layer the working group specifies:

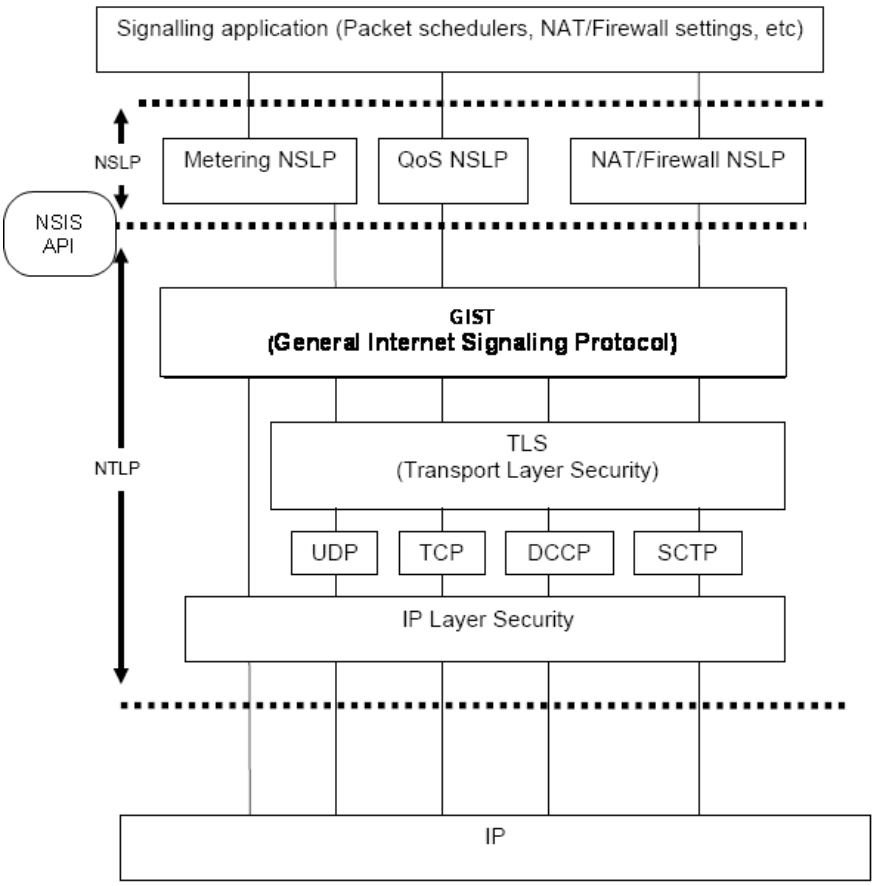

Figure 3.7: NSIS Protocol Stack

- QoS NSIS Signaling Layer Protocol (QoS NSLP) [36].
- Network Address Translation (NatFW NSLP) [44]
- NSLP for Metering (Metering-NSLP) [45]

The NSIS Transport Layer Protocol (NTLP) transports messages up and downstream along the signaling path. One or more NSLPs run on top of the NTLP. NSLPs are distinguished according to their NSLP-ID. The two-layer architecture can be seen in figure 3.7. The QoS NSLP is not the only signaling application included in NSIS stack. The other examples are, the NSLP for metering [45] and for Network Address Translation (NatFW) [44], the main application featured in this report is the Metering-NSLP. The NSIS Transport Layer Protocol operates between adjacent peers; the end-to-end messaging is handled at NSLP level. Each session is identified by a unique session identifier. Intermediate nodes are bypassed where the corresponding NSLP is not present, i.e. the NTLP only forwards messages to the upper layer NSLP where the NSLP-ID matches. Where the corresponding NSLP is not present, messages are forwarded. The NTLP can use either User Datagram Protocol (UDP) [46], or Transmission of Transport Protocol(TCP) [47]

for its message transport, the latter optionally making use of Transport Layer Protocol (TLS) [48] for security. The NTLP can use also Stream Control Transmission Protocol (SCTP) [49] or Datagram Congestion Control Protocol (DCCP) [50]. All Communications between these two layers, NSLP and NTLP, is preformed trough a predefined API.

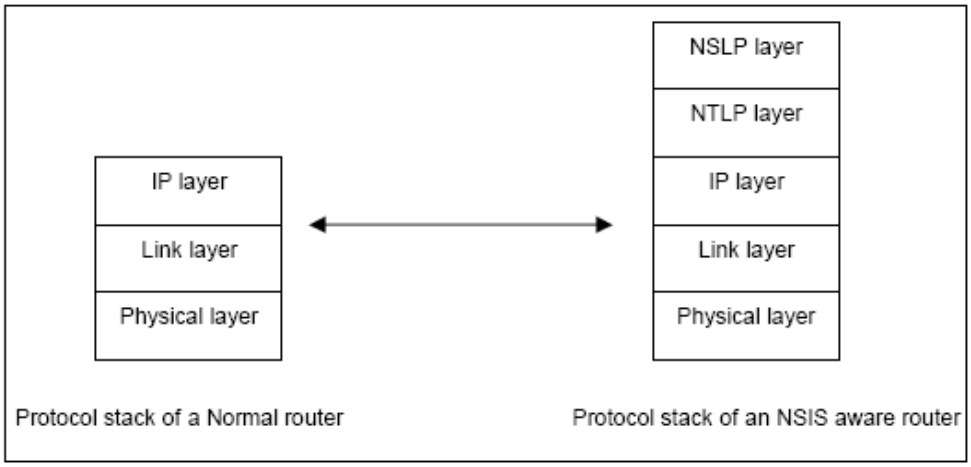

Figure 3.8: NSIS router vs normal router

The NSIS protocol suite provides support for sender and receiver-initiated measurements, uni and bidirectional operations, one measurement at time. Bidirectional measurements are useful to compare results and determine a possible bottleneck. NSIS reacts on route changes by discovering and maintaining a new route. Route changes are found in mobile and multi-homed environments where a mobile client moves from one AP to the next, but also in the fixed network. The NSIS protocol has the same drawbacks that characterize all signaling protocols, the problem of overhead [51]. This feature needs to be developed and studied, to improve NSIS performance.

3.4.3.2 GIST Transport Layer

All work realized for discovery of signaling peers and for the delivery of signaling messages along the data path, is implemented by GIST. It reuses existing transport layer protocols, like TCP, UDP and SCTP. GIST establishes and maintains Message Routing State (MRS) per session using a three-way handshake (Query, Response and Confirm). For the discovery of the next hop, a Router Alert Option (RAO) is set in the IP header of the Query message. GIST is a soft state protocol and therefore the MRS needs to be refreshed regularly. The refreshes are sent in UDP with set RAO to discover route changes. The communication between the two protocol elements is done via an Application Programmer Interface (API). NSLP messages are transported by GIST as payload and passed through the API.

The NSLP-ID in GIST messages ensure that only relevant messages are passed to the associated NSLP. The routing information, used to describe a flow, are carried in the Message Routing Information object (MRI). It carries the 5-tupel protocol, source/destination address, address prefixes and ports with some additional flags. The MRI is transferred between NTLP and NSLP over the SendMessage() and RecvMessage() API-calls.

3.4.3.3 Gist API

The Gist API is defined in terms of abstract services, three from the NSLP application toward GIST, and tree in the opposite direction. The services primitives directed from the signaling application toward GIST are **SendMessage**, **SetStateLifeTime**, **InvalidateRoutingState**. In the opposite direction the services primitives are **RecvMessage**, **MessageStatus**, **NetworkNotification**.

- SendMessage

The SendMessage primitive is used by the NSLP to initiate sending of messages. The parameters included in this message are the following:

SendMessage (NSLP-Data, NSLP-Data-Size, NSLP-Message-Handle, NSLP-Id, Session-ID, MRI,SSI-Handle, Transfer-Attributes, Timeout, IP-TTL, GHC)

NSLP-Data is the payload the NSLP application wants to transmit.

NSLP-Data-Size is the length of the NSLP-Data.

NSLP-Message-Handle is a handle that refers to this particular message. It allows GIST to refer back to it when it issues the MessageStatus service primitive.

NSLPID is 1a 6-bit unsigned integer identifying the NSLP application.

Session-ID is a 16-byte identifier unique to this session.

MRI is the Message Routing Information, which describes the flow to which the signaling pertains.

Optional arguments of the message:

SII-Handle is the Source Identification Information Handle, which can be used to bypass state stored in GIST and directly address a node.

Transfer-Attributes allow the application to convey desired transfer properties of the message. Among others this can contain whether or not the message should be sent reliably and whether or not it should be sent securely.

Timeout is the time for which GIST should attempt to keep sending the message.

IP-TTL is the Time To Live (TTL) value GIST should put in the IP header.

GIST-Hop-Count is the initial value for the GIST Hop Count (GHC) when GIST transmits the message.

- SetStateLifeTime

The SetStateLifeTime provide to NSLP control for how long the state retained within GIST for a particular session is valid. The parameters included in this message are the following:

SetStateLifeTime (NSLPID, MRI, State-Lifetime)

NSLPID is a 16-bit unsigned integer identifying the NSLP application.

MRI is the Message Routing Information, which describes the flow to which the signaling pertains.

State-Liftetime is the amount of time for which the application wishes the state to remain active in GIST.

- InvalidateRoutingState

The InvalidateRoutingState provide to NSLP request GIST to remove any state associated with a particular session. The parameters included in this message are the following:

InvalidateRoutingState (NSLPID, MRI, Status, Urgent)

NSLPID is a 16-bit unsigned integer identifying the NSLP application.

MRI is the Message Routing Information, which describes the flow to which the signaling pertains.ling pertains.

Status is a boolean which indicates how definite the routing state invalidation should be.

Urgent is a boolean which indicates whether state recovery should proceed immediately.

- RecvMessage

The RecvMessage provide to GIST deliver incoming messages to a NSLP applications. The parameters included in this message are the following:

RecvMessage (NSLP-Data, NSLP-Data-Size, NSLPID, Session-ID, MRI, Routing-State-Check, SII-Handle, Transfer-Attributes, IP-TTL, IP-Distance, GIST-Hop-Count, Inbound-Interface)

NSLP-Data is the NSLP payload of the message.

NSLP-Data-Size is the length of the NSLP-Data.

NSLPID is 1a 6-bit unsigned integer identifying the NSLP application.

Session-ID is a 16-byte identifier unique to this session.

MRI is the Message Routing Information, which describes the flow to which the signaling pertains.

Routing-State-Check is a boolean indicating that GIST is asking the NSLP application whether or not to set up state with this peer. If it is set, the application should reply to this primitive with:

- A boolean of whether it wants to set up state or if it wants the query to be propagated further downstream.
- Optionally a payload that the application wants GIST to include in the response to the querying peer or in the propagating query in case it does not want to setup state.

SII-Handle is the Source Identification Information Handle of the transmitting node, which can be used to bypass state stored in GIST and directly address this node.

Transfer-Attributes are the transfer properties with which the message was transmitted, such as reliability and security.

IP-TTL is the TTL value of the IP header of the received message.

IP-Distance is the calculated distance in IP hops between this node and the sender of the message.

GIST-Hop-Count is the value of the GHC in the received GIST message.

Inbound-Interface provides information about the physical interface on which the message was received.

- MessageStatus

The MessageStatus provide to GIST indicate to NSLP application if a message was sent correctly. The parameters included in this message are the following:

MessageStatus (NSLP-Message-Handle, Transfer-Attributes, Error-Type)

NSLP-Message-Handle is a reference to the message that was generated earlier by the application and set using the SendMessage service primitive.

Transfer-Attributes are the transfer properties with which the message was transmitted, such as reliability and security.

Error-Type indicates, if the message could not be delivered, the reason of failure, the most important one being that this node is the last NSIS-aware node in the path.

- NetworkNotification

The NetworkNotification provide to GIST indicate to NSLP application any change of network status. The parameters included in this message are the following:

NetworkNotification (NSLPID, MRI, Network-Notification-Type)

NSLPID is 1a 6-bit unsigned integer identifying the NSLP application.

MRI is the Message Routing Information, which describes the flow to which the signaling pertains.

Network-Notification-Type indicates the type of network status change, such as a change in routing state.

MRI is the core of GIST and its API, which is used to describe a flow or a set of flows. A Flow is described on the traditional 5-tuple of protocol, source address, source port, destination address, and destination port with additional information. The NSLP applications use the MRI to inform GIST about the characteristics of the flow.

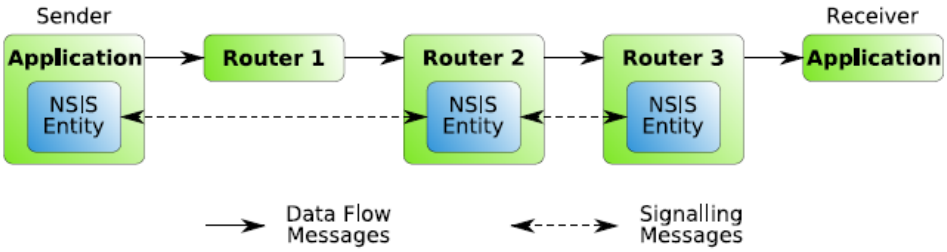

Figure 3.9: GIST Example Signaling and data flow

The Figure 3.9 illustrates an example of a GIST service provided to NSLP application. The application uses the SendMessage service to send messages o configure the flow in the MRI to its peers. All GIST flows are unidirectional. If any bidirectional operation is required it can be made using two unidirectional lows at the NSLP level.

As we can see not every nodes on the path has on the path have GIST. The sender needs to do some signaling about the flow before it can perform any measurement. For example he wants to perform any measurement. This Metering-NSLP application will construct a MRI about the flow, create a new Session Identifier (SID) and issue a SendMessage service primitive to GIST, including an M-NSLP payload to be sent and indicating what transfer properties the message should have. GIST will then check its internal state to see if knows the downstream peer, Keyed by the combination of the NSLP-ID, MRI, and SID. If this

state exists it will use it to find out if this node has a Messaging Association (MA) with the downstream peer that has the same properties that the application requested. If the state does not exist it will attempt to discover the downstream peer. Peer discovery is performed with a UDP datagram with a present destination port address toward the flow destination, gleaned from the MRI, with the Router Alert Options (RAO) [52] in the IP header set to a value corresponding to the NSLP-ID.

