Universidade de Aveiro
2012
Departamento de Electrónica, Telecomunicações e Informática

Aleksander Lechosław Pleszko

Gestão Multiplataforma de um Comutador Ethernet de Tempo-Real

Multiplatform Management of a Hard Real-Time Ethernet Switch

Universidade de Aveiro
2012

Departamento de Electrónica, Telecomunicações e Informática

Aleksander Lechosław Pleszko

Gestão Multiplataforma de um Comutador Ethernet de Tempo-Real

Multiplatform Management of a Hard Real-Time Ethernet Switch

Wieloplatformowy system zarządzania przełącznikiem Ethernetowym czasu rzeczwistego

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Mestre João Paulo Silva Barraca, Assistente Convidado do Departamento de Electrónica ,Telecomunicações e Informática da Universidade de Aveiro e co-orientação científica do Doutor Joaquim José de Castro Ferreira, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro.

Universidade de Aveiro
2012

Departamento de Electrónica, Telecomunicações e Informática

Aleksander Lechosław Pleszko

Gestão Multiplataforma de um Comutador Ethernet de Tempo-Real

Multiplatform Management of a Hard Real-Time Ethernet Switch

Wieloplatformowy system zarządzania przełącznikiem Ethernetowym czasu rzeczwistego

Dissertation submitted to the University of Aveiro as part of its Master's in Electronics and Telecommunications Engineering Degree. The work was carried out under the scientific supervision of Professor João Paulo Barraca from the Department of Electronics, Telecommunications and Informatics of the University of Aveiro, and Professor Joaquim José de Castro Ferreira from the Superior School of Technology and Management of Águeda from the University of Aveiro.

Mr. Aleksander Pleszko is a registered student of Lodz University of Technology, Lodz, Poland and carried out his work at University of Aveiro under a joint Campus Europae program agreement. The work was followed by Assistant Professor Marcin Janicki at Lodz University of Technology as the local Campus Europae Coordinator.

o júri

Presidente

Alexandre Manuel Moutela Nunes da Mota,
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

José Carlos Meireles Monteiro Metrôlho,
Professor Adjunto da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

João Paulo Silva Barraca,
Assistente Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Joaquim José de Castro Ferreira,
Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro

Marcin Janicki,
Assistant Professor of Faculty of Electrical, Electronic, Computer and Control Engineering of Lodz University of Technology

acknowledgements

I address my sincere thanks to my parents, friends, and supervisors – Professor João Paulo Barraca, Professor Joaquim José de Castro Ferreira, and Professor Marcin Janicki – for all the support and guidance I received from them. I especially want to thank for much appreciated and valuable contributions of Professor Pedro Gonçalves, and support from University of Aveiro, mainly Professor Carlos Alberto da Costa Bastos.

Stay in Portugal and the master degree was one of a kind experience and without all the people mentioned, it certainly would not be so rich. Thank you!

palavras-chave

Real-Time Systems, Real-Time Communications, Switched Ethernet, Real-Time Scheduling, Network Management, SNMP, NetConf

resumo

Ao longo dos últimos anos, o agora onipresente protocolo Ethernet, embora não dotado de mecanismos eficazes de gestão de QoS, foi ganhando uma grande aceitação no campo das comunicações industriais. Esta crescente aceitação deveu-se, em grande parte, a novos protocolos, baseados em Ethernet (por exemplo, Profinet, Ethernet Industrial, etc), capazes de fornecer comunicações com garantias deterministas ou de tempo-real.

O comutador Ethernet Hartes (Hard Real-Time Ethernet Switch), foi desenvolvido para disponibilizar uma infra-estrutura de comutação Ethernet capaz de fornecer garantias de pontualidade, de bom uso da largura de banda e para suportar, de modo eficiente, a flexibilidade operacional necessária em aplicações de tempo-real distribuídas, de sistemas embarcados dinâmicos. O desenvolvimento do comutador Hartes, foi baseado em trabalho anterior do paradigma de comunicação FTT (Flexible Time-Triggered), e teve por objetivo o projeto de um comutador Ethernet com melhor controlo de transmissão, escalonamento do tráfego e integração transparente de nodos não tempo-real.

NetConf é uma tecnologia recente de gestão de redes que tem vindo progressivamente a substituir a tecnologia SNMP (Simple Network Management Protocol), o standard de facto há muito adoptado pela indústria. A maior diferença entre NetConf e o SNMP é que o NetConf adopta um mecanismo de comunicação baseado em XML-RPC, que, graças às ferramentas desenvolvidas no âmbito de outras tecnologias web, permite ciclos mais rápidos e mais simples de desenvolvimento e de gestão.

O comutador Hartes não dispõe de uma plataforma de gestão com uma interface padronizada para os protocolos SNMP ou NetConf, de modo a permitir a sua gestão remota. Assim, o objetivo principal deste trabalho é o desenvolvimento de componentes-chave de apoio à gestão multiplataforma do comutador Ethernet Hartes, bem como a respectiva avaliação de desempenho dos componentes desenvolvidos.

keywords

Real-Time Systems, Real-Time Communications, Switched Ethernet, Real-Time Scheduling, Network Management, SNMP, NetConf

abstract

In recent years, the now ubiquitous Ethernet protocol that lacks effective QoS management functions, has gained momentum in the field of industrial communication, by means of novel, Ethernet-based protocols (e.g. Profinet, Industrial Ethernet, etc.), which are able to provide deterministic communications.

HaRTES – Hard Real-Time Ethernet Switch, aimed to develop an Ethernet switching infrastructure, able to provide timeliness guarantees, efficient bandwidth usage and support for operational flexibility as required by dynamic real-time distributed embedded systems. The project was built upon previous work on the FTT (Flexible Time-Triggered) communication paradigm to develop Ethernet switches with enhanced transmission control, traffic scheduling, and transparent integration of non-real-time nodes.

NetConf is a recent network management technology that is replacing the Simple Network Management Protocol (SNMP) – widely used and long adopted by industry standard. The biggest difference between NetConf and SNMP is that the former use a communication mechanism based on XML-RPC, which, thanks to the tools developed in the scope of other web technologies, allows a simpler and faster development and management cycle.

The HaRTES project had not provided a management platform with a standardized interface for SNMP or NetConf protocols, enabling remote switch management. Thus the main objective of this work was to develop key components for the support of the standardized multiplatform management interfaces for the HaRTES switch and their performance assessment.

słowa kluczowe

Systemy czasu rzeczywistego, komunikacja czasu rzeczywistego, Switched Ethernet, Real-Time Scheduling, zarządzanie siecią, zarządzanie zasobami sieciowymi, SNMP, NetConf

streszczenie

W ostatnich latach wszechobecny protokół Ethernet, któremu brakuje efektywnych funkcji zarządzania gwarancją jakości usług (QoS), staje się coraz popularniejszy w dziedzinie komunikacji przemysłowej, za sprawą nowych protokołów opartych o standard Ethernet (np. Profinet, Industrial Ethernet, itp.), które są w stanie zapewnić deterministyczną komunikację.

HaRTES - Hard Real-Time Ethernet Switch jest to projekt, który ma za zadanie pomóc w rozwinięciu infrastruktury pozwalającej na zapewnienie gwarancji terminowości wiadomości, efektywnego wykorzystania pasma oraz wsparcie dla dynamicznego przystosowywania się sieci, koniecznej przy zastosowaniu rozproszonych systemów wbudowanych czasu rzeczywistego. Projekt jest kontynuacją pracy traktującej o FTT (Flexible Time-Triggered), czyli modelu komunikacji rozproszonej, rozbudowującej przełącznik Ethernetowy o polepszoną kontrolę transmisji, harmonogramowanie ruchu oraz przezroczystą integrację węzłów nieobsługujących wiadomości czasu rzeczywistego.

NetConf jest technologią zarządzania zasobami sieciowymi, która zastępuje Simple Network Management Protocol (SNMP) - powszechnie stosowany i długo przyjęty standard. Największą różnicą pomiędzy standardami NetConf oraz SNMP jest to, że NetConf stosuje mechanizm komunikacji oparty o XML-RPC, który dzięki narzędziom opracowanym w ramach technologii internetowych, umożliwia prostszy i szybszy cykl rozwoju narzędzia.

Projekt HaRTES nie posiadał platformy, umożliwiającej zdalne zarządzanie jego parametrami, stosującej standardowe interfejsy do tego przeznaczone: SNMP lub NetConf. Celem tej pracy było opracowanie niezbędnych komponentów dla przełącznika HaRTES, wspierające niniejsze standardy internetowe, poparte testami wydajności.

Contents

List of Figures

List of tables

Acronyms

API	Application Programming Interface
ASN	Abstract Syntax Notation
ASW	Asynchronous Window
BAG	Bandwidth Allocation Gap
BE	Best-Effort
BS	Background Scheduling
CAN	Controller Area Network
CPU	Central Processing Unit
CPS	Cyber-Physical Systems
DES	Distributed Embedded Systems
DNS	Domain Name System
DS	Deferrable Server
EC	Elementary Cycle
EDF	Earliest Deadline First
ET	Event-Triggered
FCAPS	Fault, Configuration, Accounting, Performance, Security Management
FCS	Frame Check Sequence
FCT	Fundação para a Ciência e a Tecnologia
FIFO	First-Come-First-Served
FPGA	Field-Programmable Gate Array
FTP	File Transfer Protocol
FTT	Flexible Time-Triggered
FTT-SE	Flexible Time-Triggered - Switched Ethernet
HaRTES	Hard Real-Time Ethernet Switch
HTTP	Hypertext Transfer Protocol

IANA	Internet Assigned Numbers Authority
IFG	Inter-Frame Gap
IP	Internet Protocol
IPC	Interprocess Communication
ISO	International Organization for Standardization
LAN	Local Area Network
LEC	Length of the Elementary Cycle
LSW	Length of the Synchronous Window
MAC	Medium Access Control
MIB	Management Information Base
NE	Network Element
NetConf	Network Configuration Protocol
NMO	Network Management Objects
NRT	Non Real-Time
OID	Object Identifier
PDO	Process Data Objects
PDU	Protocol Data Unit
PHY	Physical Layer
PS	Polling Server
QoS	Quality-of-Service
RFC	Requests for Comments
RT-L	Real-Time Layer
RT-VBR	Real-Time Variable Bit Rate
RTCD	Real-Time Communication Daemon
RTE	Real-Time Ethernet
SDO	Service Data Objects
SIL	Server Instrumentation Library
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SOF	Start-of-Frame
SRDB	System Requirements Database
SRP	Stream Reservation Protocol
SSH	Secure Shell

SSL	Secure Sockets Layer
SW	Synchronous Window
TAT	Turn-Around Time
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Type-Length-Value
TM	Trigger Message
TT	Time-Triggered
TTE	TT Ethernet
TTP	Time-Triggered Protocol
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
WCET	Worst-Case Execution Time
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

Chapter 1

Introduction

Distributed Embedded Systems (DES) are widely used in many domains serving its purpose anywhere from industrial automation to vehicular system, managing both hard and soft real-time traffic. Instances of DES must follow strict timeliness, predictability and precedence constrains. In these cases, special-purpose real-time communication networks, known as fieldbuses, must be used to achieve the desired properties.

In these systems, due to its omnipresence, important role began to play Ethernet protocol. However Ethernet standard was not originally developed to meet the predictability, timeliness and reliability, which are present in Network Embedded Systems (NES). Still, there are currently available technologies, that enable mechanisms of guaranteed Quality of Service (QoS), such as MPLS [1] and RSVP [2] especially combined with IntServ [3] and DiffServ [4] models, however they only provide static guarantees.

Effective needs and a quality of service adaptation policy requires an on-line flexibility and admission control, which are not met if used previously mentioned solutions. This has motivated the development of a new generation of Ethernet switches, Hard Real-Time Ethernet Switch (HaRTES).

Over the last decade, several Ethernet-based protocols have been developed, e.g.: Ethernet-Powerlink, Profinet, EtherCAT and Ethernet/IP, which take advantage of some of Ethernet's appealing attributes e.g., large bandwidth, cheap silicon development and high availability, while removing or reducing the sources of non-

determinism arising from its MAC protocol and/or from the current switched architecture.

In this context, two research projects were founded by Fundação para a Ciência e a Tecnologia (FCT). The first one, HaRTES – Hard Real-Time Ethernet Switch [5], aimed to develop an Ethernet switching infrastructure, using FPGA technology, able to provide timeliness guarantees, efficient bandwidth usage and support for operational flexibility as required by dynamic real-time distributed embedded systems. The project was built upon previous local work on the Flexible Time-Triggered [6] communication paradigm to develop Ethernet switches with enhanced transmission control, traffic scheduling, service differentiation, transparent integration of non-real-time nodes and improved error confinement mechanisms, particularly with respect to temporal misbehaviours.

The Serv-CPS project: Server-based Real-Time Ethernet Communication Architecture for Cyber-Physical Systems, evolves the Ethernet switch developed in the scope of the HaRTES project, which already supports enhanced traffic scheduling services. The objective of Serv-CPS is to develop a networking framework, based on switched Ethernet, suitable to support Cyber-Physical Systems (CPS), by including explicit and efficient support for component-oriented design methodologies. The framework shall support: heterogeneous traffic classes with temporal isolation, partitioning and virtualization mechanisms, hierarchical multi-level server composition, dynamic adaptation and reconfiguration of servers with temporal guarantees, analytical tools for supporting the design of Cyber-Physical Systems (CPS) and middleware for service management.

The features of the Ethernet switch proposed in Serv-CPS represent a breakthrough in terms of the adequacy of complex Real-time protocols based on Ethernet technology [7], where flexibility and compensability in the time domain are design requirements. The technology proposed allows, in an innovative way, using the same network to dynamically handle multiple traffic sources (e.g. web access, file transfer, live video/audio, control data), making an efficient utilization of the resources, without compromising the performance of real-time applications. For the purpose of preforming QoS reservations, SRP [8] support was added.

NetConf and SNMP are two management technologies that handle the communication aspect of network management. The latter one is long adopted, light,

and widely used standard for managing network elements. However, due to its security issues it is used mainly for monitoring purposes. NetConf is recent technology that uses more descriptive XML-RPC standard to communicate between server and client. It has more robust security authentication and encryption mechanism than SNMP, however it introduces accordingly greater overhead of management and configuration data.

1.1 Motivation

In the context of rapid growth of networks and new technologies, unique solutions for remote device managing are needed. However, the HaRTES switch lacked such standardized interface for remote management i.e., interface for parameterizing and monitoring the switch behaviour.

Motivation for this thesis is to devise mechanisms for faster FTT-enabled switch development, and remote device testing in a real-life environment, without the need to change or monitor the switch parameters in a hardcoded way. This can reflect in new test applications of HaRTES technologies.

Therefore, this work focuses its objective on developing multiplatform (SNMP [9] and NetConf [10]) management interface, implementation and evaluation for a HaRTES switch and presents validation of the two management approaches.

As SNMP is widely adopted by most of the Real-Time Ethernet (RTE) equipment, it was also challenging to device mechanisms for managing HaRTES. Examples of RTE equipment supporting RTE protocols are: TT Ethernet (TTE), Ethernet Powerlink [11], Profinet [12], Industrial Ethernet, etc.

The TTE A664 Pro Switch [13] has a built-in management module for network monitoring and supports secure network management and allows data loading and querying of health and status information. Weidmuller IE-SWxx-M Industrial Ethernet switches [14] can also be managed via SNMP. The IE-SWxx-M switches support traps for the link-up, link-down, confirmation error, cold restart and warm restart functions. Profinet [15], also uses SNMP for maintaining and monitoring network devices.

Ethernet Powerlink adopts a proprietary protocol for the network management derived from CANopen, based on Process Data Objects (PDO), Service Data Objects (SDO), and Network Management Objects (NMO). According to [16] Ethernet Powerlink routers are managed by Powerlink SDO and optionally by the SNMP.

A common property of all previous mentioned RTE protocols is that they do not allow dynamic reconfiguration with real-time guarantees. They are fully static systems in which all operating conditions are completely defined at pre-runtime In these RTE protocols system reconfiguration involves stopping the system, apply the modifications and restart it. Since monitoring is not a time-critical activity and maintenance is performed offline, SNMP is well suited for these tasks in the case of the above mentioned RTE protocols. NetConf, a newer technology, is other valid alternative, however and to the best of our knowledge, it is not supported by current RTE equipment.

For the specific case of FTT Ethernet networks based on the HaRTES switch and supporting timely operational flexibility, it is necessary to assess the performance of network management technologies. Notice that, both SNMP and NetConf do not provide real-time guarantees, however this is no impairment for HaRTES, since modifications of the communications requirements are not made directly by the management services. Online requests to modify communication requirements are processed by the admission control and, if accepted, their timeliness is secured by real-time scheduling.

1.2 Contribution

The work developed in the scope of this thesis was oriented to the key components for the support of standardized management interfaces for the HaRTES switch, making it suitable for remote monitoring and available for currently existing management protocols, namely SNMP and NetConf.

