American Journal of Applied Sciences

Original Research Paper

Embedded Software Platform Used for Detection and Mitigation of a MAC Anomaly in IEEE 802.11n Networks

[1]Argemiro Bevilacqua, [2]Vitor Chaves de Oliveira,
[2]Gunnar Bedicks Jr., [1]Alexandre de Assis Mota and [1]Lia Toledo Moreira Mota

[1]*Department of CEATEC, Electrical Engineering Faculty,*
Pontifical Catholic University of Campinas, Campinas/SP, CEP 13086-900, Brazil
[2]*Electrical Engineering Faculty,*
Mackenzie Presbyterian University, São Paulo/SP, CEP 01302-907, Brazil

Article history
Received: 14-04-2015
Revised: 18-09-2015
Accepted: 05-10-2015

Corresponding Author:
Argemiro Bevilacqua
Department of CEATEC,
Electrical Engineering Faculty,
Pontifical Catholic University of Campinas, Campinas/SP, CEP 13086-900, Brazil
Email:
bevilacqua.argemiro@gmail.com

Abstract: IEEE 802.11 standard is largely used anywhere as a cheap way to access internet, but the majority of devices used does not provide a standard way to manage them. Mobile stations competing for access point may causes a general wireless network failure, known as the "MAC Anomaly". It is presented here an access point with modified firmware, that monitors wireless conections´s quality and has a programmable capability allowing to do network management in a standardized manner, using SNMP for example. In this article, embedded Linux "OpenWRT" installed in a very cheap and reduced size hardware was used. By means of Bourne shell script programming, it is possible to collect all important operating parameters and data of the access point. From this, its possible to gain considerable control over it. IEEE 802.11 (Wi-Fi) access point devices are routers, while embedded Linux has routing capabilities. Thus, it is easy to implement traffic policies by means of Bourne shell script programming. Traffic shapping is one of the acces point´s capabilities successfuly tested and demonstrated in this study. MAC anomaly detection in IEEE 802.11 networks can be easily implemented by means of scripts as well. It was collected and plotted network throughput data, becoming possible to observe MAC anomaly in visual charts. A matemathic model to apply on the collected data of network throughput is under study, aiming to identify the anomaly through calculations. The object of the current study is to integrate everything: Measurements of operation data (network throughput in Mbit/s), identification of the MAC anomaly and its immediate mitigation. The result was the conception of an integrated device to measure, identify and mitigate MAC anomaly in a test field and therefore restore network throughput by representing a gain of 42.5216%.

Keywords: IEEE 802.11, Network Management, Embedded Linux, Shell Programming, MAC Anomaly

Introduction

The IEEE 802.11b MAC Anomaly was first introduced by (Heusse *et al*., 2003). It was demonstrated by means of calculations and simulations of the access methods contained in the MAC layer. The MAC anomaly in IEEE 802.11 networks is explained in section 2. Heusse *et al*. (2003) gave origins to several other works all around the world. Our objects of study are the works regarding workbench setups to measure, control and mitigate this anomaly.

Branquinho *et al*. (2006) successfully identified and mitigated the MAC anomaly using Signal to Noise Rate (SNR) measures in a controlled workbench environment.

Guirardello (2008) demonstrated the possible use of QoS, Quality of Service (Oliveira *et al*., 2015), to mitigate a MAC Anomaly. An experimental workbench with modified access point firmware in a controled radio environment, with two Wi-Fi client stations connected were used.

Peris (2012) used an embedded Linux distribution running in the access point device (DD-WRT Project, 2015). It was applied bandwidth management, a resource found in DD-WRT, to mitigate a MAC Anomaly, but no method to identify it was showed. Beyond the access point device, two other wired servers were used. All operations were done manually.

Marques (2013) demonstrated a mathematic model to identify the MAC Anomaly, based on the station´s and access point´s throughput measures. For the first time, a common and accessible setup was used. This enabled a moving of the workbench from the lab to a trial field.

All the works listed above with their positive results, showed opportunities to continue studying the MAC anomaly mitigation. In this article, new equipment and methods are used to have an auto-managed access point that performs measurement, identification and mitigation of MAC anomaly automatically. Thus, a wireless network free of anomaly, in a real world environment will be possible. The workbench model in (Marques, 2013) was chosen as a starting point. Several improvements were added to build a new, cheap, accessible and manageable access point that now is being used in a trial field environment. Table 1 shows the main differences between both systems: The workbench used in this article and the workbench used by (Marques, 2013).

The main contributions of this work goes far beyond the basic system improvements, such as new hardware and the operating system as described in section 4. As in (Marques, 2013), it was also used a Linux platform, introduced with details in section 3. Similar scripts to the ones in (Marques, 2013) were also developed in order to collect connection´s data of the client stations. Data such as throughput in Mbits/s and Radio Signal Strength Indicator (RSSI) in dBm can be collected from the client stations. Data collection is described in detail in section 4.2, 4.3 and 4.4.

After checking the connection quality (Oliveira *et al.*, 2013a; 2013b), it becomes possible to perform a test to see if a given station is an offender or not (just in case the MAC Anomaly is installed). A new approach using a new mathematical calculation is being studied. "Lua", a high level scripting language running on the new platform will be used. Details are described in section 4.5.

A Bourne Shell script was successfuly tested to perform offenders´s bad effect mitigation using embedded Linux traffic control resources. To perform this, a filter was applied over the overall traffic, in order to distinguish offending station traffic from the normal one. Afterwards, to reduce the bandwidth of the offending station, "traffic shaping" was applied over the offending traffic, in order to mitigate the anomaly. Acting this way, it is possible to recover the wirelles network normal operation. "Traffic shapping" and scripts are explained in details in section 4.6.

Measurements were taken using up to five client stations connected to an access point. We visually observed up to two simultaneous offenders in the graphics plotted, as shown in section "5.1".

