EFFICIENCY ANALYSIS OF PUBLIC WI-FI NETWORKS

GRADUATE QUALIFICATION WORK
SUSU–02.04.02.2018.308-578.GQW

Supervisor
Cand. Sci., Assoc. Prof.

__________ A.T. Latipova

Author,
the student of the group CE-219

__________ M.M.H. Al-Ani

Normative control

__________ O.N. Ivanova

“___”___________ 2018

Chelyabinsk–2018
MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION

Federal State Autonomous Educational Institution of High Professional Education
South Ural State University (National Research University)
School of Electrical Engineering and Computer Science
Department of System Programming

APPROVED

Head of the department,
Dr. Sci., Prof.

__________ L.B. Sokolinsky
“___”___________ 2018

TASK
of the master graduate qualification work
for the student of the group CE-219
Al-Ani Mustafa Majid Hamid
in master direction 02.04.02
“Fundamental Informatics and Information Technologies”
(master program “Database Technologies”)

1. **The topic** (approved by the order of the rector from 04.04.2018 No. 580)
   Efficiency Analysis of Public Wi-Fi Networks.

2. **The deadline for the completion of the work:** 01.06.2018.

3. **The source data for the work**
   3.1. Open Logic (2017) How to Use Filters with Wireshark,

   3.2. Thor, J. (2016) why you need a Network Analyzer,

   3.3 IT Cornell (2015) Browsers and Drive-by-downloads,
4. The list of the development issues
   4.1. To study features of the modern analyzers of protocols of a network;
   4.2. To study methods of operation Wireshark;
   4.3. To carry out installation of Wireshark on different platforms;
   4.4. To carry out search sniffers projects on GitHub;
   4.5. To gather statistical data about Wi-Fi by means of Wireshark
       (percentage of each protocol packets, time between the first packet and
       the second packet; total size of all the packages sent and the speed in
       which the packets are sent from/received by Wi-Fi in bits per sec).

5. Issuance date of the task: 09.02.2018.

Supervisor
Cand. Sci., Assoc. Prof. A.T. Latipova
The task is taken to perform M.M.H. Al-Ani
# TABLE OF CONTENTS

INTRODUCTION .......................................................................................................................... 5

1. USAGE OF WIRESHARK ......................................................................................................... 7
   1.1. Installation and Traffic Monitoring .................................................................................. 7
   1.2. Wireshark Installer for 64-bit Windows (or 32-bit) ......................................................... 9
   1.3. Wireshark Install on Mac OS X ....................................................................................... 11
   1.4. Installation of Wireshark on Ubuntu .............................................................................. 12
   1.5. Use of Wireshark ........................................................................................................... 12

2. PROJECTS ON GITHUB ESTABLISHING AND ANALYZING Wi-Fi ....................................... 16
   2.1. A project startup on GitHub .......................................................................................... 16
   2.1.1. Creation of a local repository ..................................................................................... 16
   2.1.2. Adding of files in a local repository .......................................................................... 17
   2.1.3. Creation of the version of the project ....................................................................... 19
   2.1.4. Creation of a repository on Github .......................................................................... 22
   2.2. Review of existing projects on github ......................................................................... 23

3. COLLECTING STATISTICAL DATA FOR USING WIRESHARK ABOUT YOUR OWN Wi-Fi CONNECTIONS .................................................................................................................. 26
   3.2. Analysis of network traffic ............................................................................................ 27
   3.3. Wireshark filters ........................................................................................................... 29
   3.3. Diagnostics of problems of a network ........................................................................... 35

4. SOFTWARE IMPLEMENTATION ............................................................................................ 38
   4.1. Use case diagram ......................................................................................................... 38
   4.2. Code description .......................................................................................................... 38
   4.2.1. Python code of main program .................................................................................... 39
   4.2.2. Launching Wireshark in python code ....................................................................... 42
   4.2.3. Initialization of variables in python code ................................................................... 43
   4.2.4. Output python code .................................................................................................. 49
   4.3. Description of algorithm ............................................................................................... 51
   4.4. Functional testing ........................................................................................................ 54

CONCLUSIONS .......................................................................................................................... 59

REFERENCES ............................................................................................................................ 60
INTRODUCTION

Topicality

From all methods of a study of computer networks the analysis of traffic the most laborious and labor-consuming. Intensive flows of the modern networks generate a lot of unsolved material in which it is not simple to find particles of the useful information. During the existence the stack of TCP/IP acquired numerous applications and additions, the score which goes on hundreds and thousands. These are application and official protocols, authentication protocols, tunneling, a network access and so on. Except knowledge of elements of network interactions, the researcher of a traffic needs to be guided freely in all this legal diversity and to be able to work with specific software tools - analyzers of traffic (protocols).

The functionality of the analyzer of traffic is not only a possibility of use of promiscuous of an operation mode of the network interface card for interception. Similar software shall be able to filter effectively traffic both at a collection stage, and during a study of separate units of transmission (frames, packets, segments, datagram, messages). And the more protocols analyzers of a traffic "knows", the better.

The modern analyzers of protocols of a lot of things are able: to read traffic statistics, to draw diagrams of the course of network interactions, to derive data of application protocols, to export results of operation to different formats. Therefore selection of tools for the analysis of network traffic is a topical issue. In this operation methods of the analysis of traffic by means of Wireshark program tools are provided.

Wireshark is a powerful network analyzer which can be used for the analysis of the traffic passing through the network interface of the computer. It can be necessary for detection and the solution of problems with a network, debugging of web applications, network programs or the websites.

Wireshark allows viewing completely packet contents at all levels, so you will be able to understand better how the network at a low level works.
To conduct researches of network applications and protocols and also to find problems in network functioning, and that is important, to clarify the reasons of these problems.

It is quite obvious that in order that it is the most effective to use traffic analyzers, the general knowledge and understanding of operation of networks and network protocols is necessary at least.

All packets are intercepted in real time and are provided in a format, convenient for reading. The program supports very powerful filtering system, illumination in color, and other features which will help to find the necessary packets.

In this thesis we considered options of use Wireshark for the analysis of traffic.

**Research goal:** analysis of network protocols.

**Operation purpose:** to carry out the analysis of efficiency of the Wi-Fi network.

**Objectives:**

1) to study features of the modern analyzers of protocols of a network;
2) to study methods of operation Wireshark;
3) to carry out installation of Wireshark on different platforms;
4) to carry out search sniffers projects on GitHub;
5) to gather statistical data about Wi-Fi by means of Wireshark (percentage of each protocol packets, time between the first packet and the second packet; total size of all the packages sent and the speed in which the packets are sent from/received by Wi-Fi in bits per sec).
1. USAGE OF WIRESHARK
1.1. Installation and Traffic Monitoring

Wireshark - the program for monitoring the Internet of traffic. It intercepts TCP packets which were accepted by the computer or are sent from it. It is possible to view contents of packets, to look for errors and so on. By means of Wireshark it is possible to pull out any file from packets and to view it.

