BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

GIGABIT PASSIVE OPTICAL NETWORK ANALYSIS
ANALÝZA GIGABITOVÉ PASIVNÍ OPTICKÉ SÍTĚ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. Vladislav Mižička
SUPERVISOR VEDOUCÍ PRÁCE	Ing. Tomáš Horváth, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Vladislav Mižička	***ID:*** 195990
Ročník: 2	***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Analýza gigabitové pasivní optické sítě

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je konfigurace vysokorychlostního paketového analyzátoru Endace, konfigurace zařízení gigabitové pasivní optické sítě (GPON) a následná analýza zachycených dat. Práce by měla být soustředěna zejména na možnost detekce provozních dat jednotky GPON. V teoretické části stručně popište paketový analyzátor a jeho možnosti, dále také jednotku GPON a možnosti zapojení s analyzátorem k detekci provozu. Dále navrhněte skript či aplikaci v jazyce Python pro příjem a analýzu dat z jednotky Endace. V praktické části proveďte konfiguraci zařízení, zachycení dat a jejich analýzu pomocí jazyka Python. Aplikace či skript by měl podporovat příjem dat z jednotky Endace a možnost zpracovat informace obsažené v těchto datech. Výstup analýzy by měl být v rámci možností co nejpodrobnější s popisem významu jednotlivých položek. Pro export dat použijte analyzátor Endace na cílový server.

DOPORUČENÁ LITERATURA:

[1] VUT Brno. Zdroje pro vnitřní potřebu (Endace Guide)

[2] Sanders, Chris. Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems. 3rd ed. San Francisco: No Starch Press, 2017.

Termín zadání: 5.2.2018	***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Tomáš Horváth, Ph.D.
Konzultant: Ing. Václav Oujezský, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstract

Aim of this thesis is configuration of a high-speed packet analyzer Endace, configuration of a Gigabit Passive Optical Network (GPON) device and following analysis of captured data. Thesis should be focused particularly on possible detection of an operational data of the GPON unit. In theoretical part, the packet analyzer and its possibilities are described. Later also GPON unit and their connectivity in order to detect an operational data is provided. In practical part, all devices are configured and captured data analyzed. Application developed using a python language supports data receiving and processing. Output of an analysis should be as detailed as possible with a description of the meaning of individual items.

Keywords: GPON, Endace, operational data, packet analysis, security

Abstract [CZ]

Cílem této práce je konfigurace vysokorychlostního paketového analyzátoru Endace, konfigurace zařízení gigabitové pasivní optické sítě (GPON) a následná analýza zachycených dat. Práce by měla být soustředěna zejména na možnost detekce provozních dat jednotky GPON. V teoretické části je stručně popsaný paketový analyzátor a jeho možnosti, dále také jednotka GPON a možnosti zapojení s analyzátorem k detekci provozu. V praktické části, zařízení jsou nakonfigurovaná a zachycená data analyzovaná. Aplikace vyvinutá s pomocí jazyka Python podporuje příjem a zpracování dat. Výstup analýzy by měl být v rámci možností co nejpodrobnější s popisem významu jednotlivých položek.

Klíčová slova: GPON, Endace, provozní data, paketová analýza, bezpečnost

Declaration

sensor, information and communication systems
research centre

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

The research described in this diploma thesis has been done in laboratories supported by Sensor, Information and Communication Systems Research Centre (SIX) project; registration number CZ.1.05/2.1.00/03.0072 Operational Program Research and Development for Innovation (operační program Výzkum a vývoj pro inovace).

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj
pro inovace

Acknowledgment

First, I would like to thank my advisor Ing. Tomáš Horváth, Ph.D., who professionally guided me through every aspect of this research.

Next, I would like to thank my technical advisor Ing. Václav Oujezský, Ph.D. for his passionate willingness to help with any technical challenge I ran into.

Also, I would like to thank the SIX research center for providing the top of industrial networking devices, those were necessary for this project.

Last, but not least I would like to thank Brno University of Technology and its Faculty of Electrical Engineering and Communication for providing the environment in which it was possible and pleasure to work on this kind of research.

Contents

1 Introduction **1**

1.1 Interesting Facts . . . 2

2 Analysis **3**

2.1 Gigabit Passive Optical Network . . . 3

2.1.1 GPON architecture . . . 4

2.2 Endace Probe . . . 7

2.2.1 Architecture . . . 8

2.2.2 Performance . . . 9

2.2.3 Abilities . . . 9

2.3 Operational Data Detection . . . 10

2.3.1 Requirements Specification . . . 10

2.4 Python dpkt . . . 11

2.5 Analysis Summary . . . 12

3 Solution Proposal **13**

3.1 Topology . . . 13

3.2 Devices . . . 14

3.3 Scenario . . . 15

3.4 Data Analysis . . . 16

4 Implementation **18**

4.1 Devices connection . . . 18

4.2 Configuration . . . 19

4.2.1 GPON OLT . . . 19

4.2.2 Switches . . . 20

4.2.3 Endace . . . 21

4.3 Application . . . 22

4.3.1 ODDA Overview . . . 23

4.3.2 Data receive . . . 24

4.3.3 Rules . . . 25

4.3.4 Data processing . . . 26

4.3.5 Application Programming Interface . . . 27

5 Evaluation **28**

5.1 Data capture . . . 28

5.2 Application Testing . . . 29

5.2.1 CLI Testing . . . 29

5.2.2 API testing . . . 31

5.2.3 Testing summary 32
5.3 Results 32

6 Conclusion **34**

Bibliography **36**

Abbreviations **37**

Listings **39**

List of Figures **40**

List of Tables **41**

List of Appendices **42**

A ODDA functionality **43**
A.1 Prerequisites 43
A.2 Command-Line Interface 43
A.3 Application Programming Interface 45

B APNIC IP range **46**

C List of files on attached DVD **50**

1 Introduction

Since the Internet was created, the communication technologies have been expanding more than ever. In general, more devices are connected every day and more traffic is sent over the IP network to accommodate new features.

According to the CISCO forecast, this trend is going to continue in the future with 22% Compound Annual Growth Rate, or CAGR as might be seen in the figure 1.1 Global IP Traffic Growth [1].

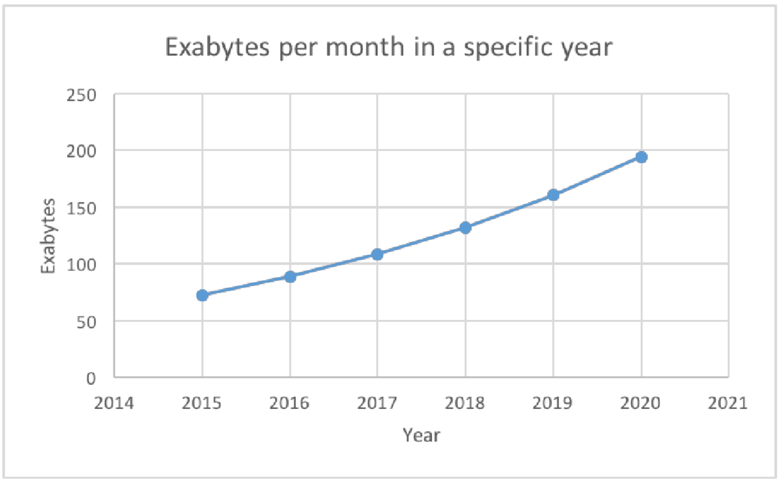


Figure 1.1: Global IP Traffic Growth

Naturally, such a big demand for bandwidth causes new technologies evolution. Nowadays, thanks to its efficiency, are fiber and passive mechanisms very popular. However, security is an important aspect, which cannot be overseen.

Gigabit Passive Optical Network, or GPON is a very beneficial technology used all around the world from Internet Service Providers, or ISPs for households, through various industries up to the Governments for their militaries.

This thesis focuses on analyzing the GPON, mainly from a security point of view, searching for an unwanted operational data to reveal any potential industrial espionage.

Chapter 1 Introduction makes a clear point about this thesis, informs about what can be find in which chapter and in the end provides some interesting facts.

Chapter 2 Analysis studies the GPON technology, high-speed packet analyzer Endace, discusses about possibilities of operational data detection and specifies requirements for this project.

Chapter 3 Solution Proposal reveals a topology, devices in it, connection between them and creates a plan for capturing and processing all the data.

Chapter 4 Implementation shows SIX research center, where entire topology is physically located and shows also configuration of main devices. Later, the chapter explains structure of an application used for data analysis.

Chapter 5 Evaluation proves, that data capturing is working, and entire communication passing through the GPON is stored in local storage of a high-speed packet analyzer. Later, it tests application functionality and provides output of a data analysis.

Chapter 6 Conclusion summarizes this thesis with its achieved results.

1.1 Interesting Facts

In compare to a traditional active Ethernet Local Area Networks, or LANs, GPON systems can reduce space requirements by 90 percent, energy consumption by 80 percent and total cost of ownership by 70 percent, according to an estimate by GPON systems provider Tellabs. [2]

According to Building Industry Consulting Service International, or BICSI, Follow on to 2010 US Army Directive directing all commands to adopt GPON technology by 2013. [3]

The Fort Huachuca project is being overseen by the U.S. Army Information Systems Engineering Command, and their systems integration contractor is NCI [1]. In a recent interview, Randy Devine, senior vice president of operations at the company's Sierra Vista, Ariz., office said: "In the project we're doing on Fort Huachuca, they are reducing from 29 telecommunications closets with active network switches down to three". [2]

[1]NCI is a leading provider of enterprise solutions and services to U.S. defense, intelligence, health and civilian government agencies.

2 Analysis

This chapter consists of a theoretical overview of Gigabit Passive Optical Network, its architecture and usage. Later, advanced packet analyzer Endace and its features are described. In the end, a module for reading a pcap file in python language is mentioned.

2.1 Gigabit Passive Optical Network

Gigabit Passive Optical Network is a very popular technology among Internet Service Providers, or ISPs delivering Fiber To The Home, or FTTH (entire access network made from optical network) for various reasons:

- Uses fiber technology
- Max. downstream rate is 2.488 Gbps
- Max. upstream rate is 2.488 Gbps
- Max. GPON split ratio is 1 : 128
- Maximal distance is 20 km

Maximum differential fibre distance is 20 km, however in ITU-T Recommendation G.984.1, maximum logical reach is defined as 60 km. [4]

Basically, the larger the split ratio is for GPON, the more attractive it is for operators. However, a larger split ratio implies greater optical splitting which creates the need for an increased power budget to support the physical reach. Split ratios of up to 1:64 are realistic for the physical layer, given current technology. However, anticipating the continued evolution of optical modules, the TC layer must consider split ratios up to 1:128. [4]

Most common passive optical networks are Broadband, Ethernet and Gigabit, which is the most deployed nowadays. Broadband PON was completely replaced by Gigabit. Ethernet PON was very popular in Asia, but it is almost displaced too. Differentiation between them might be seen in the table 2.1 PON comparison. Today, the most advanced PON is NG-PON2 or Next Generation Passive Optical Network (stage 2). Huawei has already successfully tested a 25G/100G prototype based on the MA5800, to be prepared for 5G speeds as well as UHD or Ultra High

Table 2.1: PON comparison

	BPON	EPON	GPON
Standard	ITU G.983	IEEE 802.3ah	ITU G.984
Max. Downstream	600 Mbps	1 Gbps	2.4 Gbps
Max. Upstream	150 Mbps	1 Gbps	2.4 Gbps
Protocol	ATM	Ethernet	GEM
Split ratio	1: 32	1: 32	1: 128
Max. Distance	20 km	20 km	20 km

Definition video.

Jeff Wang, President of Huawei's Access Network Product Line, said: "The success of the 25G/100G symmetric PON test indicates that Huawei is able to cope with big broadband services in the future. Huawei wants to work with industrial partners to promote the integration of next-generation PON standards in the industry, and ensure that the same set of standards and systems are used for optical access networks." [5]

2.1.1 GPON architecture

As might be seen from the figure 2.1 GPON architecture, GPON is a point to multipoint architecture, supporting all: data, voice and video services. Main components are OLT or Optical Line Termination, splitter and ONT or Optical Network Termination. ONT might be also called ONU, or Optical Network Unit. There is slight difference between ONT and ONU, but for purpose of this project, it is negligible.