The HaRTES architecture has been proposed in the past to provide implementation of the FTT paradigm on a switched Ethernet communication framework, leading to the Flexible Time-Triggered over Switched Ethernet (FTT-SE) protocol. The work described in this dissertation extends previous work in the following points:

- Support for remote enhanced real-time traffic monitoring capabilities for SNMP and NetConf;
- Support for multiple Management Information Databases;
- Novel protocol solutions for Remote Management Service;

- Performance assessment of HaRTES switch management with SNMP and NetConf.

These contributions were summarized in a paper accepted for publication in a workshop:

- Aleksander Pleszko, João Paulo Barraca, Joaquim Ferreira, and Pedro Gonçalves. Multiplatform Management of a Hard Real-Time Ethernet Switch. In *Proceedings of the IEEE Globecom 2012 Workshop: The 4th IEEE International Workshop on Management of Emerging Networks and Services*, Anaheim, California, December 3-7, 2012;

1.3 Dissertation outline

The rest of the dissertation is organized as follows. *Chapter 2* presents overview of subject of the network managements as well as the two proposed management technologies enabled to cooperate with the HaRTES switch. *Chapter 3* is dedicated to the architecture of the Real-Time Ethernet Switch and its two modules: the Master Module and the Switching Module. *Chapter 4* describes the details of the implementation of management interface and the Remote Management Protocol in management agents and Master Module side. *Chapter 5* presents and discusses results regarding the latency of two implemented management solutions. *Chapter 7* concludes the thesis and proposes some future lines of work.

Chapter 2

Network management technology

Network management may refer to a broad range of subjects. It is mostly agreed that this term is used in the context of the activities, methods, procedures, and tools that relate to the operation, administration, maintenance, and provisioning of networked systems [17]. The International Organization for Standardization (ISO) [18] created conceptual areas to help understand the major functions of network management system. It is called Fault Management, Configuration Management, Accounting Management, Performance Management, and Security Management (FCAPS).

The Internet from its birth is constantly evolving. At the beginning there were only a couple university's devices connected to each other. Later, large company's networks began to join the web. As for now, with so many devices in the Internet responsible for forwarding, managing, and controlling traffic, it would be impossible to efficiently manage those devices and traffic, without consistent protocol for remote devices' management.

Several key aspects must be taken under the consideration if designed management system is to be solid and fully operational at all possible time. A great amount of care must be taken to ensure full operational status of the network; is it running smoothly and without errors. In the event of error, provided the network is sufficiently monitored to quickly spot the problem and manage it, it is likely that small amount or no users will be affected by it.

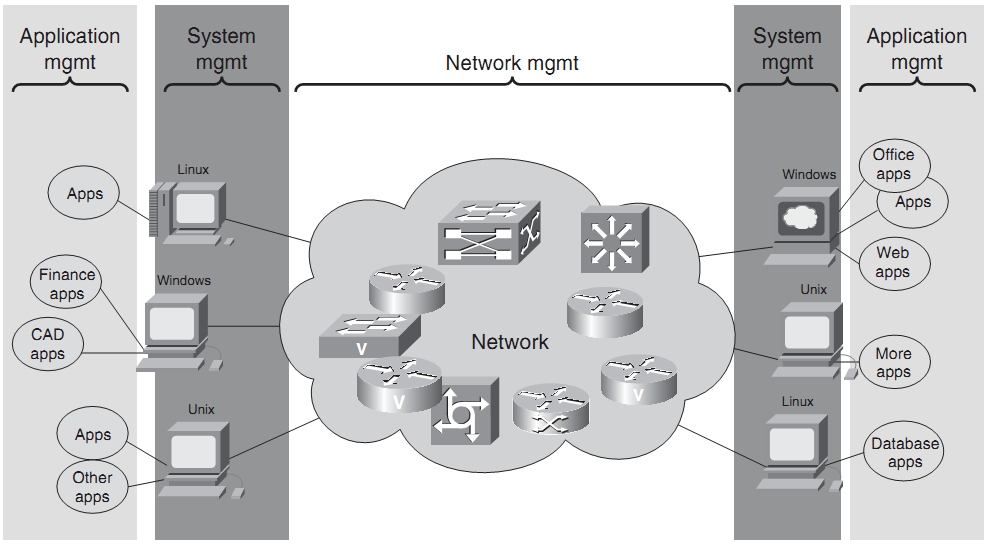

Figure 1 – Network, Systems, and Application Management [17]

Network management covers also areas of keeping network under control, i.e. keeping track of resources and the way how they are assigned in the network. This expands also to plain maintenance of the network, i.e. performing repairs and upgrades of connected devices, adding, removing devices, adjusting their parameters so that overall network performance is boosted.

Network management likewise involves configuring resources and services that are available to the users connected without adversely affecting the rest of the network. Configuration should also be flexible so that new customers can be quickly added to the pool without laborious reconfiguration of many parts of the system.

The term network management can be also narrowed to management of the networks themselves. If this is the case, the terms of system management and application management are also distinguished, as it is depicted in Figure 1. Sometimes special service can be bound to all of these three categories i.e. network, system and application, so then it is also distinguished and sum up in the term service management.

In a broader sense these categories, though may have different management practices associated with them, have also very much in common, therefore they are gathered under one general term network management.

Covering such a broad topic, even in a short run, requires several naming distinctions to be made, particularly what the network management comprises of. What is obvious, firstly we need the managed device or the network that is to be managed. Usually it is the latter, a network of interconnected devices that exchanges data amongst each other.

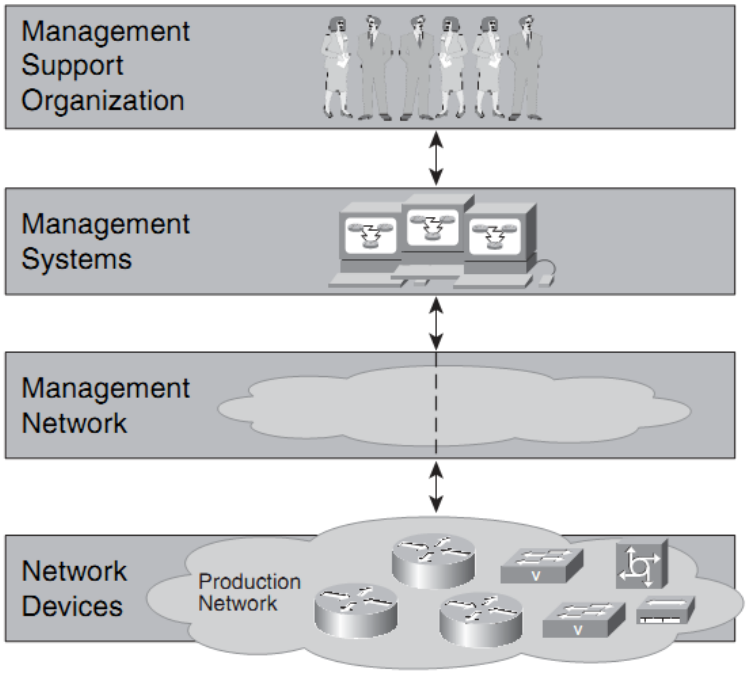

Figure 2 – Basic Components of Network Management [17]

To have any impact on this network we also need the system and application that will be used to do so – Network Management Systems (NMS). These systems encompass the management logic helping to gather, process and present collected data from managed network or device. It can also be used to send management commands to start or stop services or interfaces.

Maybe the least obvious part of the network management, that creates somewhat paradoxical situation, is the network management itself. The managing interface and network to be managed must be connected via management network, so the communication between them is even possible. That is why the ideal solution would be to have not one network, but two: one for the data traffic and one for management purposes.

The last crucial element without which the network management would not work is the management personnel, responsible for changing it and maintaining the whole structure sound and solid, as shown in Figure 2.

Closing to the topic of specific implementation of management solutions the point must be made concerning management parlance. Previously mentioned managed device can be also called network element (NE) or network node. If they are to be managed they must participate in management process.

There are also important distinctions that have to be drawn between manager/agent and client/server model alike. The communication between management interface and the managed network element is asymmetrical: the usual chain of command is that the manager (client) sends request and agent (server) responds to that plus sends asynchronous trap or notifications if selected event is triggered.

The usual place of one server, which is serving many clients, is undermined in the network management world. Figure 3 depicts the situations in which many servers (which are called agents) are serving a small number of clients (managers). Moreover is imperative that network elements provide a way to manage them, usually by implementing a management interface. This middleware is supporting a dialogue between external network management software and the managed software. It is also commonly referred as management agent.

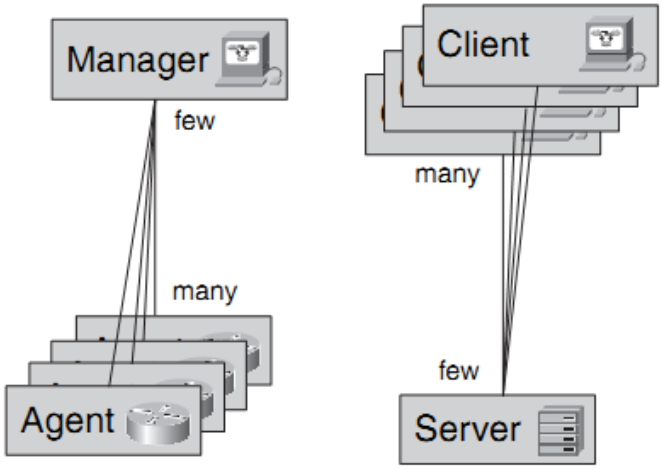

Figure 3 – Manager/Agent Versus Client/Server [17]

In general, management agent comprises three main parts:

- Management interface;
- Management Information Base;
- Core agent logic.

The management interface allows the managing application to communicate with the network elements. It allows opening and closing management sessions maintained in a specific management protocol, as well as make request, responses and traps. The management interface handles generally communication between servers and agents.

The Management Information Base (MIB) is virtual data store that represents a management view of the device being managed. It defines management information that can be retrieved and/or changed. The MIB is only a proxy for the managed parameter to be viewed by the managing application. However, it does not necessarily mean that MIB objects are defined in Structure of Management Information Version 2 (SMIv2) [19] as it is a subset of Abstract Syntax Notation One (ASN.1) [20] which is commonly used in Simple Network Management Protocol (SNMP) [9]. It can also be defined by XML [21] or Command Line Parameters, depending on implementation details.

The core agent logic is responsible for actual retrieval of the requested data from the managed device. It translates the information being encoded in the MIB to the actual register being present in the network element. It can also be infused with additional capabilities of pre-correlating raw events with each other or scheduling periodic test algorithm for validating proper functioning of the device .

The last topic to mention is the protocol standard used to manage the network devices. It should be flexible, extensive and able to manage many layers of protocols. SNMP was designed to meet these requirements and has it became the standard protocol used to monitor network devices. The OSI model [18] is depicted in Figure 4.

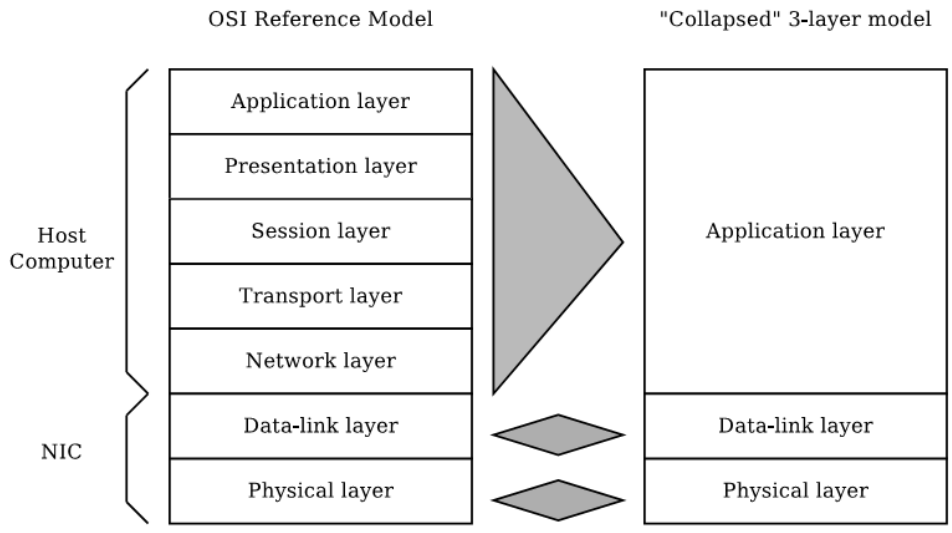

Figure 4 – The ISO/OSI reference model [22]

As it turned out in the first years of the 21st century, the SNMP protocol was used mainly to monitor networks, rather than to completely manage them. In 2002, a meeting was held between the Internet Architecture Board, and leading members of the IETF Network operators to discuss this situation. It turned out that most of network

administrators were using Command Line Interfaces (CLI) to configure their boxes, but due to unpredictable output of this command new, more flexible solution was needed. Proposed solution involved XML standard for data transmission and included SSH [23] as transfer protocol for security reasons. This became the basis of a new protocol later introduced as NetConf [10].

2.1 Simple Network Management Protocol

First version of Simple Network Management Protocol (SNMP) [9] was released in 1988. It was designed to remotely monitor and manage the network elements through the Internet Protocol (IP) [24] using sets of SNMP commands.

SNMP uses the User Datagram Protocol (UDP) [25] to transmit its data, as opposed to Transmission Control Protocol (TCP) [26] because the former is connectionless. SNMP uses UDP port 161 for sending and receiving requests data and port 162 for receiving traps (notifications) from management enabled network elements. Using connectionless protocol means that transfer of packets is unreliable, but taken the low message overhead this is somewhat reasonable.

The first version of SNMP (*SNMPv1*) was an overly simple solution. It provided a simple sets of operations transmitted to the agent. The operations types are presented in Table 1.

Table 1 – Operations provided by the SNMPv1

Operation	Description
`get`	This operation is sent from NMS to the agent. Agent process this request and responses with get-response and values obtained from the network element. However the whole message has guaranteed deliver only if it has less than 484 bytes. Most implementations allow greater size though.
`get-next`	The get-next operation allows retrieval of sequential OID values of the MIB tree. An OID is composed of integers, so it is easy for an agent to find next matching result. For each get-next request a separate get-response from the agent is generated. The NMS keep sending get-next operation until an error is returned, signifying the end of the MIB.
`set`	To change the value of an object or to create a new row in a table the set command is used. More than one object at a time can be changed.
`get-response`	This operation is send back to the NMS when the request is processed and requested information acquired.
`trap`	This is a notification message send to specific destination configured at the agent itself. Usually it is the IP address of the NMS. As UDP protocol is used the traps are prone to get lost in the network, as no backward notification is send back from the NMS that the notification was delivered.

SNMPv1 was faced with two major problems, namely: very week security and lack of tools to manage larger amount of management information data. *SNMPv2* addressed one of this issues: it introduced new operations that allowed for obtaining greater number of data OID than just one at a time. The security aspects were not addressed until version 3 of the SNMP protocol. To date, the most popular variation of the SNMPv2 is the SNMPv2c (c for "community"), indicating the community strings as a security authentication procedure. Also in version 2 of the SNMP protocol new notation was introduced, that was mostly backward compatible – SMIv2. SNMPv2 introduced some new operations described in Table 2.

Table 2 – Operations added to the SNMPv2

Operation	Description
`get-bulk-request`	This command is similar to get-next operation but unlike the former allows for retrieving more than one OID at a time. With this command, the agent sends as many information in one packet as it can. Previously mentioned message size limit is still applied.
`inform-request`	This command is similar to the trap mechanism introduced in SNMPv1, but adds support for the acknowledge response from the master node. The agent sending this command is notified if the message reaches its destination.
`response`	It is renamed version of get-response operation. It resembles the change in protocol that now response is not bound to only any of get- commands..

SNMPv3 is the newest and recommended version by the IETF. It may be considered as SNMPv2c extended to also support solid security. Strong authentication and encryption of the SNMP messages makes it less prone to security attacks. The architecture was also modularized which helped in SNMP agent implementations. However due to the increased complexity and ubiquitous use of SNMPv2c it is still unclear if this standard will have the same market acceptance as their previous versions. One final operation, which was introduced in the draft of SNMPv2, but only made it to SNMPv3 was `snmp-report` and is explained in Table 3.

Table 3 – Operation added to the SNMPv3

Operation	Description
`snmp-report`	This command enables communication between SNMP engines to exchange processing problems..

The SNMP manager, which operates on the machine called Network Management System (NMS), sends requests to the device being managed. The SNMP agent, which is located on the managed device, responds to the received requests. To establish the notion of trust between managers and agents, both SNMPv1 and SNMPv2c use community string. It is a text string that act as a authentication password between the management station and the SNMP agent. There are three configuration names that can be used to configure a SNMP agent: read-only, read-write and trap. The community string is included in every packet that is transmitted between the SNMP manager and the SNMP agent. The problem is that in the versions 1 and 2c of SNMP protocol community string (passwords) are sent in plain text. SNMPv3 addresses this issue by providing secure authentication and communication mechanisms, and that is why only version three is officially recommended by IETF, the rest are considered obsolete.