Wireshark is rather known tool for capture and the analysis of network traffic, actually the standard both for education, and for a troubleshooting. [14].

Wireshark works with the vast majority of the known protocols has the clear and logical graphic interface on the basis of GTK+ and the most powerful system of filters. Cross-platform, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, and, naturally, works with Windows in such OS as Linux. Is distributed under the license GNU GPL v2. It is available free of charge on the website wireshark.org.

Advantages:
1) cross-platform (versions for Linux, Mac, Unix, Windows);
2) the utility is free;
3) possesses a wide functionality;
4) flexibility of setup;
5) possibility of traffic filtering;
6) creation of own filters;
7) interception of packets in real time [8].

Here main opportunities of the program:
1) capture of packets in real time from wire or any other type of network interfaces and also reading from the file;
2) such interfaces of capture are supported: Ethernet, IEEE 802.11, PPP, and local virtual interfaces;
2) packets can be sifted in a set of parameters by means of filters;
4) all known protocols are highlighted in the list in different colors, for example TCP, HTTP, FTP, DNS, ICMP and so on;
5) support of capture of traffic of VoIP of calls;
6) decryption of HTTPS of traffic in the presence of the certificate is supported;
7) decryption of WEP, WPA of traffic of wireless networks in the presence of a key and handshake;
8) display of statistics of load of a network;
9) viewing of contents of packets for all network layers;
10) time display of sending and receiving packets.

It is possible to download the latest version from the official site of the program: http://www.wireshark.org/(see fig.1).

![Fig. 1. The Order of downloading of the program](https://www.wireshark.org/images/download.jpg)
1.2. Wireshark Installer for 64-bit Windows (or 32-bit)

Installation in the Windows system is trivial — next, next, next.

Installation of the utility is standard and will not cause any problems even in beginners. In an installer everything is clear (see fig. 2-5), [2].

Fig. 2. Step 1

Fig. 3. Step 2
Fig. 4. Step 3

Fig. 5. Step 4
1.3. Wireshark Install on Mac OS X

1. Install XQuartz.

Get the installer from here: http://www.xquartz.org

2. Install Wireshark.

3. Do command on the Terminal.

Type following command on the Terminal and execute it.

```
/Applications/Wireshark.app/Contents/MacOS/Wireshark
```

Then you would see the answer below (see fig. 6).

![Answer in terminal](image)

Fig. 6. Answer in terminal

Keep wait for startup Wireshark. It will take a few seconds (see fig. 7).

![The program Interface](image)

Fig. 7. The program Interface
1.4. Installation of Wireshark on Ubuntu

To Linux Wireshark it is available in the form of a packet with the source code which can be downloaded at the following link:
https://1.na.dl.wireshark.org/src/wireshark-2.2.5.tar.bz2

It is one many more convenient to set already ready packet [2].

For this purpose it is enough to execute only 2 steps:
1) sudo add-apt-repository ppa:wireshark-dev/stable;
2) apt-get update;
3) apt-get install Wireshark.

1.5. Use of Wireshark

After start you will see the main window of the program (see fig. 8).

Fig. 8. The program interface

Come into the menu Captur> Options or click Ctrl+K for further setup of parameters of the program (see fig. 9).

Fig. 9. Setup of options
You will see the Capture Options window (see fig. 10)

![Capture Options Window](image1)

**Fig. 10. The section "Setup of Options"**

In the field of Interface select from the section capture from dropdown list the adapter via which there will be a capture of packets (see fig. 11).

![Capture Interface](image2)

**Fig. 11. A button to start capture of packets**

Connect your computer to LAN port of your device. Wait your computer so far will receive the IP address from the device (if the DHCP server is switched on), or set on the computer manually the IP address from the same subnet, as LAN a device IP address.

In the Wireshark program you will see all locked packets which are present at LAN port of the device.
You make capture of network packets so that it was possible to see information on a traffic in case of origin of a problem.

For completion of capture of packets press the Wireshark program toolbar button.

Come into the menu File> Save As for saving locked data in the file.

Select location; enter file name and press the Save button for saving packets.

For convenience of search/viewing of information on the necessary packets in the Wireshark program it is possible to filter locked packets on the IP address or port number.
We will give examples: If you want to make filtering locked packets on the IP address of assignment 88.255.67.145, in the field of Filter specify the rule of the `ip.dst` filter == 88.255.67.145. (see fig. 14)

![Fig. 14. Filtering locked packets](image)

If you want to make filtering locked packets on a certain port TCP (for example, on the 80th port), in the field of Filter specify the rule of the `tcp.port` filter == 80 (see fig. 15), [3].

![Fig. 15. Filtering locked packets to a certain port TCP](image)

If you want to make filtering locked packets on two certain IP addresses (for example, on IP addresses 88.255.67.145 and 117.85.53.223), in the field of Filter specify the rule of the `ip.addr` filter == 88.255.67.145 and `ip.addr` == 117.85.53.223. (see fig. 16).

![Fig. 16. Filtering locked packets on IP addresses](image)
2. PROJECTS ON GITHUB ESTABLISHING AND ANALYZING Wi-Fi

The website github.com is positioned as a web service of a hosting of projects with use of the control system of versions of git and also as a social network for developers. Users can create unlimited number of repositories, for each of which wiki, the issue tracking-á system is provided, and there is an opportunity to carry out code review and many other things.

GitHub at the moment is the most popular service such, having overtaken Source forge and Google Code [1].

For open-source of projects use of the website is free. If necessary to have private repositories, there is an opportunity to transfer to a paid tariff plan.

The Git system appeared as a control source texts in the Linux operating system and I won a great number of admirers in the environment of the Open Source.

2.1. A project startup on GitHub

The GitHub service provides a hosting (storage) of source texts both on paid, and on a free basis. It is one of the largest systems which users love Open Source. The main difference of the paid version is a possibility of creation of private repositories (storages) of source texts and if you have nothing to hide, then you can quietly use the free version.

2.1.1. Creation of a local repository

We will assume that your project is in folder/home/user/project. Before saving the source code, it is possible to look whether there are no temporal files in the folder with the project and whenever possible to delete them.

For viewing of the folder it is convenient to use the tree command which will show not only contents of each folder, but also a tree structure of directories.

Often temporal files contain specific suffixes on which it is easy to find them and in a consequence to delete. For search of such files it is possible to use the find command. As an example we will look how to find all files which are generated by the compiler Python and have extension of .pyc.
We pass into the folder with project/home/user/project:

```
Cd /home/user/project
```

Also we show the file list with extension of .pyc:

```
Find. -name *.pyc
```

This command will output the list of all files with extension of .pyc in the current directory and in its subdirectories. For deleting the found files, it is enough to add a key - delete to this command:

```
Find. - Name *.pyc - delete
```

It is very much recommended not to hurry and directly not to add this key. The first time to cause a command for viewing of files and having only convinced that nothing was included in the list of the useful to add a deleting key.