ITU-T Recommendation G.984.1 defines ONT and ONU this way:

- **Optical Network Termination (ONT)**: A single subscriber device that terminates any one of the distributed (leaf) endpoints of an ODN, or Optical Distribution Network implements a PON protocol, and adapts PON PDUs, or Protocol Data Units to subscriber service interfaces. An ONT is a special case of an ONU. [4]

- **Optical Network Unit (ONU)**: A generic term denoting a device that terminates any one of the distributed (leaf) endpoints of an ODN, implements a PON protocol, and adapts PON PDUs to subscriber service interfaces. In some contexts, an ONU implies a multiple-subscriber device. [4]

ITU-T Recommendation G.984.3 refers both ONT and ONU as identical from the G-PON TC layer functionality point of view:
The network element interfacing the end-user access facilities and the optical distribution network (ODN) is herein referred to as an optical network unit (ONU). In the

B-PON context, clause 4 of [ITU-T G.983.1] makes a distinction between an ONT and an ONU. This Recommendation considers an ONT to be a special (single-user) case of an ONU. However, since from the G-PON TC layer functionality point of view, these two entities are identical, the term "ONU" herein applies to both of them, except for the specifically identified cases. [9]

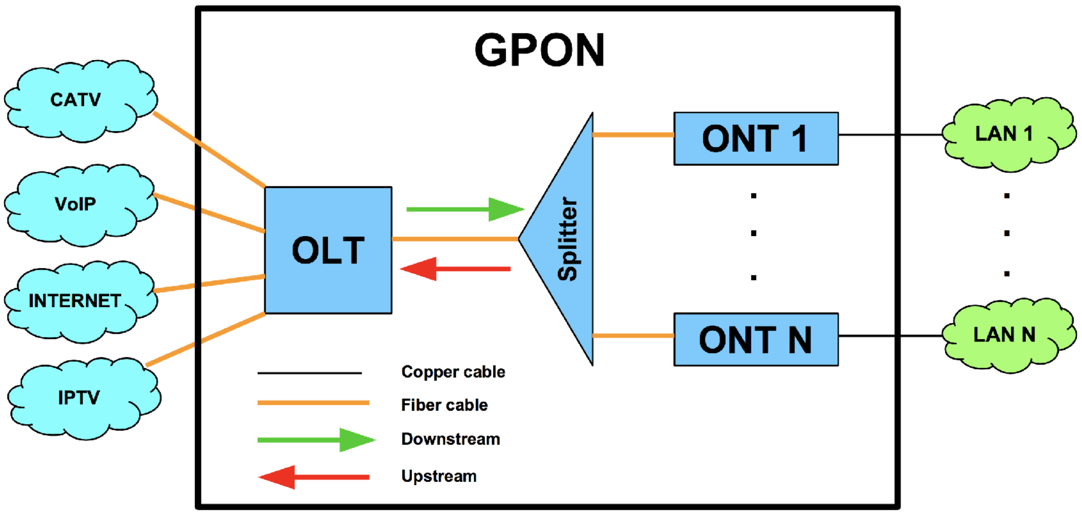


Figure 2.1: GPON architecture

This technology uses only single mode fiber and is based on WDM or Wavelength Division Multiplexing, which means every wavelength is used for something else. According to the recommendation ITU-T G.984.2 the operating wavelength range shall be as follows:

- 1260 - 1360 nm for the upstream direction on a single fibre system [6]
- 1480 - 1500 nm for the downstream direction on a single fibre system [6]

Recommendation ITU-T G.984.5 reorganizes wavelength allocation and defines space for video streaming at 1550 nm. Reserved space for upstream and downstream remains the same. [7]. Allocation might be seen in the figures 2.2 Wavelength allocation 1 and 2.3 Wavelength allocation 2.

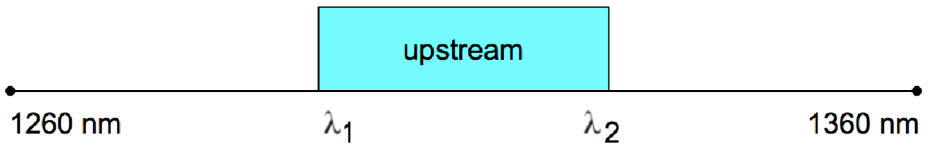


Figure 2.2: Wavelength allocation 1

λ_1 can differ from 1260 up to 1300 nm and λ_2 can differ from 1320 up to 1360 nm. It depends on used technology. Space between 1360 and 1480 nm is reserved as upstream/downstream guard and for enhanced band 1, which can be used for NGA, or Next-Generation Access. [7]

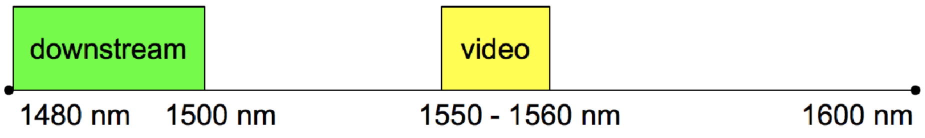


Figure 2.3: Wavelength allocation 2

Between OLT and ONU, connection oriented method with ability to fragment and transfer more than IP packets is required. Therefore, GEM or GPON Encapsulation Method was invented. [8]

G-PON encapsulation method (GEM): A data frame transport scheme used in G-PON systems that is connection-oriented and that supports fragmentation of the user data frames into variable-sized transmission fragments. [9]

G-PON Transmission Convergence (GTC) layer: A protocol layer of the G-PON protocol suite that is positioned between the physical media dependent, or PMD layer and the G-PON clients. GTC layer is composed of GTC framing sublayer and GTC adaptation sublayer. [9]

In the downstream direction, the traffic multiplexing functionality is centralized. The OLT multiplexes the GEM frames onto the transmission medium using GEM Port-ID as a key to identify the GEM frames that belong to different downstream logical connections. Each ONU filters the downstream GEM frames based on their GEM Port-IDs and processes only the GEM frames that belong to that ONU. [9]. Downstream multiplexing with shaded GEM ports represented multicast can be seen in the figure 2.4 Downstream multiplexing.

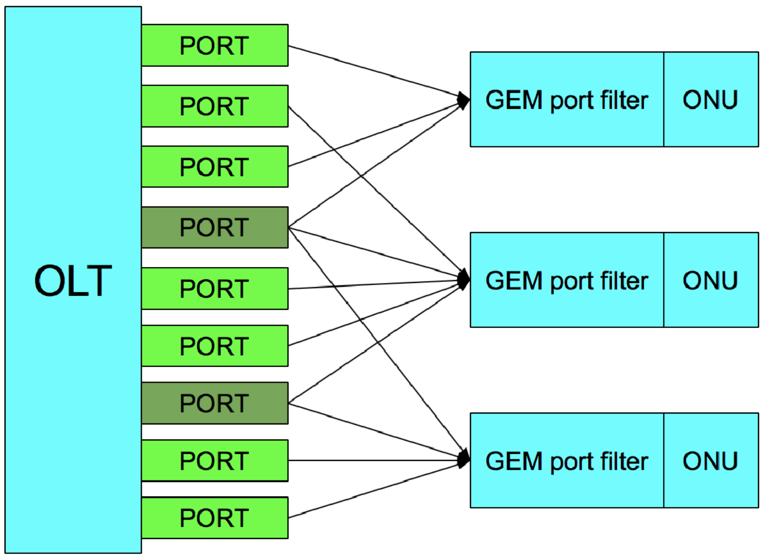


Figure 2.4: Downstream multiplexing

In the upstream direction, the traffic multiplexing functionality is distributed. The OLT grants upstream transmission opportunities, or upstream bandwidth allocations, to the traffic-bearing entities within the subtending ONUs. The ONU's traffic-bearing entities that are recipients of the upstream bandwidth allocations are identified by their allocation IDs. The bandwidth allocations to different Alloc-IDs are multiplexed in time as specified by the OLT in the bandwidth maps transmitted downstream. This means, it uses TDM, or Time Division Multiplexing. Within each bandwidth allocation, the ONU uses the GEM Port-ID as a multiplexing key to identify the GEM frames that belong to different upstream logical connections [9], as can be seen in the figure 2.5 Upstream multiplexing.

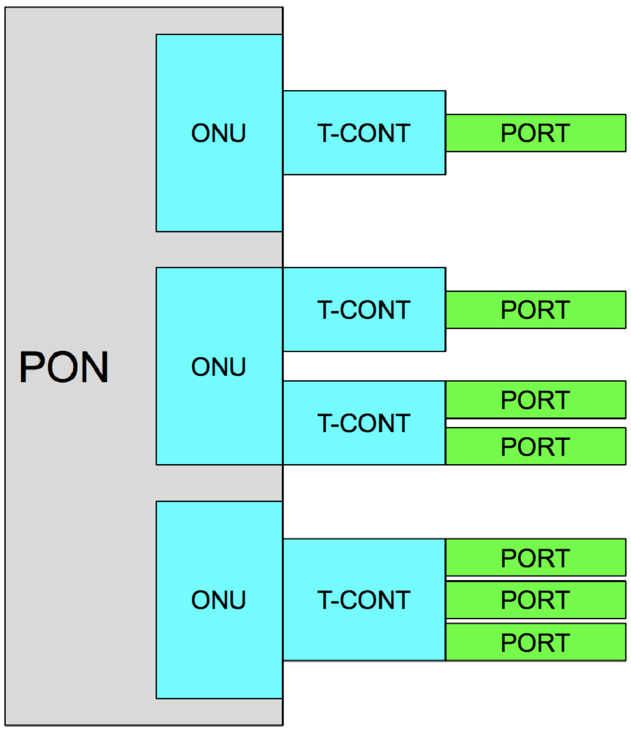


Figure 2.5: Upstream multiplexing

Transmission Container (T-CONT): A traffic-bearing object within an ONU that represents a group of logical connections, is managed via the ONU Management and Control Channel, or OMCC, and is treated as a single entity for the purpose of upstream bandwidth assignment on the PON. [9]

2.2 Endace Probe

Since 2001, Endace has provided high-speed, network visibility solutions to operators of the world's largest, most complex networks including global banks, telecommunications and mobile carriers, media and broadcast companies, health organizations, e-commerce giants, governments and defence agencies. Endace solutions deliver the actionable network data customers need for security and network performance issue identification and resolution. Endace is 100% accurate network capture and recording allows organizations to meet regulatory and governance obligations such as security monitoring and analysis, archival and lawful intercept. [10]

2.2.1 Architecture

EndaceProbes are based on a layered architecture comprising environmentally hardened server hardware tightly integrated with best-in-class Data Acquisition and Generation, or DAG technology and our own purpose-built operating system, OSm. By controlling every aspect of the hardware, firmware and software, EndaceProbes have been optimized to deliver exceptional performance even under the most extreme conditions. [11]

Endace layered architecture might be seen in the figure 2.6 Endace architecture [11].

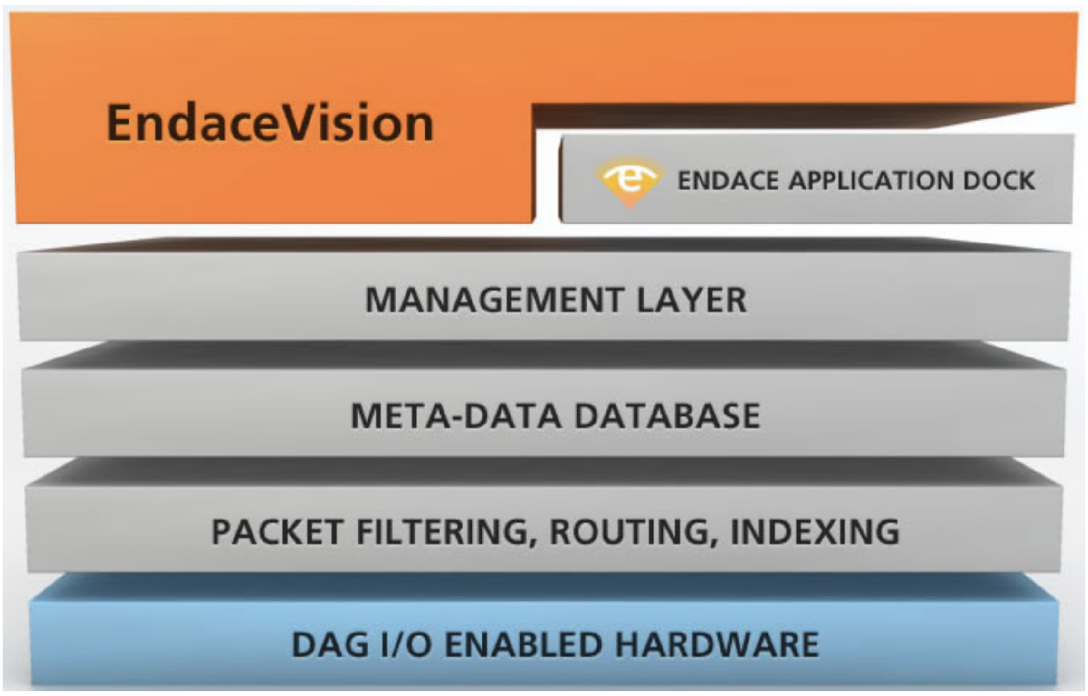


Figure 2.6: Endace architecture [11]

Endace DAG Packet Capture Cards guarantee 100% packet capture on any network regardless of packet size, interface type, or network load. DAG cards are the gold standard packet capture (sometimes called packet sniffer) cards used in appliances for security and network monitoring. DAG cards have up to four monitoring ports per card so they deliver industry-leading port density to save on rack space, power and cooling which ensures low cost of ownership. They are engineered for reliability and performance and, unlike a number of other packet capture cards, feature a fanless design that completely removes the possibility of failure due to moving parts. [10]

Looking from another position, flow architecture might be seen in the figure 2.7 Endace architecture [12], where the importance of data pipes is visible. DAG module role is to capture every single packet. Role of data pipes is to transfer specific flow the right direction.