Routers provided with NSIS will actively listening out for any UDP datagram's with the present port and the RAO set and intercept these datagram's. In this way the next downstream peer can be found and a routing state between the peers can be set up. The next step is a negotiation between the two nodes to either re-use an existing MA or set up a new MA with the desired properties. This process will continue until the last node on the path is found, in this case router 3.

Although of a node may have GIST software running, it may not have the NSLP application identified by the available NSLP-ID, or the application may even choose not the participate using the response to the RecvMessage service primitive. As already said GIST is Soft state, it means that after a certain period of disuse states will expire automatically. Although of a node may have GIST software running, it may not have the NSLP application identified by the NSLP-ID running, or the application may even choose not participate using the response to the RecvMessage service primitive.

3.4.4 GIST Transmission modes

GIST has three different modes to transfer messages to its peers, each one of which will now be illustrated.

Query Mode (Q-Mode)

This mode is used to discover downstream peers. In Query Mode a UDP datagram will be sent toward the MRI destination at a predefined destination port. The UDP source port is set to the port on which the sending GIST node will accept messages in Datagram Mode.

Datagram Mode (D-Mode)

In Datagram Mode messages are sent a UDP datagram's, addressed directly to the GIST node one wants to reach. The IP address is learn from a message it sent in Query Mode. The destination port should be equal to the source UDP port used in a previous Query or Datagram made message sent from destination node.

Connection Mode(C-Mode)

In Connection Mode a Messaging Association between two nodes is used to transmit the message. A MA is a connection between two nodes using a particular connection oriented protocol. A MA is also stored in soft state, meaning that after a certain period of disuse the connection will automatically be torn down.

3.4.5 GIST Messages

GIST protocol defines four different messages, each one with a set of GIST objects.

- Query

The Query message should be used for initial peer node discovery. This message should still be sent periodically to detect changes in routing topology. A Query message can only be sent using Query Mode. The objects included in a GIST Query message are the following:

Table 3.3: Query message

Query
Common-Header
[NAT-Traversal-Object]
Message-Routing-Information
Session-Identification
Network-Layer-Information
Message-Routing-Information
[Stack-Proposal Stack-Configuration-Data]
[NSLP-Data]

- Response

The Response message should be sent in Datagram mode, or in Connection mode if an existing messaging association is being re-used. It must echo the MRI SID and Query-Cookie of the Query message. The objects included in a GIST Response message are the following:

Table 3.4: Response message

Response
Common-Header
[NAT-Traversal-Object]
Message-Routing-Information
Session-Identification
[Network-Layer-Information]
Query-Cookie
[Responder-Cookie [Stack-Proposal Stack-Configuration-Data]]
[NSLP-Data]

- Confirm

The Confirm message should be sent in Connection mode if a messaging association is being used for this routing state, and must be sent before other messages for this routing state. If no messaging association is being used, the Confirm message should be sent in Datagram mode. The Confirm message must echo the MRI (with inverted direction), SID, and Responder-Cookie if the Response carried one. In Connection mode, the Confirm message must also echo the Stack Proposal from the Response so it can be verified that this has not been tampered with. The first Confirm message on a new association must also repeat the Stack Configuration Data from the original Query in an abbreviated form, just containing the MAHoldTime. The objects included in a GIST Confirm message are the following:

Table 3.5: Confirm message

Confirm
Common-Header
Message-Routing-Information
Session-Identification
Network-Layer-Information
[Responder-Cookie [Stack-Proposal [Stack-Configuration-Data]]]
[NSLP-Data]

- Data

This message is simply used to transfer NSLP data between nodes. It can be sent in any of the three transfer modes, in Connection mode over a MA, unreliably and insecurely in Datagram mode and in special cases in Query mode, allowing NSLP data to be sent downstream without installing state within the nodes. The objects included in a GIST Data message are the following:

Table 3.6: Data message

Data
Common-Header
[NAT-Traversal-Object]
Message-Routing-Information
Session-Identification
[Network-Layer-Information]
[NSLP-Data]

- Error

An Error message reports a problem determined at the GIST level. GIST Error messages can be sent both in Datagram and Connection mode, depending on the transfer mode of the message that caused the error. The objects included in a GIST Error message are the following:

Table 3.7: Error message

Error
Common-Header
[NAT-Traversal-Object]
[Network-Layer-Information]
GIST-Error-Data

- MA-Hello

The MA-Hello message is just used to refresh a MA, indicating that the sending node wants to keep the MA open. A GIST MA-Hello message only consists of a common header. The object included in a GIST MA-Hello message is the following:

Table 3.8: MA-Hello message

MA-Hello
Common-Header

According to Figure 3.10, NSLP located at query node wants to send some data to its downstream peer as configured in MRI. The message can be either an asynchronous message or a part of a chain of downstream message. To configure the kind of message the application most performs an API call to the GIST instance running at that node, including NSLP payload, NSLP-ID, direction bit, and SID value. If this flow is new, new SID values most be generate. As we can see in the Figure 3.10 the message will be set reliably but does not need it to be sent securely.

The GIST instance running at the querying node will process the API call and look up if there is any state installed, keyed by the NSLP-ID, MRI and SID. Considering this

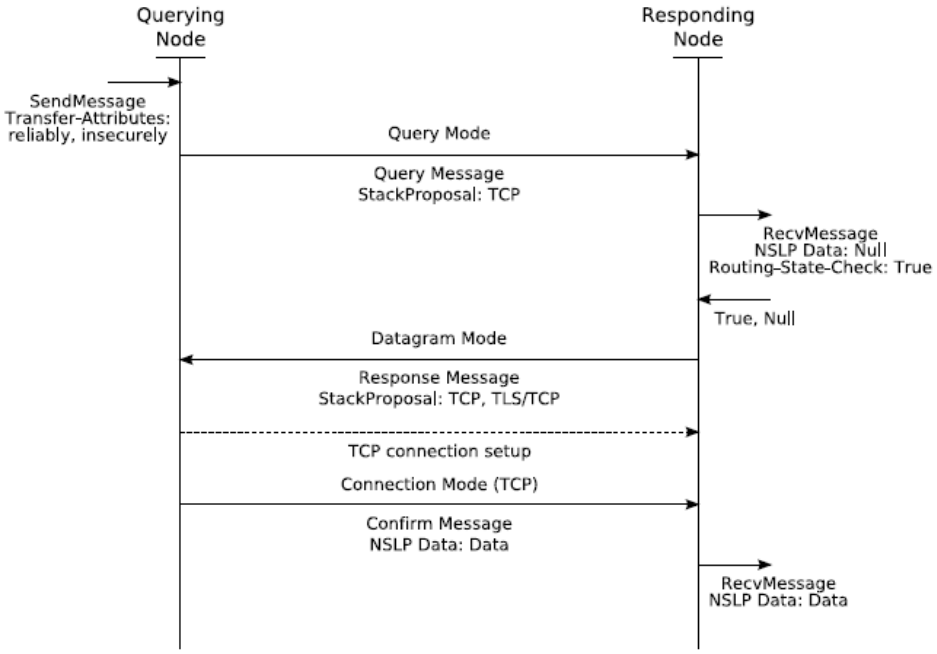

Figure 3.10: GIST message exchange between two nodes

message the first one, a new SID most be generate, so in this message, the payload cannot include in the Query message, but will include those protocols it supports that obey the desired transfer attributes in the protocol stack, in this case only TCP. The NSLP data will be queued internally. Once it has constructed the Query message, it will send it towards the MRI destination with a RAO included in the IP header.

The first node, how will intercept the Query message will be the Responding node. After this node intercept the message the GIST instance running at this node will look up if it has any state installed for this combination of NSLP-ID, MRI and SID. Once it has found that it does not, it will issue an API call to the NSLP application with no NSLP Data and with Routing-State-Check flag set to true. Next GIST will ask the NSLP application if it want to send up routing state. As we can see in this example the NSLP application will answers, that it does want to set up state and that it does not have any Payload to include in Response message. Then GIST will uses NLI included in Query message to determine the identity of the Querying node and looks up if it has a MA installed. It will ensure if TCP connection still active. In case that it does not, it will know that a new MA will be needed to set up, as all protocol stacks, and port numbers, supports in Response message.

As we can see in the example Response message will include both TCP and TLS over TCP. This message will be sent in Datagram Mode to the Querying node, having learned the destination IP address from the NLI in the Query message and the UDP destination

port from the source UDP port of the Query message.

Once the Querying node receives the Response message, it will match this with the state present within Gist and choose the MA it wants to use based on the stack-Proposal of the Response message, in this case TCP. According with these two parameters NLI, and Stack-Configuration-Data from the Responding node it will make a TCP connection to the responding node and send a confirm message over this new MA. Included in the confirm message is the NSLP payload that GIST had internally queued up for this routing state. The NSLP payload can then finally be delivered to the application at the Responding node.

3.4.6 NSLP

NSIS Signaling Layer Protocols or NSLPs, each run signaling application-specific functionality. Examples of NSLPs include the QoS NSLP for resource reservation signaling, the NAT/Firewall NSLP for middlebox configuration, and the Metering NSLP for Network measurements. In this section will be described each signaling application.

QoS-NSLP(NSIS Signaling Layer Protocol)

NSIS was meant to replace RSVP, which was designed to perform QoS reservations. QoS-NSLP application uses the services provided by GIST to install QoS reservations along the entire path a flow, informing every capable router on this path of the parameters of the QoS that is required.

QoS-NSLP also uses soft-state to install these reservations; the messages need to be refreshed, otherwise the reservation will expire after a predetermined amount of time. The reservations are performed end-to-end, meaning one message is propagated along the entire path. A reservation may also be removed by an end-to-end reservation teardown.

Unlike RSVP, QoS-NSLP decouple the reservation processing from QoS parameters, called the QSPEC (QoS Specification) [53], the configuration of this parameters must be opaque to reservation. The addition and removal of actual reservations within the traffic control subsystem of the node, called the Resource Management Function (RMF). The idea is that a flow and thus reservation must travel through several network domains, with different QoS models (QoSM).

Examples of QoSM are IntServ and DiffServ, already discussed. These different domains must receive the required QoS, signaled by the QoS Specification (QSPEC) in the reserve message. As we can see in Figure 3.11, RMF is a core of all NSLP process. This Figure illustrates the subdivision of QoS-NSLP into a processing part and the RMF. The RMF have the responsibility for interpreting the QSPEC, authorizing and admitting reservations and actually performing hem, interacting with the traffic control subsystem provided by the operating system of the host.

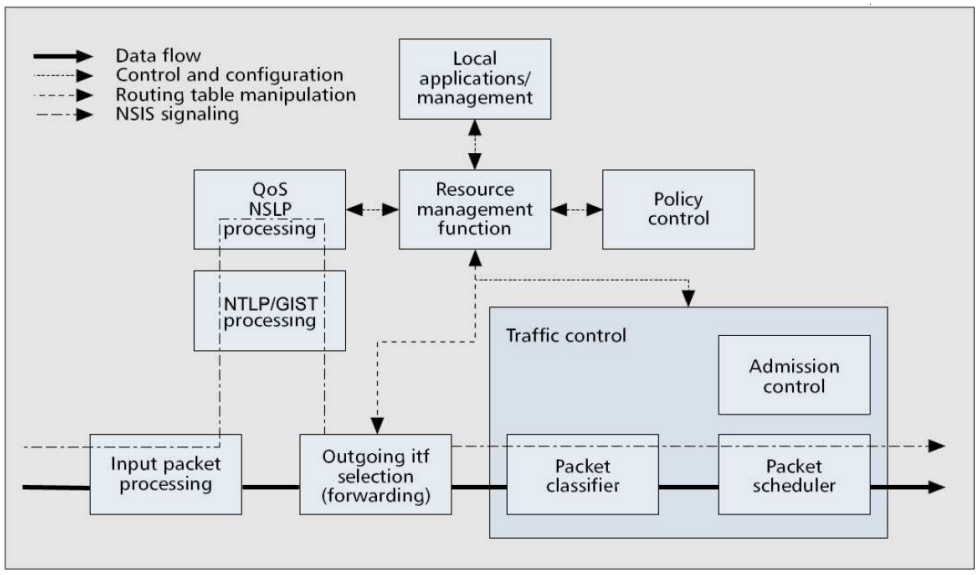

Figure 3.11: QoS-NSLP Architecture

QoS-NSLP Message Processing

The QoS-NSLP processing part uses the following messages, described in [36]:

RESERVE: The RESERVE message is the only message that can alter states in QoS-NSLP entity. This message creates, refreshes, modifies or even removes reservation state.

QUERY: A QUERY message is used to request information about the data path without making a reservation. This functionality can be used to reservations or for support of certain QoS models. The information obtained from a QUERY may be used in the admission control process of a QNE (QNE is an NSIS Entity (NE), which supports the QoS NSLP). QUERY message is not capable of change existing reservation state.

RESPONSE: The RESPONSE message is used to provide information about the result of a previous QoS NSLP message. This includes explicit confirmation of the state manipulation signaled in the RESERVE message, the response to a QUERY message or an error code if the QNE or QNR(the last node in the sequence of QNEs that receives a reservation request for a session) is unable to provide the requested information or if the response is negative. The RESPONSE message does not cause any reservation state to be installed or modified.

NOTIFY: NOTIFY messages are used to convey information to a QNE. They differ from RESPONSE messages in that they are sent asynchronously and need not refer to any

particular state or previously received message. The information conveyed by a NOTIFY message is typically related to error conditions. Examples would be notification to an upstream peer about state being torn down or to indicate when a reservation has been preempted.

QoS-NSLP Example

The example of the functionality of the QoS-NSLP, and messages exchange is illustrated in Figure 3.12. Here is suppose that some application on a NSIS capable host, in this example called QoS NSIS Initiator (QNI), is sending a flow to another host, called the QoS NSIS Receiver (QNR), and that this application wants the flow to receive a certain QoS along the path to its destination.