The availability of a low-cost platform, affordable and portable; as well as the achievement of the positive results using it, made possible to proceed with the development of a unique script to integrate:

- Measurements of client stations´s throughput
- IEEE 802.11 MAC Anomaly detection
- Application of "traffic shapping" over the offending traffic to mitigate the MAC Anomaly restoring in this way, the wireless network integrity

Table 1. Comparison of two Linux based access points

Workbench used in this article	Workbench of (Marques, 2013)
Reduced size hardware, composed of only one piece	Composed of a tower type CPU, a monitor, a keyboard and a mouse.
It costs only US$ 30,00, easy to find.	It costs about US$ 300,00, easy to find.
Embedded Linux with reduced functionality. Contains only the necessary applications to operate as an Wi-Fi access point.	Has a complete Linux installation with thousands of applications.
Wi-Fi access point is a native feature	Wi-Fi access point is configurable by means of a wireless network adapter and the "ap" software package.
Supports SNMP	Supports SNMP
Energy efficient	High energy consumption
4Mb of Flash memory and 32Mb of RAM	4Gb of RAM and 250Gb of hard disk
CPU clock of 400MHz	CPU clock of 2.4GHz
Open Source and Freeware Software	Open Source and Freeware Software
Has routing capabilities	Has routing capabilities
Very simple	A Complete Linux Workstation
Can be used in a trial field	Can be used in a trial field
Accessible, open documentation	Accessible, open documentation
Routing policies and traffic shaping can be configurable	Routing policies and traffic shaping can be configurable
IEEE 802.11n standard with MIMO	802.11b standard
Up to 300Mbps transfer rate	Up to 11Mbps transfer rate
Reaches up to 100 m	Reaches up to 70 m
Languages C, Lua and Shel	Languages C, Lua and Shel and many others

Discussions and conclusion are in section 6, references are listed in section 7.

IEEE 802.11 MAC Anomaly

As explained by (Heusse *et al*., 2003), the protocol that governs the MAC is fair to the client stations, giving them an equal oportunity to transmit their frames. The anomaly is installed when one of the connected stations has a poor connection quality, having to retransmit their frames frequently, or having a much lower transmission rate than the others. Surely, when its turn to transmit is reached, it will take longer than the others, forcing all stations to wait, including the access point. When the media is available for a normal station to transmit, it quickly transmits only one frame and starts waiting for the next opportunity. Due to the excessive delays in the media availability caused by the offending station, the well connected stations have their throughput decreased. This scenario causes a general failure in the network that is verified by a serious decrease in the total throughput. One can observe that the total network throughput is leveled beneath, having similar values to the offending station.

IEEE 802.11n standard, a very recent generation of wireless network, added several powerful technologies to the physical layer in order to enhance signal coverage and increase the transmission rate. Nonetheless, no changes were made to the core of the MAC layer and the MAC Anomaly can still be observed (Zhang *et al*., 2008; Shrivastava *et al*., 2008). Recent literature reviews is found in (Lopez-Aguilera *et al*., 2010) and (Cheng *et al*., 2007). Embedded Linux is used frequently to study IEEE 802.11 standard behavior: A study of link layer´s MAC is found in (Singh and Sohi, 2008) and a study of energy eficiency is found in (Biazotto *et al*., 2012).

Embedded Linux for Wi-Fi Routers

As described in (Hallinan, 2006), embedded Linux systems are associated with several key attributes. For example, if the system passes by an electric power outage, it can auto restart its operations without human intervention, acting as an autonomous system. User interface frequently consists of only a serial port and some LEDs.

Embedded systems are not necessarily of small sizes. They can extend to several racks in size, as storage systems does. Dedicated embedded Linux systems are largely used to control data storage. They are called Linux appliances. Embedded Linux can be found in reduced size hardware as well. A good example is a smarphone operated by “Android”. Some common characteristics found in embedded systems are:

- Is of limited resources: Small memory and no hard drive for example
- Has a very simple, or no human interface
- Contains a general-purpose microprocessor
- Applications are built-in, not installed by end-users
- Is presented ready to use, with all hardware and software preinstalled
- Human intervention is not supposed to occur, normally are autonomous systems
- Especially designed for a specific purpose
- Is not used for general purpose, is not a common computer
- Low power consuming or is powered by batteries

The hypothetical access point in Fig. 1, contains onboard:

- A 32-bit RISC processor as its core component. Often, the processor performs many other functions beyond the traditional CPU. This example contains an integrated UART for a serial interface, an integrated USB and Ethernet controllers
- Nonvolatile program and data are stored in flash memory. Main memory is Synchronous Dynamic Random-Access Memory (SDRAM) and might contain from few to hundreds of Megabytes, depending on the application
- A real-time clock module, that keeps the time of day preserved by a battery
- This example includes an Ethernet and USB interface, as well as a serial port for console access via RS-232. The 802.11 chipset implements the wireless modem function

The above text about embedded Linux and the Fig. 1, were adapted from (Hallinan, 2006), page 32 “2.2 Anatomy of an Embedded System”.

It is known that Linux is widely used in embedded systems and there has been an upward trend in their use (Palazzi *et al*., 2010). Besides not requiring user licenses, Linux has powerful programming features (Ingle and Daryapurkar, 2013).

OpenWRT is an embedded Linux distribution, oriented for network support (OpenWRT Project, 2015) and easy to configure. Besides de aforementioned features, OpenWRT was chosen for this work, because of the following:

- Native support for programming languages, such as “Lua”, Bourne Shell and C
- Native support for traffic control with bandwidth management
- “iw” command line utility to collect wireless connection data such as: Data transfer rate, Received Signal Strength Indicator (RSSI), Signal to Noise Rate (SNR) and many other data
- Has an optional software package to deploy a SNMP agent

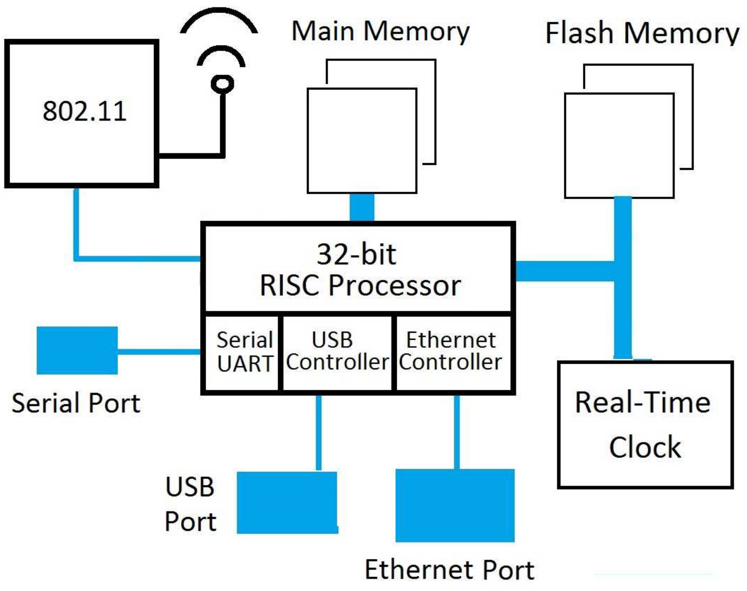
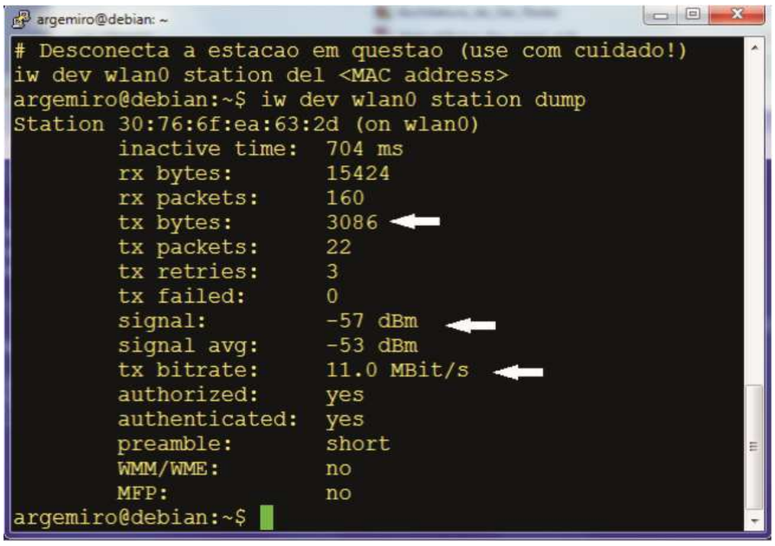