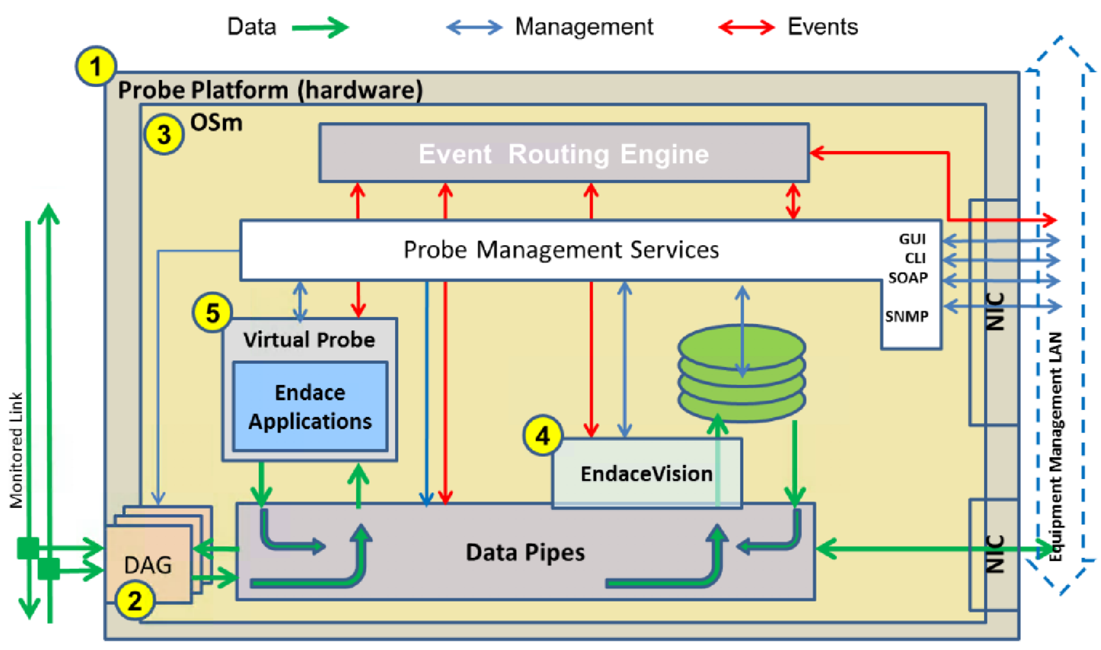


Figure 2.7: Endace architecture [12]

2.2.2 Performance

Table 2.2 shows basic data about EndaceProbe 7000 series.

Table 2.2: Endace technical data

Type	Value
Product model	NPB7000
System memory	48 GB
No. of CPUs	24
Bios version	2.1a
Data File system	8390 GB
Raid type	RAID-50
Operating system	OSm 5.1.3
Supporting speed	10 Gb/s

2.2.3 Abilities

Endace probe 7000 series offers ideal opportunity for this thesis. Thanks to its hardware construction, it might be said its capacity is endless. However, its main advantages are software-based:

- Dedicated Operating System, or OS, which allows this device to capture data for a very long time
- Full indexing of recorded traffic including application classification
- EndaceVision browser application providing network-wide visualization
- EndacePackets browser application providing packet-level investigation
- Easy integration with third-party applications via API and industry standard file formats

2.3 Operational Data Detection

To find an unwanted operational data might be a hard nut to crack. The classical approach, which will be used, is to establish GPON technology, generate random traffic through it and capture everything. Later, data flow will be analyzed in order to find such kind of data.

2.3.1 Requirements Specification

Below, there are listed ten basic requirements for a scenario in which possible unwanted operational data might be found.

1: GPON
Reason: In order to create GPON technology
Accomplishment: OLT and ONU devices
Description: Both OLT and ONU are necessary to establish GPON technology.

2: Communication generator
Reason: In order to generate random traffic
Accomplishment: Virtualization
Description: Virtualized host inside ONU's network needs to generate random traffic, which will pass through GPON.

3: Internet connection
Reason: In order to create an opportunity for GPON
Accomplishment: NAT
Description: Using Network Address Translation, the GPON will have connection to the Internet, which might motivate it to send an operational data.

4: Mirroring ability
Reason: In order to send all data towards collector
Accomplishment: Port configuration
Description: Port on the switch towards the Internet will be mirrored towards the Endace.

5: Storage capacity
Reason: In order to store all packets
Accomplishment: Endace
Description: Every packet needs to be stored for later analysis.

6: Continuous service
Reason: In order to capture exceptional data
Accomplishment: Endace
Description: Service needs to run in long term, because it is unknown, how long it might take the GPON to send operational data.

7: Packet analysis
Reason: In order to find operational data
Accomplishment: EndacePackets
Description: From entire communication, operational data needs to be filtered out.

8: Packet receiving
Reason: In order to receive data on a server
Accomplishment: Python TCP sockets
Description: To have all data available on a server, where advanced techniques might be implemented.

9: Packet parser
Reason: In order to read data from a .pcap file
Accomplishment: Python dpkt library
Description: To be able to read received data.

10: Analysis output
Reason: In order to deliver results
Accomplishment: Python script
Description: To be able to process the information.

2.4 Python dpkt

Dpkt is a python module for fast, simple packet creation / parsing, with definitions for the basic TCP/IP protocols. [13] Using this module it is possible to extract packets one by one from a pcap file and read their headers. Module is compatible with Python 3.5 and works under Windows environment. Function of the dpkt pcap Reader can be seen in the figure 2.8 dpkt pcap Reader.

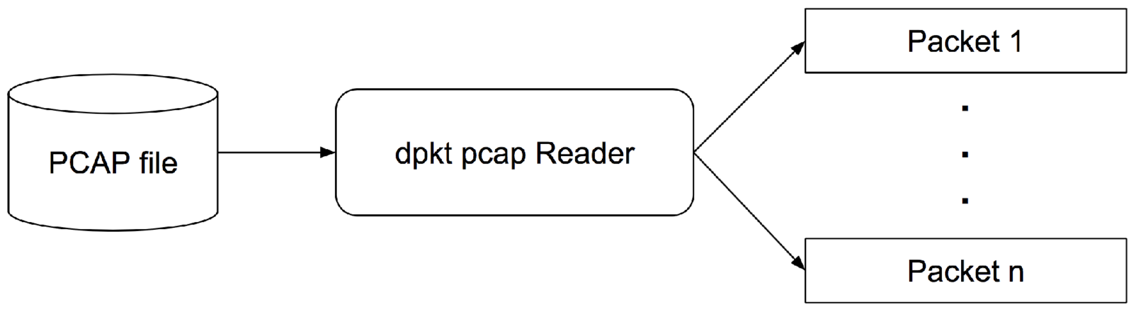


Figure 2.8: dpkt pcap Reader

2.5 Analysis Summary

This chapter provided overview of a Gigabit Passive Optical Network, explained its architecture, definition and limitations. The limitations of various GOPNs were compared in the table 2.1 PON comparison.

Later, high-speed packet analyzer Endace probe was described. Its layered architecture might be seen in the figure 2.6 and flow architecture in the figure 2.7.

Next, ten basic requirements for operational data detection were specified. The requirements start with GPON technology establishment, continue through mirroring ability and storage capacity, and conclude by analysis output.

In the end of this chapter, python module for reading a pcap file and parsing the packets was introduced.

3 Solution Proposal

In this chapter the plan of how to capture an operational data is introduced. It consists of topology, devices and scenario overview.

3.1 Topology

Simple topology diagram might be seen in the figure 3.1. Green arrow represent mirroring of all data passing through GPON. Red arrows represent potential operational data from the GPON itself. All three arrows show only data origin and destination. Path, the traffic originated form ONT is via OLT and Dell switch, where it will be mirrored together with the traffic originated from OLT and send via Cisco switch towards the DAG module of Endace probe. Computer is in reality virtualized under CentOS and its purpose is to generate traffic which will pass through the GPON.

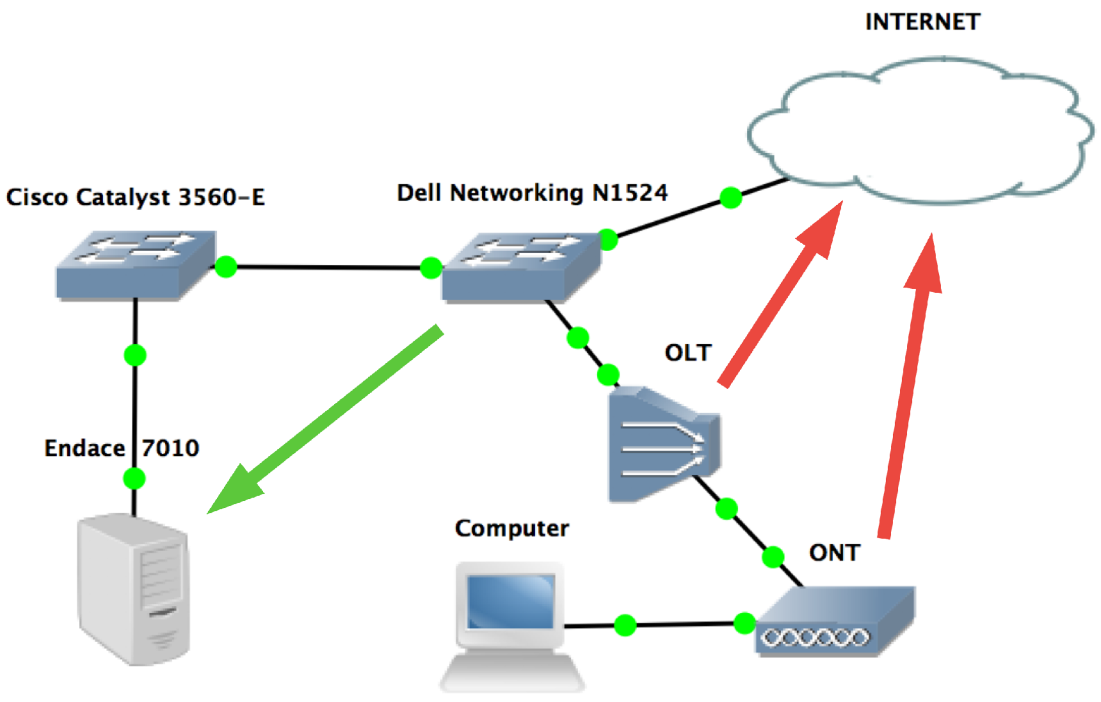


Figure 3.1: Connection Diagram

3.2 Devices

Besides other devices providing support, topology consists of four main devices:

- GPON OLT
- Endace Probe
- Dell switch
- Cisco switch

In the figure 3.2 Huawei MA5683T might be seen.

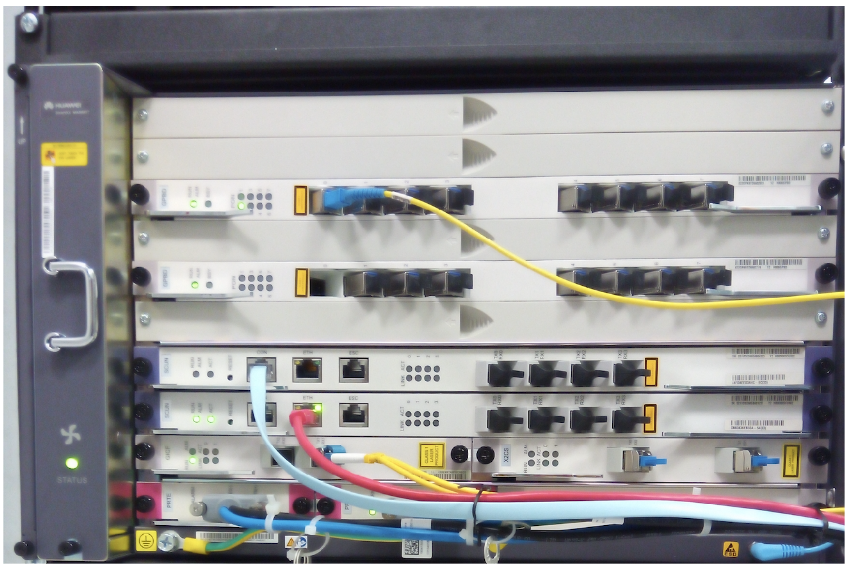


Figure 3.2: Huawei MA5683T

Table 3.1 Huawei MA5683T GPON OLT shows technical overview of Huawei MA5683T GPON OLT, which will be connected according to the topology diagram.