To initiate the reservation, the QNI will construct a RESERVE message and respective QSPEC object, describing the required QoS parameters. Before creating and sending a RESERVE message, the host will inform its own RMF about the desired QoS, so that it can approve and perform this reservation using its own QoS Model on the output interface of the flow. If the RMF indicates that it does not have enough resources the RESERVE message need not even be send and an error condition is signaled back to the application requesting the QoS. If the RMF return Successful in its reservation, the QoS-NSLP will create a new reservation state, referencing it by a newly created SID. Then the RESERVE message is passed to GIST which transport it to the next QoS NSLP node, this node could be a QoS NSIS Entity (QNE) or QNR. There it is delivered to the QoS NSLP processing which examines the message. With this processing, RMF can make decisions about policy control and admission control.

The exact processing also takes into account the QoS Model being used. Now the node can install reservation, based on the QSpec object in the message. After the successful of reserve installation and the QoS-NSLP creates state for this session, records the RII of the RESERVE to monitor the response, inserts its own RSN into it and send it along to the next downstream node. The response of a flag set in the RESERVE message header, the connection will support reduced refresh messages, it will send NOTIFY message back upstream with an INFOSPEC value that indicates if it can be supported. All of this events will continues until the flow reach the QNR, after the flow reach the QNR and RMF return Successful in response of reservation request, it will generate a RESPONSE message, indicating this success in the INFOSPEC.

Sender Initiated Reservation

This response message is send back upstream and each node will forward it, in the same way of last travel, until reaches the QNI. The QNI node will compare the RII value, and if it corresponds to the originator, it will stop forwarding it. If for any reason the

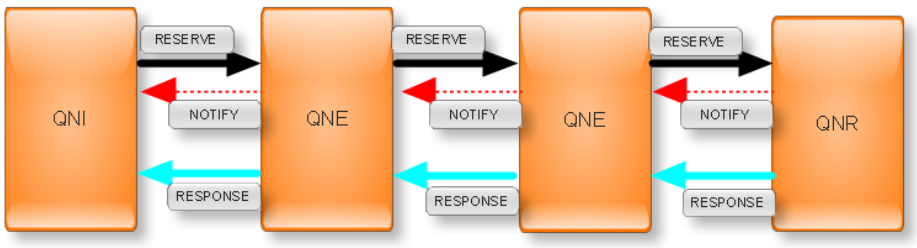

Figure 3.12: Initiated Reservation (Sender)

reservation failed anywhere along the path, a similar RESPONSE would have been sent from the node where the failure occurred, but with an IFOQSPEC object indicating this failure. When QNI receives this information, it will generate a tearing RESERVE message to remove the state from all nodes which have already installed.

Now considering if that flow is in the opposite direction, QNR to the QNI, and that the QNI wants certain QoS for this flow to be specific flag requesting the QNR to send a RESERVE message. The QNR originated RESERVE message will then behave in much the same way as the one in the example.

Receiver Initiated Reservation

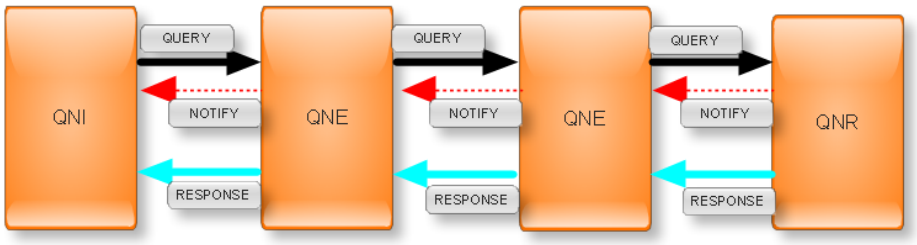

Figure 3.13: Initiated Reservation (Receiver)

NAT and Firewall NSLP(NSIS Signaling Layer Protocol)

This NSIS application will be just introduced in this report because it wasn't use to understand the NSIS mechanism in our implementation. The NAT/Firewall NSIS

Signaling Layer Protocol [44] has been worked and discussed by the IETF NSIS WG. This Signaling Protocol is used for explicit and dynamic configuration of NAT/Firewall systems along the path between two network devices. The main purpose of this application is to enable or block a data flow between any two end points. The Framework of NAT/FW NSIS have the same characteristics of the others NSIS applications already considered, as layer topology, and transport layer protocol supported by Gist.

To illustrate this scenario we will consider an example of NAT/FW NSLP topology. This network is separate into two distinct administrative domains, e.g., namely Domain A and Domain B.

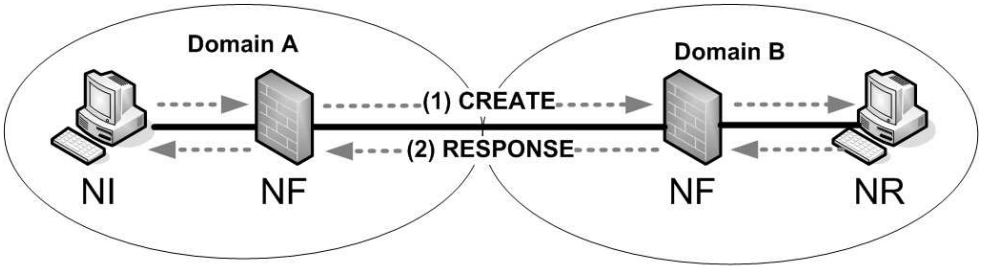

Figure 3.14: Example of Firewall Scenario

The NSLP Initiator (NI) sends NSIS NAT/FW NSLP signaling messages along the data path to the NSLP Responder (NR). Along the path, the signaling messages traverse intermediate NSLP Forwarders (NF). NFs process the messages and additional policy rules or NAT bindings might be installed for the upcoming data traffic.

Metering NSLP

This NSIS application is the solution adopted, for the main propose of this thesis. This application seems to be a good choice to perform inter/intra domains measurements. As this NSIS application is steel on draft's, we propose a possible specification, and implementation prototype. It will be described in detail in the next chapter.

Chapter 4

Metering-NSLP Design and Implementation

In this chapter we will attempt to describe exhaustively all reasons that motivated the design choices made during the implementation of Metering-NSLP in INESC Porto.

At the time of writing, all of the Metering-NSIS specifications were still in the draft stage and so had not yet reached the status of final document. We have supported all implementation work, with Draft's specifications. They have been developed and optimized by IETF WG, this working group specified all Metering-NSLP Framework. As recommended by IETF our implementation and specification of Metering-NSLP was supported in QoS-NSLP architecture.

All implementations were guided from IETF Draft's about Metering-NSLP [45]. Although all Draft's had expired the Working Group gave us assurances of that work has been done, and will be developed even more. We expect that NSIS signaling Protocol application, Metering-NSLP reach the status of final document.

4.1 Metering-NSLP Design

The design for the Metering NSLP and the processing of M-NSLP messages is in some aspects similar to QoS NSLP. Next we will define Metering-NSLP terms, logical model processing in a Metering Entity, and Metering-NSLP messages.

4.1.1 Metering-NSLP Terms

Metering Record

A Metering Record describes flow characteristics for a particular flow. Examples of such data are packet counter and time information.

Monitoring Probe

A Monitoring Probe is an entity that examines data flows in order to gather Metering Records.

Metering Entity

A Metering Entity is a node that is equipped with one or more Monitoring Probes. A Metering Entity should process the Metering Records, e.g. by aggregating them or by sampling them, before they are exported to a Collector. A Metering Entity has other functions, for example, for processing of NSIS signaling.

Collector

A Collector receives Metering Records from one or multiple Metering Entities. These Metering Records can be aggregated or correlated by the Collector. The Collector is typically not co-located with a Metering Entity. Another protocol is needed to transport the Metering Records from the Metering Entity to the Collector (in this implementation we use IPFIX).

Metering Configuration State

A State used by the Metering Manager to configure the Monitoring Probe.

Metering Manager

A unit co-located with the Monitoring Probe that communicates with M-NSLP processing. It holds Metering Configuration State, this state is used to configure the Monitoring Probe.

4.1.2 Metering-NSLP Messages

The Metering-NSLP messages are described in [45]

CONFIGURE

CONFIGURE is a M-NSLP request message. It is used to create Metering Configuration State in a Metering Entity. The format of a CONFIGURE message was implemented as follows, according to [45]:

Table 4.1: Configure message

Configure
Common HEADER
proxyFlag
scopingFlag
tearFlag
ReplaceFlag
ReducedRefreshFlag
BreakFlag
Msn
Mspec
SESSIONID
LifeTime
MNESelection
MNECounter
SecondaryMSpec

REFRESH

REFRESH is a M-NSLP request message. This message is used to extend or termination of some session by configuration of M-NSLP session lifetime. This message is also used to detect route changes at the NSLP layer. The format of a REFRESH message was implemented as follows, according to [45]:

Table 4.2: Refresh message

Refresh
Common HEADER
proxyFlag
scopingFlag
tearFlag
BreakFlag
Msn
Mspec
SESSIONID
LifeTime
MNESelection
MNECounter
SecondaryMSpec
InverDirection

OPTIONS

OPTIONS is a M-NSLP request message. Although this message wasn't specified by our implementation we will describe it, this message is used to evaluate the metering capabilities of MNEs along the signaling path.

Table 4.3: Options message

Options
Common HEADER
proxyFlag
scopingFlag
tearFlag
BreakFlag
Msn

RESPONSE

The message RESPONSE is used to confirm the request, the MSN field must be the same as in the corresponding M-NSLP in order to match with the appropriate request. Another feature of this message is the lower session lifetime value, in SessionLT, comparing with session lifetime requested by the Metering NSIS Initiator (MNI). In case of successful confirmation of the request, none MSPEC is transported in this message.

Table 4.4: Response message

Response
Common HEADER
proxyFlag
scopingFlag
BreakFlag
infoSpec
Msn
Mspec
LifeTime

NOTIFY

The NOTIFY message is an asynchronous notifications message used to convey information to a MNE. It differs from RESPONSE messages in that it is sent asynchronously and does not need to refer to any particular state or previously received message. The information conveyed by a NOTIFY message is typically related to error conditions. An example would be a notification to the MNI about state being torn down.

Table 4.5: Notify message

Notify
Common HEADER
proxyFlag
scopingFlag
infoSpec
Mspec
SecondaryMSpec

4.1.3 M-NSLP Message Objects

Here we will describe each object included in all M-NSLP messages, mentioned above.

Session Identifier (SID)

The SID object is statistically unique identifier for a MNSLP session, this object isn't carried by M-NSLP messages, it is rather carried by GIST. Each new M-NSLP session created in MNI must generate a unique SID, this identifier must be provided to GIST via the GIST API. All other Metering NSIS Entities will receive the SID via GIST API with read permission only.

Message Sequence Number(MSN)

MSN is used to detect duplicate and lost Metering NSLP messages. It is also used to match an incoming RESPONSE message to the appropriate request.

Selection of Metering Entities(MNESelection)

This object is required to determine which MNEs will actually take part in the metering. The value of MNESelection can be one of the following:

FIRSTandLAST: both the MNI and the MNR take part in the signaling.

ANY: any available MNE can perform the metering.

ALL: Each MNE capable of executing this metering request will perform it.

Session Lifetime(SessionLT)

This object carries the requested lifetime for a M-NSLP session in a CONFIGURE / REFRESH message or the granted session lifetime in a RESPONSE message, in milliseconds. When a M-NSLP session expires, the Metering Manager MUST configure the Monitoring Probe to stop the Metering. A value of zero for the 'Session Lifetime' object

leads to immediate termination of the corresponding session. We configure lifetime with a schedule, all configuration objects of schedule are included in MSPEC.

MNSLP Hop Count (MNSLPHopCount)

This object carries the number of previous MNEs on the signaling path for this session. This object is an integer which starts by zero at the MNI and must be increased by one at each M-NSLP hop on the signaling path.

Information Code (INFO)

This object carries the response code, which may be an indication for either a successful or failed request depending on the value of the 'response code' field.

Metering Specification (MSPEC)

As mentioned above, the MSPEC is an object that carries measurement specific parameters. They are interpreted in the Metering Manager Function and is opaque to M-NSLP Processing. Each MSPEC object contains a meter configuration. The MRI provides additional information to the MSPEC object on the flows/packets to be metered, as Destination Address IP, Source Address IP, Destination Port, and Source Port. The combination of the MRI and an MSPEC object contains a complete description of a requested metering task. The MSPEC object is based on IPFIX, this technology describe all measurement specifications like:

- CollectorAddress : This parameter specifies the Address of Collector where all measurements must be exported.
- CollectorPrefix : This parameter specifies the subnet of Collector.
- Mgenstate : This parameter is used to configure the MGEN, and determines if MNI will be a Transmitter, or a Receiver.
- MgenTrafficDistribution : This parameter is used to configure the MGEN, and determines the traffic distribution, the available traffic distributions are: Poisson, Burst, or Periodic.
- Metering : This parameter specifies the kind of measurement like MNESelection(Any, All, FisrtandLast).
- payload : This parameter is used to configure the MGEN and determines the payload value of each packet.

- Startdate : This parameter is used to configure the schedule, defines the beginning of measurement.(Year-Month-Day)

- StartTimeh : This parameter is used to configure the schedule, defines the time at the measurement must begin.

- StartTimem : This parameter is used to configure the schedule, defines the time at the measurement must begin.

- Stopdate : This parameter is used to configure the schedule, defines the end of measurement.(Year-Month-Day)

- Stoptimeh : This parameter is used to configure the schedule, defines the time at the measurement must finish.

- Stoptimem : This parameter is used to configure the schedule,defines the time at the measurement must finish.

4.1.4 Logical Model

The figure 4.1 shows a logical model of the operation of the Metering NSLP and the associated metering mechanism in a Metering NSIS Entity (MNE).