To monitor network element, manager has to know what kind of information actually can be managed. The Structure of Management Information Version 1 (SMIv1) [27] defines managed objects in the context of SNMP. SMIv2 [19] expands and enhances this object descriptions. The description of managed object can be represented by the three attributes:

- **Name**

  This field uniquely defines the managed object. It may appear in two forms: numerical (OID) and more "human readable." It can be used interchangeable.

- **Type and syntax**

  Objects and theirs types are described in referenced earlier ASN.1. Because of using this universal notation e.g. the bit order of the master and agent machine is not important.

- **Encoding**

  During the transmission a single managed object is encoded into a string octets using Basic Encoding Rules [28] (BER). In this form it is transported over a transport medium like Ethernet.

A tree like hierarchy organizes the managed object according to the SNMP convention. Each managed object is explicitly identified by a series of integers separated by dots. Each node has its number and name by which can be referenced to, so two definition of managed object are equal: *1.3.6.1.2.1.6* or *iso.org.dod.internet.mgmt.mib-2.tcp*

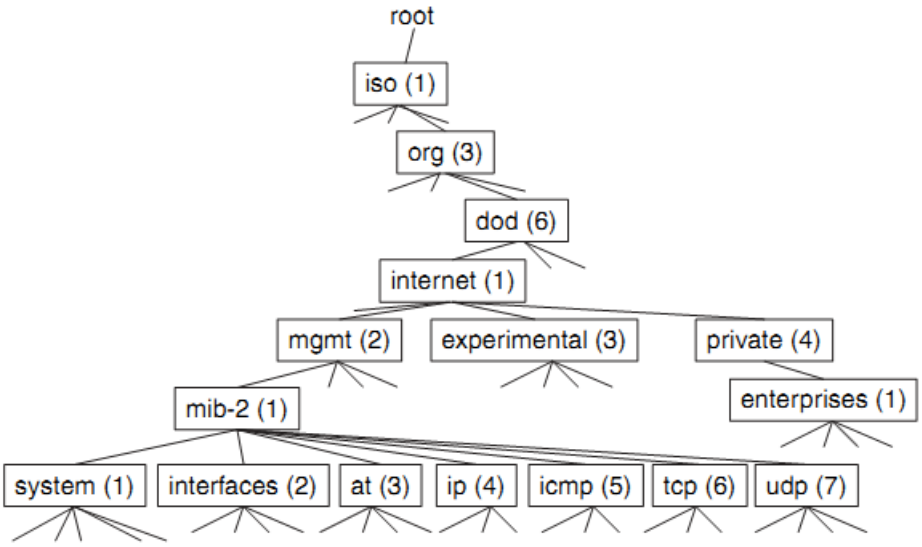

Figure 5 – SMI object tree [17]

The Internet Assigned Numbers Authority (IANA) [29] is the organization that manages association of numbers to the private customers or enterprises.

Identification information sent by SNMP is transmitted in SNMP message which is implemented at the Application Layer – Layer 7 of the OSI reference model. SNMP message structure is shown in Table – 4.

Table – 4 Structure of the SNMP PDU

IP header	UDP header	version	community	PDU-type	request-id	error-status	error-index	variable bindings

IP frame containing SNMP PDU information can be divided into four main parts:

- IP/UDP header information;
- The SNMP version number;
- A community string;
- The SNMP protocol data unit (PDU).

Since most of the fields were described previously main focus will remain in the PDU type field. However some information about version restriction must be noted also.

Because of the modular structure of SNMP agent, basic scope of supported MIBs in SNMP agent can be extended in three general ways:

- Running external commands;
- Loading new code dynamically;
- Communicating with other agents.

Each way of extending agent's functionality, if it is to be used, must be enabled during the building time of the SNMP executable file.

The first of these options uses support for the ucd-snmp/extensible and agent/extend modules. This was the earliest extension mechanism added to the agent's features. This allows for running arbitrary commands or shell scripts. Some problems may arise with more complex tasks, and with interpretation of the command output.

Most of the standard MIB are C coded modules. Such modules are compiled and then linked to the SNMP application when it is first built. The shared module is located in a separate binary file, and is loaded by the agent at runtime. This improves flexibility of adding support for the new MIB file after the agent has been compiled. The memory space is shared with the entire agent and extension module has direct access to entire API provided by SNMP. Drawback of this solution is that, since the memory is shared with the agent, a programming mistake may affect the entire agent daemon causing it to crash. Another issue is that the code always executes as the same user as the agent daemon itself, which may lead to granting too extensive privileges to the system resources, or too less. Use of this mechanism requires that the agent being built with support for the ucd-snmp/dlmod module.

The most flexible solution, which involves a communicating demon with other agents, is based on the idea of using Interprocess Communication (IPC). In this way, memory file descriptors are different from the ones of the master agent. The solution is flexible because the subagent can be started at any time, providing support for additional MIBs files. The code does not need to be changed regardless if it is to be used in dynamically loaded modules or in a separate subagent. To communicate between master agent and subagents the AgentX protocol [30] is used. To enable support for this solution, support for the AgentX module must be enabled during building time and also option enabling this must be explicitly enabled in the snmpd.conf file.

2.2 NetConf protocol

NetConf is a new network management technology standardized by IETF in December 2006 [31] and then revised in June 2011 [10]. It was designed to replace SNMP – the widely used network management technology, long adopted by industry, and to overcome some of its shortcomings. The biggest difference between NetConf and SNMP is that NetConf uses a communication mechanism based on XML-RPC [32], which, thanks to the tools developed in the scope of other web technologies, allows a simpler and faster development and management cycle. It also provides mechanisms to install, manipulate, and delete the configuration of network devices.

NetConf was designed to supports variety of transmission protocol standards and used data structure for management information. However a few are mandatory, namely SSH for transport layer, and YANG [33] data structures, which is used instead of SNMP's MIB. These are also the standards that were used in the design and implementing of this thesis protocol implementation.

NetConf server uses the IETF YANG Data Modelling Language. Its syntax and semantics are in a format that is human readable. The configuration information of the device containing the NetConf server is stored in a database in the YANG data structures. To prevent configuration loss between device reboots, these data structures are saved in non-volatile storage.

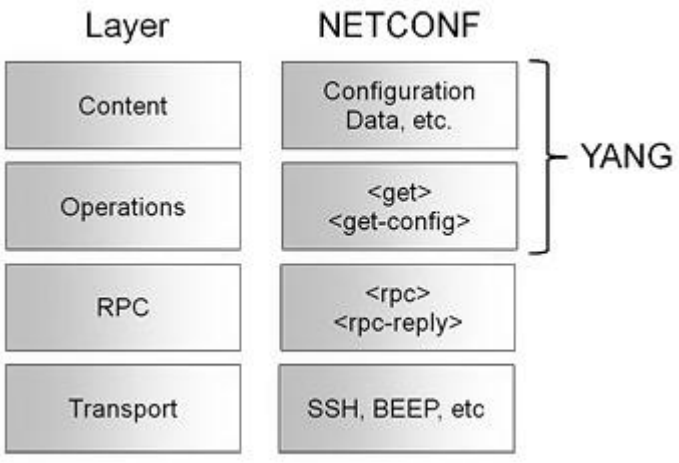

Figure 6 – NetConf Protocol Layers [34]

NetConf may be compared to XML-based, high-level version of previously discussed SNMP, and YANG module can be look at as more sophisticated counterpart of MIB file. Figure 6 shows protocol layers used by NetConf. The most widely used

protocol with NetConf connection is SSH. To work swiftly the "`netconf SSH-subsystem`" is used to listen on TCP port 80. After client-server connection NetConf is much like using CLI over SSH, but messages are exchanged in XML format, not in plain text. To authenticate and authorize session SSH user names, passwords and hosts keys are used, just as a standard SSH session would be.

Contrary to the SNMP, NetConf is a connection-oriented protocol. This means that a persistent and reliable connection must be maintained. Moreover authentication, data integrity, confidentiality, and reply protection are assumed to be provided by appropriate levels of the OSI model. One way of doing this is using mentioned earlier and default Secure Shell (SSH) protocol.

Figure 7 depicts connection handling by the NetConf application. Connection to the NetConf server can be done from any SSH compliant application. However, to simplify the management of NetConf server, clients instead of using raw SSH terminal application to direct communication with NetConf server, the NetConf client application called "`yangcli`" is used. It supports many automated features present in NetConf protocol.

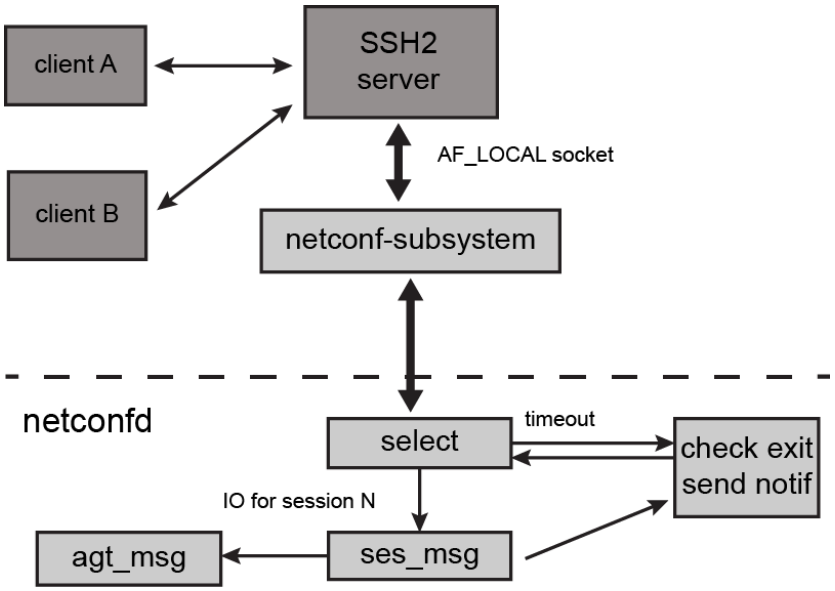

Figure 7 – NetConf input-output session [35]

Strict limitations are imposed on messages exchanged between client and server. Each NetConf message must be well-formed XML, encoded in UTF-8 [36]. If one side of pending communication receives message that is not conforming to the standard it

should reply with a “malformed-message” error or, if reply is not possible, terminate the session.

After starting the NetConf session clients sends commands to the server which processes them in order and returns results to the client. Asynchronous notification messages are also allowed, provided client requested in active session `<create-subscription>` operation.

At the beginning of each session client and server exchange a `<hello>` message to learn about each other capabilities. Thanks to that applications became aware of which operations, notification events, and database contents are supported by the other side.

Following the agreement of agent’s and server’s supported operations there are two types of data that can be transmitted: configuration and state data. The configuration data includes writable data that transforms a system from its current state into desired configuration. Statistical and read-only status information are transmitted as state data. A number of problems could arise if this distinction was not drawn. For each type distinct operation is provided, the `<get>` and the `<get-config>` respectively.

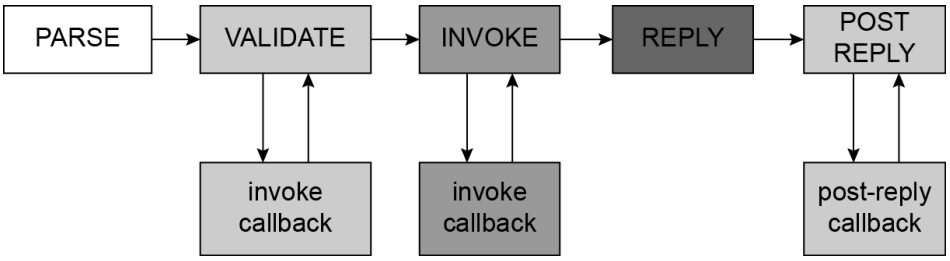

Figure 8 – RPC validate phase [35]

NetConf standard uses the RPC model to encapsulate its requests in transport protocol independent method. The client uses as a request the `<rpc>` element, servers responds with `<rpc-reply>` which is send after validation phase, as shown in Figure 8. An important attribute send in these elements is the “message-id” which is an increasingly higher integer encoded as a string and chose by the sender. Server must return the same value in the `<rpc-reply>` operation.

Important to mention are also `<rpc-error>` and `<ok>` elements. The latter is returned when no error or warning was encountered while processing the request. The `<rpc-error>` element has its own set of types and tags to indicate the location and severity of the problem.

Another extension compared with SNMP that NetConf supports is the notion of multiple datastores. A datastore holds the complete set of configuration data that takes the network element from its default state to the desired, configured state. The basic, most important and compulsory is the `<running>` configuration datastore. This element is always present on the device and can be only one copy of it. Other optional datastores, i.e. `<candidate>` and `<startup>` can be added to the device. Information regarding full support of these elements is sent in initial `<hello>` message exchanged by server and client specifying each peers' capabilities.

In Figure 9 the three-phase validation of datastore is depicted. In the first, validation phase the server determines the target nodes and database to be affected. Next, the validation of all incoming request against YANG language standard occurs. If everything checks, the node callback function is searched inside SIL libraries. If found the function is called.

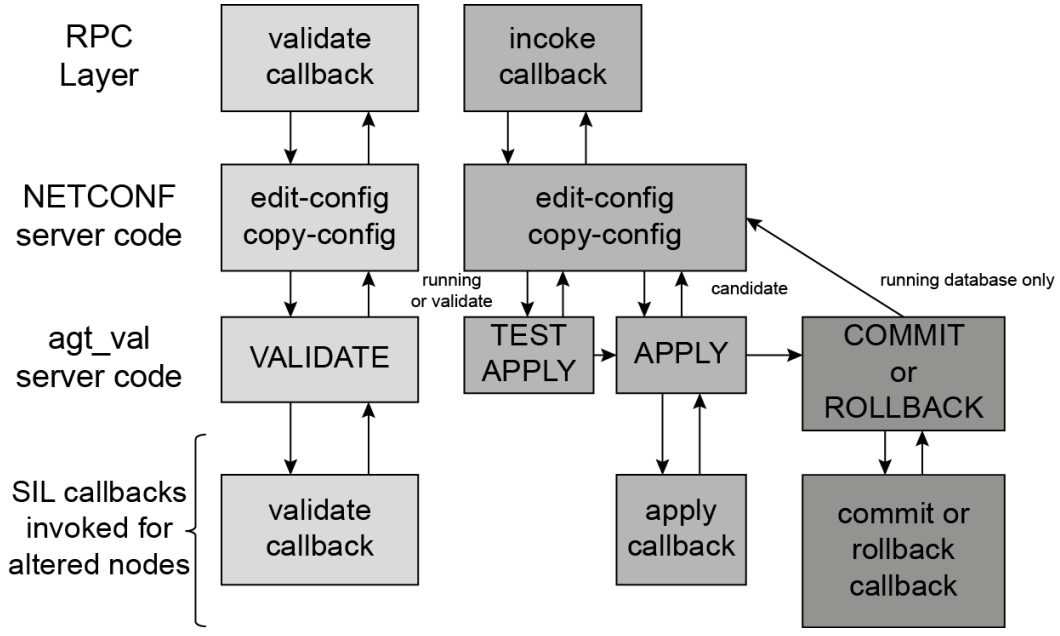

Figure 9 – NetConf database editing model [35]

If validation phase goes smoothly without errors, next phase of apply or apply-test follows. During this period, depending on configuration parameters sent with the edit-config operation the database or its copy is altered as requested. If the latter occurs, the test-apply phase is executed first, where, after altering the copy of internal data tree, it is further validated, including all cross-referential integrity tests. Then, provided no error occurs, the changes are made to the real target database.

If operations <commit>, <edit-config> or <copy-config> are requested, the server will search for SIL commit callback function for designated element(s). If earlier stages produce an error or user cancels or timeout the commit operation, the server will call a function to rollback the operation and a backup copy of database will be presumably restored.

If additional operations are supported by the device, it should be announced during exchange of the <hello> message at the beginning of session. Basic messages supported by the default NetConf installation are described in Table – 5.

Table – 5 NetConf basic protocol operations [10]

Operation	Description
<get>	Retrieves running configuration and device state information.
<get-config>	Retrieve all or part of a specified configuration datastore.
<edit-config>	Loads all or part of a specified configuration to the specified target configuration datastore.
<copy-config>	Create or replace an entire configuration datastore with the contents of another complete configuration datastore.
<delete-config>	Delete a configuration datastore. The <running> configuration datastore cannot be deleted.
<lock>	The <lock> operation allows the client to lock, usually for short amount of time, the entire configuration datastore system of a device.
<unlock>	The <unlock> operation is used to release a configuration lock, previously obtained with the <lock> operation.
<close-session>	Request graceful termination of a NetConf session. Processing this request results in releasing any locks and resources associated with the session and gracefully close any associated connections. Requests received after a <close-session> request will be ignored. If request is satisfied an <rpc-reply> with <ok> element is replied. Otherwise <rpc-error> with status information.
<kill-session>	Force the termination of a NetConf session, aborting any operations currently in process, releasing any locks and resources associated with the session, and closing any associated connections. If <commit> operation was issued the backup configuration must be restored.