Fig. 1. Representation of an ideal access point



```
# Desconecta a estacao em questao (use com cuidado!)
iw dev wlan0 station del <MAC address>
argemiro@debian:~$ iw dev wlan0 station dump
Station 30:76:6f:ea:63:2d (on wlan0)
        inactive time:  704 ms
        rx bytes:       15424
        rx packets:     160
        tx bytes:       3086
        tx packets:     22
        tx retries:     3
        tx failed:      0
        signal:         -57 dBm
        signal avg:     -53 dBm
        tx bitrate:     11.0 MBit/s
        authorized:     yes
        authenticated:  yes
        preamble:       short
        WMM/WME:        no
        MFP:            no
argemiro@debian:~$
```

Fig. 2. "iw: Utility command showing data collected

Why to Change the Original Firmware

A modified firmware was adopted in this study, instead of using a static, closed and proprietary firmware running on the access point. The modified firmware adoption will enable many features, as described in (OpenWRT Project, 2015):

- A complete file system to be freely used in read or write modes
- Software package management. As in Linux, packet management frees user from application selection and configuration
- Hundreds of software packets are available making possible a system customization to support any kind of application
- For developers, OpenWrt provides a framework to build applications having operating system support and tools, like Linux does

OpenWRT and DD-WRT

As well as OpenWRT, DD-WRT (DD-WRT Project, 2015) is an alternative, Open Source Linux embedded firmware that supports a huge list of wirelles access points, from a hundred of vendors. DD-WRT also provides bandwidth management, as used by (Peris, 2012) for MAC anomaly mitigation.

OpenWRT as a Router

OpenWRT was designed specially to work as a router: It operates the wireless access point, routing the traffic originated from the wireless client stations to the wired network. The wired network in it´s turn routes to the internet.

A router has total control over the incoming and outgoing traffic. It has buffers, where the datagrams are enqueued before a decision must be taken on what destination port the datagrams will follow. A routing table is checked to guide the decisions.

In addition to make routing, other kind of processing can be applied over the incoming or outgoing traffic. Information contained in the TCP/IP headers, can be used to identify the type of traffic. Based on this kind of information, the packets of a given type can pass by a distinguished processing.

Traffic control, that is easily configured on OpenWRT (Hubert, 2015; Lin and Hu, 2003; Vila-Carbo *et al.*, 2008; Cano and Francesco, 2015), is done using the following named objects: “filters”, “queues”, “qdiscs”, “policies” and “classes”. “Classes” are sets of “queues”.

“Filters” are applyed over the packets enqueued, based on the information contained in the packet´s headers in order to select a given type of traffic. After classifying traffic by means of a “filter”, their packets are stored into separated buffers called “qdiscs” or queue disciplines.

“Policies” are applyed over “qdiscs”, becoming possible for example:

- To reschedule or drop undesired packets
- To chose what port to deliver the packets
- Apply delays in order to reduce the throughput of a given type of traffic
- Define priority “queues” such as voice or video streaming

OpenWRT as a Programming Environment

What makes OpenWRT a good choice for data data colection, network analysis and traffic control, is it´s support for programming. OpenWRT comes with two scripting languages always pre-installed, because they are part of the system itself:

- Busybox (OpenWRT Project, 2015). A set of Linux utility programs, modified to reduce their sizes, in order enable them to run in embedded devices. Obviously, they have reduced funcionality, but have the basics necessary to run well. Busybox also have a command line shell, very similar to Bourne Shell, useful for programming
- Lua (Ierusalimschy, 2015) is a high-level and complete script programming language, with numerous features. Lua was used to build part of the OpenWRT itself and at the same time was made available to user environment
- Besides these two languages, “LuCI” (OpenWRT Project, 2015) is available as well. LuCI is a programming environment used to build WEB pages for embedded systems. The OpenWRT administrative WEB interface was built using “Lua” and “LuCI”

“C” language can be used also, but it is necessary to “Cross-compile” the code. The aforementioned languages are the most used ones. Other languages can be installed, but will require a mid-range hardware.

OpenWRT SNMP Capabilities

OpenWRT has an optional SNMP (Stallings, 1999) software package that can be easily installed and configured. SNMP on OpenWRT provides a framework to deploy remote management by means of very simple scripts. SNMP is a standard largely used for remote management.

Materials and Methods

It is necessary to measure throughput levels on all the connected stations, in order to decide if MAC Anomaly is present or not. If MAC Anomaly is present, throughput levels is forced to decrease on the offending stations, aiming to restore the normal network operation. Traffic shaping technique is used to force a decrease in throughput of the offender stations (Hubert, 2015). The lower is the throughput, the lower will be the interference caused by the offender station on the overall network performance. Thus, the normal network operation is restored, because the total network throughtput returns to the normal level.