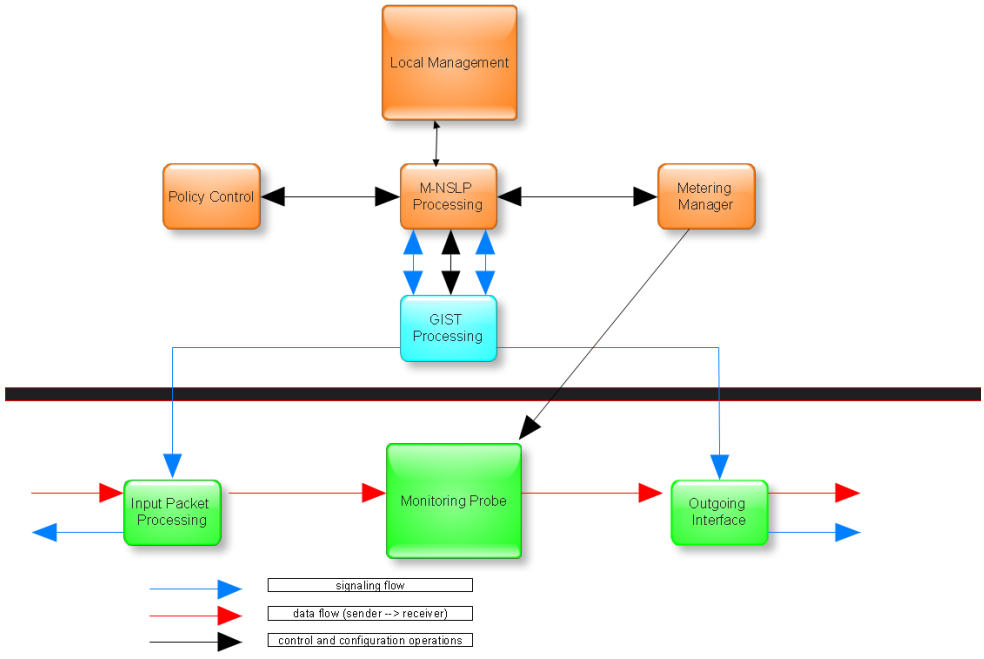

Figure 4.1: Example of an logical model in a MNE

The **Monitoring Probe** collects metering data and processes them into Metering Records, which can be sent to the Collector. The Monitoring Probe is co-located with the Input Packet Processing and the Outgoing Interface.

The **M-NSLP Processing** decides if the current MNE is addressed by this configuring message. If yes, the MSPEC (Metering Specification) objects are extracted from the M-NSLP message and passed to the **Metering Manager**, where they are interpreted and used to install **Metering Configuration State**. The Metering Manager uses this state to configure the Monitoring Probe. The **Policy Control** determines whether the sender of the M-NSLP message is authorized to configure the Monitoring Probe. The block **Gist Processing** is responsible for transporting message from the Metering-NSLP to the next node and for delivering messages coming from the network to the Metering-NSLP.

For delivering the messages the API-call RecvMessage() is used. To send messages from the Metering-NLSP to the next node over the network the API-call SendMessage() is used.

4.1.5 Metering-NSLP Standard Example

All message types were described in 4.1.2, but the better way to provide the functionality overview of the Metering-NSLP is an illustrated example. The message flow for the example described here can be seen in figure 4.2. Suppose that some application on NSIS capable host in this example called the Metering NSIS Initiator (MNI) and that this application wants to start a measurement between him self and an end host called MNR(total measurement), the measurement can be participated by other hosts in the path these hosts are called (MNE).

The Metering-NSLP application will construct a MSPEC, this object contains the measurement parameters that the MNI wants the flow to receive. This parameters can be interpreted for any selected host in SelectionMNE (configured with All, Any, or FirstandLast), and MNR. Before creating and sending a CONFIGURE message, the host will inform its own MMF about the desired measurement parameters.

If the measurement is refused by MMF the CONFIGURE message need not even be sent and an error condition is signaled back to the application requesting new parameters. If the MMF is successful in its measurement, the M-NSLP will create a new measurement state, referencing it by a newly created SID. It will use this state, which among other things contains the MSN for the downstream peer, to generate the downstream CONFIGURE message, which it will pass to GIST.

Included in this CONFIGURE is a SessionID which is unique to this reservation request. The M-NSLP will record this SessionID value to watch for in RESPONSE messages. The first downstream peer for this particular flow, which in this case will be called the Metering NSIS Entity (MNE), will receive the CONFIGURE message and check to see if it has state installed for this SID. In this example it does not, but if it had state installed

it would have done some checks on the MSN of the received message, to prevent message duplication and re-ordering, in case unreliable transport is used in GIST.

It then sends the MSPEC to the MMF, it will indicate if the object can or not start the measurement. In this example the measurement is successful and the M-NSLP creates state for this session, records the SessionID of the CONFIGURE to monitor the response, inserts its own MSN into it and sends it along to the next downstream node. Additionally, in response to a flag set in the CONFIGURE message header, that requests support for reduced refresh messages, it will send a NOTIFY message back upstream with an INFO value that indicates whether or not it supports this. The reduced refresh messages are used to ensure the soft-state characteristic. This chain of events continues until, eventually, the CONFIGURE message reaches the MNR. If the measurement is also successful at this node, it will generate a RESPONSE message indicating this success in the INFO. It will also include the SessionID of the measurement, so that nodes receiving the RESPONSE know which measurement this pertains, and a MSPEC returned from the MMF. This response message is sanded back upstream and each node will forward it until it reaches the MNI. This node determines that it was the originator for this SessionID and will stop forwarding it. If the measurement had failed anywhere along the path, a similar RESPONSE would have been sent from the node where the failure occurred, but with an INFO object indicating this failure. In this case the MNI would receive this RESPONSE and send a tearing CONFIGURE message to remove the state from those nodes that already have it installed. The measurement is then in place along the entire path. Because this measurement is soft-state it will need to be refreshed periodically, which happens asynchronously between peers. What this means is that, unless its own state expires, every node will send refreshing CONFIGURE messages to its downstream peer, which will not be propagated further downstream. Each set of peer has its own timers governing refresh intervals.

If the downstream node has indicated that it supports reduced refreshes, the upstream peer for this node need only include the MSN of the last full CONFIGURE message in the refreshing CONFIGURE. The downstream node will see that the MSN is the same as the last CONFIGURE message and will conclude that the same measurement parameters still apply.

Two other features can be highlighted at this point. Suppose that the flow is in the opposite direction, i.e. from the QNR to the QNI, and that the QNI wants a certain measurement for this flow to be applied. In this case, which is a receiver-initiated measurement instead of the sender-initiated measurement of the example, the MNI can send a REFRESH message with a specific flag requesting the MNR to send a CONFIGURE message. The MNR that triggers CONFIGURE message will then behave in much the same way as the one in the example. Another possibility is the bidirectional measurements. Although measurements in the MNSLP are always unidirectional, session binding between two separate session can be performed, one for the downstream flow and one for

the upstream flow. Each CONFIGURE or REFRESH belonging to one of the two sessions contains a reference to the other session through the use of the SessionID object. This allows for events such as errors pertaining to one of the sessions to be signaled to the other session as well.

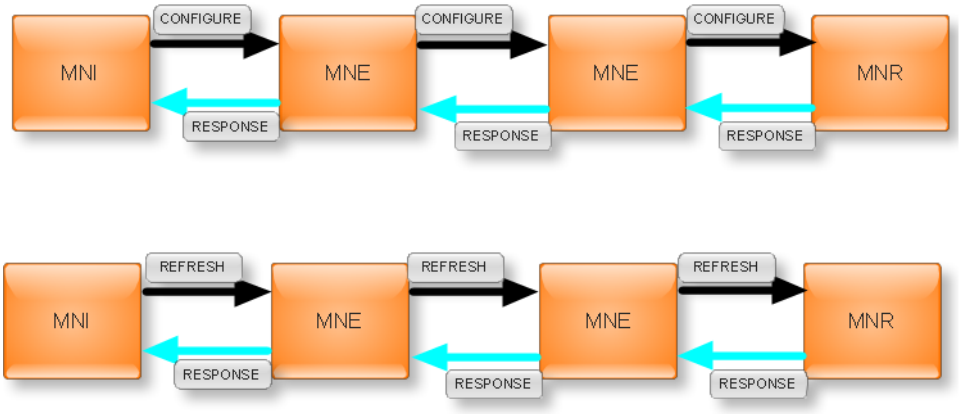

Figure 4.2: Message Flow

4.1.6 Metering-NSLP Example FIRST-and-LAST

The message flow for the example described here with only the MNI and MNR doing the measurement can be seen in figure 4.3. The MNI constructs a CONFIGURE message with the required MSPEC objects, and sends it towards the MNR. The message is interpreted by each MNEs on the data path. Each MNE forwards the message except MNI itself and MNR.

The MNR replies with a positive RESPONSE message. The RESPONSE message is interpreted by each MNE. Each MNE forwards the message except MNR and MNI. MNI and MNR can export their measurements to a collector using IPFIX Protocol, it is the implementation adopted in this work.

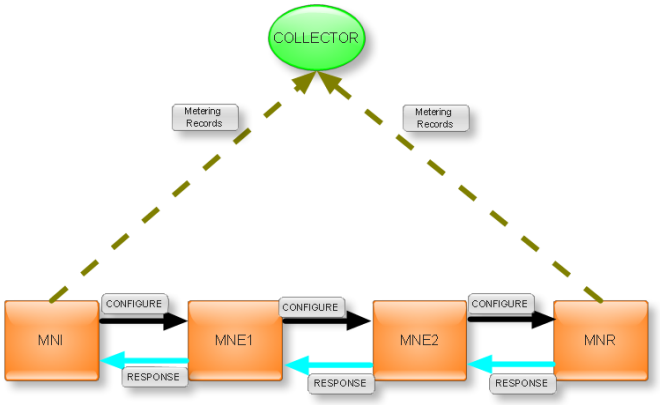

Figure 4.3: MNE First and Last

4.1.7 Metering-NSLP Example ANY

Although, in this work we don't implement this M-NSLP feature, it will be described.

Suppose that some application on the NSIS capable host in this example called the Metering NSIS Initiator (MNI) constructs a CONFIGURE message with the required MSPEC objects, and sends it towards the MNR. Assume 2 MSPEC objects are included, for example, MSPEC1 for 'counting bytes', and MSPEC2 for 'reporting the usage of the wireless link'. Assume further that MNI itself can perform MSPEC1 while MNE2 can perform MSPEC2, for example, because it is an access router managing a range of WLAN networks. MNR can be, for example, the WLAN access point. The CONFIGURE message is interpreted by the MNEs along the data path. MNI removes MSPEC1 from the MSPEC objects list, and forwards the CONFIGURE message to MNE1. MNE1 receives the CONFIGURE message with only MSPEC2 in the MSPEC objects list. However, MNE1 does not have the capabilities to meter and report the usage of the wireless link, and forwards the CONFIGURE message to MNE2.

MNE2 interprets the CONFIGURE message, and removes MSPEC2 from the MSPEC objects list. Furthermore, it notices that the MSPEC object list is now empty. Therefore, MNF2 replies with a positive RESPONSE message and becomes Responder for this session. Subsequent signaling for this session is stopped at MNE2. MNE2 starts the metering, and constructs a positive RESPONSE message and sends it to MNE1. MNF1 receives the RESPONSE message and forwards to MNI. MNI finish the measurement.

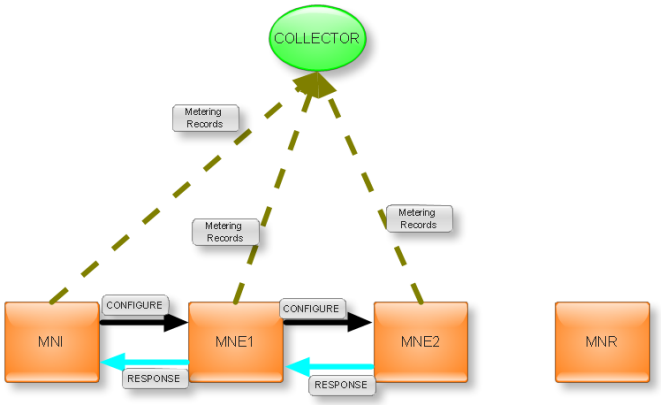

Figure 4.4: MNE ANY

4.1.8 Metering-NSLP Example ALL

The MNI constructs a CONFIGURE message with the required MSPEC objects, and sends it towards the MNR. The message is interpreted by each MNEs on the data path. Each MNE forwards the message depending on whether they are taking part of the metering process, and export results to a collector with IPFIX protocol. Each MNE uses passive measurements. The MNR replies with a positive RESPONSE message. The RESPONSE message is interpreted by each MNE and sends to MNI.

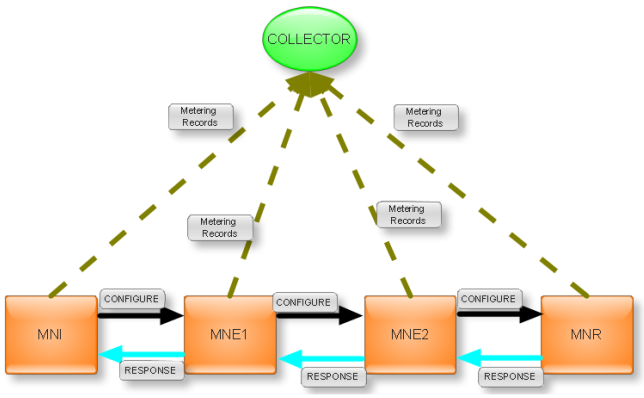

Figure 4.5: MNE ALL

4.2 Metering-NSLP Implementation

The implementation of the Metering-NSLP as well as the GIST code is written in C++ and the main target platform is Linux. The software design is object-oriented and each protocol layer is running as a single-threaded process. The communication between the NSLP and NTLP level is done via UNIX sockets.

Linux was chosen because, combined with the large number of applications available, it provides an ideal testbed for developing new network applications and protocols. Because of its open source, any adjustments that may need to be made to allow GIST to operate can be made. The version of Linux was used was Ubuntu 7.10, not for any reason in special. Initially C++ wasn't a choice, because we start the implementation based in QoS-NSLP, and it was written in C++ language, also GIST used the same language.

4.2.1 Program Structure

For a conceptual overview of the components within the Metering-NSLP application, the figure **??** is a good example.