We will create a local repository in the folder with the project:

```
Gitinit
```

After execution of this command the new folder with the name .git will appear. In it there will be several files and. At the moment the version management system does not see our files yet.

### 2.1.2. Adding of files in a local repository

For adding of files the command is used:

```
Git add readme
```

After command execution, the readme file will be added to management system of versions (of course if it already was that it in the project). When adding the file the hash value which looks approximately so is generated:

```
9f2422325cef705b7682418d05a538d891bad5c8
```

We will create a local repository in the folder with the project of networks.
The added files are stored in the .git/objects/xx/y folder, at the same time the first 2 digits for specifying of a directory, and remaining a hash the value is file name. Our added file will be is here:

```
.git/objects/9f/2422325cef705b7682418d05a538d891bad5c8
```

That it is easy to see by means of a command:

```
ls .git/objects
```

The file is archive which is easy for unpacking and displaying, having specified complete value of a hash.

```
Git cat-file -p 9f2422325cef705b7682418d05a538d891bad5c8
```

To add all files from the current directory enter:

```
Git add
```

If it is necessary to add files from the current directory and from all subdirectories, then use:

```
Git add -all
```

In order that temporal files did not get to system, it is possible to enter them in the .git ignore file which needs to be created independently and to place in the root directory of the project (at the same level, as .git directory).

For example, if in to add the following line of *.pyc to the .git ignore file, then all files with extension of .pyc will not be added to a repository.

After adding of files, all changes are in so-called staging (or cached) area. This some temporal storage which is used for accumulation of changes and from which versions of projects (commit) are created actually.

For viewing of a current status it is possible to use a command:

```
Git status
```

After command execution we will see that in stage area there is our file of sniffa:
If you continue to make changes to the readme file, then after git status instruction fetch you will see two versions of the file.

To add new changes it is enough to repeat a command. git add command not only adds new files, but also all changes of files which were added earlier.

It is possible to cancel adding of the readme file staging area by command:

After command execution, the readme file marked not changed by system.

2.1.3. Creation of the version of the project

After we added the necessary files to staging area we can create the version of the project. By means of a command:

Each new version is followed by the comment.

After the commit, we will be able to find two new objects in repository.git.

We will look that inside:
The key - t shows an object type. As a result we see:

```
For the second object:
```

```
Git cat-file - t dac6721c3b75fcb3c9d87b18ba4cef2e15e0a3d3
```

Result:

```
Tree
```

```
For the very first file:
```

```
Git cat-file - t 9f2422325cef705b7682418d05a538d891bad5c8
```

We see:

```
blob
```

If we study further contents of these files, then we will find a tree structure. From each comment it is possible to follow the links according to all changed files. It is not really necessary for practical application, but perhaps it will be so easier to understand what happens by operation to the Git system.

The very first version cannot be cancelled. It can only be corrected. If you want to add change to the latest version, then after execution of the command of commit, you add necessary changes and you cause:

```
Git commit - m "comment" - amend
```

Or so:

```
Git commit - amend - no-edit
```

The key - no-edit is necessary not to enter again the comment.

It is possible to view changes which you made the last commit:
Or so:

```
Git show - name-only
```

The key - name-only is necessary to show only names changed files. Without it according to each changed file the list of all changes will be issued.

If you continued to work and changed only those files which were already added to system by the git add command, you can make commit one command:

```
Git commit -a -m "comment"
```

For viewing of the list of all comment, use a command:

```
Git log
```

Or so:

```
Git log - online
```

The key - online is necessary to reduce amount of information given out on the screen. With this key everyone commit is shown in one line. For example:

```
2b82e80 update
657ab4c first
```

To view changes on a specific commit, there is enough in the git show do bait command the hash value of the commit which can be received by means of the previous command.

```
Git show 657ab4c
```

For canceling of the last commit (except very first) it is possible to use the following command:

```
Git reset HEAD ~ 1
```

To delete all files in the folder which do not belong to the project and are not saved in a repository, it is possible to use a command.
2.1.4. Creation of a repository on Github

Till a present situation we worked with a local repository which remained in the folder on the computer. If we want to have a possibility of saving the project on the Internet, we will create a repository on Github. For a start it is necessary to register on the website github.com under the name of my user (in your case it can be any other name).

After registration we click a button "+" and we enter the name of a repository. We select the Public type (a repository always of Public for the free version) and we click Create [2].

As a result we created a repository on the website Github. On the screen we will see the instruction how to connect our local repository to again create. A part of commands is already familiar to us.

We add a remote repository (according to the SSH protocol) under the name of origin (instead of origin it is possible to use any other name).

```
git remote add origin git@github.com:myuser/project.git
```

We can view result of adding by means of a command:

```
Git remote - v
```

If everything was correctly made, then we will see:

```
Origin git@github.com:myuser/project.git (fetch)
Origin git@github.com:myuser/project.git (push)
```

To cancel registrations of a remote repository enter:

```
Git remote rm origin
```

It can be necessary if you want to change SSH access to HTTPS. After that it is possible to add it again, for example under the name of github and the HTTPS protocol.

```
Git remote add github https://github.com/myuser/project.git
```
The following command you will skip all changes which were made in a local repository on Github.

```
Git push -u github master
```

The key -u is used to establish connection between a remote repository of github and your branch of master. You can transfer all further changes to a remote repository the simplified command.

```
Git push
```

We considered basic scenarios of operation with the Git and Github systems. Each command given above has much more keys and according to opportunities.

**2.2. Review of existing projects on github**

In the github system there are not a lot of projects similar to the Wireshark program. In this paragraph several programs for the analysis and testing of the Wi-Fi networks are described.

    Wi-Fi Test Suite Introduction Wi-Fi Test Suite is a software platform originally developed by Wi-Fi Alliance, the global non-profit industry association that brings you Wi-Fi, to support certification program development and device certification. Non-proprietary components are provided under the ISC License and can be accessed at this open source project on GitHub. Wi-Fi Alliance members can access the full software package, including proprietary components, on the Wi-Fi Alliance member site.

    The Unified CAPI Console (UCC) provides the overall control console for the Wi-Fi Test Suite.

    UCC runs tests defined by input text files containing CAPI commands (CAPI specification). The individual CAPI commands within the input files are handled by UCC to perform functions such as DUT configuration, traffic stream definition, and test execution. UCC will direct specific CAPI commands to the appropriate device. This is accomplished via the Control Network.
Wi-Fi Test Suite 10.0.0-beta can be downloaded through the open source repository or by members in the most recent Wi-Fi Test Suite package.

Dependencies: Wi-Fi Test Suite requires python version 2.7 to be installed (see fig. 17), [6].

![GitHub repository](image)

Fig. 17. Wi-Fi Test Suite - 10.0.0-betas

Wi-Fi Test Suite Introduction Wi-Fi Test Suite is a software platform originally developed by Wi-Fi Alliance, the global non-profit industry association that brings you Wi-Fi®, to support certification program development and device certification. Non-proprietary components are provided under the ISC License and can be accessed at this open source project on GitHub. Wi-Fi Alliance members can access the full software package, including proprietary components, on the Wi-Fi Alliance member site.