Table 3.1: Huawei MA5683T GPON OLT

Type	Value
Product model	MA5683T
Backplane bus switching capacity	1.5 Tbps
Number of boards	6/13
Maximum number of GPON ports in a subrack	96
Maximum Split Ratio	1:128
Tx rate	2.488 Gbps
Rx rate	1.244 Gbps
Optical Fiber Type	Single-mode
Reach	20 km
Weight (empty chassis)	7.0kg

Other devices used in topology are:

- Server Dell T330 running Windows Server 2012
- Server Dell R550 running ESXI
- WiFi modem

Those devices are providing necessary support such as:

- Providing environment to be able to access and configure all main devices
- Providing Internet access
- Acting as a host in LAN and generating traffic
- Acting as GPON ONT and creating LAN
- Providing environment for a customized application

3.3 Scenario

The plan consists of five steps:

- Connect and configure all devices inside the topology the way to meet the requirements listed in Chapter 2.
- Host (virtualized CentOS) will generate random traffic from ONU's LAN via both GPON's units to the Internet. Switch will mirror the port connected to the Internet and send all data towards the probe.
- Endace will capture all data and store it in the local storage. Basic analysis will be done using its internal filtering mechanisms.
- Endace will transfer selected data to server, where customized application can be run.
- This application written in python will take a closer look at the data and could filter them based on the pre-defined rules.

3.4 Data Analysis

Endace will capture and store entire traffic. The transfer to another server will be done using TCP protocol. Data will be stored on a server in .pcap format. Once data are transferred, an analysis over them will be done. Because the data are stored in .pcap file, they could be read using dpkt module. This module will be imported in application and will read the packets one by one from the file. Simplified diagram can be seen in the figure 3.3 Data transfer.

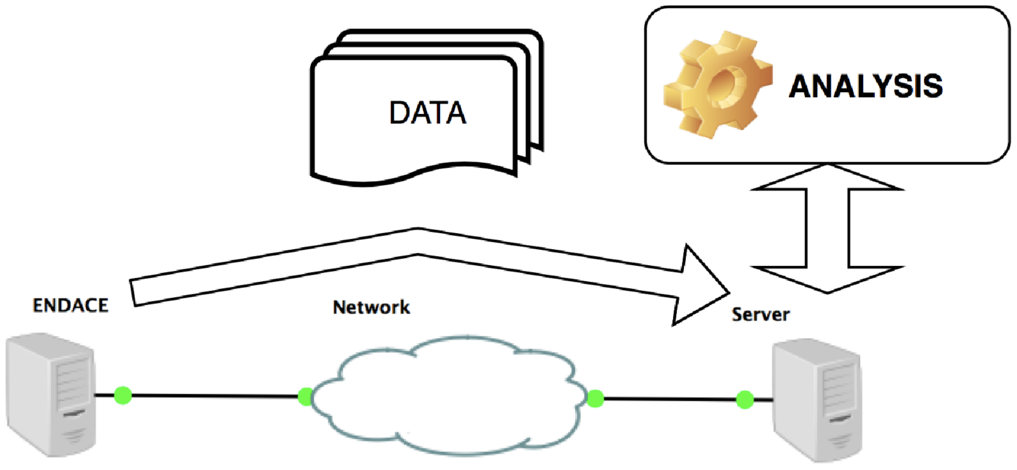


Figure 3.3: Data transfer

The analysis over this data will be done using filtering mechanisms (rules). Those rules could be defined within the application, or could be written based on template and then loaded. The application will be controlled via CLI, which will support various commands. The complete list of commands for it can be found in the Appendix A ODDA functionality.

ODDA as Operational Data Detection Application is the name given to the application, which will be developed. In the figure 3.4 Basic overview, a hierarchy of the ODDA might be seen. There will be two ways of controlling the application, which is build based on four main pillars covering main functionality.

Automatic control will be a simplified API, or Application Programming Interface providing an option to use core of the application by another application.

Manual control will be representing a CLI, or Command Line Interface allowing a user to fully control every aspect of the application.

Data receive will be one of the core functionality supporting data receiving from the Endace probe via both: TCP, or Transport Control Protocol and UDP, or User Datagram Protocol.

PCAP will be one of the core functionality supporting PCAP, or Packet Capture file creation and handling.

Rules will be one of the main functionality, designed in order to implement and use rules, based on whose the ODDA can process the data.

Data process will be one to the main functionality processing the selected data one packet after another in form of a pcap file according to the pre-defined rules.

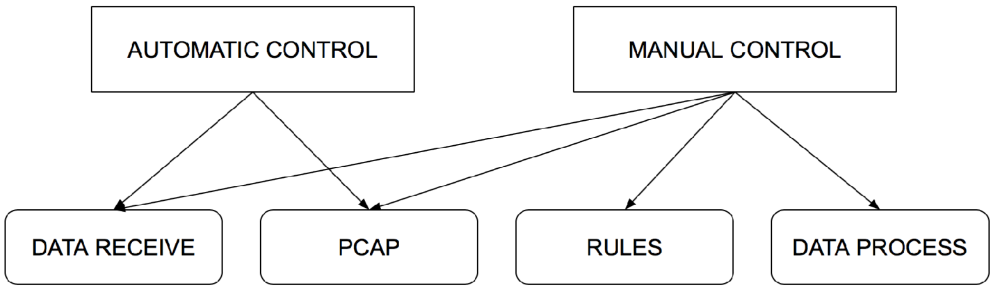


Figure 3.4: Basic overview

4 Implementation

This chapter consists of connecting the devices between each other and their configuration. Later, implementation of an application written in python for packet analysis is described.

4.1 Devices connection

All devices were racked, stacked and connected according to the topology diagram. In the figure 4.1 the GPON OLT unit might be seen in one of the rack of SIX research center in the Faculty of Electrical Engineering and Communication.

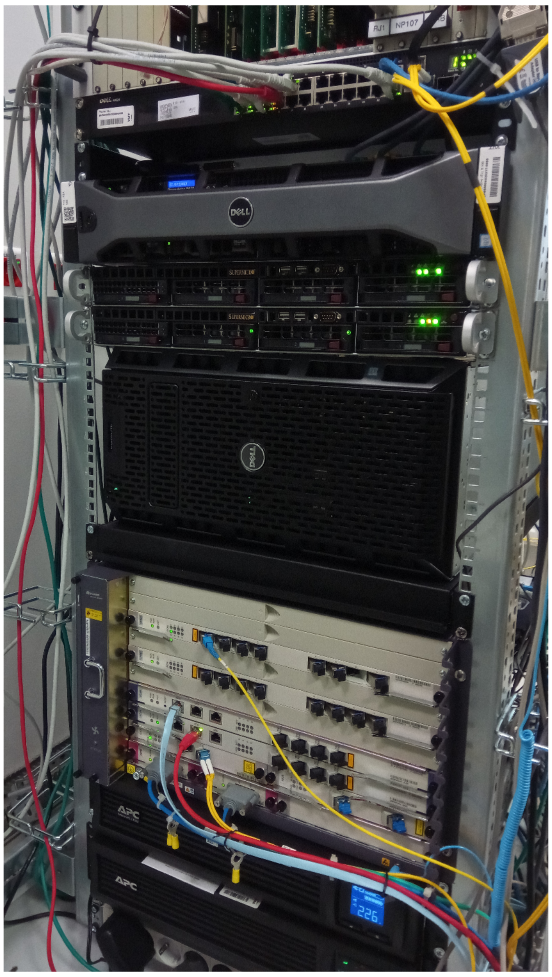


Figure 4.1: Racked GPON OLT

4.2 Configuration

This section shows configuration of four main devices in the topology: GPON OLT, Cisco and Dell switches and Endace Probe.

4.2.1 GPON OLT

Main configuration part of the OLT unit interface 0/2 might be seen in the listing 4.1 GPON OLT configuration. For purpose of this thesis, ONU with ip address of 10.0.0.200/24 was chosen. Authentication commands were censured for security reasons. GPON OLT was configured by Dr. Oujezský.

Listing 4.1: GPON OLT configuration

```
#
 [gpon]
   <gpon-0/2>
  interface gpon 0/2
  port 0 ont-auto-find enable
  port 0 fec enable
  port 1 ont-auto-find enable
  ont add 0 0 sn-auth "****************" omci
  ont-lineprofile-id 1
 ont-srvprofile-id 1 desc "ONT_NO_DESCRIPTION"
  ont ipconfig 0 0 static ip-address 10.0.0.200 mask
  255.255.255.0 vlan 2
 priority 5 gateway 10.0.0.30
  ont add 0 1 sn-auth "****************" omci
  ont-lineprofile-id 1
 ont-srvprofile-id 1 desc "ONT_NO_DESCRIPTION"
  ont ipconfig 0 1 static ip-address 10.0.0.201 mask
  255.255.255.0 vlan 2
 priority 5 gateway 10.0.0.30
  ont add 0 2 sn-auth "****************" omci
  ont-lineprofile-id 1
 ont-srvprofile-id 1 desc "ONT_NO_DESCRIPTION"
  ont ipconfig 0 2 static ip-address 10.0.0.202 mask
  255.255.255.0 vlan 2
 priority 5 gateway 10.0.0.30
  ont port native-vlan 0 0 eth 1 vlan 2 priority 0
  ont port native-vlan 0 1 eth 1 vlan 2 priority 0
  ont port native-vlan 0 2 eth 1 vlan 2 priority 0
 #
```

4.2.2 Switches

Mirroring configuration on both switches might be seen in the listing 4.2 CISCO switch configuration, respectively 4.3 DELL switch configuration. This part of configuration ensures, that traffic passing through GPON towards the Internet and vice versa will be mirrored and send towards the Endace. Switches were configured by Dr. Oujezský.

Listing 4.2: CISCO switch configuration

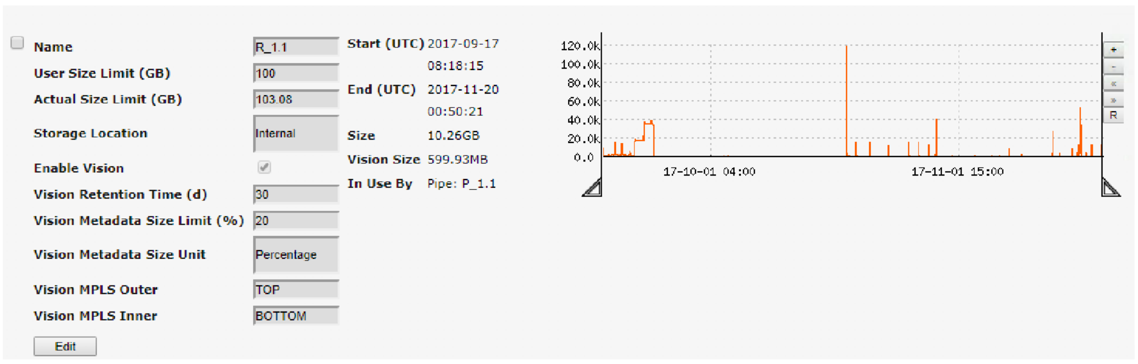
```
!
 interface GigabitEthernet0/26
  description ENDACE RSPAN
  switchport trunk encapsulation dot1q
  switchport trunk allowed vlan 100
  switchport mode trunk
  load-interval 30
 !
 interface TenGigabitEthernet0/2
  switchport access vlan 4
  switchport mode access
 !
 monitor session 1 destination interface Te0/2
 monitor session 1 source remote vlan 100
 !
```

Listing 4.3: DELL switch configuration

```
 !
  vlan 100
  name "RSPAN"
  remote-span
  monitor session 1 destination remote vlan 100
  reflector-port Te1/0/1
  monitor session 1 source interface Gi1/0/24
  monitor session 1 mode
  interface Te1/0/1
  description "ENDACE_MIRROR"
  spanning-tree portfast
  switchport mode trunk
  switchport trunk allowed vlan 100
  udld enable
  no lldp transmit
  no lldp receive
  no lldp med
 !
```

4.2.3 Endace

Endace configuration started with defining available space for the data. For this, rotation file *R_1.1* was configured as might be seen in the figure 4.2 Rotation file. This file there will be stored all the traffic, which passed through GPON. Therefore, a sufficient limit of 100 GB, or Gigabytes was chosen.