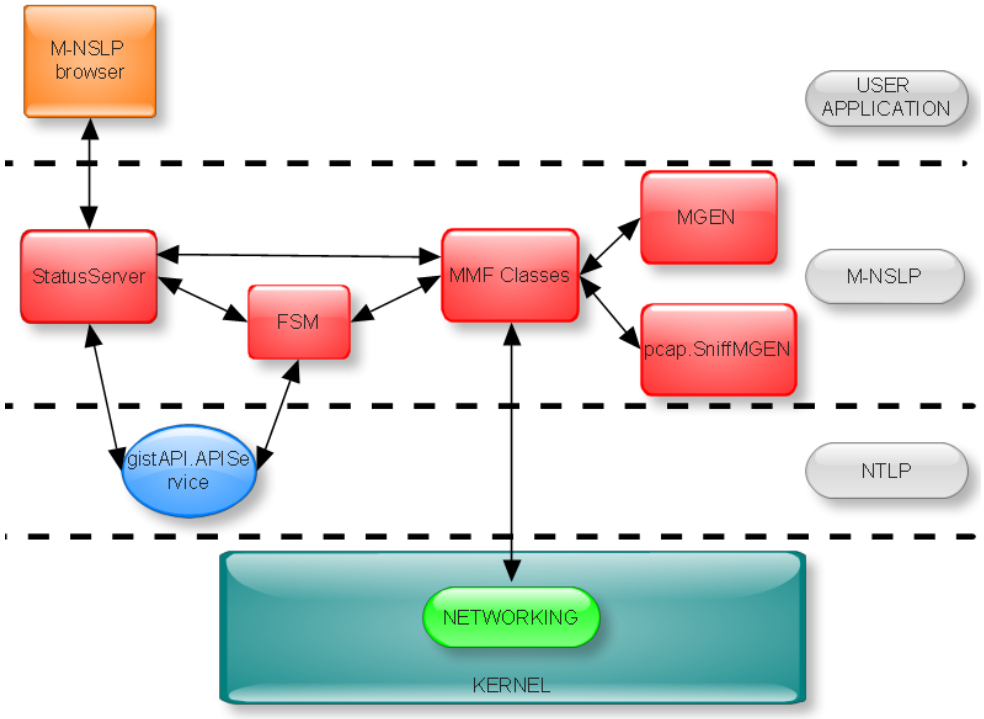

Figure 4.6: Metering-NSLP System Diagram

This same diagram can be seen as Metering-NSLP layers, which are mostly those dealing with MSPEC processing and traffic control interaction. The message handling component is represented in figure 4.7, by the StatusServer class, while the Measurement

Management Function component can be represented by any class in the meteringMMF module.

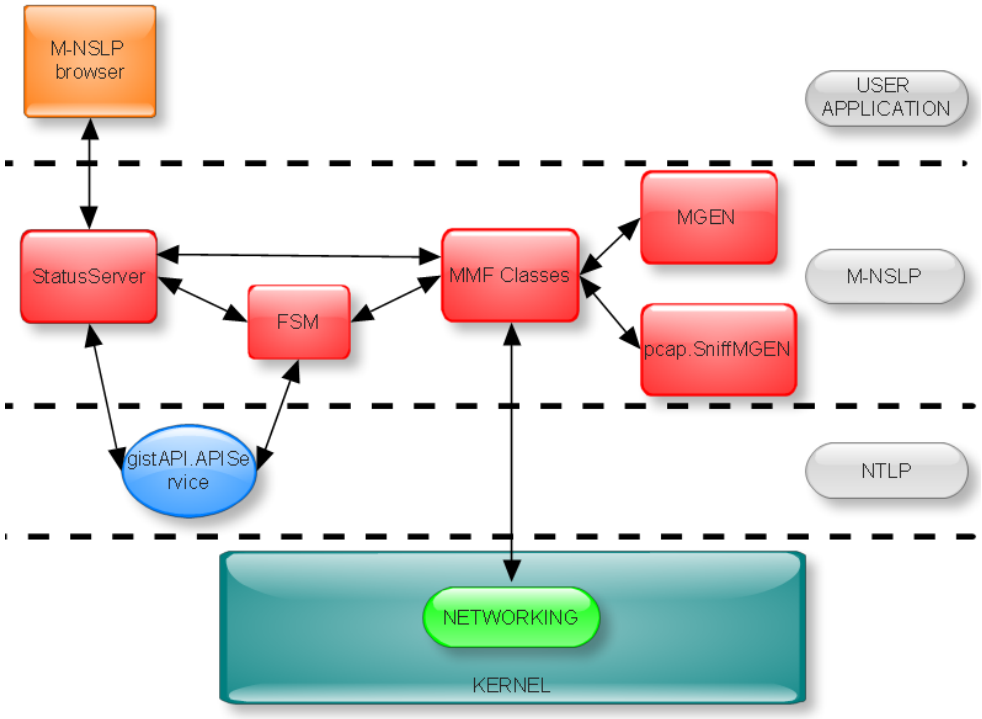

Figure 4.7: Metering-NSLP Implementation Diagram

4.2.2 User Application API

The **user application** used in M-NSLP implementation was written in php, the php code was exported with C++, using Sockets. This interface application is used to configure all MSPEC, to ensure the measurement.

When all fields from the form are completed, they are parsed in statusServer and MSPEC can be built, the next step is the calling of MMF, who will interpret all the information, and therefore will process it.

Depending on what is specified on the form, and the kind of node (MNI, MNE, or MNR), different decisions will be taken if the field included in MNESelection was FirstandLast only MGEN will be called and processed, performing the measurements, between end-host MNI and MNR. If the field included in MNESelection was ANY or ALL, MGEN was also executed between end-host but also SniffMGEN will be executed in each selected MNE, performing passive measurements in these nodes. All this process is run by a State Machine(FSM). A detailed description of FSM is given in next section.

For the partial implementation currently in place, the interface methods are the following:

- **processRequest** This instructs the MMF to evaluate what kind of message will be processed in case of Measure message, MMF constructs an initiating measurement message, with objects and a MSPEC with all measurement configuration as Payload, MNESelesction, Protocol, Metering Distribution, Destination IP and port, Source IP and port, and schedule configuration start/stop time for the measurement. This method is generally started if the application API is called. In case of Response message RMF inspects, if the MSPEC returned and either approves or disapproves it. Upon disapproval a tearing MEASURE is sent.

- **toRefresh** This instructs to built refreshing measure message for a particular session, taking into account information gathered about reduced refresh support of the downstream node. These messages are generated asynchronously, because if the timer expires within the persistent state in each node.

- **tear** The MMF will construct a tearing measurement with NULL MSPEC value for a particular session. This can be either as a result of application API interaction, following an error condition, or an end of measurement time.

4.2.3 Finite State Machine (FSM)

A **Finite State Machine (FSM)** can be useful with even two states when having many events. FSMs consist of four main elements: states, state transitions, input events and a set of rules to allow state transitions. For a given state, a certain event triggers an action: the execution of a callback function (the event handler).

All possible states, events and resulting callback functions are stored in an array, the jumpmatrix. The FSM can be operated using two functions. TriggerEvent() executes a state event. SetState() manipulates the internal state of the FSM. In this implementation we will adapt the QoS-NSLP FSM to our M-NSLP specification, another reason is the poor specification released in [45].

The following Fluxograms will illustrate our FSM.

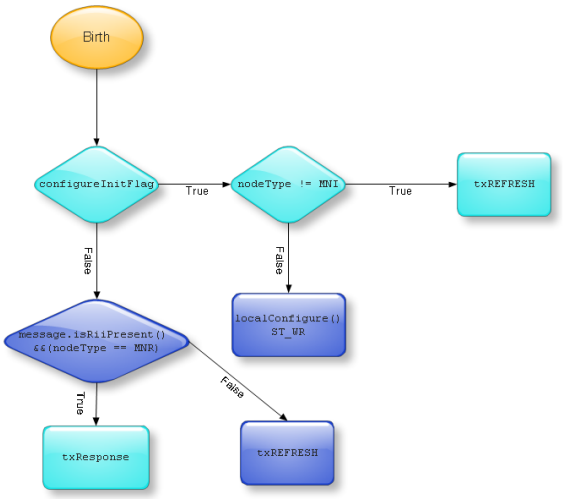

Figure 4.8: Refresh message FSM

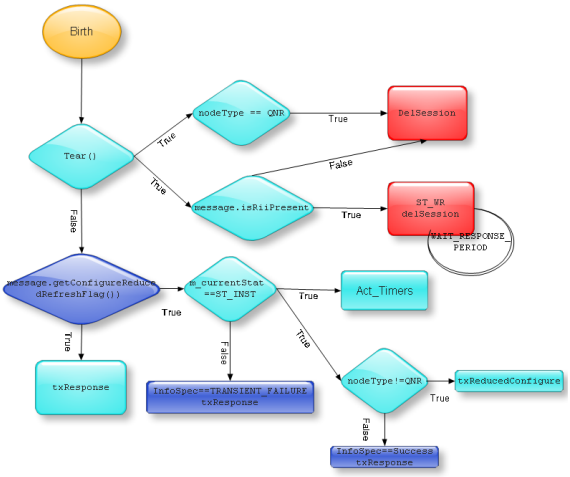

Figure 4.9: Configure message FSM

The other diagrams are very similar, next we will illustrate the FSM suggested by [45], it could be improved in future with a new M-NSLP update.

The State Machine of an M-NSLP session has three main states [45], illustrated above, and described below. The three main States are:

- closed : At this state, the MNE does not keep any state for the session
- pending : At this state a CONFIGURE message has been received. The pending state consists of 2 sub-states

  1. pending.forward : The MNE is not taking part in the metering process. Instead, it is just expecting a confirmation to know that the session is active

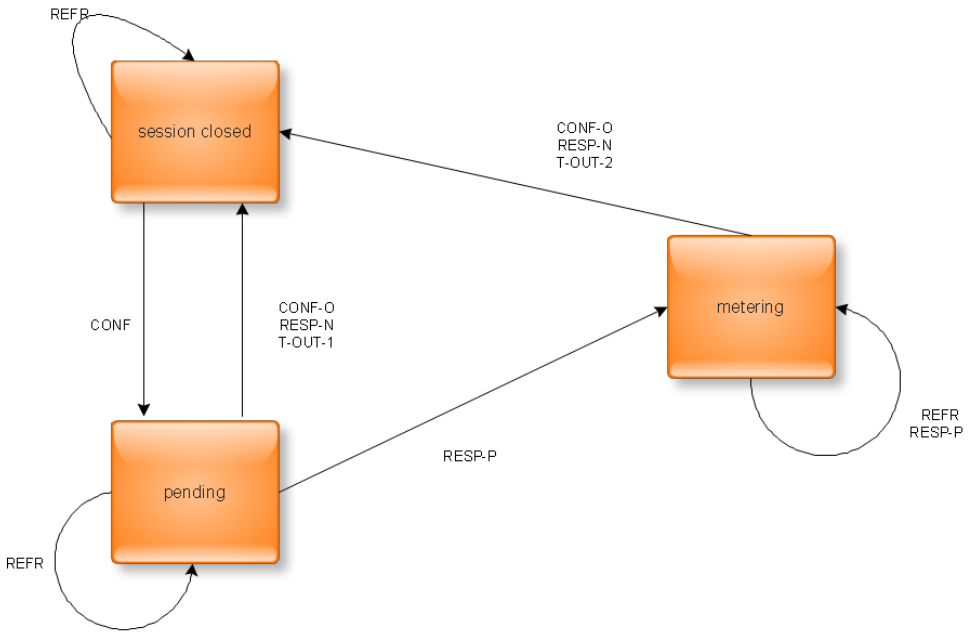

Figure 4.10: M-NSLP Session State Machine

and some other MNEs will be metering. It can be an example of FirstandLast measurement, where each MNE is configured with this state.

2. pending.participating : With this state MNE is actively taking part in the metering process. The MNE has been configured. But MNE is not ready for measurement process yet.

- metering : At this state the configuration request has been successfully confirmed with an appropriate RESPONSE message. The metering state consists also of 2 sub-states.

  1. metering.forward : The MNE is not actively taking part in the metering process. Instead, it is informed that the session is active and there are some other MNEs along the signaling path for this session that took over the metering process.
  2. metering.participating : With this state MNE is actively taking part in the metering process. The MNE is metering according to a metering configuration received by a CONFIGURE message.

Each state changes are caused by the following events:

- CONF : This transition is triggered by a CONFIGURE message that is received and processed successfully by the MNE and that identifies a new session.
- CONF-O : This transition is triggered by a CONFIGURE message that is received and processed successfully by the MNE and that identifies an existing session in

state 'pending' or 'metering'. This transition leads always to the state 'closed' in order to avoid inconsistencies with old configurations.

- REFR : This transition is triggered by a REFRESH message that identifies an existing session in state 'pending' or 'metering'.

- RESP-P : This transition is triggered by a positive RESPONSE message that indicates that a metering configuration request or a refreshing request can be activated. This transaction always leads to state 'metering'.

- RESP-N : This transition is triggered by a negative RESPONSE message. This transaction always leads to state 'closed'.

- T-OUT-1 : This transition is triggered by a timeout of a session in state 'pending'. The time interval for this timeout is typically large enough in order to tolerate network failures, network congestion and slow MNEs along the signaling path. This transition always leads to state 'closed'.

- T-OUT-2 : This transition is triggered by a timeout of a session in state 'metering'. The time interval for this timeout is defined by the MNSLP session lifetime. This transition always leads to state 'closed'.

4.2.4 Measurement Management Function (MMF)

The **MMF** listens for incoming messages from itself and for messages coming over the network from GIST. For each new SessionID (SID), a new State Machine (FSM) is created, a table is searched for the given SID. If it is not found, a new FSM is created and its address together with the SID is added to the table. If the SID is found, the address of the FSM is returned. Incoming Messages are parsed into objects.

The FSM is triggered with the corresponding event to the message. If, e.g. a CONFIGURE message has arrived the event rxNslpMessage is triggered. If the type of the message is a REFRESH with set R-Bit, a receiver-initiated measurement is requested and the node position is MNR; in all other cases, the node position is MNI. If the FSM returns with STIDLE, it is deleted together with its table entry.