Control Agents Control agents are a proxy in which a CAPI control command (CAPI specification) is converted for the device into the device’s native control interface. APs, DUTs, sniffers, and STAs may require control agents.
The Linux control agent can be downloaded through the open source repository.

Installation from sources Refer to the Install Guide for instructions on setting up a Linux DUT (see fig. 18), [3].

![Fig. 18. Wi-Fi Test Suite Linux Control Agent](image)

![Fig. 19. Wi-Fi Test Suite Linux Control Agent](image)
3. COLLECTING STATISTICAL DATA FOR USING WIRESHARK ABOUT YOUR OWN WI-FI CONNECTIONS

3.1. Data Using Wireshark

Wireshark is a powerful network analyzer which can be used for the analysis of the traffic passing through the network interface of your computer. It can be necessary for detection and the solution of problems with a network, debugging of web applications, network programs or the websites.

Wireshark allows viewing completely packet contents at all levels, so you will be able to understand better how the network at a low level works.

All packets are intercepted in real time and are provided in a format, convenient for reading. The program supports very powerful filtering system, illumination in color, and other features which will help to find the necessary packets. In this instruction we will consider how to use Wireshark for the analysis of traffic. Recently developers passed to operation over the second branch of the Wireshark 2.0 program, in it the set of changes and improving, especially for the interface was made. We will use it to apply.

Before passing to reviewing of methods of the analysis of traffic it is necessary to consider what possibilities the program supports in more detail with what protocols it can work and what to do. Here main opportunities of the program:

1) capture of packets in real time from wire or any other type of network interfaces and also reading from the file;

2) such interfaces of capture are supported: Ethernet, IEEE 802.11, PPP, and local virtual interfaces;

3) packets can be sifted in a set of parameters by means of filters;

4) all known protocols are high lit in the list in different flowers, for example TCP, HTTP, FTP, DNS, ICMP and so on;

5) Support of capture of a traffic of VoIP of calls;

6) decryption of HTTPS of a traffic in the presence of the certificate is supported;
6) decryption of WEP, WPA of traffic of wireless networks in the presence of a key and handshake [6];
7) display of statistics of load of a network;
8) viewing of contents of packets for all network layers;
9) time display of sending and receiving packets;
10) the primary window of the program is partitioned into three parts; the first column contains the list of network interfaces, available to the analysis, the second - options for opening of files, and the third - the help (see fig. 20).

![Fig. 20. The analyzer interface](image)

### 3.2. Analysis of network traffic

To start the analyses select the network interface, for example, of eth0 and press the Start button (see fig. 21).

![Fig. 21. The beginning of the analysis](image)
After that the following window, already with a flow of packets which pass through the interface will open. This window is partitioned into several parts too:

1) the upper part is a menu and a panel with different buttons;

2) the list of packets - is displayed further a flow of network packets which you will analyze;

3) packet contents - are located contents of the selected packet slightly below, it is broken on categories depending on the transport layer;

4) real representation - is at the very bottom displayed packet contents in a real look and also in the form of HEX [17].

It is possible to select any packet to analyze its contents (see fig. 22).

![Fig. 22. A packet choice for the analysis](image)

We see a request packet to DNS to receive the website IP address, in the request the domain goes, and in a response packet, we receive our question and the response.

For more convenient viewing it is possible to open a packet in a new window having executed double click of a mouse on record.
3.3. **Wireshark filters**

To touch packets manually to find necessary very inconveniently, especially in case of the active flow. Therefore, for such task it is better to use filters. For input of filters under the menu, there is a special line. You can click Expression to open the designer of filters, but there is a lot of them therefore, we will consider the most main:

1) ip.dst - the target IP address;
2) ip.src - the IP address of the sender;
3) ip.addr - ip of the sender or the receiver;
4) ip.proto - the protocol;
5) tcp.dstport - the port of destination;
6) tcp.srcport - port of the sender;
7) ip.ttl - the filter on ttl, defines network distance;
8) http.request_uri - the required address of the website.

For specifying of the relation between a field and value in the filter, it is possible to use such operators:

9) == - equally;
10) != - not equally;
11) < - it is less;
12) > - it is more;
13) <= - it is less or equally;
14) >= - it is more or equally;
15) matches - the regular expression;
16) contains - contains.

It is possible to apply to combining of several expressions:

1) && - both expressions shall be correct for a packet;
2) || - there can be correct one of expressions.

Now we will consider in more detail on examples several filters and we will try to consider all signs of the relations.
At first we will filter all packets sent on 194.67.215.125 (losst.ru). Collect a line in the field of the filter and click Apply [5]. For convenience wireshark filters can be saved, by means of the Save button (see fig. 23).

\[
ip.dst == 194.67.215.125
\]

Fig. 23. Filtering packets

To receive not only the sent packets but also received in reply from this node we will integrate two conditions (see fig. 24).

\[
ip.dst == 194.67.215.125 \mid \mid ip.src == 194.67.215.125
\]

Fig. 24. Filtering on a condition
We will select packets with ttl less than 10 further (see fig. 25).

\[ \text{ip.ttl} < 10 \]

Fig. 25. Selection on ip.ttl condition <10

Also we can select the transferred big files:

\[ \text{Http.content_length} > 5000 \]

Having filtered Content-Type, we can select all pictures which were loaded; we will make the analysis of traffic of wireshark, packets which contain the word image:

\[ \text{Http.content_type contains image} \]

To clean it is necessary to press the Clear button. Happens not always you know all information, necessary for filtering, and just you want to study a network. You can add any field of a packet as a column and look at its contents in the general window for each packet [4].
For example, we need to remove in the form of the column ttl (lifetime) of a packet. For this purpose open information on a packet, find this field in the section IP. Then open the shortcut menu and select the option applies as column: (see Fig. 26).

![Fig. 26. Operation with the option Apply As Column](image)

Further you will see the necessary column after up-dating (see fig. 27).

![Fig. 27. Result of filtering](image)
It is also possible to create the filter on the basis of any necessary field. Select the necessary field and open the shortcut menu, then click Apply as filter or Prepare as filter, then to select Selected to remove only the selected values or Not selected that to move away them (see fig. 28), [10].

![Fig. 28. Filter Options](image)

The specified field and its value will be applied or in the second case is added in the field of the filter (see fig. 29).

![Fig. 29. Operation of the filter](image)
In such a way you can add a field of any packet or a column to the filter. There too is this option in the shortcut menu. For filtering protocols you can use also simpler conditions. [11]

For example, we will make the analysis of traffic of Wireshark for the HTTP and DNS protocols:

```
Http | | dns
```

We will use the Wireshark tools to tracing of a certain session between the computer of the user and the server. For this purpose we will open the shortcut menu for a packet and we will select Follow TCP stream (see fig. 30).