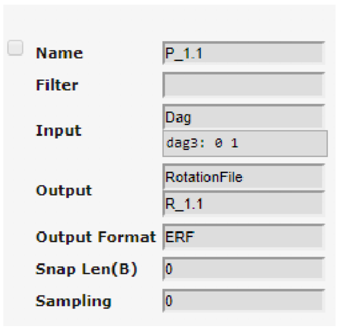
Name	R_1.1
User Size Limit (GB)	100
Actual Size Limit (GB)	103.08
Storage Location	Internal
Enable Vision	✓
Vision Retention Time (d)	30
Vision Metadata Size Limit (%)	20
Vision Metadata Size Unit	Percentage
Vision MPLS Outer	TOP
Vision MPLS Inner	BOTTOM

Edit

Start (UTC)	2017-09-17 08:18:15
End (UTC)	2017-11-20 00:50:21
Size	10.26GB
Vision Size	599.93MB
In Use By	Pipe: P_1.1

Figure 4.2: Rotation file

Second, data pipe *P_1.1* was configured in order direct received data from DAG module to the rotation file *R_1.1*. Configuration might be seen in the figure 4.3 Data pipe. Extensible Record Format, or ERF was chosen as a output format, because it was defined by Endace and their DAG modules and cards are designed to support it.

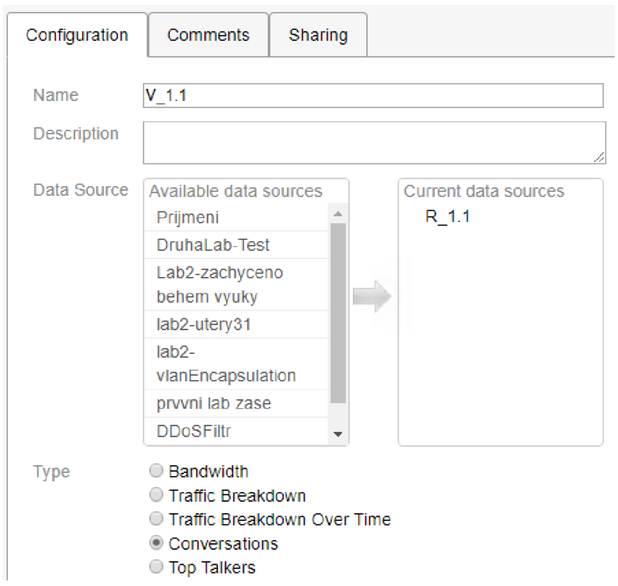


Name	P_1.1
Filter	
Input	Dag dag3: 0 1
Output	RotationFile R_1.1
Output Format	ERF
Snap Len(B)	0
Sampling	0

Figure 4.3: Data pipe

Next, visualization *V_1.1* was configured in order to have basic visibility over the captured data. Its configuration might be seen in the figure 4.4 Visualization. As a data source, the rotation file *R_1.1* was chosen.

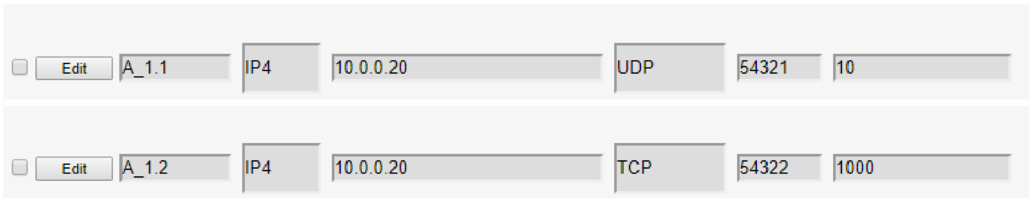
There are more types of visualizations carts: Bandwidth, Traffic Breakdown, Traffic Breakdown Over Time, Conversations and Top Talkers. Everyone has its own purpose and displays same source data from various point of views. The most useful for this thesis was Conversations type, because it can display flow of data as a list of conversations between two hosts.



Name	V_1.1
Description	
Data Source	Available data sources: Prijmeni, DruhaLab-Test, Lab2-zachyceno behem vyuky, lab2-utery31, lab2-vlanEncapsulation, prvni lab zase, DDoSFiltr; Current data sources: R_1.1
Type	Bandwidth, Traffic Breakdown, Traffic Breakdown Over Time [✓ Conversations], Top Talkers

Figure 4.4: Visualization

In order to send data data towards the server for an advanced analysis, new pipes, needs to be configured. As a destination, the address, protocol and port need to be specified. In the figure 4.5 Network Addresses there might be seen two destinations: *A_1.1* and *A_1.2* for both: UDP and TCP transfer. Ports reserved for this purpose are 54321 and 54322. With similar to the *P_1.1*, but adjusted data pipes *P_1.2* and *P_1.3*, Endace can transfer data to a server, where those ports need to be enabled in firewall.



☐	Edit	A_1.1	IP4	10.0.0.20	UDP	54321	10
☐	Edit	A_1.2	IP4	10.0.0.20	TCP	54322	1000

Figure 4.5: Network Addresses

4.3 Application

Application for processing the data was written in python language version 3 and was given a name: ODDA, as Operational Data Detection Application. Its functionality is more described in the Appendix A ODDA functionality.

Application can be operated via both: CLI and API. Both share some common resources of the application, but they are not designed to do the same. CLI was designed to operate the whole application, while the purpose of its API is to use the application's core for online data receiving and then build and return dedicated data.

4.3.1 ODDA Overview

Structure of the application is represented using a module diagram which can be seen in the figure 4.6 Module Diagram.

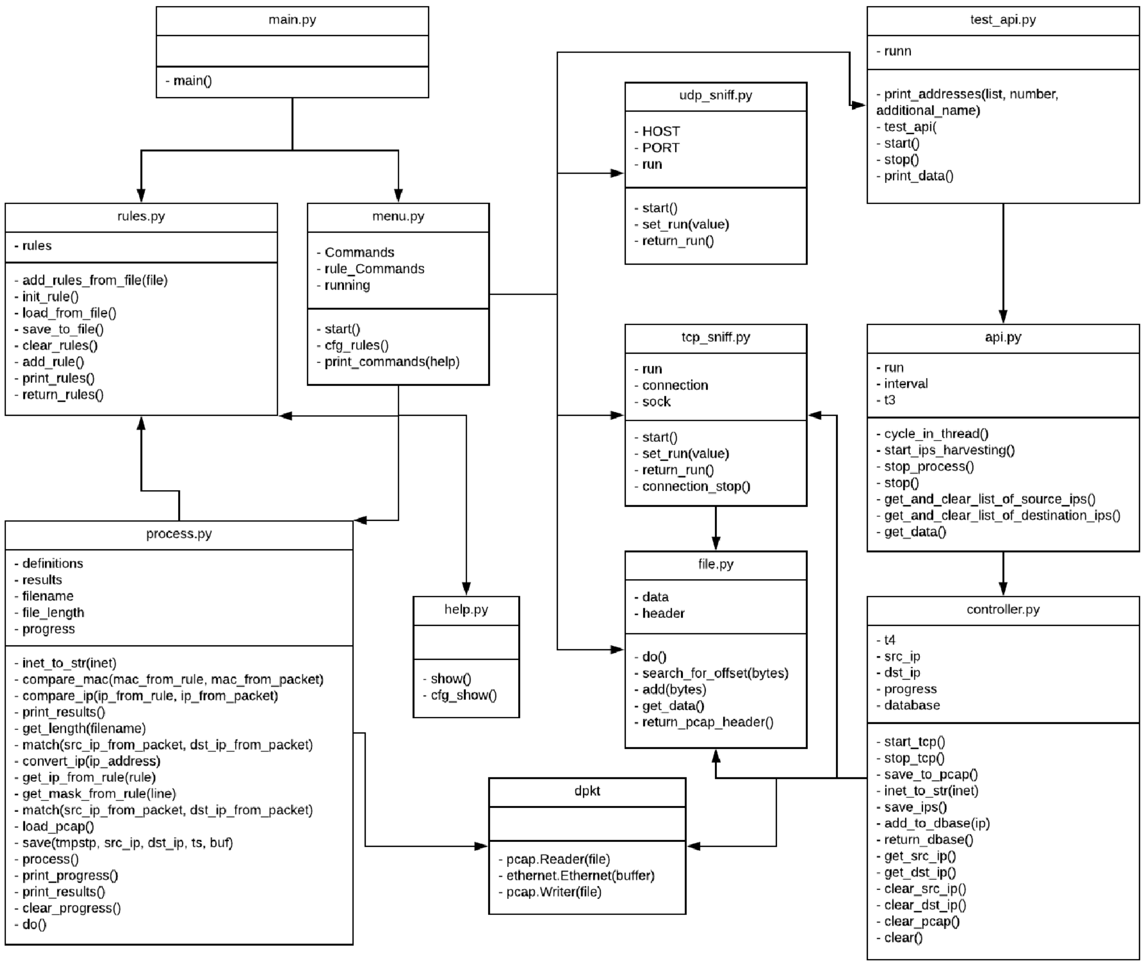


Figure 4.6: Module Diagram

main.py: is used for starting the application. In the beginning it tries to initialize rules form a default *rule.cfg* file and opens a CLI mode.

rules.py: is used to handle operations with the rules, such as: initialize, print, create, load from a file, save to a file, clear and return.

menu.py: is a CLI mode of ODDA. From this mode, user can control every aspect of the application. All the commands are listed in the Appendix A ODDA functionality.

process.py: is used for analyzing the data from a pcap file according to the rules. This process can take more minutes to complete. It depends on pcap file length and rule file complexity. Therefore, it is run in separate thread and a progress status can be shown. This module is also responsible for saving and printing the results.

help.py: this module can print all available commands, their description and one useful note, that using a *"?"* can help filter and show available commands.

dpkt.py: is an imported library for raw operations with a pcap files such as opening a pcap file, buffering the packets or writing them to a pcap file.

udp_sniff.py: is a module, which was developed just for transfer testing. It is simpler than TCP and can verify if data sent from source are coming.

tcp_sniff.py: is one of the core modules, which was developed to support data receiving via TCP protocol. Using a TCP is important in order to achieve data loss protection.

file.py: is used by both: CLI and API for operations with received data and creating files.

test_api.py: is used to automatically test API functionality and also provides ability to use API functions from CLI mode.

api.py: is designed to use core of the application for dedicated purpose and be imported as a module by another application.

controller.py: is used as a mid-module between api.py and core to maintain both: api simplicity and functionality.

4.3.2 Data receive

Data from Endace probe are received via Transmission Control Protocol, or TCP and port 54322, which was allowed in the server's firewall. In the listing 4.4 TCP listening a code can be seen in the python language opening up a TCP socket for the specific port for listening. The application has in disposal also a UDP, but it was implemented only for testing purposes.

Data receiving can be started and stopped from both: CLI as well as API. Received data are later processed by other modules such as file.py for a pcap file creation, which can be that read by dpkt module for either processing the data according to the rules, or building up an API's database.

Listing 4.4: TCP listening

```
# Create a TCP socket
try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print ('Socket was created')

except socket.error as msg :
        print ('Failed to create socket.')
        sys.exit()

# Bind the socket to the port
port = 54322
server_address = ("", port)
print('starting up on {} port {}'.format(*server_address))

try:
        sock.bind(server_address)

except socket.error as msg:
        print ('Bind failed.')
        sys.exit()

# Listen for an incoming connections
sock.listen(1)
print ('Listening on a TCP port ', port)
```

4.3.3 Rules

Packets are filtered according to the pre-defined rules. Format of the rules can be seen in the listing 4.5 Rules format. Although it is possible to set up a rule via console, it is simpler to write them on their own, save to a file and then import or load them to the application. When importing or loading the rules form an own file, ODDA requires to have them is the exact format as in the following listing. Another file *apnic.cfg* with lots of rules can be seen on an attached DVD.

Rules are designed to support IPv4 only, or Internet Protocol version 4 only and VLSM or Variable-Length Subnet Masking. They are also bi-directional, so it means it is possible to distinguish between source and destination address. However, only positive logic was implemented.