The embedded Linux, with programming capability, along with its native traffic control utilities, installed in a reduced size hardware, motivated the choice for this platform. All the three following processes, running in this order, were implemented to run automatically as an integrated program forming an autonomous system:

- Bourne shell scripts collect throughput data from all connected stations (Mbits/s) and save to disk files
- Run manually the method described in section 4.5 to check if MAC anomaly is present or not. If MAC anomaly is present, identify the offender statation. A plan for near future is to program a Lua script that reads throughput data from file to perform MAC Anomaly detection and identification of the offender stations. SciLab

simulation software was used succesfully to validade the future Lua scripts´s logic

- If MAC anomaly is present, a Bourne shell script is called passing the hardware address of the offending station as argument. Mitigation is then started. Traffic shaping technique described in section 4.6 is used to force a decrease of throughtput in the offending stations

Process 3 is still being started by hand due to the lack of phase 2 that is being performed off-line by graphics visual inspections only. Lua scripts are not running yet. This setup will be used in real world, not at the laboratory. Being of reduced size and wireless, it can be moved to anywhere to perform wireless signal quality analysis.

All of this work is performed above the link layer. MAC sub-layer of link layer, remains untouched in this study. Only network data throughput was measured.

If MAC anomaly is present, the method described in section 4.5 proved to be 100% efficient to detect it. Anomaly can be evoked by placing one connected station far away from the access point, or by setting its transmission rate by hand to a lower value than others. Attenuation of radio signal due to high distance from the access point automatically causes the station´s data transfer rate to decrease. A transfer rate lower than normal is enough to cause an anomaly.

Hardware

There are hundreds of vendors and devices listed in (OpenWRT Project, 2015) hardware compatibility list. Some of them come with an USB interface, where it is possible to connect a pen drive or even an external hard disk. The one chosen for this work has the following characteristics:

- Two antennas, MIMO technology
- Maximum throughput of up to 300Mbps
- Model TL-WR841ND
- 4Mb of Flash Memory, 32Mb of RAM
- Atheros AR7241 main processor, clock of 400MHz
- Atheros AR9287 onboard network interface
- OpenWRT firmware

The Utility Command "iw"

"iw" is a very useful program made available by OpenWRT, that was used in this study to take measures on the wireless connections, as shown by Fig. 2:

Iperf Network Performance Tool

"Iperf" was installed on OpenWRT access point as a daemon using the "-s"option. On the client stations´s side, it was run in client mode, using the "-c" option. A traffic simulation is necessary before measuring the network´s throughtput.

Scripts Used to Perform Measurements

"BusyBox" shell programming (Bourne Shell alike with reduced functionality) and "iw" command were used to write the scripts that measures network throughput of each connected stations (Mbits/s). The scripts are simple and self explanatory, having ("#") commented lines inside to explain their logic. Only Mbits/s measurements were taken, but many others can be implemented in similiar way these scripts do, as shown by Fig. 3-5.

MAC Anomaly Detection Method

Throughput data files collected by these measures is used to detect the presence of "MAC Anomaly", identifying the offending station as well. Throughput data files are loaded manually in SciLab. Curves of each connected station are plotted. The method relays on the similarity between two curves: If a station´s curve is similar to the network´s total throughtput curve, then the station is not an offender; otherwise it is the offender station and is causing the MAC anomaly. The offending station has a particularity: Comparing it´s throughput curve with the network´s total throughput curve, a very clear opposite image is observed. Only visual inspections of the graphs is used as a MAC Anomaly identification method.

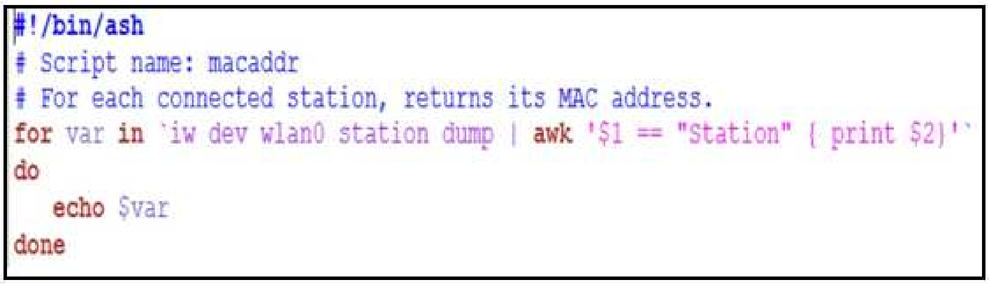
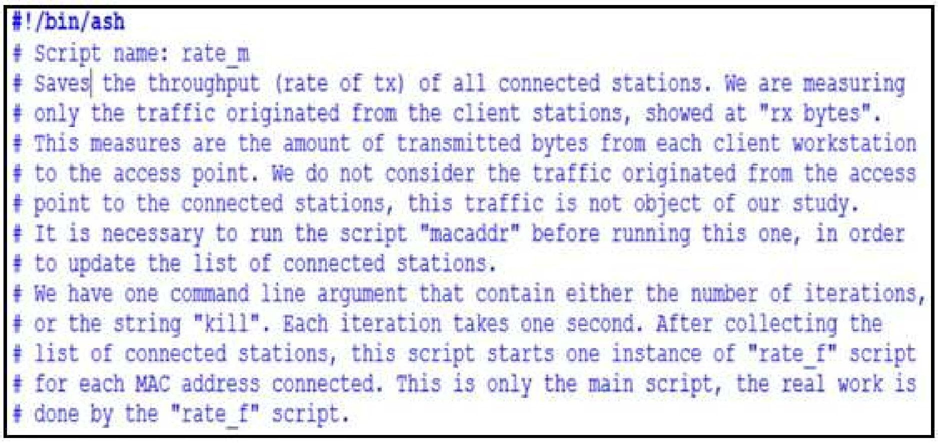
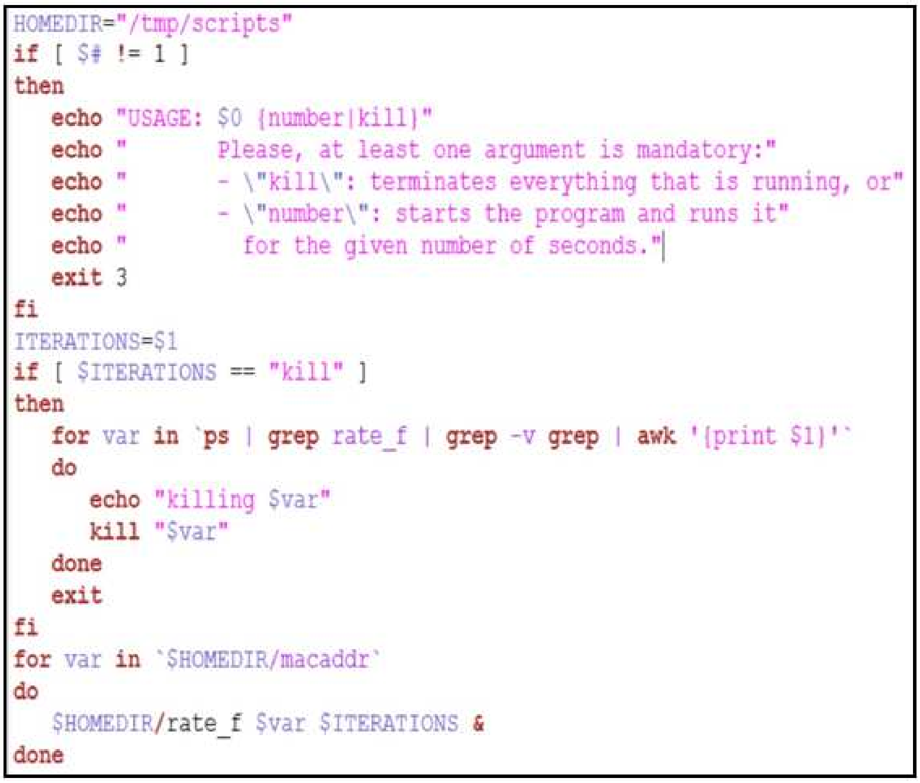

```
#!/bin/ash
# Script name: macaddr
# For each connected station, returns its MAC address.
for var in `iw dev wlan0 station dump | awk '$1 == "Station" { print $2}'`
do
   echo $var
done
```