![Fig. 30. Use of Follow TCP stream](image)

Then the window in which you will find all data transferred between the server and the client will open (see fig. 31).

![Fig. 31. The data transferred between the server and the client](image)
3.3. Diagnostics of problems of a network

For detection of problems on a network in the lower left corner of a window there is a round button, when clicking it the Expert Tools window opens. In it Wireshark collects all error messages and malfunctions on a network, (see fig. 32), [4].

![Fig. 32. The expert tools window](image)

The window is partitioned into such tabs as Errors, Warnings, Notices, and Chats. The program is able to filter and find a set of problems with a network and here you can see them very quickly. Here too wireshark filters are supported (see fig. 33), [7].

![Fig. 33. Detection of problems of a network](image)
The program successfully analyzes traffic. You can very just understand that users downloaded and what files they watched if connection was not ciphered. The program very well copes with extraction of content.

For this purpose at first it is necessary to stop capture of traffic by means of a red square on a panel. Then open the File menu-> Export Objects-> HTTP (see fig. 34).

![Fig. 34. Capture of a traffic](image1)

Further, in the opened window you will see all available intercepted objects. It is enough to you to export of them to file system. You can equally save both pictures, and music (see fig. 35), [15].

![Fig. 35. The review of locked files](image2)
Further you can make the analysis of network traffic of wireshark or directly open the received file other program, for example, a player.

In this chapter we considered how to use Wireshark 2 for the analysis of network traffic and also the solution of problems with a network. It is very powerful utility which has many functions.

All the statics below this line are made by wireshark’s “statistics” menu, so the first step for every statistic is to open wireshark and sniff, of open a pcap file in wireshark. Then go to the Statistics menu and choose the right option.
4. SOFTWARE IMPLEMENTATION

Before writing a code for my application a use case diagram was created.

4.1. Use case diagram

This Use case diagram shows seven use cases with two actors (Admin, User), these actors connected with this use case by relationship to show the capability for user in the system (see fig. 36).

![Use case diagram](image)

Fig. 36. Use case diagram

These use cases should be implemented in my code.

4.2. Code description

The main purpose of my code is to analyze a hotspot Wi-Fi Network. It analyzes the network interface live for 15 seconds and lets us know the percentage of each protocol packets and also information like time between the first packet and the second packet, the total size of all the packages sent and the speed in which the packets are sent from the system in bits per sec. It also sends a packet to www.google.com and prints the response in the output [16].
To execute this code, we have to install Python 2 along with the modules needed for this specific code.

We utilize the python libraries dpkt and socket. Only the dpkt need to be installed. This can be installed by executing the command – ‘pip install dpkt’ in the Windows Power Shell/Command Prompt.

This python code is executed in the Command Prompt/Windows using the command-

```
python<filename>.py
```

```
example- python execute.py
```

4.2.1. Python code of main program

**Listing 1. Code of main program**

```python
from __future__ import division
import dpkt #importing the dpkt package for parsing the pcapng file
import socket # importing socket package
import sys # importing sys module
import os # importing os module( to open wireshark in silent mode)

inter_ip = socket.gethostname()
inter_ip = socket.gethostbyname(inter_ip)

os.system("{tshark_loc} -i Wi-Fi -a duration:15 sniff_output.pcapng".format(tshark_loc=r'"C:\Program Files\Wireshark\tshark"')) #Starting wireshark in silent mode
f = open("sniff_output.pcapng", mode= "rb") #pass the file argument to the pcapng.Reader function (file)
pcap = dpkt.pcapng.Reader(f) # parsing the pcapng file using dpkt package (iterable)
size_sum = 0 #size_sum is the total size of the package (it is initialized to zero as there is no package in the beginning)
first = True # flag to see if it's the first package
(ts_first = 0 #first timestamp (integer)
(ts_last = 0 #last timestamp (integer)
(arp = 0
https = 0
http = 0
dns = 0
udp = 0
tcp = 0
els = 0
tot = 0
```
forts, buf in pcap:
    # loop for all the requests in the pcapng File (a
tuple of buffer and timestamp respectively)
eth = dpkt.ethernet.Ethernet(buf)  # converting it to eth object
ip = eth.data  # to read the source IP in src (IP object)
protocol=ip.data  # to read which protocol the packet is in

for tot+=1
if(type(ip)==type(dpkt.arp.ARP())):  # if packet is in ARP protocol
    arp+=1
elif(type(protocol)==type(dpkt.udp.UDP())):
    # if packet is in UDP protocol
    if(protocol.dport==53 or protocol.sport==53):
        # if packet is a dns service
dns+=1
    elif(protocol.dport==443 or protocol.sport==443):
        # if packet is a https service
https+=1
else:
    udp+=1
elif(type(protocol)==type(dpkt.tcp.TCP())):
    # if packet is in TCP protocol
    if(protocol.dport==443 or protocol.sport==443):
        # if packet is a https service
https+=1
    elif(protocol.dport==80 or protocol.sport==80):
        # if packet is a http service
http+=1
else:
tcp+=1
else:
els+=1
if
    (ip == dpkt.ip.IP and socket.inet_ntoa(ip.src) == inter_ip):  # we are
checking if the source of the package is the connection we want
    if first:
        # checking it it's the first package
        first =False  # (boolean)
    ts_first = ts  # assigning the first timestamp
try:
    size_sum += ip.len  # analyzing the total size of the packages by
    adding the size of individual package
except:pass
    ts_last=ts  # assigning the last timestamp
ts_diff = ts_last - ts_first # taking the timestamp difference to know the total time taken (integer)
speed = size_sum/ts_diff*1000/1024  # dividing the timestamp difference with the total size of the packages to get the speed (integer)
arps=float(arps/tot)*100
https=float(https/tot)*100
http=float(http/tot)*100
dns=float(dns/tot)*100
tcp=float(tcp/tot)*100
udp=float(udp/tot)*100
els=float(els/tot)*100
print ("Percentage of ARP:%.3f" %arp)
print ("Percentage of HTTPS:%.3f" %https)
print ("Percentage of HTTP:%.3f" %http)
print ("Percentage of DNS:%.3f" %dns)
print ("Percentage of other TCP:%.3f" %tcp)
print ("Percentage of other UDP:%.3f" %udp)
print ("Percentage of other:%.3f" %els)
print ("total size %.3f" %size_sum)
print("time diff %.3f" %ts_diff)
print("Kilobit per second %.3f" %speed)
f.close()  # closing the file which opened pcapng file
try:
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
# opening a socket connection
except socket.error:
    print("Failed to send and receive packets")

print("Socket Created")
host= "www.google.com"  # sending packets to google.com
port=80  # since it's an http request
try:
    remote_ip = socket.gethostbyname(host)  # reading host ip
except socket.gaierror:
    print("Hostname could not be resolve")

print("IP Address"+ remote_ip)
s.connect((remote_ip, port))  # connecting to the remote_ip through that port

print("Socket Connected to " + host + " using IP" + remote_ip)
message = "GET / HTTP/1.1\r\n\r\n"
try:
    s.sendall(message.encode())  # message is encoded and sent

except socket.error:
    print("Did not send packet successfully")
    print("Message sent successfully")
reply = s.recv(4096)  # message is received at port 4096
# print(reply.decode())  # the received message is decoded and printed
s.close()

4.2.2. Launching Wireshark in python code
Listing 2. Code to launch Wireshark

os.system("{tshark_loc} -i Wi-Fi -a duration:15
sniff_output.pcapng".format(tshark_loc=r'C:\Program
Files\Wireshark\tshark'))  # Starting wireshark in silent mode

The program reads the IP address of the computer it’s running from and
assigns it to inter_ip. Then it open’s Wireshark’s tshark application within the
code. This application records the network activity in the Wi-Fi network
interface for 15 seconds and saves the progress to a file named
sniff_output.pcapng. The tshark is opened using the python module ‘os’.