Figure 10 depicts the set of available Yuma tools which helps during the development process. It also shows possible course of action during developmental process. Presumably, if we are implementing functionality described previously in SMIv2 or other information database format "smidump" converts it to YANG modules. These files are very flexible and are interpreted directly without further translation by all Yuma applications and are parsed straight by "yangcli" and

NetConf executable. "`Yangdump`" translator renders them to C source files. For ease of implementation callback functions associated with modules and node names are automatically generated there. After implementation is ready files are compiled into its binary format called Server Instrumentation Library (SIL). Placed in an appropriate catalogue can be automatically loaded into the NetConf server at run-time.

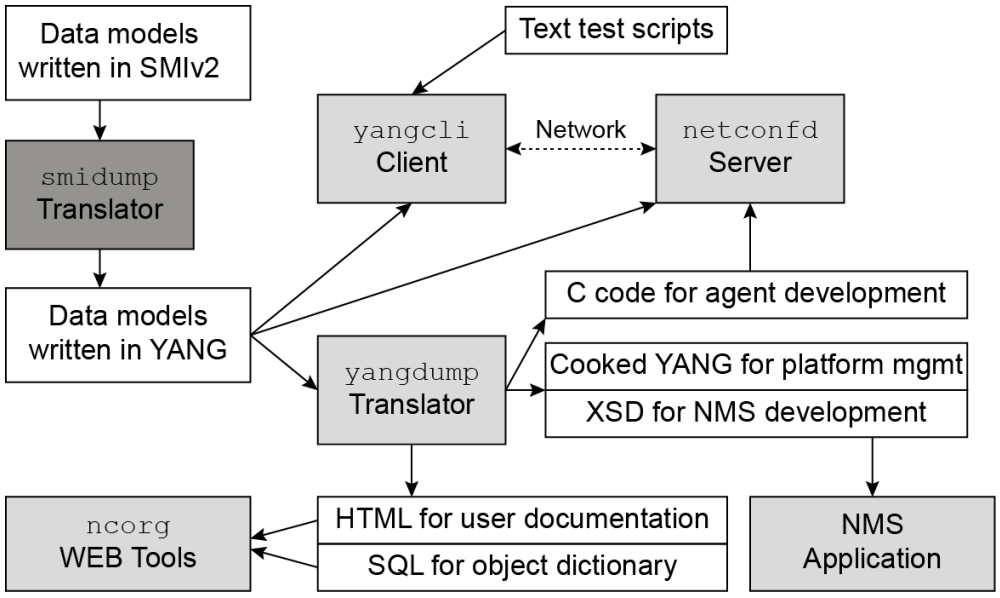

Figure 10 – Yuma Tools and Server Instrumentation Library [35]

Chapter 3

Hard Real–Time Ethernet Switch

HaRTES is a modified Ethernet switch based on the FTT paradigm. The idea behind developing the FTT-compliant switch was to extend the structural limitations of the FTT-SE protocol, which cannot be resolved with standard Ethernet switches. The Flexible Time Trigger-Switched Ethernet (FTT-SE) [22] protocol needs all nodes to be completely respecting the EC-schedules provided by TM. To enable such system to work in each node must be present a specific network device driver. The driver might not be available in several operating systems. Moreover misbehaving or broken nodes which do not respects the timeliness strict traffic can introduce several communication problems including completely jeopardizing timing guarantees. Solution to this was to introduce temporal control to the switch. Therefore FTT master was inserted into the switch providing seamless support for synchronous and asynchronous traffic as depicted in Figure 11.

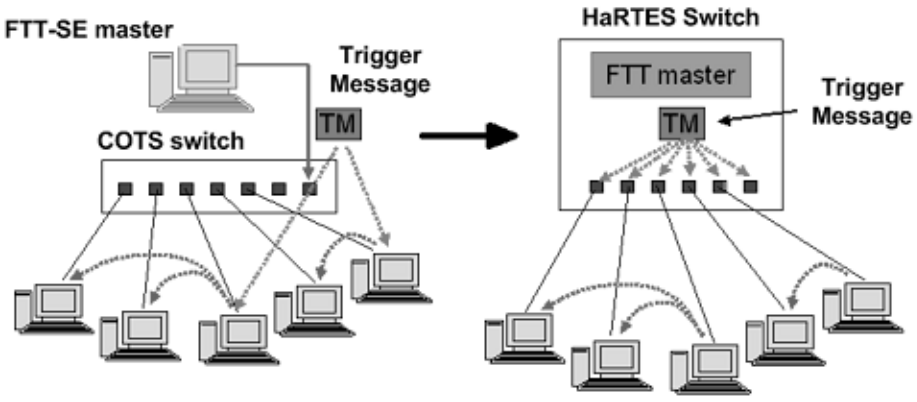

Figure 11 – HaRTES, an FTT-enabled Ethernet switch [37]

Due to inclusion the FTT master into HaRTES switch several the most relevant features of FTT-SE were preserved, which fostered several facets. First of them is the handling of asynchronous traffic is simpler. The nodes can autonomously trigger the traffic instead of being polled by the master node, together with maintaining per-stream temporal isolation. Next, the system integrity is increased. The switch input ports can block unauthorized real-time transmissions, isolating it from the rest of the system. Seamless integration of non FTT-compliant nodes without threatening the real-time traffic was also fostered. Lastly, superior transmission parameters with TM higher precision, lower jitter and latency. Because of that overall network synchronization is improved.

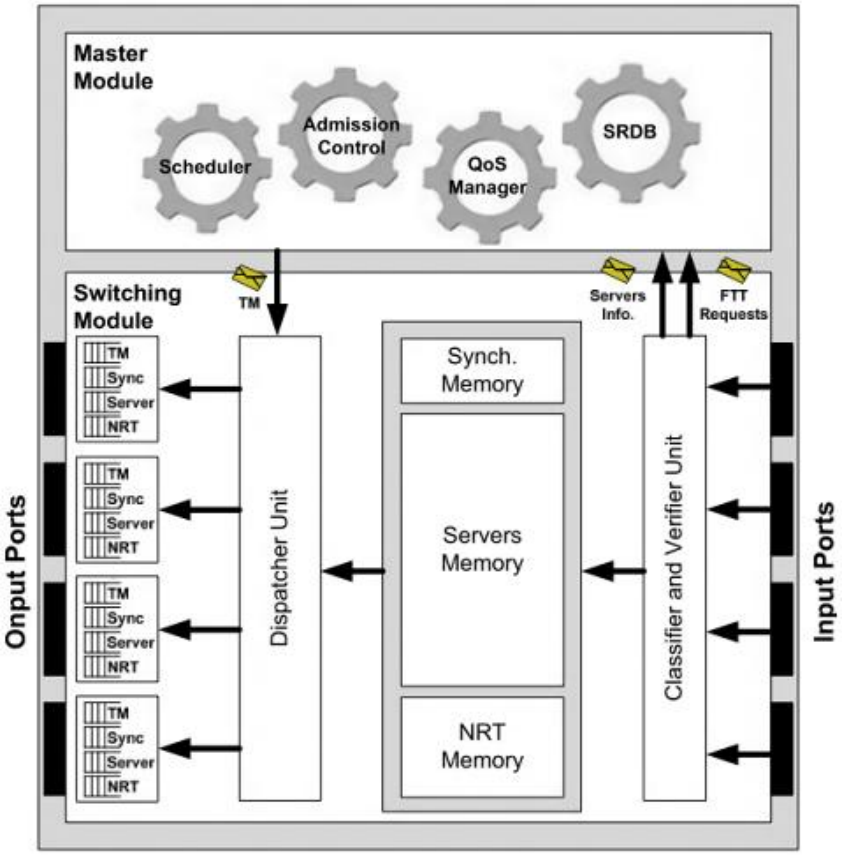

Figure 12 – Functional architecture of the server-based [38]

The switch is split on two main components: the software component and the hardware component as depicted in Figure 12. These components of the system are: Master Module, and the Switching Module. The Switching Module units were modelled at register transfer level (RTL) with the Field-Programmable Gate Array (FPGA) and

written in VHDL hardware description language. In this configuration, it is easily reconfigurable for testing purposes and the overall cost of the system is greatly reduced.

The Master Module and Switching Module are, in this implementation, connected using Ethernet port. This allows for fast and flexible communication between the two parts of the system.

3.1 The Master Module

Thanks to the modular implementation of the FTT-SE, the master and slave component can coexists in the same node, complementing each other and saving on the hardware platform costs.

The Master Module implements the FTT master which is depicted in Figure 13. The shaded area represents management services responsible for the FTT traffic. This part of the switch is algorithmically complex and tends to use much of the available system resources. Sophisticated sets of operations included in the system requirements database, the scheduler, the admission control and the QoS manager included in this module make broad use of dynamic lists. Those elements are more resource demanding and needs more resources. Because of it, and mentioned earlier, potentially grater costs of the hardware the Master Module was, in this case, implemented in software.

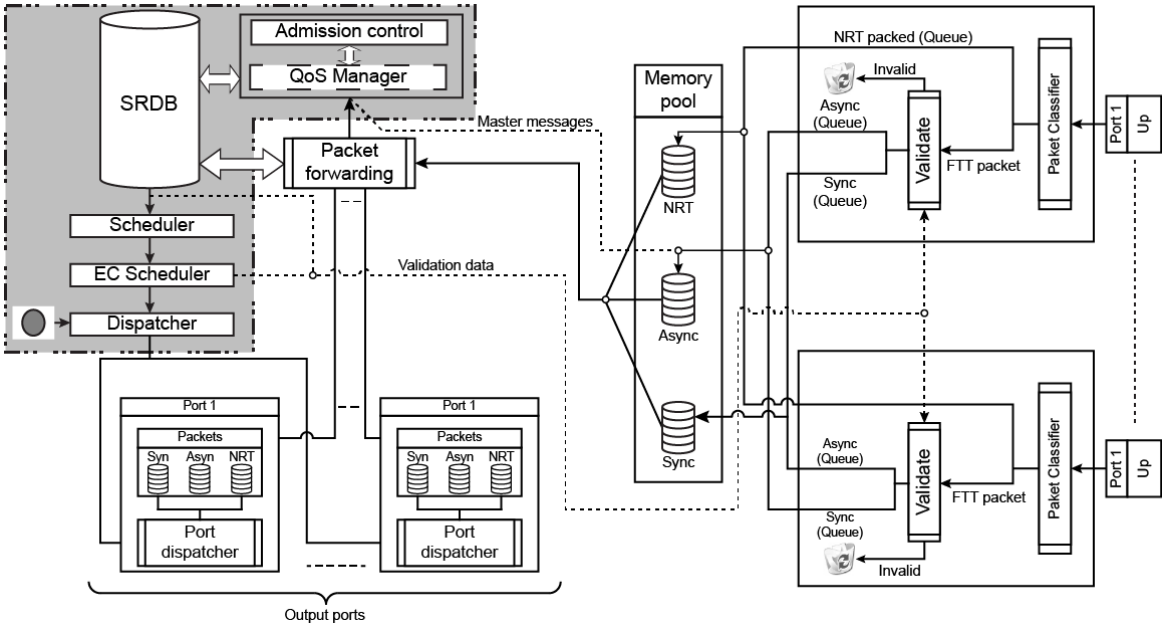

Figure 13 – HaRTES Functional architecture [22]

Running many services in many nodes as it take place in distributed systems, introduce a challenge to efficiently manage it. That is why the middleware between user application and the node service is introduced. It also makes efficient opportunity to introduce Stream Reservation Protocol (SRP) [8].

The FTT-SE middleware introduce a binding application interface common to all nodes, available at all of them, managing communication details and bringing forth seamless design composition. Thanks to the middleware synchronization mechanisms and a logical abstraction responsible for associating services with the nodes are assured to have consistent network loading accounting and resource allocation. This is especially essential in open and dynamic environments. An abstraction layer also fosters application development and its deployment in the field. Better application integration is based upon the middleware abstraction layer which is crucial for retrieving HaRTES data values thus several of its features are described next.

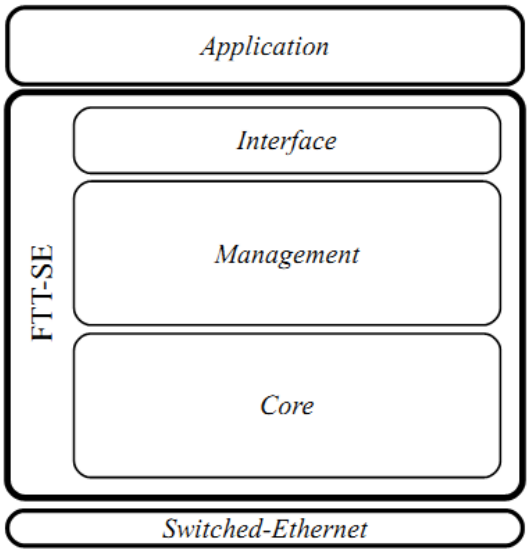

Figure 14 – FTT-SE internal layering [22]

Figure 14 presents several building blocks which detailed overview is depicted there. In the bottom of the stack is the Network Interface Card (NIC) controller which interacts with full-duplex Switched Ethernet network. Inside the FTT-SE core layer one can find basic protocols for communication mechanisms i.e. the traffic scheduling and the transmission control. The management layer performs a high level session control, which includes establishing connections and differentiation of the streams. This leads to decoupling the streams and endpoint threads and makes possible to register them

independent of their location in the network and regardless of their position as producer or consumer. Also in this layer a QoS management module guards proper distribution and adjustments of the network online capacity splitting it evenly between stream. The rules are set by the application based on the needs and importance of the streams. The last discussed part is the interface layer. In there rests the communication and management services available to the application allowing for sending, receiving, binding, resource reservation and de-registration in the FTT-SE protocol standard.

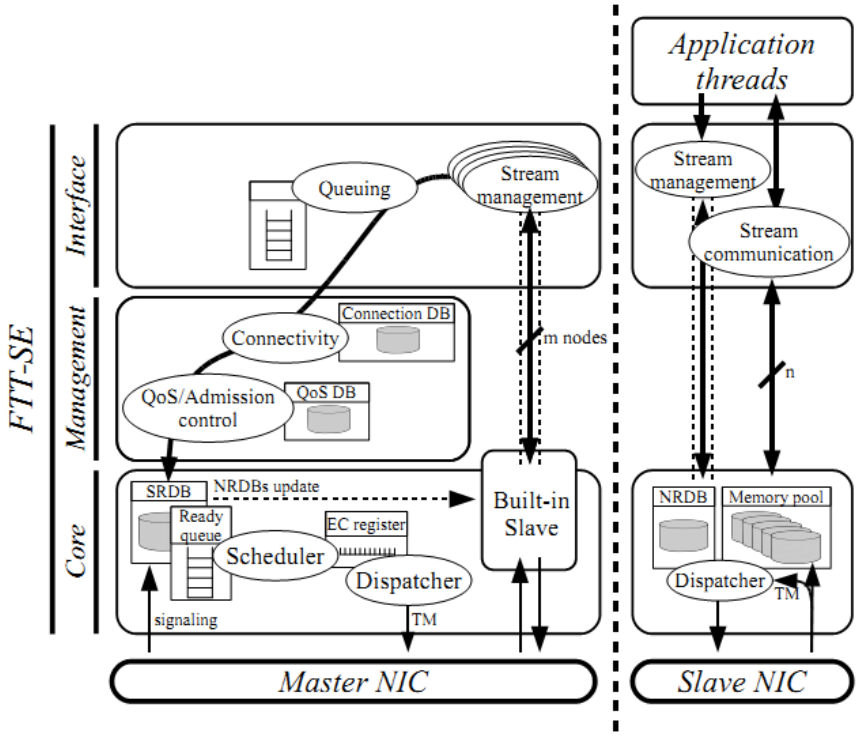

Figure 15 – FTT-SE internal details [22]

Figure 15 presents more detailed implementation of the internal services cooperation. As mentioned earlier, one of the most important parts of the whole switch is the System Requirements Database (SRDB) where information about synchronous and asynchronous messages (e.g. minimum/period inter-arrival time, deadline, priority, length) and running traffic properties are kept alike. Also information about global configuration (e.g. data rate, synchronous window duration, elementary cycle duration) and each traffic class allocated resources information (e.g. maximum amount of buffer memory, phase durations) are stored there. A scheduler is a module which periodically scans the SRDB in search for data necessary to build TM for synchronous traffic. Each sequential EC is built based on a list of synchronous messages (EC-Schedule) stored in

EC-register. Module responsible for this is the scheduler which monitors the SRDB and responds to the changes in it. The messages are pushed into the ready queue and transaction plan is created. Next the EC-schedule is broadcasted within the TM by the Dispatcher module. Independent scheduler for every traffic class and ready queue is present. Each of them follows different policies established by the length of the EC (LEC).