Fig. 3. Script used for data collection



```
#!/bin/ash
# Script name: rate_m
# Saves the throughput (rate of tx) of all connected stations. We are measuring
# only the traffic originated from the client stations, showed at "rx bytes".
# This measures are the amount of transmitted bytes from each client workstation
# to the access point. We do not consider the traffic originated from the access
# point to the connected stations, this traffic is not object of our study.
# It is necessary to run the script "macaddr" before running this one, in order
# to update the list of connected stations.
# We have one command line argument that contain either the number of iterations,
# or the string "kill". Each iteration takes one second. After collecting the
# list of connected stations, this script starts one instance of "rate_f" script
# for each MAC address connected. This is only the main script, the real work is
# done by the "rate_f" script.
```

(a)



```
HOMEDIR="/tmp/scripts"
if [ $# != 1 ]
then
   echo "USAGE: $0 {number|kill}"
   echo "       Please, at least one argument is mandatory:"
   echo "       - \"kill\": terminates everything that is running, or"
   echo "       - \"number\": starts the program and runs it"
   echo "         for the given number of seconds."
   exit 3
fi
ITERATIONS=$1
if [ $ITERATIONS == "kill" ]
then
   for var in `ps | grep rate_f | grep -v grep | awk '{print $1}'`
   do
      echo "killing $var"
      kill "$var"
   done
   exit
fi
for var in `$HOMEDIR/macaddr`
do
   $HOMEDIR/rate_f $var $ITERATIONS &
done
```

(b)

Fig. 4. (a) Script used for data collection (b) Script used for data collection

Comparisons between each stations´s throughput curves with the networks´s total throughput curve indicates if MAC Anomaly is installed or not and identifies the offending station as well.

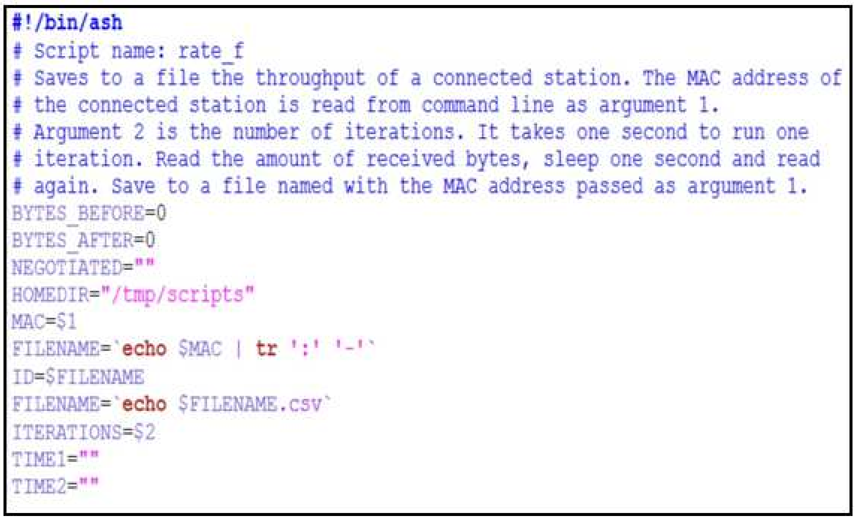
If total network´s total throughput decreases while a certain station´s throughput increases, or vice-versa, we can say that anomaly is installed and this is the offending station.

The throughput of the offending station varies in the opposite way to the network´s total.

This method is well described by (Marques, 2013). This behavior can be observed visually in the figures presented in section 5.

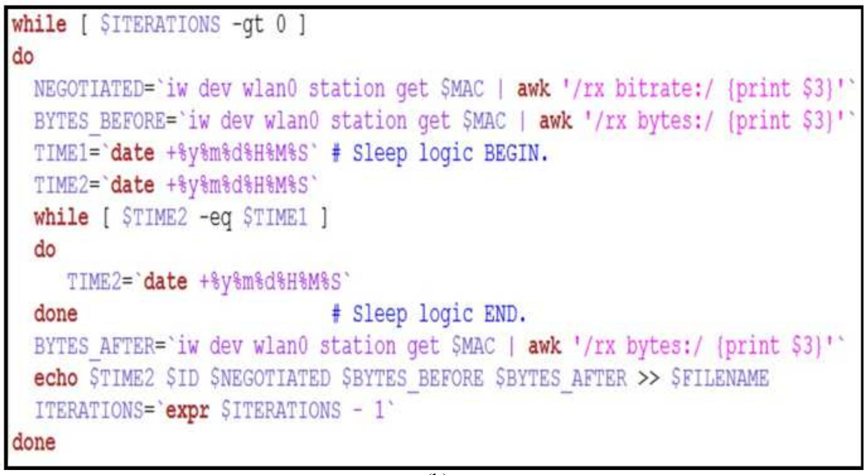
Traffic Shaping Script

The more a offending station generates traffic, the more it will cause anomaly. Traffic shaping is the method used here to mitigate the MAC Anomaly restoring the normal network operation. This action will increase total network throughput to the same level as before.



```
#!/bin/ash
# Script name: rate_f
# Saves to a file the throughput of a connected station. The MAC address of
# the connected station is read from command line as argument 1.
# Argument 2 is the number of iterations. It takes one second to run one
# iteration. Read the amount of received bytes, sleep one second and read
# again. Save to a file named with the MAC address passed as argument 1.
BYTES_BEFORE=0
BYTES_AFTER=0
NEGOTIATED=""
HOMEDIR="/tmp/scripts"
MAC=$1
FILENAME=`echo $MAC | tr ':' '-'`
ID=$FILENAME
FILENAME=`echo $FILENAME.csv`
ITERATIONS=$2
TIME1=""
TIME2=""
```