When the program’s executed, we see that tshark is opened within the
terminal (see fig. 37).

Fig. 37. Running main program

The program parses the pcapng file with the help of dpkt module. The
program opens the .pcapng file in ‘rb’ mode (Read mode as a binary file). The
variable pcap reads the contents using the function of dpkt module
dpkt.pcapng.reader() function. We can now iterate through the pcap object and
access each packet contained in the dump.
4.2.3. Initialization of variables in python code

1. size_sum=0

This variable gives us the total sum of all the packages sent from the source ip that we provided. It’s initialized to zero because at the start of the program, no packages are sent.

2. first = True

This variable acts like a flag to point out the first package processed. Once the first package (of the same source ip as the inter_ip) is processed, this variable changes to False.

3. ts_first = 0

This variable stores the timestamp of the first package processed. This variable is set to the timestamp of the first package. This is done along with making the variable first to False as mentioned before.

4. ts_last = 0

This variable shows the timestamp of the last package processed.

5. arp=0

This variable shows the number of packages of the ARP protocol.

6. https=0

This variable shows the packages of the https service which may be under TCP/UDP protocol.

7. http=0

This variable shows the packages of the http service which may be under TCP/UDP protocol.

8. udp=0

This variable shows the number of packages of the other UDP protocol except the HTTPS/DNS request.

9. dns=0

This variable shows the number of packages of the DNS service which is under UDP protocol.

10. tcp=0
This variable shows all the other packages which are not in https service or http service but under TCP protocol.

11. els=0

This variable shows all the other protocols which do not fall under the category of TCP or UDP or ARP protocol.

12. tot=0

This variable shows the total number of packages in this pcapng file.

After the initialisation of the above variable, the processing of the packages is done one by one using for loop. As mentioned above, the variable pcap contains the contents of the .pcapng file as an iterable of tuples containing two components – Timestamp and buffer.

Listing 3. Looping in pcap.

for ts, buf in pcap:
    eth = dpkt.ethernet.Ethernet(buf)
    ip = eth.data

As the pcap is an iterable object, the Timestamp and the buffer of each package is stored to the variable ts and buff respectively. It would be a lot more useful to parse the packet data into a more friendly, usable form. Using dpkt, we can simply pass a raw buffer to the appropriate dpkt class and have its contents automatically parsed and decoded into friendly python objects.

Passing the packet data to dpkt's Ethernet class will parse and decode it into the eth object. Since dpkt's Ethernet class also contains some extra magic to parse higher layer protocols that are recognized, we see that both the IP and TCP layer information has been decoded as well [8].

As we can see from the output, eth is the Ethernet object, eth.data is the IP object, and eth.data.data or ip.data is the TCP object.

Socket methods inet_aton and inet_pton. inet_pton converts a 32-bit packed IPv4 address (a string of four characters in length) to its standard dotted-quad string representation.
Listing 4. Analyzing packet’s type

```python
if(type(ip)==type(dpkt.arp.ARP())):
    arp+=1
elif(type(protocol)==type(dpkt.udp.UDP())):
    if(protocol.dport==53 or protocol.sport==53):
        dns+=1
    elif(protocol.dport==443 or protocol.sport==443):
        https+=1
    else:
        udp+=1
elif(type(protocol)==type(dpkt.tcp.TCP())):
    if(protocol.dport==443 or protocol.sport==443):
        https+=1
    elif(protocol.dport==80 or protocol.sport==80):
        http+=1
    else:
        tcp+=1
else:
    els+=1
```

We are parsing each package of the pcapng file one by one to check what protocol/service the particular package is in. As mentioned above, in the `ip.data (protocol)` contains the data about the protocol which the package is under.

We check the type of the object ip first to determine whether if it’s an ARP object or not, since the ARP object does not contain the `ip.data`. If it’s not an ARP object, it will check if it’s an UDP object or a TCP object or of any other protocol.

If it’s a UDP protocol, the type of the service is determined by the source port and destination port of the particular package. A DNS request listens or sends through the port 53, so we check if the package utilizes port 53 either as a source port or as a destination port, and if that’s the case we increment the dns variable since it’s a dns request. Similarly, we check whether the particular package of UDP protocol is an HTTPS request (port 443). If it does not belong to either of the service the udp variable is increment as it’s some other service in the UDP protocol.
Next, we check if the protocol object is a TCP object. As we mentioned above, we determine the type of the service by the source and destination port. We check for HTTP service (port 80) or HTTPS protocol (port 443) or any other TCP protocol service.

Hence, we determine the number of packets of each protocol/service.

In the next part we check for all the packages that were sent from a particular IP (inter_ip).

**Listing 5. Checking IP**

```python
if type(ip) == dpkt.ip.IP and socket.inet_ntoa(ip.src) == inter_ip:
    if first:
        first = False
        ts_first = ts
    try:
        size_sum += ip.len
    except:
        pass
    ts_last = ts
```

In the outer if condition, we check that if the `eth.data` or the IP object is equal to the dpkt.

When this condition turns out to True, the package is processed. The inner if checks for the first package and sets the variable `first` and `ts_first` to False and the corresponding `ts` (the timestamp of the first package). The `size_sum` is added to length of the `ip` object of the current object. Also, the `ts_last` is assigned to the `ts` of the current package.

After the program control is out of the loop, the following variables are defined:

1. `ts_diff = ts_last - ts_first`
   
   This variable contains the timestamp difference between the first package and the last package. This determines the time taken to process all the packages with the source ip as the given IP (inter_ip).

2. `Speed = size_sum/ts_diff`
   
   This variable contains the speed in which the packages were received in bits per second. This is determined by dividing the total size of the packages
(size_sum) by the time difference between the first and the last package (ts_diff).

Finally, we come to the section of the program which sends a packet to www.google.com and receives a response.