Synchronous and asynchronous messages are activated differently depending on the model adherence. The synchronous are activated in a periodic basis. The asynchronous are triggered based on the triggering mechanism of the node. They are waiting for master to transmit permission order. Meanwhile, the signalling message is sent from node to the master node informing about the messages in the queue. The master during building following EC takes these information into account and, based on the traffic class, enforces the nodes triggering mechanism to send the messages. The asynchronous traffic without the real-time compliance is transmitted transparently in background in a best-effort basis. The nodes uses FIFO queues for handling this kind of traffic. The transmission frame is filled based on no schedulability polices or overflow-free guarantees. The rest of the message operations are similar with no regards to the messaging model.

In this type of communication it is very important to ensure that streams are registered consistently across the master and slave nodes. The mechanism ensuring that the master SRDB and the nodes NRDB are the same, must be present and sound. FTT-SE protocol uses mechanism that transparently synchronizes every database which is bound e.g. the NRDB databank sores the model properties of all streams passing through that node, thus storing and sending to the master node the long-lasting information about it. On the other hand, if the SRDB is updated every bounded NRDB is also notified. Moreover, all configuration command that modify the network requirements are distributed from the master node in one EC. Therefore consistency and unity of configuration between all nodes in the network is guaranteed, suppressing the need for additional synchronization mechanism.

Present synchronization mechanism uses the built-in slave component in the master to create an asynchronous broadcast channel between the master and the slaves.

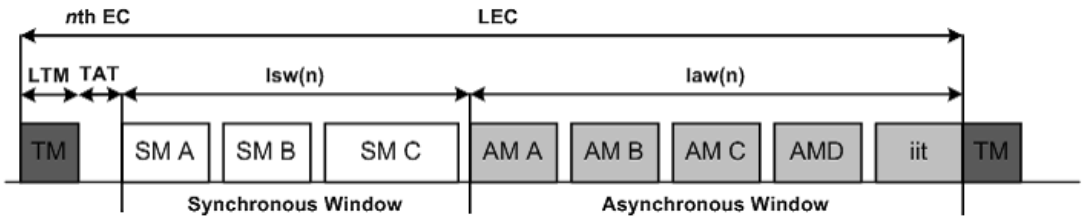

Figure 16 – HaRTES elementary cycle (EC) structure [37]

The SRDB with admission control and an optional QoS manager cope with the sustention of the continued real-time traffic timeliness. Admission control module ensures that request for a change in traffic flow will not jeopardise any of the timing guarantees previously negotiated between the master node and slaves.

Also important issue to address, concerning HaRTES, is an Elementary Cycle (EC). It is used to communicate between nodes in a scheme of constant infinite sequence of windows. The FTT-SE protocol, in modules described previously, builds the EC frames respecting their class, streams and model. The frame, structure of which is presented in Figure 16, consist of synchronous and asynchronous windows which carry out synchronous and asynchronous traffic correspondingly. Due to simplicity sake the organization of the inside of the frame is fixed – the asynchronous window every time comes once the synchronous window period is over.

Each EC frame starts with a control message, which is called Trigger Message (TM). It is broadcasted to all other slave nodes in the network. The objectives of this message is to synchronize all the network, because this message is always broadcasted in the precise intervals of time, with low jitter. The other objective is to transmit the EC-schedule – time when the slaves can transmit and their identification data. Upon receiving the frame by slave internal algorithms checks if this node is the producer or consumer of any data, and act respectively during scheduled time interval.

The communication structure is configurable in a way of changing several temporal parameters. The most prominent parameter is an Length of Elementary Cycle (LEC). This is de facto system working resolution, because all other parameters e.g. periods, deadlines, windows, are relative to this interval.

In the EC structure three main slots are identified as depicted in Figure 16. The LTM and TAT form the time required to actually transmit the TM to all slaves and the time associated with different propagation and decoding time, which is essential by the nodes to prepare data to be sent. The first is known as transmission time of the TM

(LTM) and the other as Turn-Around Time (TAT). Next two time slots are the synchronous and the asynchronous windows correspondingly. The maximum duration of the synchronous window (LSW) can be defined be the FTT master scheduler. Because of that, a minimum duration can be guaranteed for the asynchronous window.

The strict temporal isolation enforced by the schedulers, between these both types of traffic, imposes that no traffic which could end after the Synchronous or asynchronous time window limit is initialized. In the asynchronous window all the messages are transmitted before the next EC. In case of synchronous window the scheduler guards the temporal isolation in a way that messages each time fit inside the maximum time of synchronous window duration (LSW).

3.2 The Switching Module

The Switching Module is responsible for the reception, switching and transmission process. It also handles the memory management. For the HaRTES to work efficiently the switching logic require speed during the execution, determinism and predictability, hence hardware implementation of this component was carried out.

The implementation was based FPGA technology allowing an easy and flexible integration of all required components. Because of this design prototype of HaRTES switch is easy to parameterize giving the possibility of changing network communication speeds, number of ports, core operating frequency or width of internal databus.

Figure 17 depicts the hardware architecture of the Switching Module with Master Module. Each of showed components is briefly described below.

Ethernet PHY

This device, known also as Ethernet physical receiver, operates at the lowest layer of the OSI model – the physical layer. It is connected directly to the Medium Access Controller (MAC). PHY purpose is to communicate through cable with other Ethernet PHY, exchanging data. Due to electrical characteristics and timing requirements this component is outside the main FPGA board. Each port has its own instance of this device.

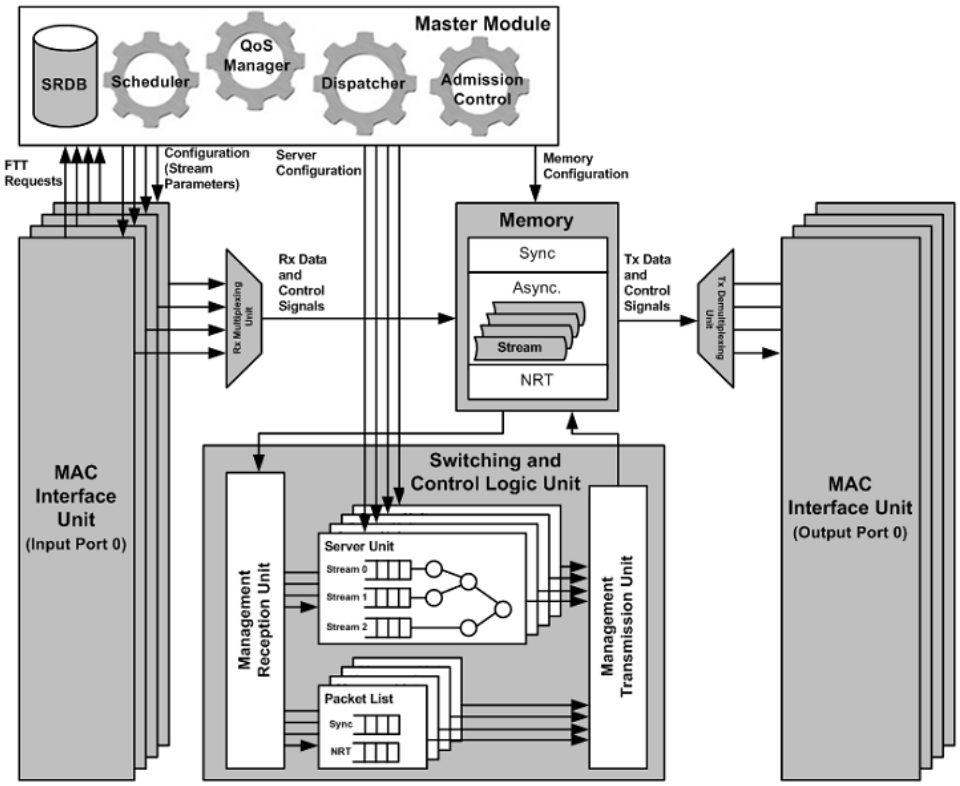

Figure 17 – Switching Module hardware architecture [37]

Medium access controller (MAC)

The MAC Interface Unit is integration of two blocks, namely MAC Interface and MAC IP Core module. The firs one operates at data link layer of the OSI model and is responsible for handling and decoding physical Ethernet frames. The following module handles packets data flow and, in accordance with traffic class, redirects it to different modules of the witch: non real-time traffic is passed directly to the Rx Multiplexing unit, traffic with the commands of the Master Unit are passed there, and the rest of traffic is verified – real-time packets are checked for time constrains. If time requirements are not met, the packets are deleted. Because of strict verification of packets in this early stage of traffic management the switch and network integrity in the realm of timeliness guarantees in guaranteed.

Rx Multiplexing Unit

The Rx Multiplexing Unit is shared among all switch ports. It is a TDMA wheel which ensures that all packets from all ports, even if they arrive simultaneously, are delivered to the main Memory Unit with no packets drop.

Memory Pool

The Memory Unit consist of a dual port static Synchronous Random Access Memory (SRAM). Memory port is accessed by decoding signal of control, address and data bus data. It is important to note that one memory port is shared with all switch ports.

As for the upcoming traffic, the memory leakage for synchronous packets is never the issue – the Master and MAC Interface Module ensures that by controlling the amount of information sent by slaves in EC-schedules. With non-real-time traffic no such guarantee can be provided and overfilled packages are deleted.

Control and Switch Logic Unit

The Control and Switching Logic Unit performs a crucial tasks of reception, switching and transmission packets to different destinations. It has a central role in the switch. It is controlled by the control and status signals of the MAC Interface Unit. This unit is also responsible for generating synchronization signals which coordinates all the switch and protocol operations.

The Management Reception Unit is responsible for packets forwarding to the specified output port. To utilize that process real-time and non-real-time packets are placed in separate queues, waiting to be fitted into frame and send out. The TM supplies the synchronous packet list that is put into the queue. For asynchronous real-time traffic the validation of packets proceeds the insertion into its queue. As for non-real-time traffic first the memory overflow condition is checked. If no such error occurs the NRT packets are queued and send out in a FIFO policy.

The Management Transmission Unit carries out, in any given EC, transmission of the packets gathered in queues. Special policies are applied to ensure that transmitted packets from asynchronous and NRT queues fit into the time windows they are provided, same not blocking TM and synchronous traffic.

Tx Demultiplexing Unit –TDU

The role of this module is somewhat similar to the Rx Multiplexing Unit. This module works as demultiplexer and is also shared among all switch ports. It reads flow of the packets data from the memory pool and redirects it to the corresponding output ports of the switch.

3.3 Real-time management systems

A real-time system is defined as a computation that responds to internal or external events within a dedicated periods of time [39]. There are two categories of a real-time system: hard and soft. If failing to meet a deadline is catastrophic to the functioning of the system it is called hard real-time system. Catastrophic in a sense that failing meeting the deadline may involve huge material losses or human lives. This systems need to be very predictable and carefully designed considering pessimistic scenarios. If missing the deadline causes non-catastrophic consequences e.g., multimedia streaming or interactive computer games, it is called soft real-time system.

Networked Embedded Systems (NES) are widely disseminated in many application domains ranging from industrial automation to building automation and vehicular system, serving in both soft and hard real-time areas. Some application domains exhibit strict timeliness, predictability and precedence constraints. In these cases, special-purpose real-time communication networks, known as fieldbuses, must be used to achieve the desired properties.

One network technology that became widely used in these systems is Ethernet, however, it was not originally developed to meet the requirements of NES, namely in what concerns key aspects such as predictability, timeliness and reliability. There are currently available technologies, that enable mechanisms of guaranteed Quality of Service (QoS), such as MPLS [1] and RSVP [2] especially combined with IntServ [3] and DiffServ [4] models, however they only provide statistical guarantees.

The timeliness guarantees provided by those protocols are essentially static, based on pre-run-time analysis. On-line admission control is not generally available and neither is on-line adaptation of the communication requirements according to effective needs or to a quality-of-service (QoS) adaptation policy. This has motivated the development of a new generation of Ethernet switches, Hard Real-Time Ethernet Switch (HaRTES). For the purpose of preforming QoS reservations, SRP support was added.

However, HaRTES new technology was lacking remote management support which in a context of extensible testing and constant development was a major disadvantage, making it hard to monitor its internal values in a real-life implementations.

Developed in scope of this dissertation management solution extends HaRTES capabilities providing an internal interface with use of the custom HaRTES Management Protocol to communicate with management clients. Additional Remote Management Service was added to the Master Module of the HaRTES device. Extra subagent and Server Instrumentation Library were also compiled as to enable support for the newly proposed HaRTES Management Information Database.

Chapter 4

Multiprotocol management interface

To allow other application communicate with the Hard Real Time Ethernet Switch through network, capabilities of the SNMP and NetConf technology were extended. Integrating HaRTES monitoring functions to the management agents implied the design and incorporation of suitable subagent or additional Server Instrumentation Library. Extensions to the management tools were built with designed from the scratch Main, Coding and Retrieval Modules. Also during this work, the added module to the Master Module, the Remote Management Service, was designed from the ground. Its purpose was to expose internal values of the switch , by communicating with different modules already existing in the server, to the external world – in this case with subagent or SIL. A custom communication interface, the HaRTES Protocol, was developed to manage it. The HaRTES Protocol provides a convenient way to communicate with the Remote Management Service and to access the internal state variables, based on the existence of known data structures. Managing a given attribute (e.g. getting its value) starts by issuing a command with the module and the node name, explicitly indicating the attribute to be retrieved. Currently, it is supported manipulation of attributes, as well as support for the generation of events (i.e. traps, notifications). While this interface aims to provide an efficient way of communicating with the internal components of the HaRTES switch, it lacks standardization. Therefore, an architecture was devised to interface the HaRTES switch with standard management tools.

Both of the management tools need to have their version of the management software installed on the machine running the Master Module. In case of SNMP it is the

agent, the latter uses NetConf server. To allow agent and server software to monitor the Master Module extensions to both applications must be provided. The HaRTES implementation of the remote management technologies is split between two main components: the extension to the SNMP subagent or the NetConf server, and the component running in the Master Module of the switch.

As for SNMP agent an additional subagent application was developed. NetConf provides tools for developing the Server Instrumentation Library (SLI) which extends NetConf server capabilities for managing additional modules. The SNMP subagent and additional Server Instrumentation Library for NetConf are responsible for communication with the Remote Management Service module running in the Master Module.

Figure 18 depicts the scenario which takes place during the retrieval of HaRTES data values. The main work was done in modules marked by the box.

In this scenario remote administrator using SNMP or NetConf protocol compatible application communicates with the subagent or server correspondingly. The request is processed by the application. The SNMP daemon gets the request form Net-SNMP transport and processing layers. Then the SNMP core based on which registered module it possesses indicates how it can respond to a particular OID sub-request and core suitable agent helper to handle request that request it called. The helper communicates through AgentX protocol and retrieves the data.

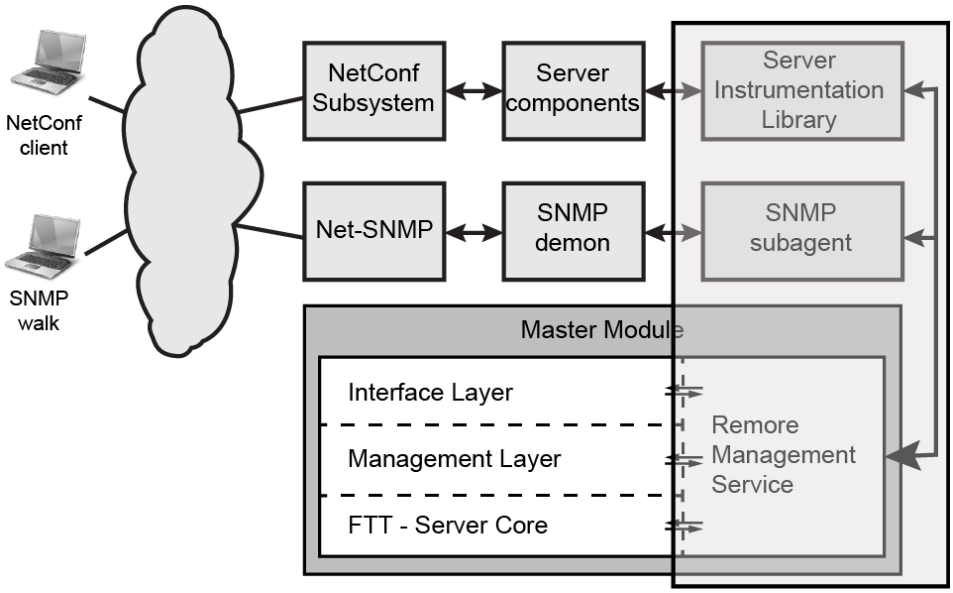

Figure 18 – Scenario of remote access the HaRTES data

For NetConf sessions, NetConf application calls XML parsing components decoding request information. After verification by the access control module the permission is granted and using suitable mechanism, YANG database Server Instrumentation Library and suitable callback function is called.

The SNMP handler and NetConf SLI runs custom module which sends the request to the Master Module of the switch. The request is coded in HaRTES Remote Management Protocol. In the Master Module custom module opens the server socket and listens for incoming request for data.