Listing 4.5: Rules format

```
ID: 10
SOURCE MAC: any
DESTINATION MAC: any
SOURCE IP: 10.0.0.2
SOURCE MASK: 32
DESTINATION IP: any
DESTINATION MASK: 0

ID: 20
SOURCE MAC: any
DESTINATION MAC: any
SOURCE IP: any
SOURCE MASK: 0
DESTINATION IP: 42.0.0.0
DESTINATION MASK: 8
```

4.3.4 Data processing

Data are processed using a core function *process()*, which can be seen in the listing 4.6 Process function. During the process, a specific PCAP file is opened and divided through the dpkt module into packets. Later, function *match()* saves the selected packets based on pre-defined rules. Because of the size of some PCAP files, the progress calculator was implemented too, so it is possible to check how far the processing went at a specific time.

Function *save()* saves packets matching the rules and later, those rules can be printed. Output of the results is provides in a three way: summary on the screen, timestamps with IP addresses to a text file *output.txt* and packets to a pcap file *output.pcap*.

Listing 4.6: Process function

```
# Open the specific file
file = open(filename, 'rb')
pcapfile = dpkt.pcap.Reader(file)

#Load the definitions
definitions = rules.return_rules()

# For every packet in file
```

```
for timestamp, buf in pcapfile:

        progress += 1
        eth = dpkt.ethernet.Ethernet(buf)

        # Check if packet has network header
        if not isinstance(eth.data, dpkt.ip.IP):
                continue

        ip = eth.data

        # Grab the data
        tmpstp = str(datetime.datetime.utcfromtimestamp(
           timestamp))
        src_ip = inet_to_str(ip.src)
        dst_ip = inet_to_str(ip.dst)

        # Save the data if they match the rules
        if match(src_ip, dst_ip):
                save(tmpstp, src_ip, dst_ip, timestamp, buf)

# Close the file
file.close()
```

4.3.5 Application Programming Interface

Application Programming Interface was designed to use core of the application in order to continuous receive real-time data via TCP and read source and destination IP addresses of all packets. For this, a dedicated controller was implemented. The controller can be operated via API, which needs to be imported as a module.

API can return the data either as list of source IP addresses, list of destination IP addresses or a database with list of IP addresses and count of packets.

Because of the API was developed to for a single purpose and has a controller at its disposal, it consists of few seamless functions. For example, in the listing 4.7 Starting the API can be seen, how it can be started using just one command. That function will create the thread in which the API will be running.

Listing 4.7: Starting the API

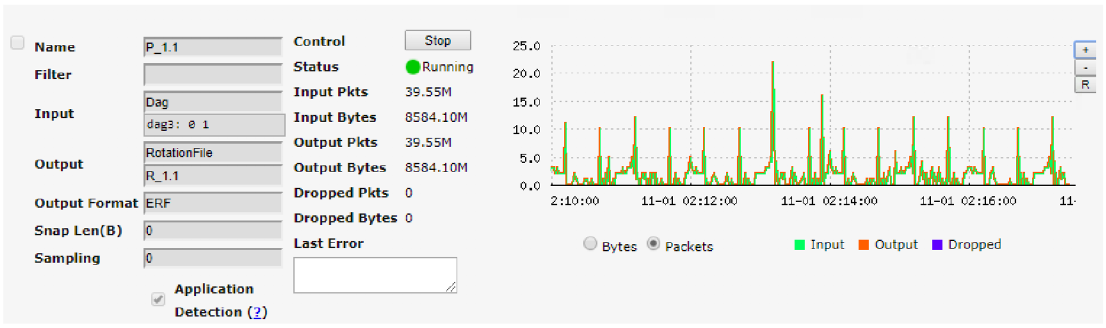
```
def start():
        api.start_ips_harvesting()
        print ('API was started.')
```

5 Evaluation

This chapter evaluates implemented solution as well as reveals the results. Section Data capture evaluates ability to capture all data passing through the GPON towards the internet and vice versa. Next, the application is tested using manual tests and automatic testing feature. Later, achieved results are revealed.

5.1 Data capture

Figure 5.1 Data Capture shows data pipe *P_1.1* in running state. This pipe receives all packets from a DAG module. Using this pipe it is possible to save these packets into internal storage.



Name	P_1.1
Filter	
Input	Dag dag3: 0 1
Output	RotationFile R_1.1
Output Format	ERF
Snap Len(B)	0
Sampling	0
✓ Application Detection (?)	

Control	Stop
Status	● Running
Input Pkts	39.55M
Input Bytes	8584.10M
Output Pkts	39.55M
Output Bytes	8584.10M
Dropped Pkts	0
Dropped Bytes	0
Last Error	

Figure 5.1: Data Capture

Main advantage here is the capacity of Endace probe. Thanks to its ability to be operational nonstop, powerful Central Processing Unit, or CPU and huge internal storage, it might be said that for purpose of this thesis it's capacity is endless.

Similary to data pipe *P_1.1*, there are also data pipes *P_1.2*, *P_1.3* and WEBGUI for sending data from *R_1.1* towards the server via UDP, TCP and for sending the real time data directly from the DAG module.

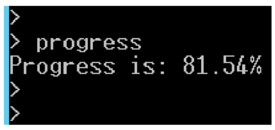
On the other side, the server, or specifically the application needs to be running and set for listening to receive any packets via either TCP or UDP protocol. For a successful start of data transfer, the specific ports on a firewall needed to be allowed.

5.2 Application Testing

Testing of an application was done for both modes: CLI and API separately. Both modes were tested the way, by which they were designed to operate. This means, that CLI was tested manually, because it was designed to be operated by user, whereas for API, an automatic testing system was developed.

5.2.1 CLI Testing

Below, there is a list of tests performed by user in CLI mode of the application in order to verify its functionality and stability. One example of testing can be seen in the figure 5.2 Progress, where is output showing progress of an data analysis. Taking into consideration possible size of a pcap file, process can last many minutes, so knowing the progress is a very handy information.



```
>
> progress
Progress is: 81.54%
>
>
```

Figure 5.2: Progress

1: Starting the UDP listening
Reason: In order to test packet receiving
Accomplished via command: *runu*
Result: Successful.

2: Stopping the active UDP listening
Reason: In order to stop active UDP listening
Accomplished via command: *stopu*
Result: Successful.

3: Stopping the non-active UDP listening
Reason: In order to test application stability
Accomplished via command: *stopu*
Result: Unsuccessful.

4: Starting the active TCP listening
Reason: In order to test packet receiving
Accomplished via command: *runt*
Result: Successful.

5: Stopping the active TCP listening
Reason: In order to stop active TCP listening
Accomplished via command: *stopt*
Result: Successful.

6: Stopping the non-active TCP listening
Reason: In order to test application stability
Accomplished via command: *stopt*
Result: Unsuccessful.

7: Saving the received data
Reason: In order to test pcap file creation
Accomplished via command: *file*
Result: Successful.

8: Loading a pcap file
Reason: In order to test pcap file reader
Accomplished via command: *load*
Result: Successful.

9: Processing the data
Reason: In order to test data analysis
Accomplished via command: *process*
Result: Successful.

10: Showing a progress
Reason: In order to test process progress
Accomplished via command: *progress*
Result: Successful.

11: Printing the results
Reason: In order to test results reachability
Accomplished via command: *results*
Result: Successful.

12: Adding new rule
Reason: In order to create a new rule
Accomplished via command: *add*
Result: Successful.

13: Saving the rules
Reason: In order to save configured rules to a file
Accomplished via command: *save*
Result: Successful.

14: Loading the rules from a file
Reason: In order to load the rules from a file
Accomplished via command: *load*
Result: Successful.

15: Loading the rules from a non-existing file
Reason: In order to test application stability
Accomplished via command: *load*

Result: Successful.

5.2.2 API testing

Testing of the API was done automatically using test_api function. Part of the source code can be seen in the listing 5.1 API testing. The function starts API and every interval, which was set for 60 seconds during the testing, grabs IP addresses and saves output to a file. Testing was performed in two fifty minutes cycles, those outputs are in folders *api test 1* and *api test 2*, and are attached on Digital Video Disc, or DVD as can be seen in the Appendix C List of files on attached DVD.

Listing 5.1: API testing

```
api.start_ips_harvesting()

for i in range (1, 50):
        time.sleep(interval)

        list_of_source_ip_addresses = api.
           get_and_clear_list_of_source_ips()
        list_of_destination_ip_addresses = api.
           get_and_clear_list_of_destination_ips()

        print_addresses(list_of_source_ip_addresses, i, '
           _list_of_source_ip_addresses_')
        print_addresses(list_of_destination_ip_addresses, i,
            '_list_of_destination_ip_addresses_')

print_data()
api.stop()
```

In the listing 5.2 Database print is the example on how it is possible to get and print API's database. This function was also used during the testing.

Listing 5.2: Database print

```
def print_data():

        data = api.get_data()

        print('| Address          | count ')
        for item in data:
                print('| {0:15s} | {1:d}'.format(item[0],
                   item[1]))
```

5.2.3 Testing summary

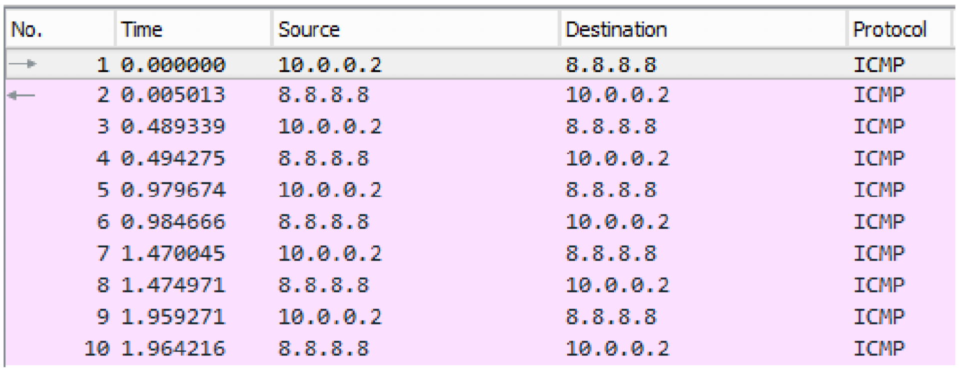
Application testing successfully finished. Application is fully operational and processes, what it is designed for. However, minor stability tests revealed, that application is not totally immune against not valid actions, or in another words: it is possible to crash the application using invalid procedure. Therefore, it is important for user to have at least a basic knowledge about the applications commands. API tests verified, that API is really capable of delivering the results it is dedicated for. The example of database output can be seen in the listing 5.3 API database output.

Listing 5.3: API database output

```
| Address         | count
| 192.168.10.2    | 11746
| 192.168.10.96   | 11746
| 10.0.0.110      | 458
| 239.255.255.250 | 226
| 10.0.0.200      | 68
| 216.58.201.110  | 10
| 10.0.0.81       | 219
| 10.0.0.255      | 122
| 224.0.0.252     | 266
| 10.0.0.2        | 394
| 10.0.0.20       | 88
| 10.0.0.10       | 30
| 10.0.0.11       | 34
| 169.254.19.65   | 67
| 10.0.0.99       | 44
| 10.0.0.76       | 6
| 0.0.0.0         | 18
| 255.255.255.255 | 18
| 8.8.8.8         | 28
| 192.168.0.100   | 6
| 10.0.0.50       | 57
| 169.254.255.255 | 1
| 54.191.46.28    | 30
>
```

5.3 Results

Searching for an unwanted operational data of the GPON OLT was done using adjusted rules for the following IP addresses: *192.168.200.1* and *10.0.0.2*, because they are configured on the OLT device. Only few packets originated communicated by OLT left the Local Area Network. The result might be seen in the figure 5.3 Data investigation.

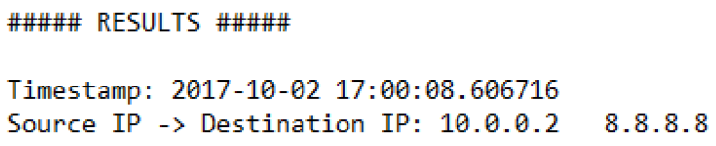


No.	Time	Source	Destination	Protocol
1	0.000000	10.0.0.2	8.8.8.8	ICMP
2	0.005013	8.8.8.8	10.0.0.2	ICMP
3	0.489339	10.0.0.2	8.8.8.8	ICMP
4	0.494275	8.8.8.8	10.0.0.2	ICMP
5	0.979674	10.0.0.2	8.8.8.8	ICMP
6	0.984666	8.8.8.8	10.0.0.2	ICMP
7	1.470045	10.0.0.2	8.8.8.8	ICMP
8	1.474971	8.8.8.8	10.0.0.2	ICMP
9	1.959271	10.0.0.2	8.8.8.8	ICMP
10	1.964216	8.8.8.8	10.0.0.2	ICMP

Figure 5.3: Data investigation

There are 5 ICMP requests originated from GPON OLT with destination address of the Google's DNS server. Source Media Access Control, or MAC address is 48:FD:8E:E0:3A:4F, which is globally unique address and manufacturer part of it indicates it belongs to the Huawei. Whether it was generated automatically or manually, which is not possible to conclude from the output, it might be concluded, that these packets do not belong to a category of unwanted operational data.