### Listing 6. Sending packets

```python
try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error:
    print("Failed to send and recieve packets")
print("Socket Created")
host= "www.google.com"
port=80             #since it's an http request
try:
    remote_ip = socket.gethostbyname(host)       #reading host ip
except socket.gaierror:
    print("Hostname could not be resolved")

print("IP Address"+ remote_ip)
s.connect((remote_ip, port))

print("Socket Connected to " + host + " using IP" + remote_ip)
message = "GET / HTTP/1.1\r\n\n"
try:
    s.sendall(message.encode())
except socket.error:
    print("Did not send packet successfully")

print("Message sent succesfully")
reply = s.recv(4096)
# print(reply.decode())
s.close()
```

A socket connection is obtained in the try segment using socket.socket() function and if it fails it prints the inactivity. In the coming lines we define the host and the port through which we are connecting.

In the next try segment, we get the IP address of the host (www.google.com) and assign to the variable `remote_ip`. A connection is made to this IP through the port in the `port` variable (80) using the function `s.connect`. 

47
We define the message to be sent in the variable *message*. In the next try segment we send this message after encoding it by using the function `s.sendall(message.encode)`. The reply is captured through the function `scrreech()` and assigned it to *reply*. The reply is captured through the port 4096. The reply is then decoded and printed in the output [12].

Interface python code we can analysis of Wi-Fi network from this interface automatically (see fig. 38).

![Interface analysis Wi-Fi networks](image)

**Fig. 38. Interface analysis Wi-Fi networks**

When I click on capture on Wi-Fi and analysis the request we get interface (see fig. 39).

![Interface capture Wi-Fi and analysis](image)

**Fig. 39. Interface capture Wi-Fi and analysis**
4.2.4. Output python code

Finally, we come to the part where it gives the output. After it open the code in sublime editor, and update the following things.

Interface num update it to your Wi-Fi interface number, you can check it by running in CMD the following lines (see fig. 40).

```
Cd "C:\Program Files\Wireshark"
```

```
tshark -D
```

![Image of Command Prompt showing Wireshark interface]

**Fig. 40. Results of python code**

Then update you IP in the code - you can get your Wi-Fi IP by typing "ipconfig" in CMD (see fig. 41).

![Image of Command Prompt showing Wi-Fi IP address]

**Fig. 41. Get IP address**
Finally resulting output of main program will be as follows (see fig. 42).

The output of the program prints:
1) percentage of the number of packages of the ARP protocol;
2) percentage of the number of packages of the HTTPS service (HTTPS);
3) percentage of the number of packages of the HTTP services (HTTP);
4) percentage of the number of packages of the DNS service (DNS);
5) percentage of the other packages which are not in https service or http service but under TCP protocol;
6) percentage of the other UDP protocol except the (HTTPS/DNS);
7) percentage of the other protocols which do not fall under the category of TCP or UDP or ARP protocol;
8) the total size of all the packages, It’s given by the variable size_sum;
9) the timestamp difference between the first and the last package. It’s given by the variable ts_diff as mentioned above;
10) the TCP speed or the network speed. It is given by the variable speed as mentioned above, the IP address of the host address;
11) the received message from www.google.com (decoded) After the output of the program we make sure that the file and the socket connection is closed by f.close() and s.close() respectively.

4.3. Description of algorithm

1. Start.
2. Read IP address of the computer to inter_ip.
3. Start tshark for 15 seconds and write to file name sniff_output.pcapng.
4. Open the file sniff_output.pcapng file in read mode.
5. Read the contents of the sniff_output.pcapng file to pcap.
6. Initialize size_sum = 0.
7. Initialize first = True.
8. Initialize ts_first = 0.
9. Initialize ts_last = 0.
10. Initialize arp=0.
11. Initialize https=0.
12. Initialize http=0.
13. Initialize dns=0.
14. Initialize udp=0.
15. Initialize tcp=0.
16. Initialize els=0.
17. Initialize tot=0.
18. For ts, buf in pcap, move to step 19, else go to step 40.
19. Read the ethernet object from buf to eth.
20. Read the ip object to ip.
21. Read the tcp/udp object to protocol.
22. Set tot=tot+1.
23. If type(ip)=type(arp object), go to step 24, else if type(protocol)==
type(udp object), go to step 25, else if type(protocol)==type(tcp object) step 29.
24. Set arp = arp + 1, go to step 34.
25. If protocol’s destination port=53 or protocol’s source port = 53, go to step 26, else if protocol’s destination port=443 or protocol’s source port = 443, go to step 27, else go to step 28.
   26. Set dns= dns +1, go to step 34.
   27. Set https = https + 1, go to step 34.
   28. Set udp=udp+ 1, go to step 34.
   29. If protocol’s destination port=443 or protocol’s source port = 443, go to step.
   30. else if protocol’s destination port=80 or protocol’s source port = 80.
   31. Set https = https +1, go to step 34.
   32. Set http = http + 1, go to step 34.
   33. Set tcp = tcp +1, go to step 34.
   34. Set els = els+1, go to step 34.
   35. If the ip’s source ip is equal to inter_ip, go to step 35.
   36. If first=True, go to step 36, else go to step 38.
   37. Set first=False.
   38. Set ts_first = ts.
   39. Set size_sum = size_sum + ip.len.
   40. Set ts_last = ts, go to step 18.
   41. Set ts_diff = ts_last – ts_first.
   42. Set speed = size_sum/ts_diff*1000/1024.
   43. Set arp=(arp/tot)*100.
   44. Set https=(https/tot)*100.
   45. Set http=(http/tot)*100.
   46. Set dns=(dns/tot)*100.
   47. Set tcp=(tcp/tot)*100.
   48. Set udp=(udp/tot)*100.
   49. els=(els/tot)*100.
   50. Print ("Percentage of ARP:%.3f" %arp).
   51. Print ("Percentage of HTTPS:%.3f" %https).
52. Print ("Percentage of HTTP:%.3f" %http).
53. Print ("Percentage of DNS:%.3f" %dns).
54. Print ("Percentage of other TCP:%.3f" %tcp).
55. Print ("Percentage of other UDP:%.3f" %udp).
56. Print ("Percentage of other:%.3f" %els).
57. Print ("total size %.3f" %size_sum).
58. Print ("time diff %.3f" %ts_diff).
59. Print ("Kilobit per second %.3f" %speed).
60. Close the pcapng file.
61. Try to open socket connection, else print ("Failed to send and receive packets") and STOP.
62. Print ("Socket Created").
63. Set host= "www.google.com".
64. Set port=80.
65. Try to read ip address of the host, else print and STOP.
66. Print ("IP Address"+ remote_ip).
67. Start a connection to remote_ip via port number = port.
68. Print ("Socket Connected to " + host + " using IP" + remote_ip).
69. Set message = "GET / HTTP/1.1\r\n\n".
70. Try to encode the message and send it to host, else print and STOP.
71. Print ("Message sent successfully").
72. Read the received message from port number 4096 to reply.
73. Decode the reply and print it.
74. Socket connection is closed.
75. Stop.
4.4. Functional testing

Functional testing is considered a kind of testing which investigates that each function of the software application operates in conformance with the requirement specification. This testing can mainly involve black box testing and it is not concerned about source code of the application.