4.2.5 MGEN

In this Metering-NSLP implementation **MGEN** is a key component for the active measurements between all nodes participating for all MNESelection configuration i.e. FirstandLast, Any, and All. This tool that generates real-time traffic, and packets with payload variable, can be all configured by the **user application**, these configuration objects are included in MSPEC. This tool can be configured to generate or to receive traffic,

so MNI can be one of these i.e. MNI can be the generator and MNR the receiver or the other way around.

This process is triggered by FSM and MMF in conjunction. MMF will evaluate each message and the MSPEC objects, when a CONFIGURE message is required, and a measurement is scheduled, MNI will have one of two states already referenced, generator or receiver. The opposite process will be taken by the MNR. If the MNI is the generator MNR will be the receiver, MNR will receive generated packets and log them for analysis. This log data can be used to calculate performance statistics on throughput, packet loss rates, communication delay, and more.

The MGEN can be set with various parameters that define a particular traffic, like video, audio, and bulk data. This application uses this tool to generate the required traffic with varying characteristics. With this characteristic we can simulate various types of streams for our experiments and validation. For example multimedia video traffic can be defined as having bursts at irregular intervals through MGEN's parameters. The bursts can be generated with varying parameters. The BURST pattern generates bursts of other MGEN pattern types at a specified average interval.

Another kind of traffic can be simulated with PERIODIC, and POISSON distributions. When the measurement ends the M-NSLP reads the logfile generated by MGEN and count the packet loss rates, communication delay, and Jitter. After measurement has finish, the information is exported to a collector. The IP address of this collector is included in MSPEC object.

4.2.6 SniffMGEN

This packets capture application, is used by each MNE participating in the measurement. This application is triggered by the reception of CONFIGURE message in MNE.

The SniffMGEN was built on top of Sniffex [26]. The main objective of this tool is capturing all MGEN packets, the specifications of these packets are defined bellow, with this capture a log file is built with all the measurement information's. The data in the logfile provides the elements to calculate the packet loss rates, communication delay, and Jitter. This can be also exported to a collector.

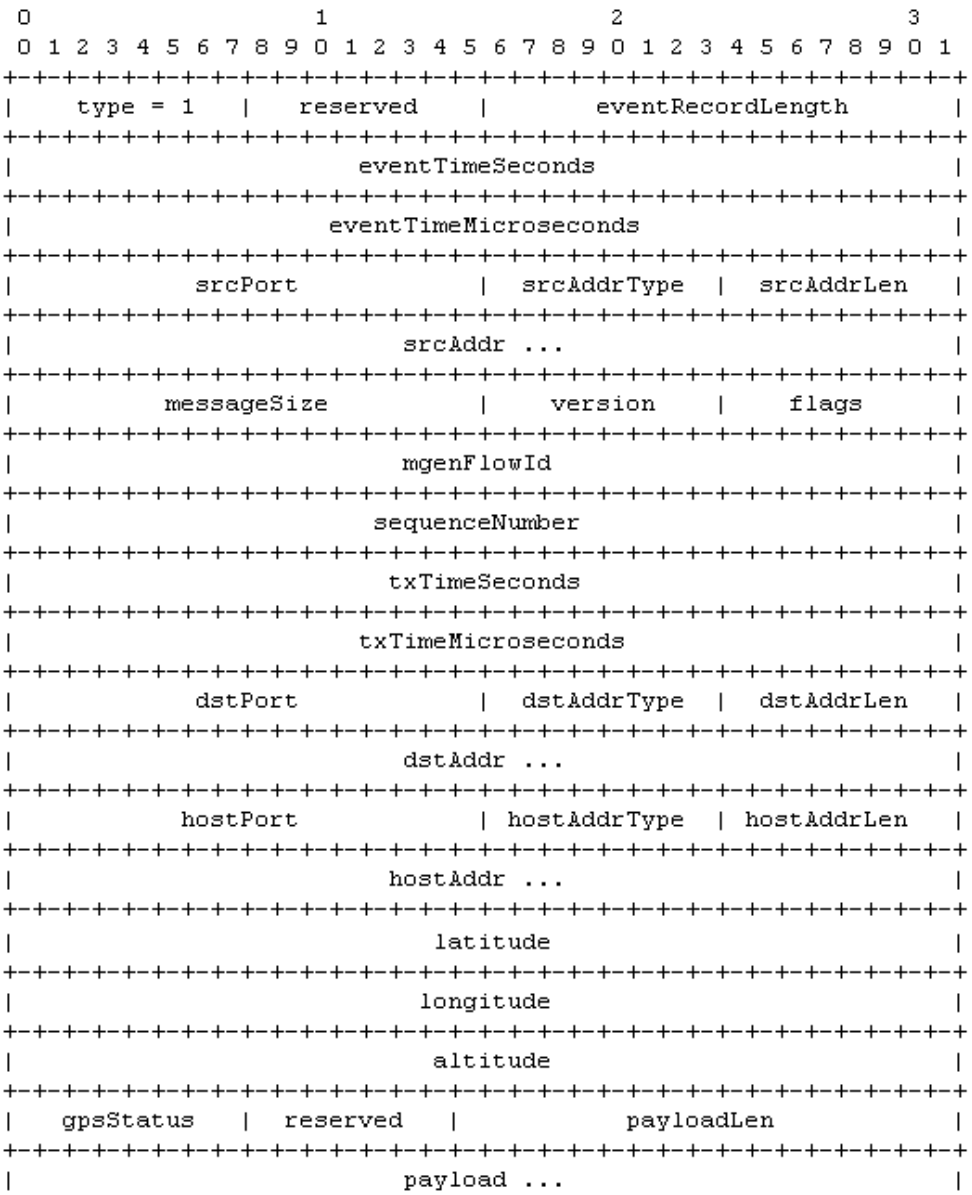

Figure 4.11: The format of MGEN message

Chapter 5

Functional Experiments

This chapter presents the experiments that have been performed on the M-NSLP prototype implementation discussed in Chapter 4. The M-NSLP need to be tested to confirm its usability and software design evaluation. During implementation many tests were performed to remove any bugs that may appear.

The simple setup used for performing these tests will be described first, after this there will be a brief discussion of early tests performed during implementation.

The main section of this chapter will be comprised of the tests performed with the final implementation. After this there will be a final discussion about all results.

5.1 Testbed Setup

5.1.1 Tests during Implementation

To perform tests with M-NSLP, different elements need to be configured such as network configuration, and a NTP server for synchronization of clock between all network elements. However, because of resource limitations, it was not possible to have a real-life testing network. For these early tests we only use three standards PC and one wireless router. The synchronization is a key component in terms of network performance analysis. We used an NTP server to synchronize the clock of each computer, and we reference them to NTP server in our network. It provides accuracy typically within a millisecond on LAN.

The Network configuration can be seen in the following Figures.

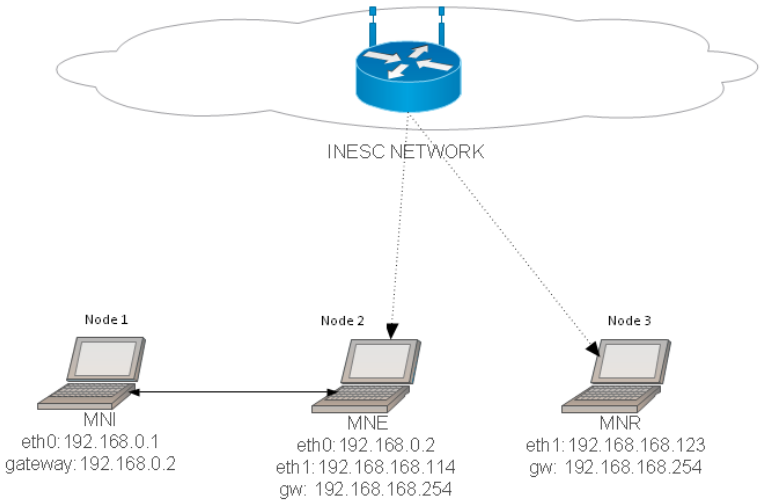

Figure 5.1: Implementation Test Configuration

(Node 1)

The first element is configured as a GIST aware node. It will play as NSIS Initiator, which basically means that it will initialize NSIS signaling, as a Transmitter. This node will configure MGEN tool to generate traffic, it will cross all network, until the NSIS receiver. This node is further configured as follows:

- network interface: eth0
- IP address: 192.168.0.1
- gateway: 192.168.0.2
- Supported NSLPs and measurement tools: M-NSLP, MGEN, SniffMGEN(if required, if node type change to MNE)

(Node 2)

The element in the middle is configured as a GIST aware node. It will play as NSIS Entity, which basically means that it will forward NSIS signaling, or perform passive measurements, if configured, with SniffMGEN and then forward signaling. This node will sniff MGEN packets to compare timestamps, if it not configured to perform measurements it only forwards signaling.

This node is further configured as follows:

- network interface: eth0, eth1
- IP address: eth0:192.168.0.1, eth1:192.168.168.114

- gateway: 192.168.168.254
- Configured with IPForwarding
- supported NSLPs and measurement tools: M-NSLP, MGEN, SniffMGEN(if required)

(Node 3)

The last element is configured as a GIST aware node. It will play as NSIS receiver, which basically means that it will receive NSIS signaling. This node will configure MGEN tool to capture traffic from NSIS Initiator. This node is further configured as follows:

- network interface: eth1
- IP address: 192.168.168.123
- gateway: 192.168.168.254
- supported NSLPs and measurement tools: M-NSLP, MGEN, SniffMGEN(if required, if node type change to MNE)

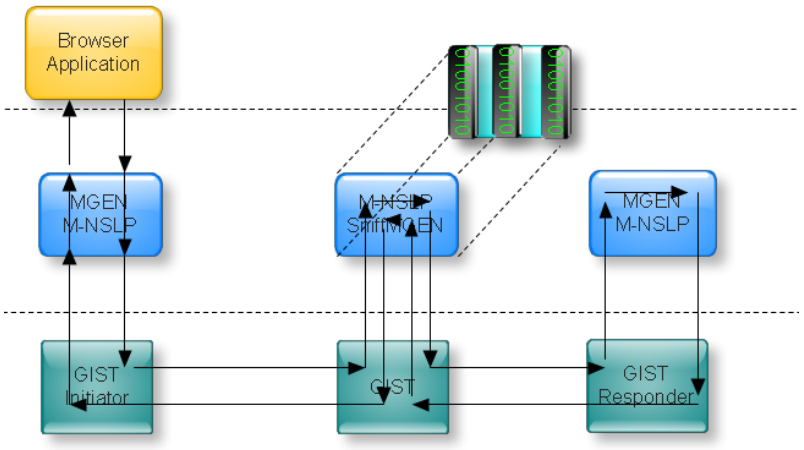

Figure 5.2: Example of MGEN traffic flow

The application interface is exported, it can be displayed at port:9999. The interface has all fields to configure a measurement. The following picture shows how it looks like. Figure 5.3 displayed the front page of application.

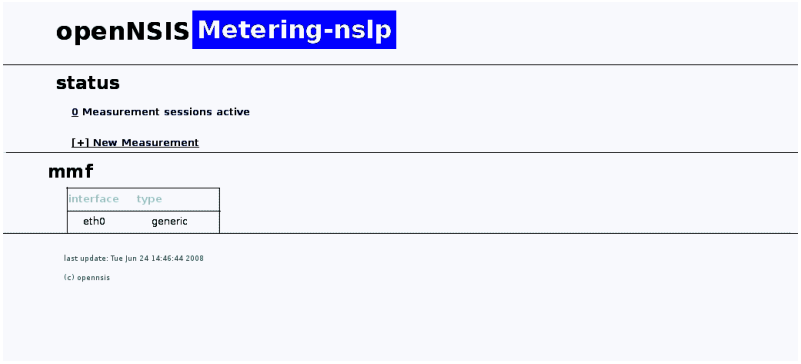

Figure 5.3: Interface Front Page

After starting the configuration, another form will be present in figure 5.4, now MSPEQ can be configured, to perform some kind of measure.

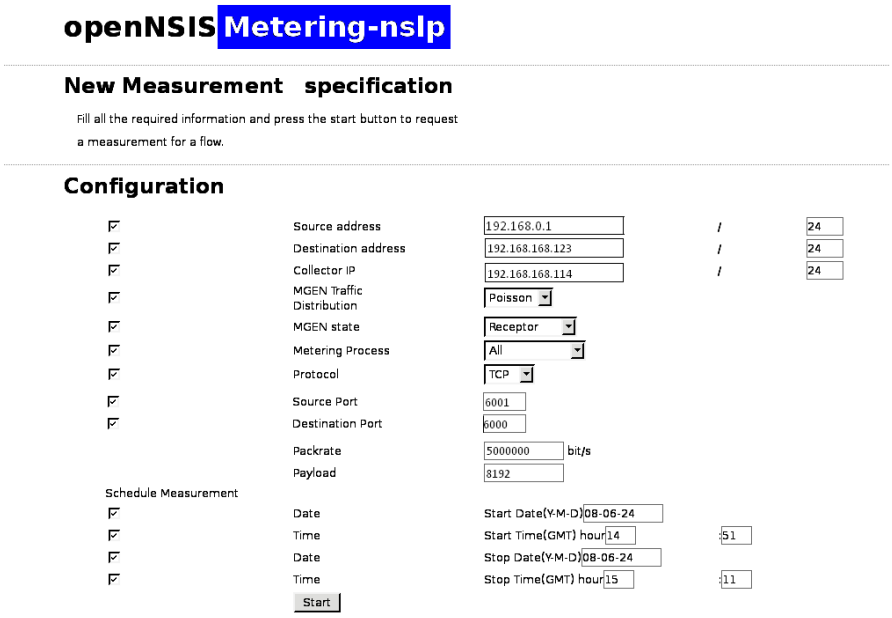

Figure 5.4: Interface Measurement Configuration

The result of connectivity is provided be the following picture 5.5, With the information provided by this picture is possible to know how many sessions are actives.

In the next section will be discussed all results achieved in final tests.

Figure 5.5: Interface Active Sessions

5.1.2 Final Tests

In this section we will present all results archived in Testbed platform, located in University of Aveiro.