The application provides results in a three way. First, it prints out basic summary such as packet count on the screen. Second, it writes to a file *output.txt* timestamps and ip addresses. The example can be seen in the figure 5.4 Results. Finally, it them in the pcap file *output.pcap*.



```
##### RESULTS #####

Timestamp: 2017-10-02 17:00:08.606716
Source IP -> Destination IP: 10.0.0.2   8.8.8.8
```

Figure 5.4: Results

Searching for an unwanted operational data of the GPON ONT device started defining its IP address *10.0.0.200* in rules. However, the ONT is performing NAT and there are other devices behind it, so it is not possible to define true origin of the data. Then it continued using another approach and it was selecting range of 'suspicious' destination address ranges, in this case from the APNIC region, which is shown in the Appendix B APNIC IP range. Those IP addresses were transformed to the rules in files *apnic.cfg*, which is without ONT IP address *10.0.0.200*, and *apnic2.cfg*, which is also with *10.0.0.200*. Although with this result it cannot be said whether the ONT is sending some operational data or not, it was possible to narrow the selection of data.

6 Conclusion

Gigabit Passive Optical Networks was analyzed and established inside one of the laboratories of the faculty. Between the GPON and Internet access, mirroring session was positioned to mirror all traffic towards a high-speed network analyzer. This analyzer, Endace probe, was configured to capture all packets and store it in the local storage.

Later, an advanced tool for unwanted operational data searching was developed in form of an application written in Python language. In order to make captured data available for the application, the transfer mechanism was established. This mechanism consists of Endace probe configuration in order send the data proper way towards the application, allowing the traffic to pass through a firewall and programing a listening module for the application.

Using dpkt library it was possible to open a pcap file and search for a packets using pre-defined rules. This rules can be either configured or prepared and imported. At this point, the application is universal. It can process any packet according to any rules. Results are provided in a triple way: it prints out summary on the screen, writes timestamps with addressing to a text file and creates new pcap file only with packets, those match the pre-defines rules.

Moreover, there is a simplified API developed for a dedicated purpose: to use core of an application for a real time traffic listening, reading the IP addresses and returning them to the function caller.

The analysis in order to find unwanted operational data originated from GPON devices was done using the developed application named ODDA. This application is fully operational and delivered the results, those can be found in the Chapter 5 Evaluation. The solution has met the assignment.

Bibliography

1. CISCO. *Cisco Visual Networking Index IP Traffic Growth*. 2016. Available also from: `https://newsroom.cisco.com/press-release-content?articleId=1771211` visited on 2017-11-10.
2. KIMIMEDIA. *http://www.kmimediagroup.com/articles2/430-articles-mit/army-s-optical-tryout* [online] [visited on 2017].
3. MICHAEL WILSON, RCDD. *MYTHBUSTERS The Honest Facts About Gigabit Passive Optical Networks*. 2017. Available also from: `https://www.bicsi.org/uploadedfiles/pdfs/presentations/region_events/Northeast_LaurelMD_03-14-13/MythbustersGigabitPON.pdf` [visited on 2017].
4. ITU-T. *G.984.1GPON General characteristics*. 2008. Available also from: `https://www.itu.int/rec/T-REC-G.984.1-200803-I/en` [visited on 2017].
5. JEFF WANG, President of Huawei's Access Network Product Line. *Huawei and Openreach Test 25G/100G IP Traffic Growth*. 2017. Available also from: `http://www.huawei.com/en/news/2017/7/Openreach-Test-Symmetric-PON-Prototype` [visited on 2017].
6. ITU-T. *CG.984.2 Physical Media Dependent (PMD) layer specification*. 2003. Available also from: `https://www.itu.int/rec/T-REC-G.984.2-200303-I/en` [visited on 2017].
7. ITU-T. *G.984.5 Enhancement band*. 2014. Available also from: `https://www.itu.int/rec/T-REC-G.984.5/en` [visited on 2017].
8. JIRACHARIYAKOOL, R.; SRA-IUM, N.; LERKVARANYU, S. Design and implement of GPON-FTTH network for residential condominium. In: *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 2017, pp. 1–5.
9. ITU-T. *G.984.3 Transmission convergence layer specification*. 2014. Available also from: `https://www.itu.int/rec/T-REC-G.984.3-201401-I/en` [visited on 2017].
10. ENDACE. *https://www.endace.com/* [online] [visited on 2017].
11. ENDACE. *EndaceProbe architecture*. 2017. Available also from: `https://www.endace.com/endace-high-speed-network-recorders.html` [visited on 2017].
12. ENDACE, Technology Ltd. Endace Probe Technical Overview. 2012, vol. 29.
13. SONG, Dug. *dpkt module Documentation*. 2015. Available also from: `https://dpkt.readthedocs.io/en/latest/index.html` [visited on 2018].

14. DELL. *Dell Networking N1500 Series Switches*. 2017. Available also from: `http://www.dell.com/en-us/work/shop/cty/pdp/spd/networking-n1500-series` [visited on 2017].
15. CISCO. *Cisco Catalyst 3560E-24TD-S SwitchG*. 2017. Available also from: `https://www.cisco.com/c/en/us/support/switches/catalyst-3560e-24td-s-switch/model.html#~tab-documents` [visited on 2017].
16. APNIC. *APNIC Resource ranges allocated by*. 2018. Available also from: `https://www.apnic.net/manage-ip/manage-resources/address-status/apnic-resource-range/` [visited on 2018].
17. NRO. *Regional Internet Registries Overview*. 2018. Available also from: `https://www.nro.net/about-the-nro/regional-internet-registries/` [visited on 2018].

Abbreviations

API	Application Programming Interface
APNIC	Asia-Pacific Network Information Centre
ATM	Asynchronous Transfer Mode
BICSI	Building Industry Consulting Service International
CAGR	Compound Annual Growth Rate
CATV	Cable Television
CLI	Command-line Interface
CPU	Central Processing Unit
DAG	Data Acquisition and Generation
DVD	Digital Video Disc
ERF	Extensible Record Format
FTTH	Fiber to the Home
GB	Gigabyte
Gbps	Gigabit per second
GEM	GPON Encapsulation Method
GTC	GPON Transmission Convergence
GPON	Gigabit Passive Optical Network
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPv4	Internat Protocol version 4
IPTV	Internet Protocol Television
ITU	International Telecommunication Union

ISP	Internet Service Provider
LAN	Local Area Network
MAC	Media Access Control
Max	Maximum
Mbps	Megabit per second
NAT	Network Address Translation
NGA	Next-Generation Access
NG-PON2	Next Generation Passive Optical Network (phase 2)
ODN	Optical Distribution Network
OLT	Optical Line Termination
OMCC	ONU Management and Control Channel
ONT	Optical Network Termination
OS	Operating System
PCAP	Packet Capture
PDF	Portable Document Format
PDU	Protocol Data Unit
PMD	Physical Media Dependent
PON	Passive Optical Network
SDH	Synchronous Digital Hierarchy
T-CONT	Transmission Container
Tbps	Terabit per second
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
UDP	User Datagram Protocol
UHD	Ultra High Definition
VLSM	Variable-Length Subnet Masking
VoIP	Voice over IP
WDM	Wavelength Division Multiplexing

Listings

4.1 GPON OLT configuration 19
4.2 CISCO switch configuration 20
4.3 DELL switch configuration 20
4.4 TCP listening 25
4.5 Rules format 26
4.6 Process function 26
4.7 Starting the API 27
5.1 API testing 31
5.2 Database print 31
5.3 API database output 32

List of Figures

1.1 Global IP Traffic Growth 1

2.1 GPON architecture 5
2.2 Wavelength allocation 1 5
2.3 Wavelength allocation 2 6
2.4 Downstream multiplexing 6
2.5 Upstream multiplexing 7
2.6 Endace architecture [11] 8
2.7 Endace architecture [12] 9
2.8 dpkt pcap Reader 12

3.1 Connection Diagram 13
3.2 Huawei MA5683T 14
3.3 Data transfer 16
3.4 Basic overview 17

4.1 Racked GPON OLT 18
4.2 Rotation file 21
4.3 Data pipe 21
4.4 Visualization 22
4.5 Network Addresses 22
4.6 Module Diagram 23

5.1 Data Capture 28
5.2 Progress 29
5.3 Data investigation 33
5.4 Results 33

A.1 Run the application 44

B.1 Regional Internet Registries [17] 46

List of Tables

2.1 PON comparison 4
2.2 Endace technical data 9

3.1 Huawei MA5683T GPON OLT 14

B.1 APNIC IP range part 1 46
B.2 APNIC IP range part 2 47
B.3 APNIC IP range part 3 48
B.4 APNIC IP range part 4 49

List of Appendices

This thesis includes following appendices:

- Appendix A ODDA functionality
- Appendix B APNIC IP range
- Appendix C List of files on attached DVD

A ODDA functionality

This appendix consists of a functionality information for delivered application. The guide is written separately for Command-line interface part as well as Application programming interface part.

A.1 Prerequisites

Basic instructions, those should be followed when running the application:

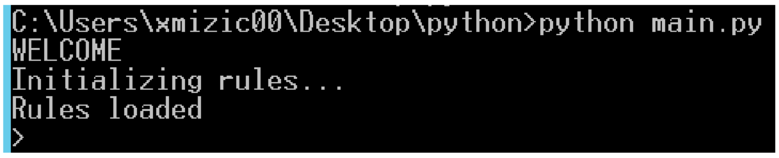
- The application is written in python version 3, therefore it can be run only in an environment which is capable of running python version 3.
- Source code consists of more files, therefore it is crucial to have all the files together.
- Although it should be possible to run it under more operating systems, it was tested only under Windows Server 2012.
- Application supports data receiving and creates/removes files. For proper functionality, specific port on firewall needs to be allowed and program should be run with appropriate privileges.
- Running the application can cause following files creation/modification: output.pcap, output.txt, data.pcap, pcapfile.pcap and rule.cfg
- Application requires none installation

A.2 Command-Line Interface

To run the application standard way, the file main.py needs to be run using python language, as can be seen in the figure A.1 and the user will automatically be in the CLI mode of the application.

There are two modes in CLI of the application: standard mode and configuration mode. Configuration mode is primary designed for rules configuration and standard mode for everything else such as: starting the activity, stopping the activity and printing the output. Following commands can be issued in standard mode:

api print ..... Print API results



```
C:\Users\xmizic00\Desktop\python>python main.py
WELCOME
Initializing rules...
Rules loaded
>
```

Figure A.1: Run the application

api start	Start API
api stop	Stop API
api	test Start API testing
conf rules	Enter configuration mode
exit	Exit the application
file	Creates a pcap file from received data
help	Print help
load	Load a .pcap file
print results	Print results of a process
print rules	Print current rules
progress	Print progress
process	Start process
runt	Start TCP sniffing
runu	Start UDP sniffing
stopt	Stop TCP sniffing
stopu	Stop UDP sniffing

Following commands can be issued in configuration mode:

add	Add rule
clear	Clear rules
exit	Exit configuration mode
help	Print help
load	Load rules file
print	Print rules
save	Save current rules

A.3 Application Programming Interface

API of the application is not universal and is specially designed to use core of the application to for a dedicated purpose. To run this API, *api.py* file needs to be imported. API consist of four seamless function:

- start_ips_harvesting() - function will start the API process in separate thread
- get_and_clear_list_of_source_ips() - function returns list of next source IP addresses
- get_and_clear_list_of_destination_ips() - function returns list of next destination IP addresses
- stop() - function stops the API process
- get_data() - function returns database, which consist of ip addresses and their counts

B APNIC IP range

This appendix consists of list of IP addresses assigned to APNIC region. APNIC region might be seen in the Figure B.1 Regional Internet Registers. IP addresses assigned for that region are listed in the tables B.1, B.2, B.3, B.4, APNIC IP range part 1 - 4. Data are from [16].