(a)



```
while [ $ITERATIONS -gt 0 ]
do
  NEGOTIATED=`iw dev wlan0 station get $MAC | awk '/rx bitrate:/ {print $3}'`
  BYTES_BEFORE=`iw dev wlan0 station get $MAC | awk '/rx bytes:/ {print $3}'`
  TIME1=`date +%y%m%d%H%M%S` # Sleep logic BEGIN.
  TIME2=`date +%y%m%d%H%M%S`
  while [ $TIME2 -eq $TIME1 ]
  do
     TIME2=`date +%y%m%d%H%M%S`
  done                       # Sleep logic END.
  BYTES_AFTER=`iw dev wlan0 station get $MAC | awk '/rx bytes:/ {print $3}'`
  echo $TIME2 $ID $NEGOTIATED $BYTES_BEFORE $BYTES_AFTER >> $FILENAME
  ITERATIONS=`expr $ITERATIONS - 1`
done
```

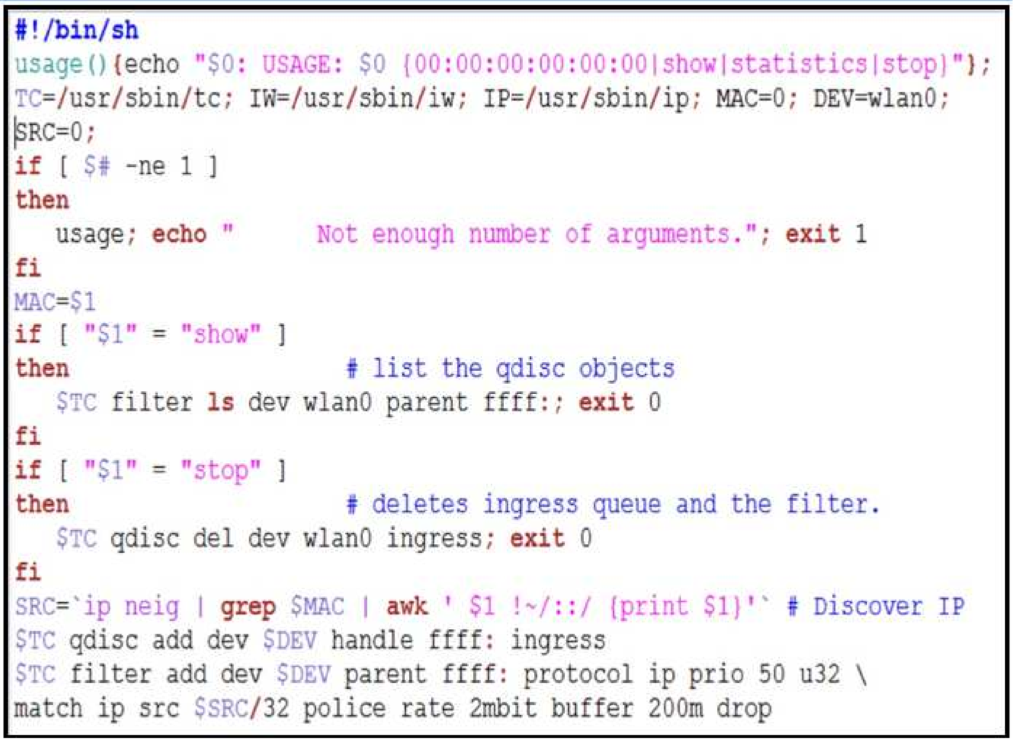
(b)

Fig. 5. (a) Script used for data collection (b) Script used for data collection

The objective is to reduce the throughput of the offending station to a lower level. Thus, it will not affect the rest of the network, still having opportunitty to transmit.

A complete explanation of the traffic shaping process can be found in (Hubert, 2015), one of the most referenced documents about traffic control. A small and simple script to implement traffic shapping was developed and presented in Fig. 6. The script logic is explained as follows:

- First of all, it is necessary to have the MAC address of the offending station to be passed as command line argument to the script. It is used the MAC address of the station identified as a offender by the method described in section 4.5.
- Create a table using the command "iptables". This table will pick-up all incoming packets and mark them if the MAC address matches with the one passed as command line argument
- Create an ingress "qdisc" (queue discipline). Ingress qdiscs are used to specify traffic of incoming type
- Create a filter and apply it to qdisc. The relationship between the filter and the qdisc is specified using the "parent" clause that contais the qdisc id "ffff:". The filter will select only the packets marked by "iptables", i.e., from the offender station. The relation between "filter" and "iptables" is specified by the "handle 1" clause
- Apply a police on these packets. The police will do the following: Put all packets in a buffer of size 10k bytes. Forward them only if the buffer is not full at a rate of 250kbit/s. If the buffer is full and more packets arrive, they will be dropped. The incoming dropped packets, won´t be lost. After not receiving an ACK during a period of time, the transmitter will lower its transmition rate and retransmit the packets dropped by this polic



```
#!/bin/sh
usage(){echo "$0: USAGE: $0 {00:00:00:00:00:00|show|statistics|stop}"};
TC=/usr/sbin/tc; IW=/usr/sbin/iw; IP=/usr/sbin/ip; MAC=0; DEV=wlan0;
SRC=0;
if [ $# -ne 1 ]
then
   usage; echo "      Not enough number of arguments."; exit 1
fi
MAC=$1
if [ "$1" = "show" ]
then                        # list the qdisc objects
   $TC filter ls dev wlan0 parent ffff:; exit 0
fi
if [ "$1" = "stop" ]
then                        # deletes ingress queue and the filter.
   $TC qdisc del dev wlan0 ingress; exit 0
fi
SRC=`ip neig | grep $MAC | awk ' $1 !~/::/ {print $1}'` # Discover IP
$TC qdisc add dev $DEV handle ffff: ingress
$TC filter add dev $DEV parent ffff: protocol ip prio 50 u32 \
match ip src $SRC/32 police rate 2mbit buffer 200m drop
```

Fig. 6. Traffic shaping script

Putty, WinSCP and SciLab

Putty: Running as ssh, it was used to start and stop the measuring and traffic shape scripts. Also to do all the configurations on the OpenWRT. The Luci Graphical User Interface was not used.