Upon receiving the request the Remote Management Service checks it for validity and decodes it. Then adequate layer and function is called to retrieve the data. Providing no errors occurred the data are coded into the HaRTES Remote Management Protocol and sent back to the subagent or NetConf server.

After receiving the coded frame, module is called to decode it and check for validity. If everything checks, the data are copied back to the memory of SNMP subagent or NetConf server.

The data are then handled by the cashing mechanisms of the subagent and NetConf server. After gathering all of the requested values data are prepared, coded and send back to the Network Management System issuing the request.

4.1 The SNMP subagent and the NetConf server module

After the wrappers of management application (SNMP or NetConf) process the request, the custom build module is called to handle the retrieving of the HaRTES data. In case of SNMP, as described in section 2.1, additional subagent must be connected to fulfil request, which is depicted in Figure 19. NetConf uses its own Server Instrumentation Library, described in section 2.2, to retrieve HaRTES data, which is shown in Figure 20. Components of this system were design as modules so to ensure flexible programing solution and more stable functioning and error handling.

The Main Module in the SNMP subagent and NetConf SIL, bind together other custom modules added to the applications. This allows for easy data flow control inside the extended SNMP and NetConf wrappers. It is responsible for invoking modules in control of coding, decoding, setting, passing to and receiving from the HaRTES its data values. Most of the parameters are passed as addresses to save the resources and to limit the load of the system.

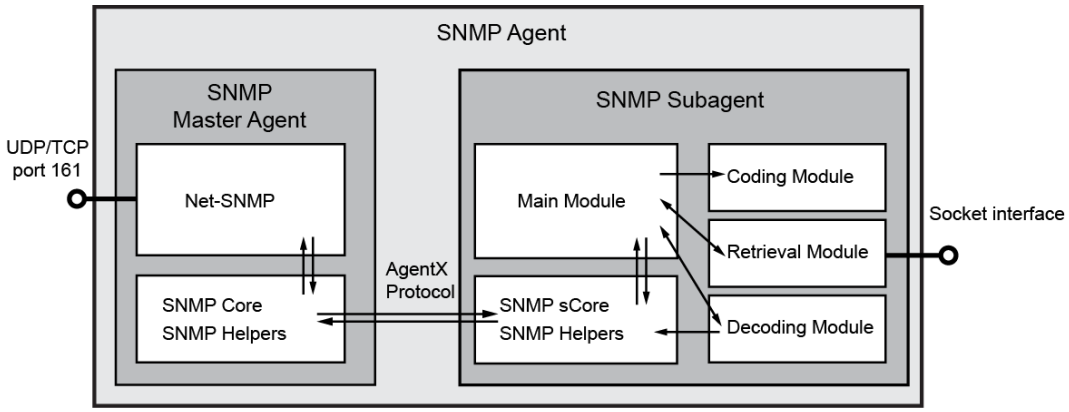

Figure 19 – SNMP Agent architecture with connected SNMP subagent

The Main Module gets two most important parameters: the node name and the requested variable. The node name is the text identifier representing the OID being processed. That convention was used explicitly to be fully compatible with the NetConf server variable identification. The second parameter points to the address of memory to where the retrieved variable should be wrote to. The numbers of bytes to be wrote are defined in the Remote Management Service of the switch. This prevents the memory leaks during passing the values to the management application. The Main Module also statically reserves the memory for the frame being sent down to the switch and for the response frame being sent up to the waiting module.

This parent module passes frame to the Coding Module to code the request info into the HaRTES Management Protocol. After the protocol frame is built, it is passed to the Retrieval Module which is responsible for sending and receiving data from the Remote Management Service.

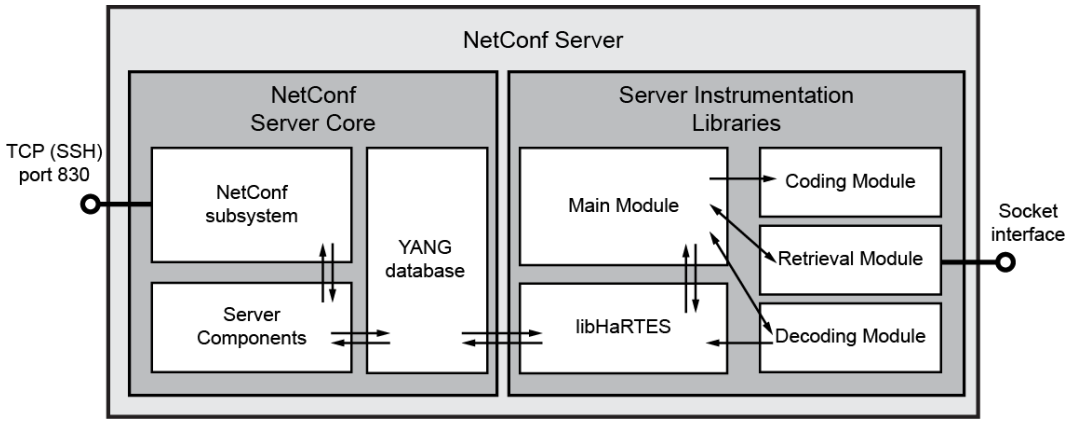

Figure 20 – NetConf server with Server Instrumentation Library implemented

The Main Module also compares the sent and received module and node names coded into HaRTES Management Protocol. If they are equal, the Decoding Module is called with received frame as parameter. There actual decoding of the FTT-Enabled switch values takes place. After translation of the protocol, the data are copied to the memory of the SNMP subagent or NetConf server.

In case of receiving values for not this module or node name, the error is issued, passed to the SNMP agent or NetConf, and no data are copied.

4.2 The Remote Management Service

The Remote Management Service is part of the Master Module and it is the module that SNMP subagent and NetConf server SIL are interacting with.

Communication starts in the management application. The SNMP subagent and the Server Instrumentation Library of the NetConf server are translating the request form agents core to the HaRTES Management Protocol. Then the data is sent to the server through internal socket interface (Figure 19 and Figure 20). The Remote Management Service, when initialized by the Master Module, opens the socket and listens to the requests for the switch data. The use of internal socked communication interface was devised due to several reasons.

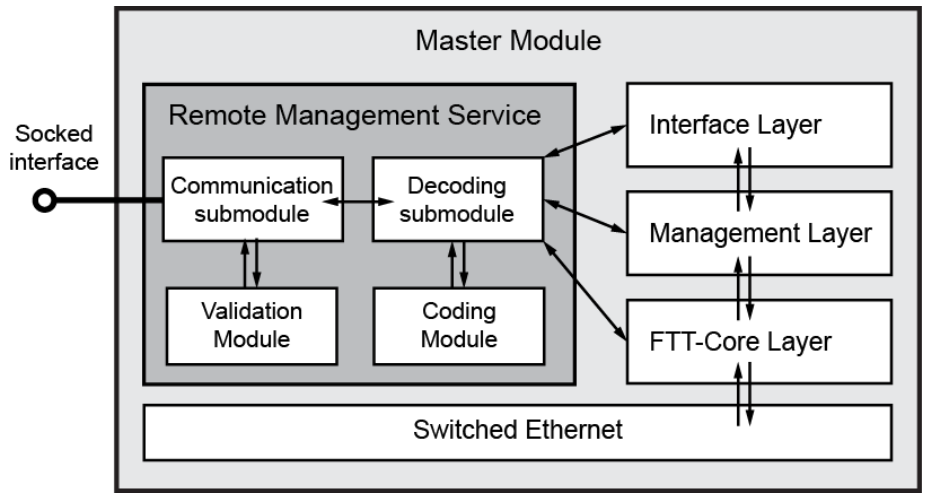

Figure 21 – Master Module architecture with the Remote Management Service

Sock interfaces better isolates operation of the management application wrappers and the Remote Management Service. This aims at reducing the impact of management operations to the Master Module of the HaRTES switch. Isolation also

reduces the number of exploits through the management interface, as if required, firewall characteristics can be added to the Remote Management Server. The second characteristic of this interface is that it allows reuse of the same management interface to several different management solutions. In this case the same interface can be easily used with both SNMP and NetConf, while relying on components requiring a small number of modifications from the components auto-generated by the support tools.

In Figure 21, the architecture of the Master Module with its submodules is depicted. Two main components of the Remote Management Service are Communication and Decoding submodule. The Communication submodule is responsible for opening the socked, listening to the incoming requests, accepting the HaRTES Protocol data and sending response back to the remote management application. The work of the Decoding submodule is twofold: first it decodes module and node name, and compares it with the names that it can handle. Next, if the module and node name is found, it calls appropriate method to get the value from the HaRTES corresponding layer.

The Remote Management Service uses in internal data structure which provides improved handling, gathering, validation and coding of the data passed from one layer to another. This structure holds information about the module and node name currently being processed. During the processing period it also gathers other information, such as: module name, node name, protocol data type, data length, HaRTES data value.

Implementing internal data structure allowed for easier managing the parameters between the switch layers. Table 10 shows full listening of the values accepted by the structure. Worth noting is that data value uses union, which reduces use of memory space. This can become crucial during running this code in embedded system where memory resources are limited.

The Communication submodule passes received protocol frame to the Decoding submodule. It is then checked for validity of the frame. If no protocol marks are found, an error code is returned. If the frame is valid, the module and node name are recognized and marked in the structure's suitable variable. Next the protocol (and variable) data type are specified and also defined in the data structure, before adequate HaRTES layer is called. The structure passed as the argument.

Depending on the implementation used for retrieving the data from FTT-Enabled switch, the module in server layer uses that method to access the switch data. If no error is returned, the information held in the data structure are used to call the

Coding module. This module uses passed data to build protocol frame, respecting protocol data type previously defined.

The structure of the protocol frame are shown in Table 6. For the most cases the frame is built as modification of Type-Length-Value [40] (TLV) data communication protocol.

When the frame is built it is lastly validated and its length is determined based on the protocol data type used. After this, the frame is ready to be sent.

After sending the frame and checking if the transmission commenced properly, the connection is closed. The management application receives the frame, checks for validity and writes it to the application core, ending the same the process of getting the data from FTT-Enabled switch.

4.3 The HaRTES Management Protocol

To ensure robust data exchange between the management applications and the Remote Management Service new HaRTES Management Protocol was developed. Basic structure of the HaRTES Management Protocol which is used to explicitly indicate the node of which the data value is needed is showed in Table 6.

Table 6 – Structure of the basic HaRTES Management Protocol

Size		8 bit		8 bit
Field name	Module name	Separation Mark	Node name	End Mark

To clearly indicate the module and node name the data frame containing these fields are sent to Remote Management Service in the Master Module of the FTT-Enabled switch. To prevent any ambiguity during frame parsing, separation mark also has the length of 1 byte. The 8-bit code used to define marks are the ASCII codes of 058 and 038 in octal number system. This corresponds to separation and ending mark being ‘:’ and ‘$’ respectively. Symbols are taken from extended list of ASCII characters according to the “code page 437” standard introduced in 1981 by IBM corporation [41]. These 8-bit codes were chose due to low probability of node and module name containing these signs.

The second part (value or message) of the HaRTES Management Protocol which is used to transport the HaRTES data back to the management applications is based on Type-Length-Value. TLV is simple method used as data communication. However, HaRTES protocol most commonly uses for its internal communication a modified version of this protocol, skipping the Length value, therefore the Type field explicitly informs of the length of the variable. This reduces, even more, the size of protocol overhead, making it faster and less demanding.

Table 7 – HaRTES Management Protocol return frame with modified version of TLV protocol

Size		8 bits		8 bits	16 bits	
Field name	Module name	Separation Mark	Node name	End Mark	Data Type	Data Value

The Data field consists of two bytes which is interpreted as unsigned short based on data type name of C language in most common implementations. This gives sufficient data types to choose from. Table 8 points out the currently supported data types which are compatible with the newest SMIv2 data types. HaRTES Management Protocol also defines types for referencing status information and error messages.

Table 8 – Protocol data types supported by the HaRTES Management Protocol

Variable transmitted	Length of transmitted variable
Integer	32 bits
Octet String	An array containing the characters and terminated with a null character
Bits	An array containing the characters and terminated with a null character
OID	Specified in the length field
IP address	32bits (4 octets)
Counter 32	32 bits
Gauge 32	32 bits
Unsigned 32	32 bits
Time ticks	32 bits
Counter 64	64 bits
Float	32 bits
Information	An array containing the characters and terminated with a null character

Error	An array containing the characters and terminated with a null character

The scheme of communicating between modules of management applications was chose to be as presented, because in this way easy parsing as well as small overhead over the data transferred is introduced.

Support for OID data type in HaRTES Management Protocol was developed by expanding TLV protocol to its full length. The protocol in this case passes additional information about the length of the OID being sent in the frame. The full frame in this case presents Table 9 below.

Table 9 – HaRTES Management Protocol return frame with full version of TLV protocol

Size		8 bit		8 bit	2 bit	2 bit	
Field name	Module name	Separation Mark	Node name	End Mark	Data Type	Data Length	Data Value

The protocol is built in the Coding Modules, which is present in all application and modules responsible for managing the FTT-Enabled switch namely: SNMP subagent module, NetConf Server Instrumentation Libraries and Remote Management Service in the Master Module.

In the initialization phase of management application the request for status info is sent to the Master Module of the FTT-Enabled switch. The Remote Management Service then checks if all the supported layers are operational. If that checks the confirmation message is issued back to the management application. After that the initialization phase is over.

When the request for values of the switch are obtained, the SNMP subagent internally transcodes that PDU into frame of HaRTES Management Protocol with the related module and node name. Upon receiving the protocol frame, suitable module in picked out level is chosen. The module responds providing value in question for node name. When no error is detected the protocol frame is generated with gathered HaRTES data and send back to the SNMP subagent. After the verification phase, data is copied to the memory of the SNMP Master Agent, and transmitted to the Network Management System (NMS).

In the case of having to shutdown management application, or the Remote Management Service, there is a possibility of sending graceful shutdown tag through the HaRTES protocol. This lets to know the other component that is working in parallel on the other side of socket interface that this instance is going to be shut down.

The trap tag source can be located in one of the switch layers. If the event occurs, the function is called that triggers the sending of HaRTES protocol with trap tag which is then received by SNMP subagent trap handler. The information regarding the event are transcoded and passed to the SNMP Master Agent which sends SNMP strap to the SNMP sink. Figure 23 depicts this process.

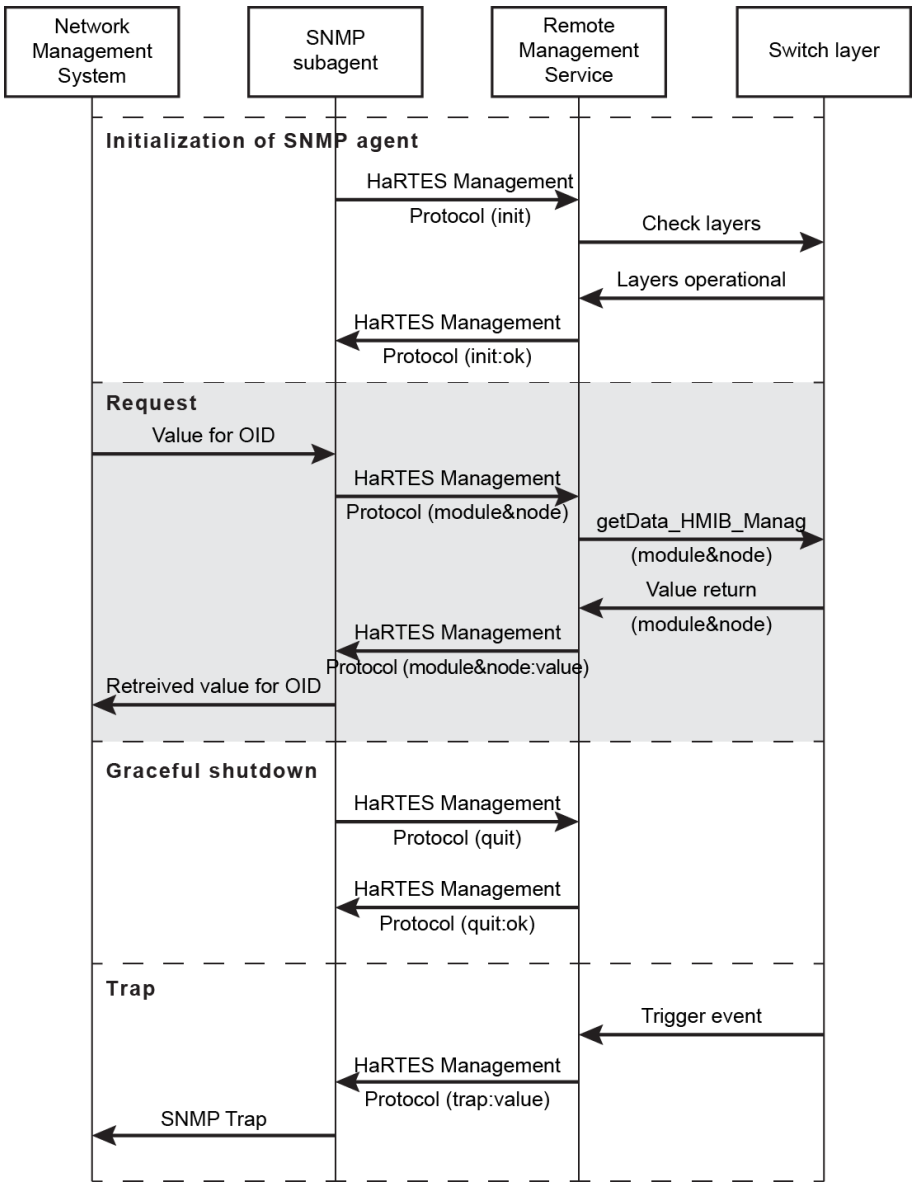

Figure 22 – Sequence diagram of SNMP agent and the Master Module of the switch

The sequence diagram differs slightly in case of NetConf protocol. Although order and form of the HaRTES Management Protocol frames stays very similar to the one used in SNMP protocol case, there are differences concerning connecting to the NetConf server which are shown in Figure 23. Initialization and graceful shutdown phases may be concerned the same as of SNMP protocol in regards to message sequence.