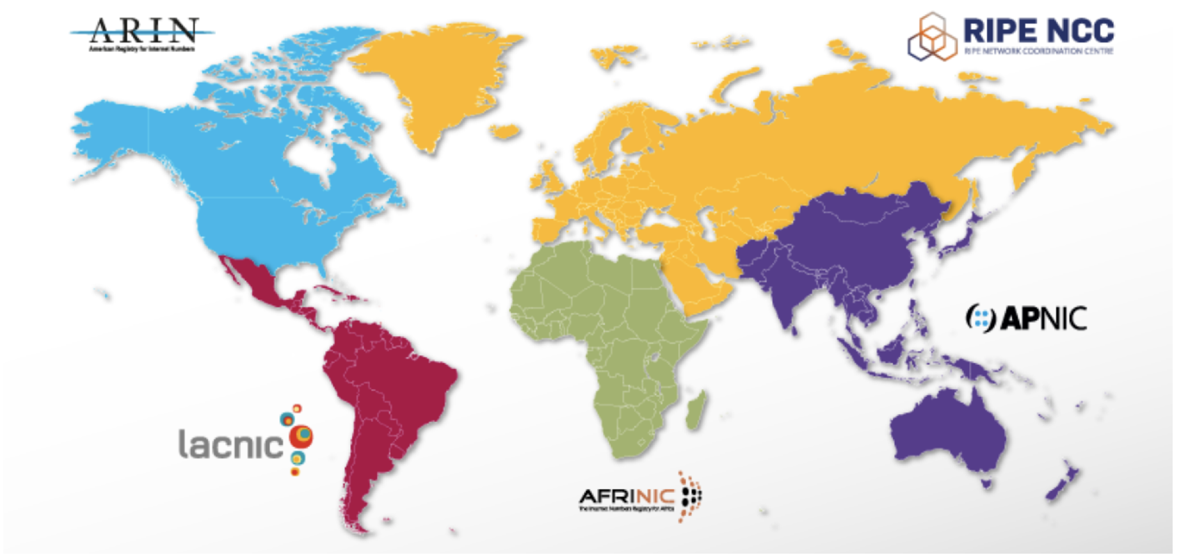

ARIN
American Registry for Internet Numbers
RIPE NCC
RIPE NETWORK COORDINATION CENTRE
APNIC
lacnic
AFRINIC

Figure B.1: Regional Internet Registries [17]

Table B.1: APNIC IP range part 1

Range	First IP address	Broadcast	Subnet mask
1.0.0.0/8	1.0.0.1	1.255.255.255	255.0.0.0
14.0.0.0/8	14.0.0.1	14.255.255.255	255.0.0.0
27.0.0.0/8	27.0.0.1	27.255.255.255	255.0.0.0
36.0.0.0/8	36.0.0.1	36.255.255.255	255.0.0.0
39.0.0.0/8	39.0.0.1	39.255.255.255	255.0.0.0
42.0.0.0/8	42.0.0.1	42.255.255.255	255.0.0.0
43.224.0.0/13	43.224.0.1	43.231.255.255	255.248.0.0
43.236.0.0/14	43.236.0.1	43.239.255.255	255.252.0.0
43.240.0.0/14	43.240.0.1	43.243.255.255	255.252.0.0
43.245.0.0/16	43.245.0.1	43.245.255.255	255.255.0.0

Table B.2: APNIC IP range part 2

Range	First IP address	Broadcast	Subnet mask
43.246.0.0/15	43.246.0.1	43.247.255.255	255.254.0.0
43.248.0.0/14	43.248.0.1	43.251.255.255	255.252.0.0
43.252.0.0/16	43.252.0.1	43.252.255.255	255.255.0.0
43.254.0.0/15	43.254.0.1	43.255.255.255	255.254.0.0
45.64.0.0/16	45.64.0.1	45.64.255.255	255.255.0.0
45.65.0.0/20	45.65.0.1	45.65.15.255	255.255.240.0
45.65.16.0/20	45.65.16.1	45.65.31.255	255.255.240.0
45.65.32.0/19	45.65.32.1	45.65.63.255	255.255.224.0
45.112.0.0/12	45.112.0.1	45.127.255.255	255.240.0.0
45.248.0.0/13	45.248.0.1	45.255.255.255	255.248.0.0
49.0.0.0/8	49.0.0.1	49.255.255.255	255.0.0.0
58.0.0.0/8	58.0.0.1	58.255.255.255	255.0.0.0
59.0.0.0/8	59.0.0.1	59.255.255.255	255.0.0.0
60.0.0.0/8	60.0.0.1	60.255.255.255	255.0.0.0
61.0.0.0/8	61.0.0.1	61.255.255.255	255.0.0.0
101.0.0.0/8	101.0.0.1	101.255.255.255	255.0.0.0
103.0.0.0/8	103.0.0.1	103.255.255.255	255.0.0.0
106.0.0.0/8	106.0.0.1	106.255.255.255	255.0.0.0
110.0.0.0/8	110.0.0.1	110.255.255.255	255.0.0.0
111.0.0.0/8	111.0.0.1	111.255.255.255	255.0.0.0
112.0.0.0/8	112.0.0.1	112.255.255.255	255.0.0.0
113.0.0.0/8	113.0.0.1	113.255.255.255	255.0.0.0
114.0.0.0/8	114.0.0.1	114.255.255.255	255.0.0.0
115.0.0.0/8	115.0.0.1	115.255.255.255	255.0.0.0
116.0.0.0/8	116.0.0.1	116.255.255.255	255.0.0.0
117.0.0.0/8	117.0.0.1	117.255.255.255	255.0.0.0
118.0.0.0/8	118.0.0.1	118.255.255.255	255.0.0.0
119.0.0.0/8	119.0.0.1	119.255.255.255	255.0.0.0
120.0.0.0/8	120.0.0.1	120.255.255.255	255.0.0.0
121.0.0.0/8	121.0.0.1	121.255.255.255	255.0.0.0
122.0.0.0/8	122.0.0.1	122.255.255.255	255.0.0.0
123.0.0.0/8	123.0.0.1	123.255.255.255	255.0.0.0
124.0.0.0/8	124.0.0.1	124.255.255.255	255.0.0.0
125.0.0.0/8	125.0.0.1	125.255.255.255	255.0.0.0
126.0.0.0/8	126.0.0.1	126.255.255.255	255.0.0.0
137.59.0.0/16	137.59.0.1	137.59.255.255	255.255.0.0
139.5.0.0/16	139.5.0.1	139.5.255.255	255.255.0.0
144.48.0.0/16	144.48.0.1	144.48.255.255	255.255.0.0
146.196.32.0/19	146.196.32.1	146.196.63.255	255.255.224.0
146.196.64.0/18	146.196.64.1	146.196.127.255	255.255.192.0
150.107.0.0/16	150.107.0.1	150.107.255.255	255.255.0.0
150.129.0.0/16	150.129.0.1	150.129.255.255	255.255.0.0
150.242.0.0/16	150.242.0.1	150.242.255.255	255.255.0.0
157.119.0.0/16	157.119.0.1	157.119.255.255	255.255.0.0

Table B.3: APNIC IP range part 3

Range	First IP address	Broadcast	Subnet mask
160.19.20.0/22	160.19.20.1	160.19.23.255	255.255.252.0
160.19.48.0/21	160.19.48.1	160.19.55.255	255.255.248.0
160.19.64.0/22	160.19.64.1	160.19.67.255	255.255.252.0
160.19.208.0/20	160.19.208.1	160.19.223.255	255.255.240.0
160.19.224.0/22	160.19.224.1	160.19.227.255	255.255.252.0
160.20.0.0/20	160.20.0.1	160.20.15.255	255.255.240.0
160.20.40.0/21	160.20.40.1	160.20.47.255	255.255.248.0
160.20.48.0/20	160.20.48.1	160.20.63.255	255.255.240.0
160.20.72.0/22	160.20.72.1	160.20.75.255	255.255.252.0
160.20.222.0/23	160.20.222.1	160.20.223.255	255.255.254.0
160.202.8.0/21	160.202.8.1	160.202.15.255	255.255.248.0
160.202.32.0/19	160.202.32.1	160.202.63.255	255.255.224.0
160.202.128.0/17	160.202.128.1	160.202.255.255	255.255.128.0
160.238.0.0/24	160.238.0.1	160.238.0.255	255.255.255.0
160.238.12.0/22	160.238.12.1	160.238.15.255	255.255.252.0
160.238.16.0/22	160.238.16.1	160.238.19.255	255.255.252.0
160.238.34.0/23	160.238.34.1	160.238.35.255	255.255.254.0
160.238.58.0/23	160.238.58.1	160.238.59.255	255.255.254.0
160.238.64.0/19	160.238.64.1	160.238.95.255	255.255.224.0
162.12.208.0/21	162.12.208.1	162.12.215.255	255.255.248.0
162.12.240.0/21	162.12.240.1	162.12.247.255	255.255.248.0
163.47.4.0/22	163.47.4.1	163.47.7.255	255.255.252.0
163.47.8.0/21	163.47.8.1	163.47.15.255	255.255.248.0
163.47.16.0/23	163.47.16.1	163.47.17.255	255.255.254.0
163.47.18.0/24	163.47.18.1	163.47.18.255	255.255.255.0
163.47.20.0/23	163.47.20.1	163.47.21.255	255.255.254.0
163.47.32.0/21	163.47.32.1	163.47.39.255	255.255.248.0
163.47.40.0/22	163.47.40.1	163.47.43.255	255.255.252.0
163.47.44.0/23	163.47.44.1	163.47.45.255	255.255.254.0
163.47.47.0/24	163.47.47.1	163.47.47.255	255.255.255.0
163.47.48.0/20	163.47.48.1	163.47.63.255	255.255.240.0
163.47.64.0/18	163.47.64.1	163.47.127.255	255.255.192.0
163.47.128.0/17	163.47.128.1	163.47.255.255	255.255.128.0
163.53.0.0/16	163.53.0.1	163.53.255.255	255.255.0.0
169.208.0.0/12	169.208.0.1	169.223.255.255	255.240.0.0

Table B.4: APNIC IP range part 4

Range	First IP address	Broadcast	Subnet mask
175.0.0.0/8	175.0.0.1	175.255.255.255	255.0.0.0
180.0.0.0/8	180.0.0.1	180.255.255.255	255.0.0.0
182.0.0.0/8	182.0.0.1	182.255.255.255	255.0.0.0
183.0.0.0/8	183.0.0.1	183.255.255.255	255.0.0.0
192.26.110.0/24	192.26.110.1	192.26.110.255	255.255.255.0
192.75.137.0/24	192.75.137.1	192.75.137.255	255.255.255.0
192.135.90.0/23	192.135.90.1	192.135.91.255	255.255.254.0
192.135.99.0/24	192.135.99.1	192.135.99.255	255.255.255.0
192.140.128.0/17	192.140.128.1	192.140.255.255	255.255.128.0
192.144.78.0/23	192.144.78.1	192.144.79.255	255.255.254.0
192.144.80.0/20	192.144.80.1	192.144.95.255	255.255.240.0
192.145.228.0/23	192.145.228.1	192.145.229.255	255.255.254.0
192.156.144.0/24	192.156.144.1	192.156.144.255	255.255.255.0
192.156.220.0/24	192.156.220.1	192.156.220.255	255.255.255.0
192.188.82.0/23	192.188.82.1	192.188.83.255	255.255.254.0
199.21.172.0/22	199.21.172.1	199.21.175.255	255.255.252.0
202.0.0.0/8	202.0.0.1	202.255.255.255	255.0.0.0
203.0.0.0/8	203.0.0.1	203.255.255.255	255.0.0.0
210.0.0.0/8	210.0.0.1	210.255.255.255	255.0.0.0
211.0.0.0/8	211.0.0.1	211.255.255.255	255.0.0.0
216.250.96.0/20	216.250.96.1	216.250.111.255	255.255.240.0
218.0.0.0/8	218.0.0.1	218.255.255.255	255.0.0.0
219.0.0.0/8	219.0.0.1	219.255.255.255	255.0.0.0
220.0.0.0/8	220.0.0.1	220.255.255.255	255.0.0.0
221.0.0.0/8	221.0.0.1	221.255.255.255	255.0.0.0
222.0.0.0/8	222.0.0.1	222.255.255.255	255.0.0.0
223.0.0.0/8	223.0.0.1	223.255.255.255	255.0.0.0

C List of files on attached DVD

Below is a list of files, those are on attached DVD:

- Thesis in Portable Document Format:
  - thesis.pdf
- Application source code files:
  - api.py
  - controller.py
  - file.py
  - help.py
  - main.py
  - menu.py
  - process.py
  - rules.py
  - tcp_sniff.py
  - test_api.py
  - udp_sniff.py
- Configuration files with rules:
  - rule.cfg, apnic.cfg, apnic2.cfg
- Folder with dpkt library:
  - dpkt
- Folders with output files from the API tests
  - api test 1, api test 2
- Small pcap file for testing purposes:
  - workfile.pcap
- Folders with results:
  - output 1, output 2