WinSCP: The results were saved in OpenWRT itself, but due to its reduced memory, WinSCP was used to transfer the data, in order to process them in SciLab.

SciLab: It was used to merge all the station´s data and build the graphics using its programming language.

Results

MAC Anomaly Detection

Set of Fig. 7-9 are showing a scenario with two stations connected: A and B. The total network throughput curve (A+B) is superposed with the station A and "B" throughput curves in Fig. 7. Figure 8 compares station "A" throughput curve with network´s total separately and Fig. 9 does the same with station B. As explained in section 4.5, our goal is to check if anomaly is present or not, identifying the offender as well. Anomaly is shown at time interval of 130-150s, were a throughput level decrease is observed. The station under 5 Mbits/s is the offender, because his curve is exactly the opposite image of the network´s total curve (Fig. 9). The other station above 20 Mbits/s has its curve very similar to the network total, then it is not an offender (Fig. 8).

Set of Fig. 10-15 are showing a scenario with five stations connected: Two are offenders and 3 are normal. Anomaly is shown at time interval of 60-120s, when a total network throughput decrease is observed (Fig. 10). Separately comparisons between each station´s throughput curves against network´s total throughput are shown in Fig. 11-15; aiming to identify the ofending stations that are causing the MAC anomaly.

MAC Anomaly Mitigation

Set of Fig. 16-18 shows a scenario with two stations connected, anomaly installed. Throughput curves were superposed: Network total, normal station and offending station in Fig. 16. Figure 17 compares a normal station against network, showing very similar curves. Figure 18 compares offending station against network, showing oposite images. At the moment of 200s, traffic shapping was applied to mitigate anomaly restoring the network´s normal operation. A strong total network throughput is observed. Offending station throughput was forced to decrease by traffic shaping. Stopped traffic shapping to demonstrate that the anomaly came back at 360 sec.

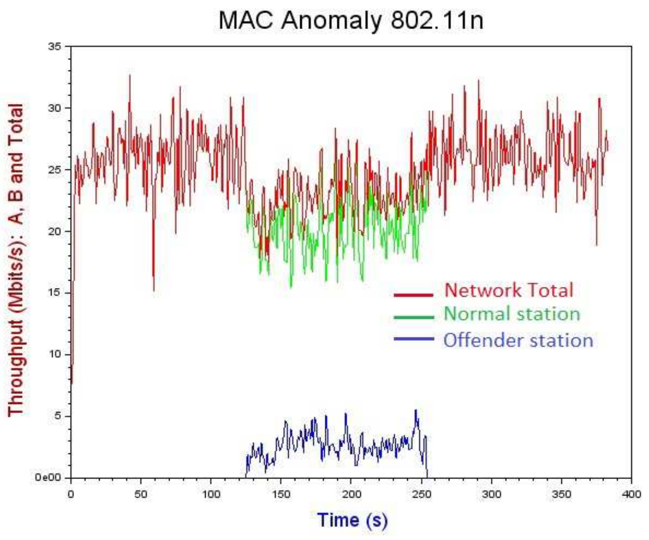


Fig. 7. Two stations, one is the offender. Superposed throughput curves: A, B and network total