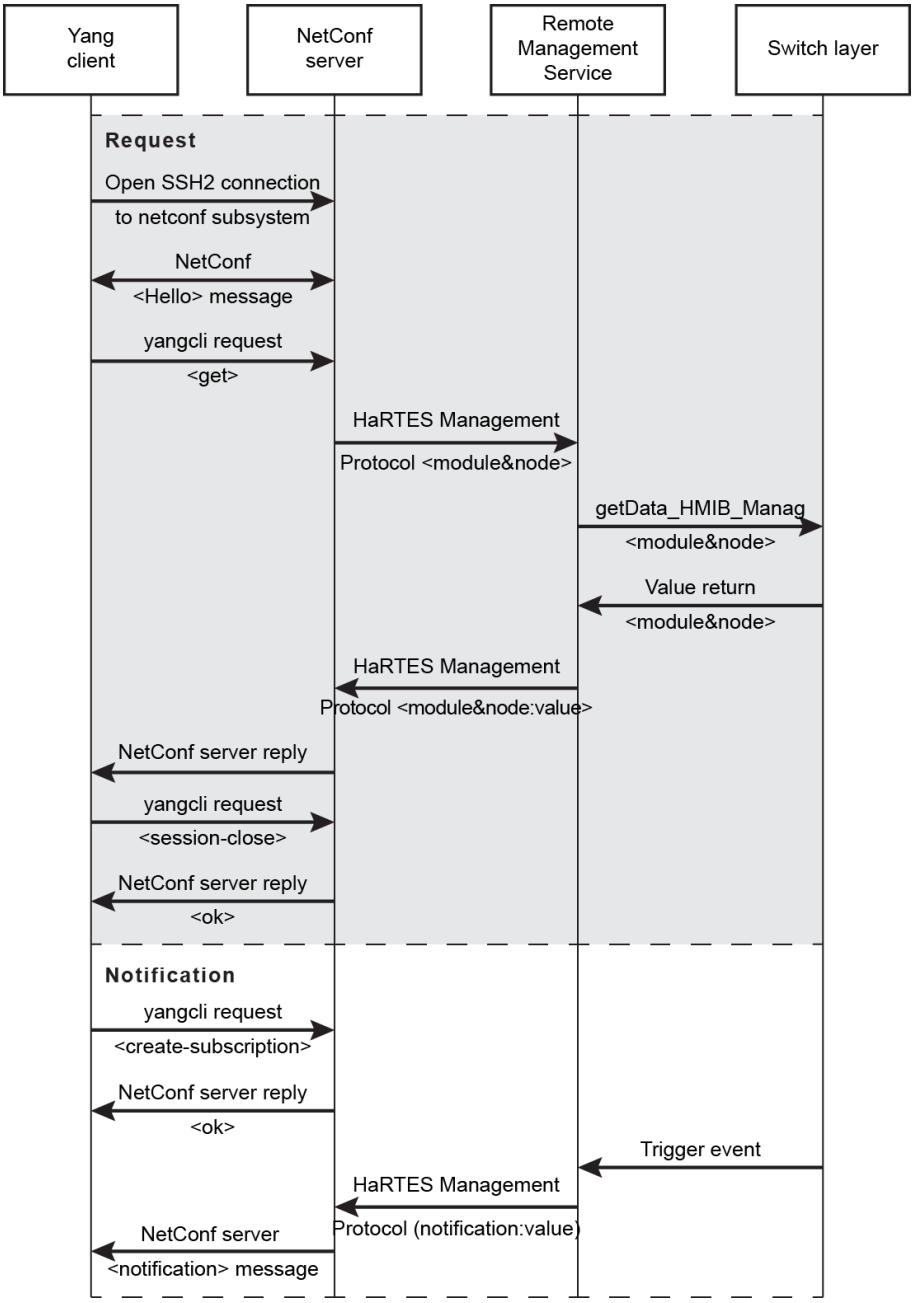

Figure 23 – Sequence diagram of NetConf protocol and Master Module of the switch

Important distinction has to be made on the topic of notifications. Contrary to the previous case this time the yang client must first be subscribed to the notification

one wish to receive. This means previous connection and session establishment. Also after disconnection from server no further notifications will be acquired by the NetConf client.

4.4 Proposed MIB and YANG module

After analysis of the IF-MIB file and the FTT-Enabled switch parameters described in Chapter 3 it became clear that the new custom MIB file must be created. The ATNOG-HARTES-MIB file adds values that can be retrieved from HaRTES and that are not readily available in the IF-MIB or other MIBs files. Thanks to that, many internal values are gathered in one MIB file for ease of use and management. MIB data are showed in Table 10.

Table 10 – MIB data names and corresponding FTT-Server data

MIB data	Corresponding value in FTT-Server and its explanation
eCycleDuration	*Elementary Cycle Duration* Length of communication slots, determines the system periodic granularity. This time can be tailored to suit the needs of a specific application. EC contains two consecutive phases dedicated to synchronous and asynchronous traffic, carrying time-triggered and event-triggered packets.
sWindowDuration	*Synchronous Window Duration* Length of the window for slave's synchronous messages. TM pools that information every EC.
asWindowDuration	*Asynchronous Window Duration* Length of the window for slave's asynchronous messages. TM pools that information every EC.
turnaroundTime	*Turnaround Time* It is the time between the end transmission of a TM and the beginning of a asynchronous window. During this time the impact of a propagation delay is reduced and slaves prepare gathered data to be sent.
switchingDelay	*Switching Delay* In the FTT Enabled Switch, it is the delay that takes the packet to be switched from input queue from source node to the queue of destination node.
transmissionTime	*Message Transmission Time* It is the time from the beginning until the last bit of a message, that has left the source node.
idleTime	*Idle time* If a synchronous message cannot be transmitted within current periodic traffic window, it is rescheduled for the next window, and the gap filled with the idle time.

qosType	*Scheduling (QoS) type* Discriminating policy, determining the most important services (highest qos), and assigning free bandwidth to them.
maxPeriodicMsg	*Maximum number of periodic message* The maximum number of periodic message transmitted in periodic window of EC.
maxAPeriodicMsg	*Maximum number of aperiodic message* The maximum number of aperiodic message transmitted in aperiodic window of EC.
mtu	*Maximum message size (fragmentation)* It is the maximum transmission unit in bytes. It influences the memory fragmentation and the real message size.
maxPckSize	*Maximum packet size* The maximum size of packet sent in one EC. This reduces the probability of blocking the higher priority traffic by the lower one, but with greater size. Long messages can be broken in packets and send sequentially.
bRate	*Link speed (baud rate)* The number of periodic messages per second.

The UA-HARTES-MIB file is described in the newest version of ASN, ASNv2 language description. It is defined as follows:

```
+--hartes(123321)
   |
   +--hartesObjects(1)
   |  |
   |  +-- -R-- Integer32 eCycleDuration(1)
   |  +-- -R-- Integer32 sWindowDuration(2)
   |  +-- -R-- Integer32 asWindowDuration(3)
   |  |
   |  +--gwTable(4)
   |  |  |
   |  |  +--gwEntry(1)
   |  |     |  Index: turnaroundTime, switchingDelay, transmissionTime
   |  |     |
   |  |     +-- -R-- Integer32 turnaroundTime(1)
   |  |     |        Range: 1..2147483647
   |  |     +-- -R-- Integer32 switchingDelay(2)
   |  |     |        Range: 1..2147483647
   |  |     +-- -R-- Integer32 transmissionTime(3)
   |  |              Range: 1..2147483647
   |  |
   |  +-- -R-- Integer32 idleTime(5)
   |  +-- -R-- Integer32 qosType(6)
   |  +-- -R-- Integer32 maxPeriodicMsg(7)
   |  +-- -R-- Integer32 maxAPeriodicMsg(8)
   |  +-- -R-- Integer32 mtu(9)
   |  +-- -R-- Integer32 maxPckSize(10)
   |  +-- -R-- Integer32 bRate(11)
   |
   +--hartesNotifs(2)
      |
      +--hartesTrap(1)
```

NetConf is newly introduced standard with different architecture to describe device capabilities, for this purpose it uses the YANG data modules. To convert MIB file described in SMIv2 directly to YANG language, Yuma Tools uses the "`smidump`" tool to ease these process.

Once YANG module is created "`yangdump`" tool is used to parse created file and generate C program files. Files are directly modified to support the retrieval of FTT-Server values. Then modules are compiled into Server Instrumentation Libraries and loaded into NetConf server during startup. The main part of the generated YANG module is showed below.

```
module UA-HARTES-MIB {

  namespace "urn:ietf:params:xml:ns:yang:smiv2:ATNOG-HARTES-MIB";
  prefix "atnog-hartes";

  import ietf-yang-smiv2 {
    prefix "smiv2";
  }

  organization
   "UA";

  contact
   "Aveiro";

  description
   "MIB for HARTES, developed in UA.";

  revision 2012-07-20 {
    description
     "[Revision added by libsmi due to a LAST-UPDATED clause.]";
  }

  revision 2012-05-25 {
    description
     "Initial version.";
  }


  container UA-HARTES-MIB {
    config false;

    container hartesObjects {
      smiv2:oid "1.3.6.1.4.1.123321.1";

      leaf eCycleDuration {
        type int32;
        description
         "Elementary Cycle Duration";
        smiv2:max-access "read-only";
        smiv2:oid "1.3.6.1.4.1.123321.1.1";
      }

      leaf sWindowDuration {
```

```
        type int32;
        description
         "Synchronous Window Duration";
        smiv2:max-access "read-only";
        smiv2:oid "1.3.6.1.4.1.123321.1.2";
      }
      …

        leaf transmissionTime {
          type int32 {
            range "1..2147483647";
          }
          description
           "";
          smiv2:max-access "read-only";
          smiv2:oid "1.3.6.1.4.1.123321.1.4.1.3";
        }
      }
    }
  }
```

Due to properties of the FTT-SE layers described in chapter 3.2 the values of IF-MIB and UA-ATNOG-MIB are assigned corresponding layer to which the Remote Management Server during the retrieval of the HaRTES information is calling.

Chapter 5

Performance assessment

The management interface was evaluated to determine its suitability for the task of managing a distributed system with real time constrains. SNMP and NetConf protocols are not designed to be real time aware. However, the inherent latency and variations observed when using these management methods will have impact on the remaining systems of a distributed management infrastructure if used there. Performed tests gave a better comprehension of delays introduced by the designed protocol.

Work and test were performed on a machine which consisted of a Pentium(R) Dual-Core CPU T4400 @ 2.20GHz and 2GB of RAM, running Ubuntu 12.04 LTS 32 bits. Software modules included SNMP version 5.7.1, NetConf version 2.2-3 and FTT-Server version 2.5.3-1. Also the Master Module clients were launched as a separate processes and the FTT-Server Remote Management Service was running on a separate thread.

Evaluation focused in triggering a series of management commands as fast as possible, while a passive monitoring application was capturing all communications being made. In this way management traffic was monitored without causing much interference to the management process. Values of jitter and communication latency experienced by the client application were afterwards extracted from the log. All analysis was done offline after the experiments were completed. The aim was to minimize external delay factors. Therefore, no network was actually used and all communication was done to the localhost. If an Ethernet connection were to be considered, a fixed amount of latency would have been observed due to transmission

buffers, reception buffers, as well as transmission to the medium and other existing queues. Also, jitter would be added in non-easily deterministic way [42].

The main work was focused in the development of a protocol management interface, enabling the HaRTES switch to be managed both through SNMP (v2c and v3) and through NetConf. Therefore, the performance of both approaches were evaluated. In order to have a clear idea about the overhead introduced by each protocol, and the absolute minimum bounds we can expect to experience given SNMP the current hardware, the internal communication delay was also measured.

Each management operation, for each protocol, was repeated 150 times. Afterwards, the top 10 values (roughly 6.67%) with higher difference from the average of the entire set of experiments were discarded. The remaining 140 values were considered for analysis, and consider a 95% confidence interval.

As a reference for the analysis, we measured the internal latency from the time the request was received, until the switch management software was queried and a result was provided to the management protocol. The internal latency values obtained are depicted in Figure 24. Internal queries were always lower than 350µs, and presented an average of 205.45±31.31µs. In order to calculate the penalty of SNMP or NetConf, this average value should be deducted. While observing some jitter in internal communications, 95% of the results stayed within 15% of the average value. Because the delays from managements agents are in microseconds, and given the hardware available, we expect this jitter to have minimum impact in management.

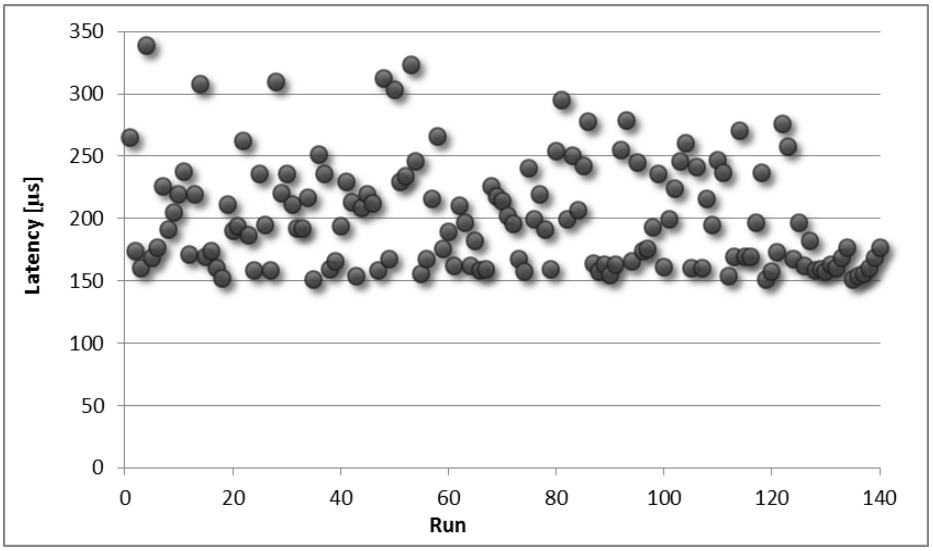

Figure 24 – Internal communication latency

SNMP was evaluated using its two most used varieties: SNMP v2c and SNMP v3. First tests considered request time for all HaRTES database values. As for SNMP this involved using tool called "`snmpwalk`" which issues sequentially operation "`get-next`" until management agent jumps to another network node, then the tree is considered to be complete.

Figure 25 presents scatter graph of all, SNMPv2c, SNMPv3 and NetConf, solutions. Earlier version of the SNMP presented lowest average query latency levelling at 19,43±4,11ms. The latest version presented lightly higher average query latency staying in range of 20.72±2.22ms. NetConf, presenting different management technologies as described in section 2.2, significantly stand out of SNMP two technologies average values, placing itself far above them at the level of 527±18ms. This value is over 25 times higher than corresponding SNMP values, but, again, substantial gains in management capabilities which this technology allows, makes it reasonably outcome anyway.