The Testbed simulate connectivity inter/intra domains. Its architecture is better understood with a scheme 5.6. Despite it is very complete and well structured, in these measurements only the access router was configured with M-NSLP, supported by Gist as under layer.

The tests were performed from host *Haitian* to host *Sylar*, MNI and MNR respectively. The hosts in the middle, *Linc* and *Petrelli*, were MNE's when they are participating in the measurement, when the MSPEC parameter MNE-Selection is configured as ANY or ALL. The Measurement path can be sub-divided in four different ones, and each one with different measurement, for example passive measurement between *Sylar* and *Petrelli*, this measurements are performed by SniffMGEN, still in passive measurements are *Petrelli* and *Linc*. Active measurements are only performed between *Haitian* and *Sylar*.

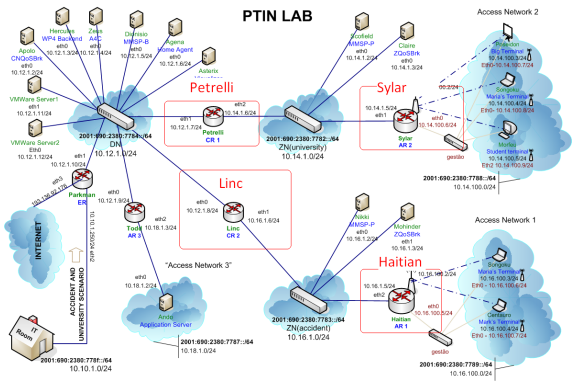

Figure 5.6: PTIN LAB Testbed

All Measurements was configured as follows, they are the common parameters between the different measurements:

- Source IP :10.16.1.5/24
- Destination IP :10.14.1.5/24
- Collector IP :10.100.1.2/24
- Source Port :6001
- Destination Port :6000
- Payload :8192 bytes
- Time of measurement:5 min

To configure each characteristic of M-NSLP, such as the monitoring of nodes in the middle characterized by the flags ALL, ANY and FirstandLast, packet rate, and traffic distribution like Poisson distribution, or traffic periodic, or even burst to simulate streaming. With all these parameter the process needs to be configured every time. To perform all tests we will use two volumes of traffic, to observe the reaction of our software to the increase of volume Network traffic.

5.2 Results

- First and Last with 5 MB/s

In this measurement between *Haitian* and *Sylar* only these two nodes are participating, this measurement is Active.

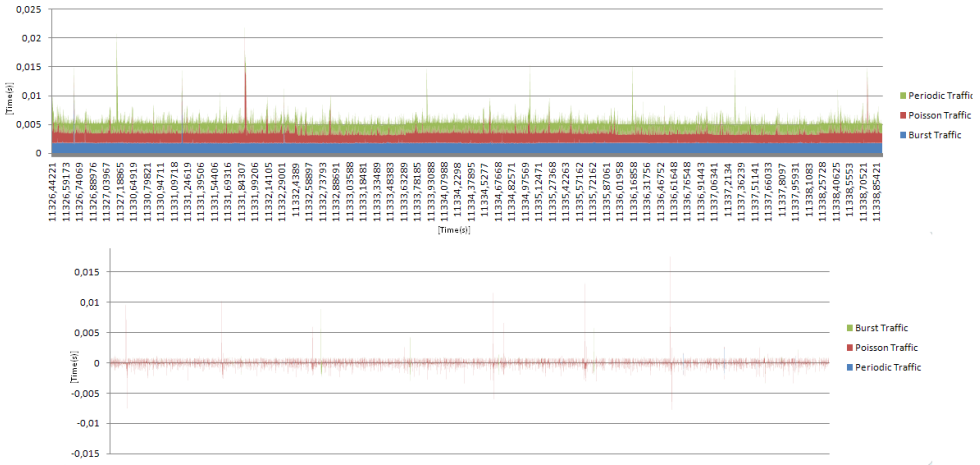

Figure 5.7: Measurement:First and Last. Rate=5 MB/s

Table 5.1: Haitian to Sylar 5 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	2,2(ms)	1,7(ms)	1,7(ms)
Jitter(Average)	0,28(μ s)	0,29(μ s)	0,30(μ s)
Sent Packets	183129	183000	71953
Processed Packets	183129	183000	71953
Diff	0	0	0

In this measurement between these two end hosts, we did not have packet loss. It must be due to good optimization of MGEN for this task. The results of delay and jitter for each kind of traffic are very acceptable.

- Measurements whit MNEselection=ALL, Rate=5 Mb/s.

Passive Measurements in *Petrelli*

Figure 5.8: Petrelli: Measurement All. Rate=5 Mb/s

Table 5.2: Passive measurement Petrelli 5 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	0,79(ms)	0,6(ms)	0,8(ms)
Jitter(Average)	0,08(μ s)	0,07(μ s)	0,02(μ s)
Sent Packets	183098	182707	101999
Processed Packets	20767	37281	21740
Diff	162331	145426	80259

As we expected, with this bandwidth the SniffMGEN don´t have capabilities to catch all packets. The variability of results between the three kinds of tested traffic, is normal because Poisson and Burst Traffic generate allot of packets in short interval of time.

Passive Measurements in *Linc*

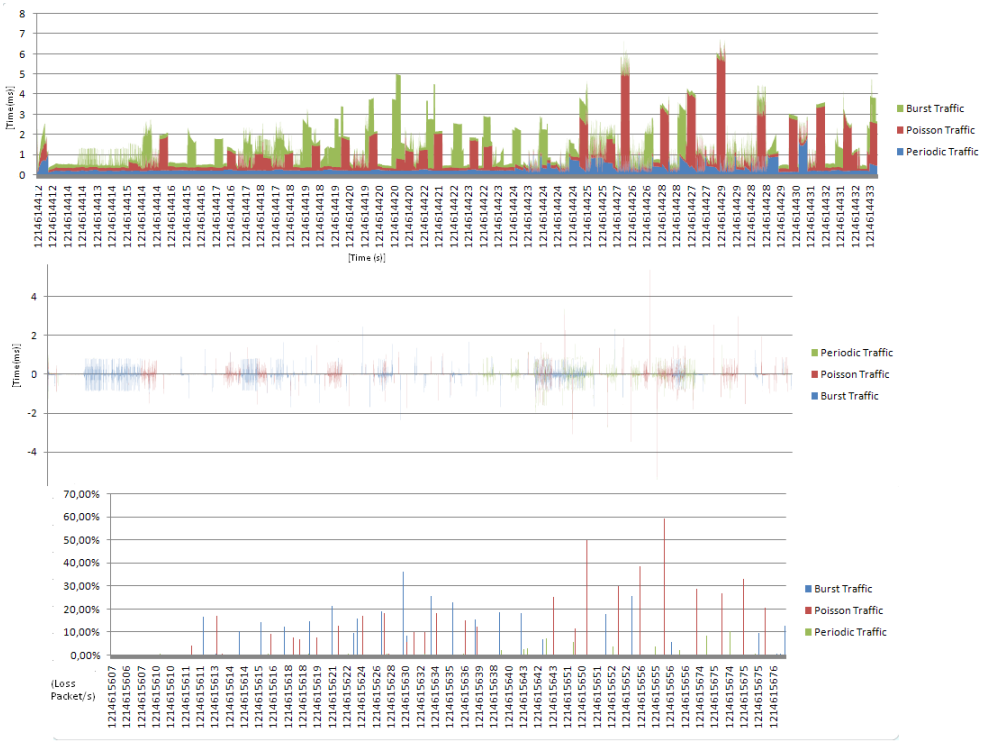

Figure 5.9: Linc: Measurement All. Rate=5 MB/s

Table 5.3: Passive measurement Linc 5 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	0,71(ms)	0,29(ms)	0,59(ms)
Jitter(Average)	0,38(μ s)	0,08(μ s)	0,02(μ s)
Sent Packets	183098	182707	101999
Processed Packets	24864	55115	22639
Diff	158234	127592	79360

This case is very similar to the last measurement in MNE *Petrelli*. The better results in this MNE can be explained by the better performance of this machine.

Active Measurements in *Haitian*

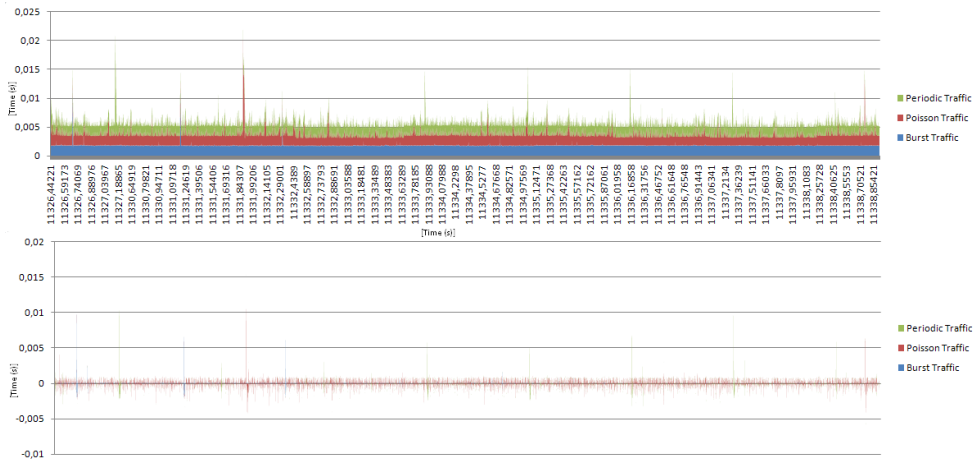

Figure 5.10: Haitian: Measurement All. Rate=5 MB/s

Table 5.4: Active measurement Haitian 5 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	2(ms)	1,8(ms)	1,7(ms)
Jitter(Average)	0,34(μ s)	0,28(μ s)	0,31(μ s)
Sent Packets	183098	182707	101999
Processed Packets	183098	182707	101999
Diff	0	0	0

In this graphic and table, can be observed that the results of delay and jitter are the sum of results in the two MNE's in the path that are participating in the measurement.

- **Measurements whit MNEselection=ALL, Rate=2 MB/s.**

Passive Measurements in *Petrelli*

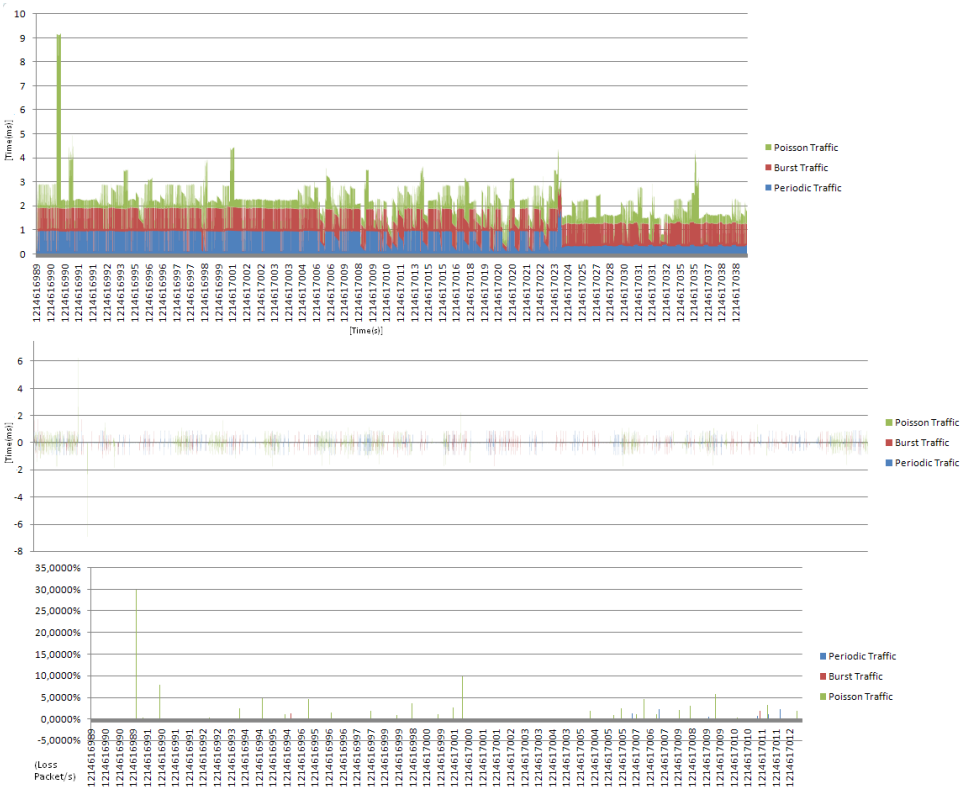

Figure 5.11: Petrelli: Measurement All. Rate=2 MB/s

Table 5.5: Active measurement Petrelli 2 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	0,63(ms)	0,85(ms)	0,83(ms)
Jitter(Average)	0,036(μ s)	0,17(μ s)	0,17(μ s)
Sent Packets	73449	70591	38108
Processed Packets	42054	49132	31116
Diff	31395	21459	6992

The better result in this test is because we test with a lower bandwidth. In this test the MNI generates 245 packets per second, with 8192 of payload.