Each system’s functionality can be tested by supplying appropriate input, checking the output and comparing the actual results with the expected results. This testing includes checking of the working of the program that is associated with the parsing of each network packet. The testing can be done either manually or using automation (see tab. 1), [8].

Table. 1. Functions and Testing Results.

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>Expected Result</th>
<th>Obtained Result</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Collect live information about packages sent and received.</td>
<td>Packages sent and received from the Network should be recorded for 10 seconds and written into a .cap file.</td>
<td>Packages sent and received from the Network is recorded for 10 seconds and written into a .cap file.</td>
<td>Passed</td>
</tr>
<tr>
<td>2</td>
<td>Sent a packet to any server and receive back a packet.</td>
<td>1 HTTP GET request should be made to <a href="http://www.google.com">www.google.com</a> and the response packet should be decoded and shown in the output.</td>
<td>1 HTTP GET request is made to <a href="http://www.google.com">www.google.com</a> and the response packet is decoded and shown in the output.</td>
<td>Passed</td>
</tr>
</tbody>
</table>
### Table 1

<table>
<thead>
<tr>
<th>No.</th>
<th>Function</th>
<th>Expected Result</th>
<th>Obtained Result</th>
<th>conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Obtain the total size of all the packages sent from the specified IP address.</td>
<td>The total size of all the packages sent from the specified IP should be in the output.</td>
<td>The total size of all the packages sent from the specified IP is in the output.</td>
<td>Passed</td>
</tr>
<tr>
<td>4</td>
<td>Obtain the time difference between the first and the last package of the cap file.</td>
<td>The timestamp difference between the first and the last package should be in the output.</td>
<td>The timestamp difference between the first and the last package is in the output.</td>
<td>Passed</td>
</tr>
<tr>
<td>5</td>
<td>Obtain the speed in which the packets are sent IP.</td>
<td>The speed in Bits/second should be displayed in the output.</td>
<td>The speed in Bits/second is displayed in the output.</td>
<td>Passed</td>
</tr>
<tr>
<td>6</td>
<td>Obtain the percentage of the number of packages with ARP protocol.</td>
<td>The percentage of the number of packages with ARP protocol should be obtained in the output.</td>
<td>The percentage of the number of packages with ARP protocol is obtained in the output.</td>
<td>Passed</td>
</tr>
<tr>
<td>No</td>
<td>Function</td>
<td>Expected Result</td>
<td>Obtained Result</td>
<td>Conclusion</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>7</td>
<td>Obtain the percentage of the number of packages with HTTPS service (HTTPS requests)</td>
<td>The percentage of the number of packages with HTTPS service /HTTPS requests should be obtained in the output.</td>
<td>The percentage of the number of packages with HTTPS service /HTTPS requests should be obtained in the output.</td>
<td>Passed</td>
</tr>
<tr>
<td>8</td>
<td>Obtain the percentage of the number of packages with HTTP service (HTTP requests)</td>
<td>The percentage of the number of packages with HTTP service /HTTP requests should be obtained in the output.</td>
<td>The percentage of the number of packages with HTTP service /HTTP requests should be obtained in the output.</td>
<td>Passed</td>
</tr>
<tr>
<td>9</td>
<td>Obtain the percentage of the number of packages with DNS service (DNS requests)</td>
<td>The percentage of the number of packages with DNS service /DNS requests should be obtained in the output.</td>
<td>The percentage of the number of packages with DNS service /DNS requests is obtained in the output</td>
<td>Passed</td>
</tr>
</tbody>
</table>
No | Function | Expected Result | Obtained Result | Conclusion |
---|----------|-----------------|-----------------|------------|
10 | Obtain the percentage of the number of packages of other UTP services which are not HTTPS requests or DNS requests. | The percentage of the number of packages of other UTP services which are not HTTPS requests or DNS requests should be obtained output. | The percentage of the number of packages of other UTP services which are not HTTPS requests or DNS requests is obtained in the output. | Passed |
11 | Obtain the percentage of the number of packages of other TCP services which are not HTTPS requests or HTTP requests. | The percentage of the number of packages of other TCP services which are not HTTPS requests or HTTP requests should be obtained output. | The percentage of the number of packages of other TCP services which are not HTTPS requests or HTTP requests is obtained in the output. | Passed |
12 | Obtain the percentage of number packages other protocols which do not under category TCP protocol or UDP | The percentage of the number of packages of other protocols which do not fall under the category of TCP protocol or UDP protocol should obtained. | The percentage of the number of packages of other protocols which do not fall under the category of TCP protocol or UDP protocol is obtained. | Passed |
During functional testing collected Wi-Fi, statistics were analyzed and the following conclusions were made:

1) at average, one of every 1000 packets is an ARP packet;
2) the network speed depends not only by the network connection established on user computer, but also the server network connection;
3) most of the network communication is over pure TCP/UDP protocols, without the 5-th layer;
4) in fixed number of packets, the UDP percentage will be much bigger than the TCP;
5) live stream or any other video streams in the network are sent plain-text over udp, because udp is much faster;
6) the network speed is in direct proportion to the physical distance between the ips.

Analysis of Wi-Fi networks using the mobile, we connect the Wi-Fi network to the phone with the computer and then we analyze Wi-Fi network (see fig. 44).

Fig. 44. Connect Wi-Fi using mobile
CONCLUSIONS

Every day internet effects more and more on our lives, we use it for any little thing we do, from reading news and to programming at work. That is why the goal of my research is to create analysis software for one of the most popular network connection types – the Wi-Fi. Nowadays almost everyone has a Wi-Fi adapter, and Wi-Fi is more popular than other connections like Ethernet.

So a python program was developed which collects various statistics about current Wi-Fi connection: number of UDP/TCP packets, types of packets, network speeds according to various IPs, etc. Python Language was chosen for coding, because it is one of the most popular languages for network connection establishing and analyzing. Also open source python libraries were used (scapy, dpkt and requests).

The developed program takes in the name of the pcapng file that is to be analyzed and an IP address. The program analyzes the package one by one and extracts the data like percentage of the number of different protocols/services like ARP, HTTPS, UDP, TCP etc. It also gives the total size of all the packages and the time stamp difference between first and the last package of the pcapng file, which is essentially the time difference between the first and the last package. The output gives us the rate at which the packages are sent in bits per second. Finally, it sends a message to www.google.com whose reply is also shown in the output.

After analysis of collected Wi-Fi statistics, the following conclusions were made:

- at average, one of every 1000 packets is an ARP packet;
- the network speed depends not only by the network connection established on user computer, but also the server network connection;
- most of the network communication is over pure TCP/UDP protocols, without the 5-th layer;
- in fixed number of packets, the UDP percentage will be much bigger than the TCP;
- live stream or any other video stream in the network is sent plain-text over udp, because udp is much faster; the network speed is in direct proportion to the physical distance between the ips.
REFERENCES