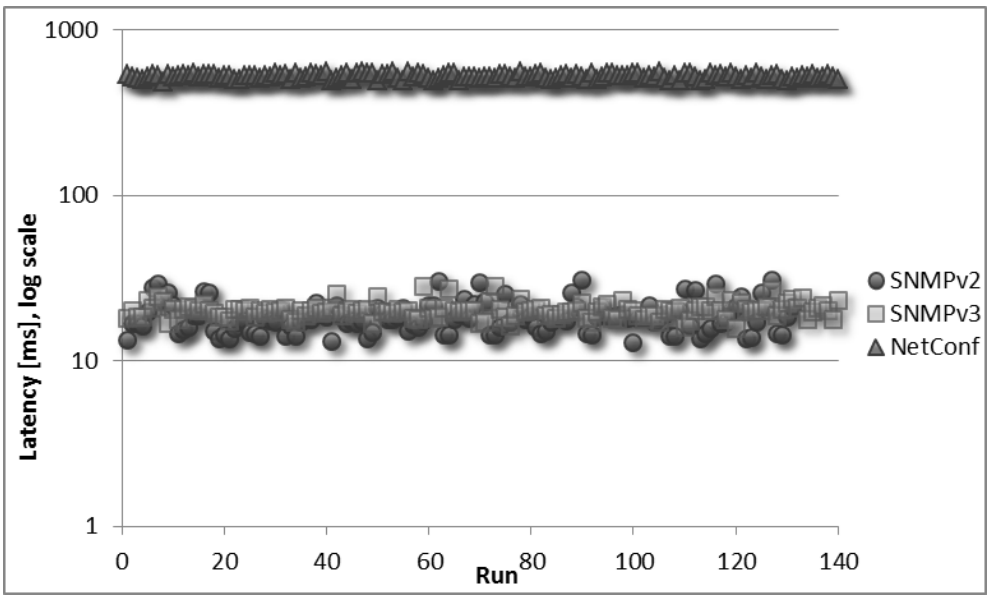

Figure 25 – Time scatter of all management platforms during the retreival of all HaRTES values

Next test focused at all management getting a single value. As described in section 2.1, SNMP considers UDP communications and small individual transactions following a request/response approach with two packets for each transaction. All information required in contained in the request, and the reply also takes a single UDP packet. SNMP v3 follows the same approach but introduces changes to messages in

order to add support for shared key authentication. Still, each management action is composed by a simple exchange of messages through UDP.

Figure 26 depicts the values obtained for both SNMPv2c and SNMPv3. Due to the slightly higher complexity of SNMPv3, the management latency was also a little higher than SNMPv2c. The first presented an average query latency of 1.41±0.29ms, while SNMPv2c presented an average query latency of 0.95±0.16ms. It is interesting to observe that SNMPv2c presents lower average latency, but also less error. In part, the smaller number of packets of each transaction when using SNMPv2c can explain this behaviour.

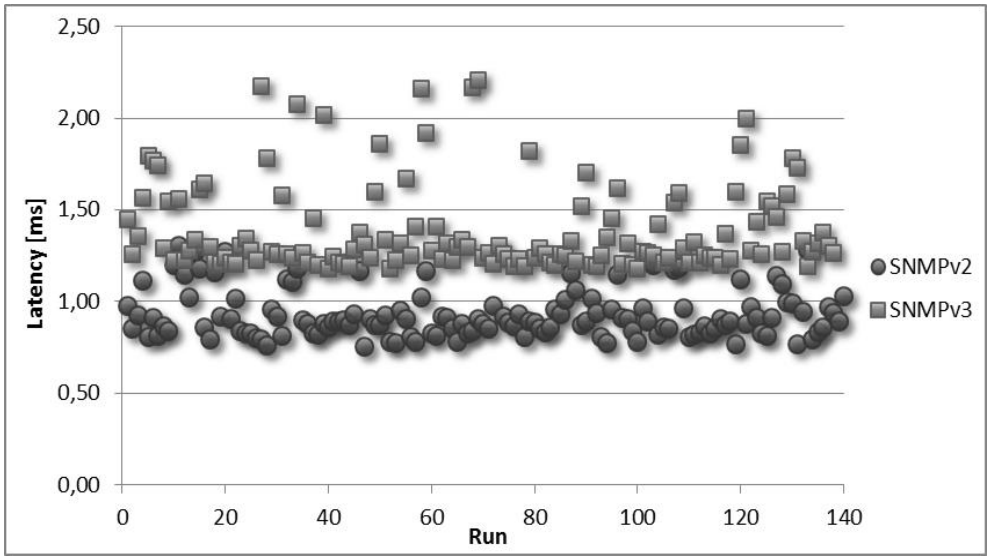

Figure 26 – Time scatter chart of SNMPv2 and SNMPv3

The same methodology was followed in order to evaluate NetConf technology. Many transport methods are supported by this standard. As SSH is mandatory, and the one most commonly available, that is why it was decided to focus in evaluating NetConf over SSH. Because SSH is a secure protocol supporting both peer authentication and private communications, the overhead produced is expected to be much higher. Figure 27 depicts the latency observed for the best 140 queries for one and multi values queries. Query time of NetConf technology for one HaRTES value was almost the same (the average value differs only 0,6%), as for query time for all (muti) HaRTES values. As shown, NetConf introduces much higher latency into management functions, one and multi query times averaging at 525±18ms and 527±18ms, respectively. The added overhead is so high that the values obtained for internal latency are three orders of magnitude lower, and therefore negligible. The error margin itself is

two orders of magnitude higher, which reflects the high overhead of this management protocol.

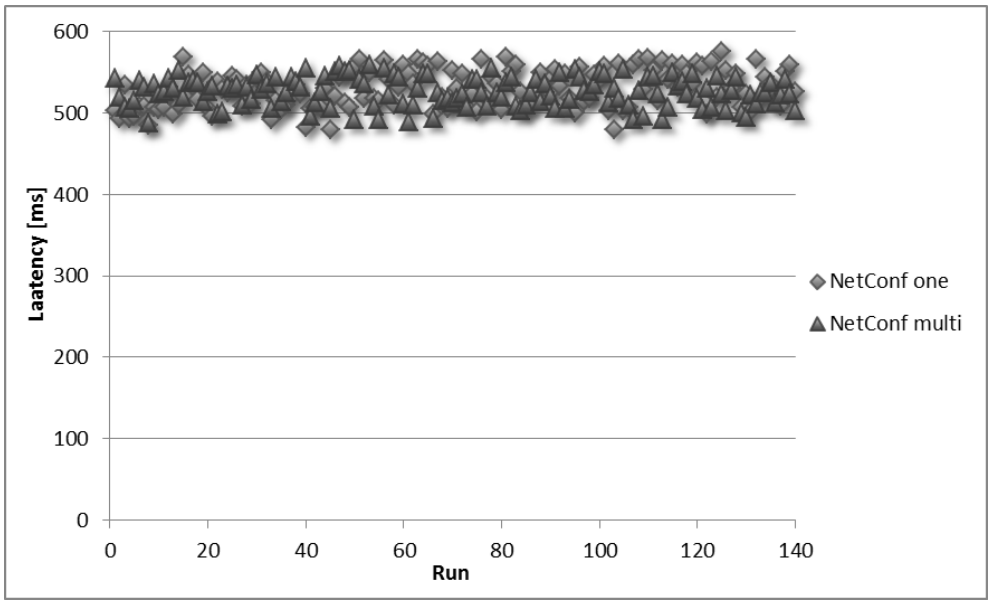

Figure 27 – Comparison of NetConf protocol time resolving one and multiple queries

From the evaluation of these three management approaches, it is clear that the use of any standard compliant management protocol will introduce very high latency and variability in processes. Figure 28 compares gathered data with the pure internal latency. This picture also shows that the lightest solution is SNMPv2c, followed by SNMPv3, and then NetConf. However, the term lightest can be very misleading as it increases latency by a factor of 5. NetConf falls in a completely different bucket, increasing management latency by a factor of almost 4000. But the advantages of strong authentication of peers communication and secure management process which NetConf well supports, should be taken into consideration also.

However, as mentioned in chapter 2, the best practices state that management traffic should be transported in networks completely isolated from clients (e.g. dedicated VLANs), these advantages become less clear.

Nevertheless, considering Real-Time systems, with dependencies in remote managed systems, lower latency will result in higher scalability for the system, as well as higher levels of determinism. In scenarios where critical timings must be observed, such as automotive or industrial scenarios, SNMP still proves itself to be the best management approach if standards are to be respected. Custom developed solutions

show to provide higher performance, at the cost of interoperability and eventually future evolution.

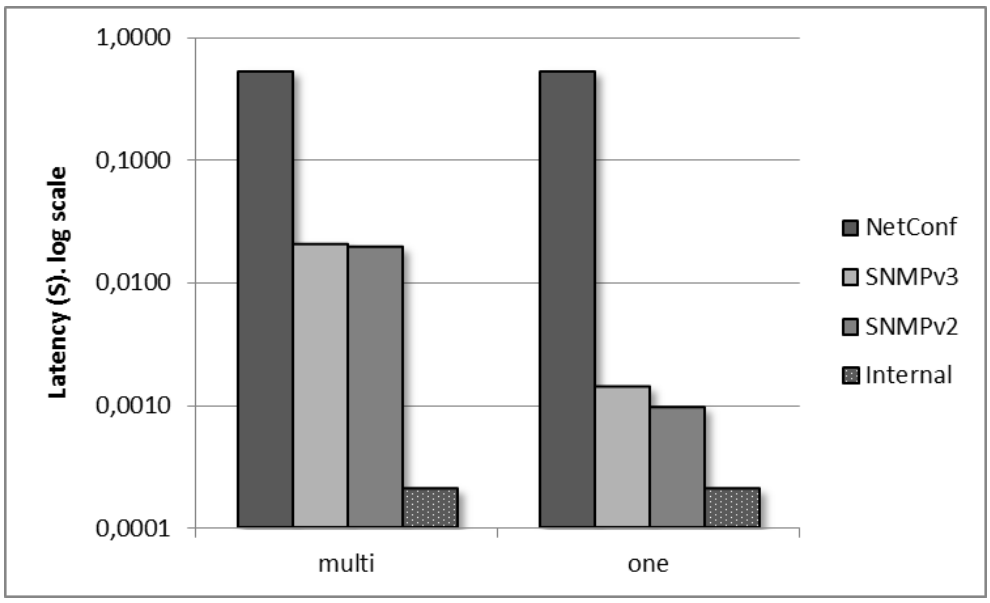

Figure 28 – Latency of the different management methods handling one and multi requests with respect to the internal latency

Chapter 6

Conclusions and future research

As the technical world become more expansive and sophisticated, new more reliable and effective solutions are required to be integrated. Ethernet standard became ubiquitous feature in many devices. To satisfy real-time requirements in switched Ethernet networks, several solutions were introduced, but most of them lacked operational flexibility. Managing network that guarantee the quality of service that is used in Embedded Systems and which allow for coexistence of Ethernet traffic requires a special approach. The Hard Real-Time Ethernet Switch (HaRTES) meet the task well, allows using the same network to handle multiple traffic flows, without compromising the performance of real-time applications.

Network managers who had to efficiently manage large chunks of networks, had to have a possibility to resourcefully handle these demands. Over past decades several solutions were introduced in service to resolve these problems. Long-adopted SNMP standard which turned out to be insufficient in facing the devices control needs is slowly being catch up by or even replaced with new NetConf technology, which although struggles with larger message overhead, as we have seen on chapter 5, is more flexible and technology advanced solution.

The Hard Real-Time Ethernet Switch lacked a standard management interface to configure its parameters and view its status. The created interface was required to help monitor the switch performance and foster its further development. Now it is easier and convenient to handle multiple traffic flows. Smaller delays introduced by SNMP system make this solution nearer to be able to monitor the switch in real-time.

6.1 Future research

Presented in this thesis management solution extends the flexibility and capabilities of HaRTES switch. Future lines of research may include:

Full support for Management of Stream Reservation Protocol.

Technologies responsible for handling these types of protocols are now being developed to expand FTT-Enabled switch abilities. With convenient API provided, the remote management solutions can also be expanded, through adding new MIB and YANG module, to fully support this technology.

Full management support.

As for now HaRTES Remote Management Interface allows for monitoring and notification/trap issuing. Because of shaky FTT-SE support for managing states of its internal configuration this solution has been postponed until save and tested internal value manipulation is provided by FTT-SE API.

Performance improvements.

Remote Management Interface of HaRTES switch efficiently handles queries issued by the management applications. Steps may be taken to handle multiple queries at the same time by the Remote Management Service. Cache mechanisms and HaRTES values configuration database can be developed. NetConf full capabilities of several datastores being managed on one device at the same time, could be supported.

Bibliography

[1] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, 2001.

[2] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, 1997.

[3] B. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, 1994.

[4] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, B. Braden, B. Davie, E. Felstaine, and J. Wroclawski, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, 2000.

[5] R. Santos, "Enhanced Ethernet Switching Technology for Adaptable Hard Real-Time Applications", PhD Thesis, Department of Electronics Telecommunications and Informatics University of Aveiro, 2011.

[6] R. Marau, P. Pedreiras, and L. Almeida. "Enhancing Real-Time Communication over COTS Ethernet Switches", In Proceedings of the 6th IEEE International Workshop on Factory Communication Systems (WFCS'06), pages 295–302, June 2006.

[7] W. W. Diab. (2012). Available: http://www.ieee802.org/3/

[8] IEEE Standard for Local and Metropolitan Area Networks--Virtual Bridged Local Area Networks Amendment 14: "Stream Reservation Protocol (SRP)", IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005), vol., no., pp.1-119, Sept. 30

[9] J. Case, M. Fedor, M. Schoffstall, and J. Davin, Simple Network Management Protocol (SNMP), RFC 1157, 1990.

[10] R. Enns, M. Bjorklund, T.-f. Systems, J. Schoenwaelde, and A. Bierman, "Network Configuration Protocol (NetConf)", RFC6241, 2011

[11] Ethernet Powerlink - online information. http://www.ethernet-powerlink.org/, May 2012.

[12] PROFInetIRT. Real-Time PROFInet IRT. http://www.profibus.com/pn, May 2012.

[13] TTESwitch A664 Pro 24. TTTech Computertechnik AG. http://www.tttech.com/en/products/ttethernet/flight-and-rugged-hardware/switch-a664-pro-24/- Accessed June 29, 2012

[14] Industrial Ethernet. Weidmüller. http://www.weidmuller.ru/news/pi_ie_en.pdf - Accessed June 27, 2012.

[15] PROFIBUS International: PROFINET Specification, Profinet IO Application Layer Service Definition

[16] EPSG Draft Standard 301, Ethernet POWERLINK Communication Profile Specification Version 1.1.0, EPSG 2008.
[17] A. Clemm, *Network Management Fundamentals*. United States of America: Cisco Press, 2006.
[18] H. Zimmermann. "OSI reference model: The ISO model of architecture for open systems interconnection", IEEE Transactions on Communications, 28(4):425–432, 1980.
[19] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, and S. Waldbusser, "Structure of Management Information Version 2", RFC2578, 1999
[20] Telecommunication Standarization Sector of ITU, "OSI networking and system aspects – Abstract Syntax Notation One (ASN.1)" in Series X: Data Networks and Open Systems Communications, Published as ISO/IEC 8824-1, 2002
[21] T. Bray, J. Paol, C. M. Sperberg-McQueen, S. M. Eve Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0*. (2008). Available: http://www.w3.org/TR/REC-xml/
[22] R. R. D. Marau, "Real-time communications over switched Ethernet supporting dynamic QoS management," Ph.D., Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, Portugal, 2009.
[23] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC4253, 2006
[24] Information Sciences Institute University of Southern California, "Internet Protocol", RFC791, 1981
[25] J. Postel, "User Datagram Protocol", RFC 768, 1980
[26] Information Sciences Institute University of Southern California, "Transmission Control Protocol", RFC793, 1981
[27] M. Rose and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC1155, 1990
[28] S. Legg, "Abstract Syntax Notation X (ASN.X) Representation of Encoding Instructions for the Generic String Encoding Rules (GSER)", RFC4913, 2007
[29] *The Internet Assigned Numbers Authority*. (2012). Available: http://www.iana.org/
[30] M. Daniele, B. Wijnen, M. Ellison, and D. Francisco, "Agent Extensibility (AgentX) Protocol", RFC 2741, 2000
[31] R. Enns, "NetConf Configuration Protocol", RFC4741, 2006
[32] Sperberg-McQueen, C., Bray, T., Paoli, J., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium REC-xml-20001006, October 2000.
[33] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol", 2010
[34] tail-f. *What is YANG*. Available: http://www.tail-f.com/what-is-yang
[35] YumaWorks, "Yuma Developer Manual," 2.2 ed, 2012.
[36] F. Yergeau, "UTF-8, a transformation format of ISO 10646", RFC3629, 2003
[37] R. G. V. d. Santos, "Enhanced Ethernet Switching Tecnology for Adaptibe Hard Real-Time Applications," Master, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, Portugal, 2010.
[38] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida, "Designing a costumized Ethernet switch for safe hard real-time communication", Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on, 2008, pp. 169-177.

[39] H. Kopetz. "Real-Time Systems: Design Principles for Distributed Embedded Applications". Kluwer Academic Publishers, 1997.
[40] T. Przygienda, "Reserved Type, Length and Value (TLV) Codepoints in Intermediate System to Intermediate System", RFC3359, 2002
[41] IBM Corporate Specification, "Graphic Character Sets and Code Pages", Avaliable (2012): ftp://ftp.software.ibm.com/software/globalization/gcoc/attachments/CP00437.pdf
[42] G. Cena, I. C. Bertolotti, and A. Valenzano, "Experimental analysis of latencies in ethernet communications", Factory Communication Systems, 2006 IEEE International Workshop on, 2006, pp. 303-312.