Passive Measurements in *Linc*

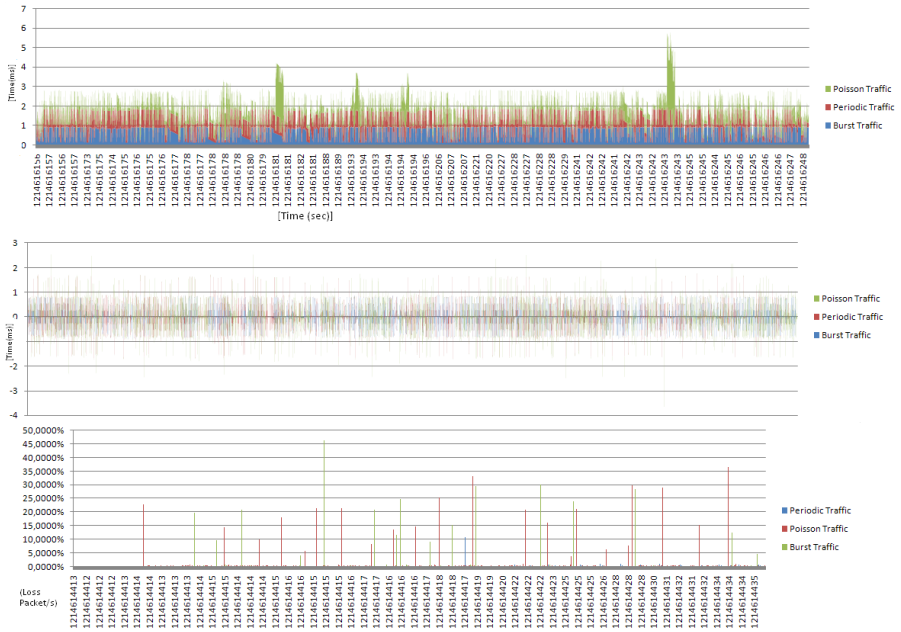

Figure 5.12: Linc: Measurement All. Rate=2 MB/s

Table 5.6: Active measurement Linc 2 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	0,52(ms)	0,62(ms)	0,64(ms)
Jitter(Average)	0,15(μ s)	0,15(μ s)	0,16(μ s)
Sent Packets	73449	70591	38108
Processed Packets	55819	56018	33082
Diff	17630	14573	5026

Active Measurements in *Haitian*

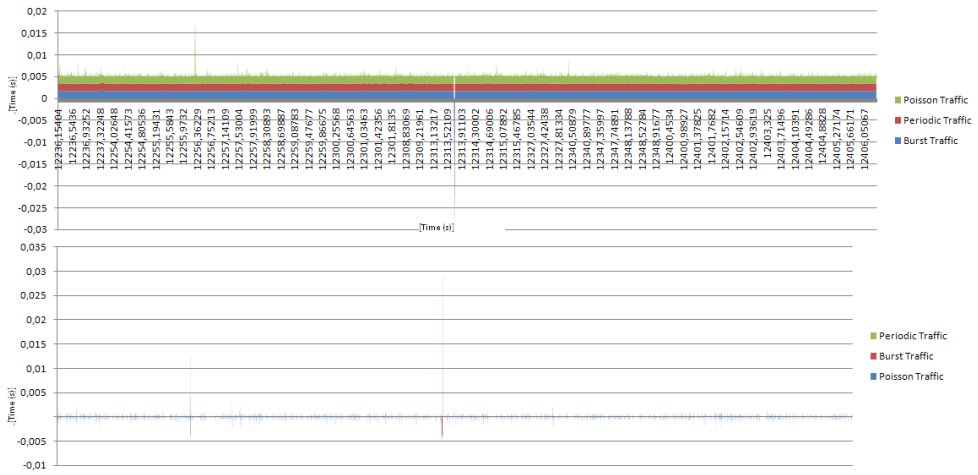

Figure 5.13: Haitian: Measurement All. Rate=2 MB/s

Table 5.7: Active measurement Haitian 2 MB/s

	Poisson	**Periodic**	**Burst**
Delay(Average)	1,5(ms)	1,5(ms)	1,6(ms)
Jitter(Average)	0,56(μ s)	0,38(μ s)	0,3(μ s)
Sent Packets	73449	70591	38108
Processed Packets	73449	70591	38108
Diff	0	0	0

5.3 Evaluation

After an exhaustive analysis of all results it is possible to observe that, if we compare the sum of results provided by passive measurements in each MNE with result of End-to-End Measurement, is the same. We can conclude that the measurement process is worked as expected. With this result we can say that, this Signaling Protocol is adequate to Metering-NSLP. As expected with high volume of data flows, the SniffMgen is a bottleneck. But it can be improved.

Chapter 6

Conclusion and Future Work

This thesis is the final result of a project concerning the design and implementation of Metering-NSLP, in order to enable performance measurements between hosts. It describes the NSIS protocol, and some examples of export protocols such as IPFIX. The main goal of this thesis was given a first contribution on the Metering-NSLP implementation. Different functional experiments were performed in order to test the functionality of the implemented protocol. This final chapter will evaluate the work performed in order to have the perception of which objectives have been achieved, and recommend future work that can be done.

6.1 Conclusion

In this section we will discuss all objectives, as was discussed in section 1.1 of this thesis, listing all goals proposed that were implemented. The first objective proposed was studied current specifications and implementations of the NSIS protocol, this goal was reached.

The implementation of Metering-NSLP was built whit help of QoS-NSLP developed by INESC Porto. This implementation of QoS-NSLP uses GIST as Signaling protocol. We also use GIST in our implementation, and it seems to be very efficient. Although, we didn't need to redesign and re-implement GIST, the work developed in Metering-NSLP was started without any kind of support for result analysis. With this fact we tried to follow strictly the recommendation of Drafts. Some of these requirements, and recommendations were tested in chapter 5.

A Metering application API was designed, with objective to cover all Metering requests, included in MSPEC. This Metering specification, MSPEC, was implemented as specified in [45]. All demonstrations show that the basic functionality of the implementation, such as exportation, performing and tearing down Measurements, works as expected.

The implementations of SniffMGEN to do passive measurements, complete the M-NSLP specification, with active and passive measurements.

With these results the achievement of the main goal of this implementation, use of the NSIS protocol for Metering between hosts in an IP Network, was demonstrated.

The work in this thesis was performed while all specifications were still in draft; it means that some specifications can change. We hope that all work done in this thesis can contribute to the standardization process.

6.2 Future Work

Recommended future work that could be done is the following:

- Updating the implementation performed to the latest specifications.
- Review the FSM, the state machine implemented is an adaption of FSM from QoS-NSLP. This adaptation was caused by lack of time.
- Updating MSPEQ specification, and work for its standardization.

References

[1] Bruce A. Mah. Ravi S. Prasad, Constantinos Dovrolis. The effect of layer-2 store-and-forwarddevices on per-hop capacity estimation. Technical report, IEEE Infocom conference, 2002.

[2] Mayi Zoumaro-Djayoon. Next step in signaling transport protocol/general internet signaling protocol. Master's thesis, University of Twente, 2005.

[3] William B. Norton. Internet service providers and peering. Technical report, May 2001.

[4] i380. Internet protocol data communcation service - ip packet transfer and availability performance parameters. Technical report, ITU-T, February 1999.

[5] M. Mathis Pittsburgh Supercomputer Center V. Paxson, Lawrence Berkeley National Lab; G. Almes Advanced Network & Services;J. Mahdavi. Framework for ip performance metrics. Technical report, RFC 2330, May 1998.

[6] E.800. Terms and definitions related to quality of service and network performance including dependability. Technical report, ITU-T, August 1994.

[7] X.140. General quality of service parameters for communication via public data networks. Technical report, ITU-T, September 1992.

[8] Fraunhofer FOKUS; B. Claise Cisco Systems; S. Zander Swinburne University J. Quittek, NEC Europe Ltd; T. Zseby. Requirements for ip flow information export (ipfix). Technical report, RFC 3917, October 2004.

[9] V. Paxson. *Measurement and Analysis of End-to-end Internet Dynamics*. PhD thesis, University of California - Berkeley, 1997.

[10] Intel; P. Ford Microsoft; F. Baker Cisco; L. Zhang UCLA Y. Bernet, Microsoft; R. Yavatkar. A framework for end-to-end qos combining rsvp/intserv and differentiated services. Technical report, Internet Engineering Task Force, March, 1998.

[11] T. Culler D. Roscoe T. Peterson, L.Anderson. A blueprint for introducing disruptive technology into the internet. Technical report, PLANET LAB, 2002.

[12] Braun H.-W. e Brown J. McGregor, A. The nlanr network analysis infrastructure. *IEEE Communications Magazine*, pages 122–128, 2000.

[13] A. J. Luckie, M. J. e McGregor. Ipmp: Ip measurement protocol. In *In Proc. of the Passive and Active Measurement Workshop - PAM'2002, Fort Collins, CO, EUA*, 2002.

[14] Moon S. Lyles B. Cotton C. Khan M. Moll D. Rockell R.; Seely T. Diot C. Fraleigh, C. *Packet-level traffic measurements from the Sprint IP backbone*, chapter 17(6), page 6–16. IEEE Network, 2003.

[15] CAIDA. Cooperative association for internet data analysis (caida). Technical report, http:// www.caida.org, 1997.

[16] N. Brownlee; C. Mills; G. Ruth. Traffic flow measurement: Architecture. Technical report, RFC 2722, October 1999.

[17] N. Brownlee. Traffic flow measurement: Meter mib. Technical report, RFC 2720, October 1999.

[18] N. Brownlee. Srl: A language for describing traffic flows and specifying actions for flow groups. Technical report, RFC 2723, October 1999.

[19] J. Quittek. Requirements for ip flow information export (ipfix). Technical report, RFC3917, 2004.

[20] Seunghyun Yoon Byungjoon Lee, Hyeongu Son and Youngseok Lee. End-to-end flow monitoring with ipfix. Technical report, IETF, 2007.

[21] Matt Mathis Pittsburgh Supercomputing Center. Treno bulk transfer capacity. Technical report, Internet Draft, 1999.

[22] S. Leinen. Evaluation of candidate protocols for ip flow information export (ipfix). Technical report, RFC 3955, 2004.

[23] Nick Duffield. A framework for passive packet measurement. Technical report, AT&T Labs, 2002.

[24] Mikkel Thorup Nick Duffield, Carsten Lund. Properties and prediction of flow statistics from sampled packet streams. Technical report, AT&T Labs–Research, 2002.

[25] Darryl Veitch Nicolas Hohn. Inverting sampled traffic. Technical report, Australian Research Council Special Research Center for UltraBroadband Information Networks Department of Electrical and Electronic Engineering.

[26] Craig Leres Van Jacobson and Steven McCanne. *TCPDUMP*, 15 June 2007.

[27] Gerald Combs. *Wireshark*. http://www.wireshark.org, 2005.

[28] M. E. Crovella R. L. Carter. Measuring bottleneck link speed in packet-switched networks. Technical report, Computer Science Department, Boston University, 111 Cummington St., Boston, 1996.

[29] P. Moore D. Dovrolis, C. Ramanathan. What do packet dispersion techniques measure? Technical report, Wisconsin Univ, 2001.

[30] Constantinos Dovrolis Manish Jain. Pathoad: a measurement tool for end-to-end available bandwidth. Technical report, PAM workshop, 2002.

[31] Mary Baker Kevin Lai. Nettimer: A tool for measuring bottleneck link bandwidth. Technical report, Department of Computer Science, Stanford University, 2001.

[32] Mary Baker Kevin Lai. Measuring link bandwidths using a deterministic model of packet delay. Technical report, Department of Computer Science, Stanford University, 2000.

[33] W. Stevens M. Allman, V. Paxson. Tcp congestion control. Technical report, RFC 2581, 1999.

[34] Federico Montesino-Pouzols. Comparative analysis of active bandwidth estimation tools. Technical report, Instituto de Microelectrnica de Sevilla.

[35] C. Dovrolis K. Claffy R. S. Prasad, M. Murray. Bandwidth estimation: metrics, measurement techniques, and tools. Technical report, CAIDA, 2003.

[36] A. McDonald J. Manner, G. Karagiannis. Nslp for quality-of-service signaling. Technical report, IETF, http://www.ietf.org/internet-drafts/draft-ietf-nsis-qos-nslp-16.txt, February 7, 2008.

[37] D. Clark; Xerox PARC S. Shenker ISI, R. Braden; MIT. Integrated services in the internet architecture: an overview. Technical report, RFC 1633, June 1994.

[38] EMC Corporation; M. Carlson Sun Microsystems; E. Davies Nortel UK; Z. Wang Bell Labs Lucent Technologies; W. Weiss Lucent Technologies S. Blake, Torrent Networking Technologies; D. Black. An architecture for differentiated services. Technical report, RFC2475, December 1998.

[39] V. Jacobson; Bay Networks K. Poduri Cisco Systems, K. Nichols. An expedited forwarding phb. Technical report, RFC 2598, June 1999.

[40] F. Baker Cisco Systems; W. Weiss Lucent Technologies; J. Wroclawski MIT LCS J. Heinanen, Telia Finland. Assured forwarding phb group. Technical report, RFC 2597, June 1999.

[41] Department of Information Engineering Niigata University Shigeo Shioda, Department of Urban Environment & Systems Chiba University; Kenichi Mase. Performance comparison between intserv-based and diffserv-based networks. Technical report, Tokyo Institute of Technology, 2005.

[42] J. Loughney S. Van den Bosch R. Hancock, G. Karagiannis. Next steps in signaling (nsis): Framework rfc4080. Technical report, IETF, June 2005.

[43] R. Hancock H. Schulzrinne. Gist: General internet signalling transport. Technical report, IETF, 2008.

[44] C. Aoun E. Davies M. Stiemerling, H. Tschofenig. Nat/firewall nsis signaling layer protocol (nslp). Technical report, IETF, February 15, 2008.

[45] F. Dressler J. Quittek C. Kappler H. Tschofenig A. Fessi, G. Carle. Nslp for metering configuration signaling. Technical report, IETF, March 5, 2007.

[46] J. Postel. User datagram protocol. Technical report, RFC 768, 28 August 1980.

[47] DARPA INTERNET. Transmission control protocol. Technical report, RFC 793.

[48] C. Allen T. Dierks. The tls protocol. Technical report, RFC 2246, January 1999.

[49] K. Morneault C. Sharp H. Schwarzbauer T. Taylor I. Rytina M. Kalla L. Zhang V. Paxson V. R. Stewart, Q. Xie. Stream control transmission protocol rfc2960. Technical report, IETF, October 2000.

[50] S. Floyd E. Kohler, M. Handley. Datagram congestion control protocol (dccp) rfc4340. Technical report, IETF, March 2006.

[51] Hannes Tschofenig Christian Dickmann Xiaoming Fu, Henning Schulzrinne and Dieter Hogrefe. Overhead and performance study of the general internet signaling transport (gist) protocol. Technical report, (INFOCOM 2006), Barcelona, Spain,, 2006.

[52] D. Katz. Ip router alert option rfc2113. Technical report, IETF, February 1997.

[53] C. Kappler D. Oran G. Ash, A. Bader. Qos nslp qspec template. Technical report, IETF, April 3, 2008.