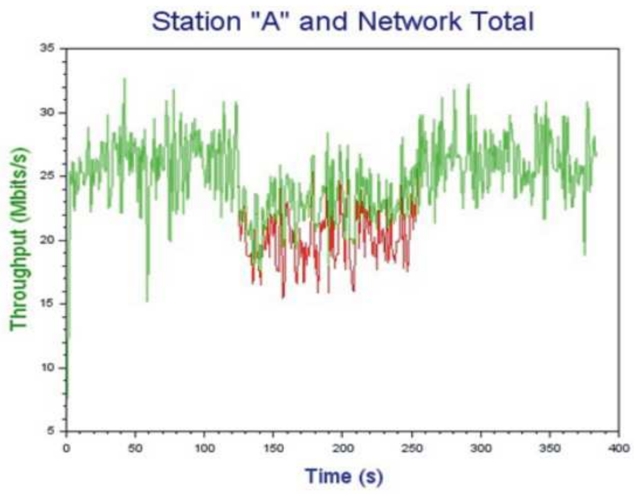


Fig. 8. Network throughput compared with a normal station´s throughput

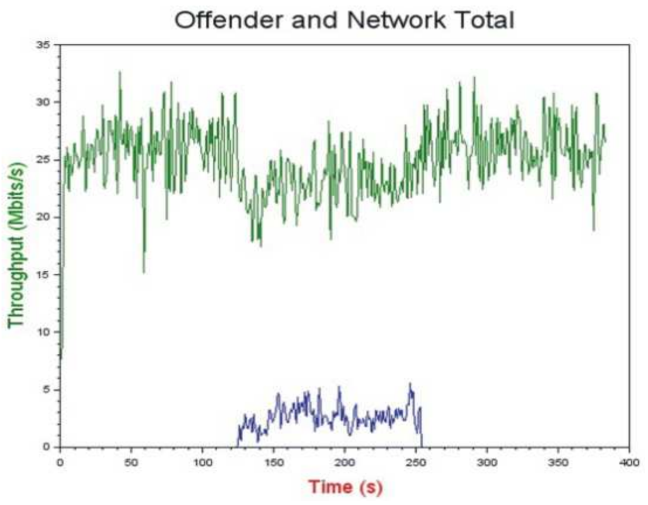


Fig. 9. Network total throughput compared with the offender´s throughput

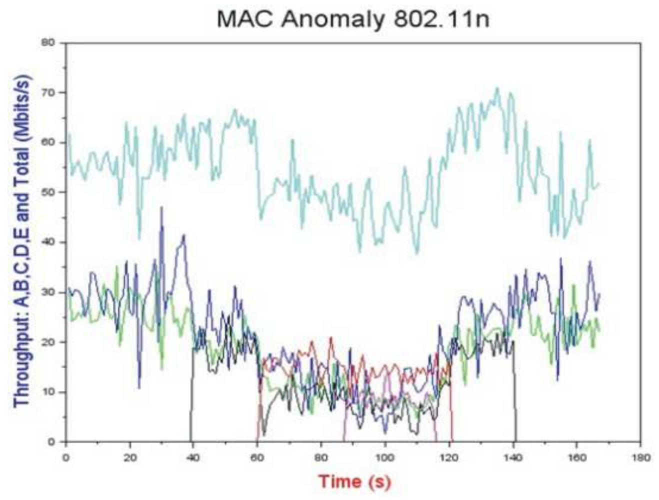


Fig. 10. Five station´s, two are offenders. Superposed throughput curves

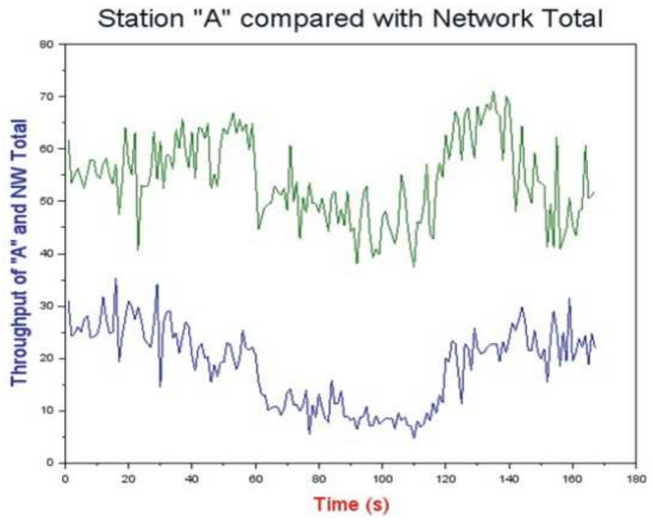


Fig. 11. Total network throughput compared with a normal station´s throughput. A similarity is observed

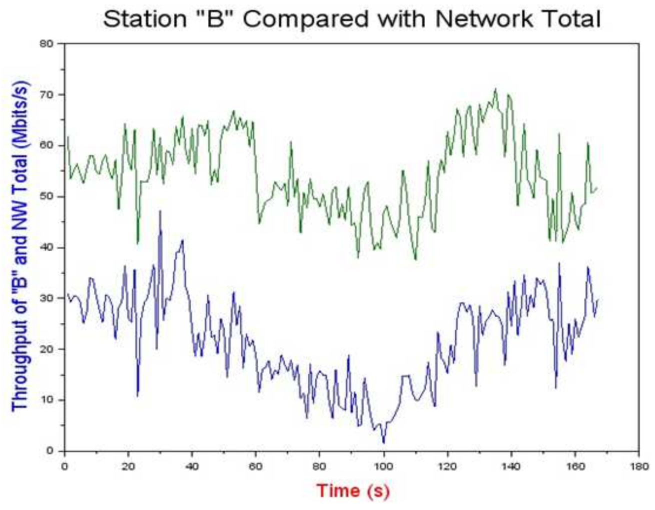


Fig. 12. Total network throughput compared with a normal station´s throughput. A similarity is observed

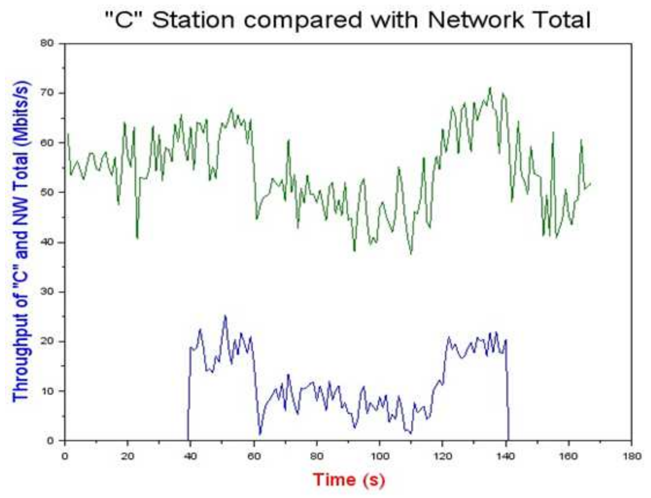


Fig. 13. Total network compared with a normal station´s throughput. A similiary is observed

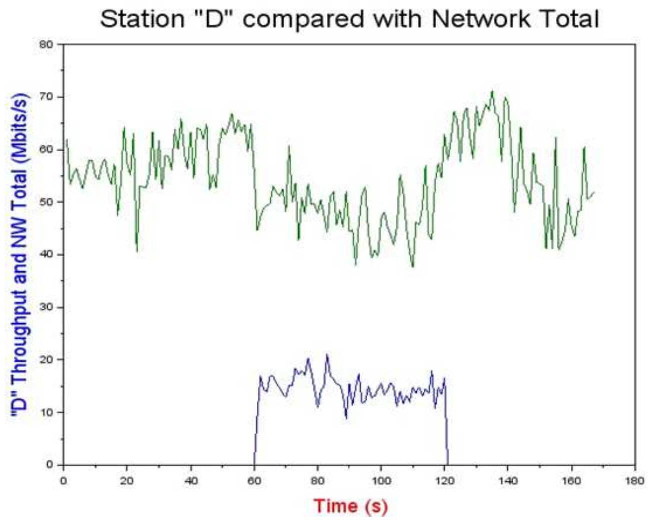


Fig. 14. First offending station´s throughput compared with total network´s throughput. Oposite images are observed

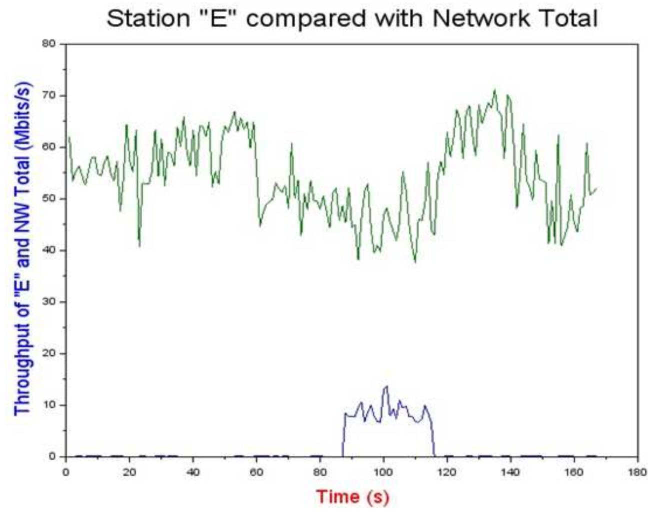


Fig. 15. Second offending station throughput compared with network´s total throughput. Oposite images are observed

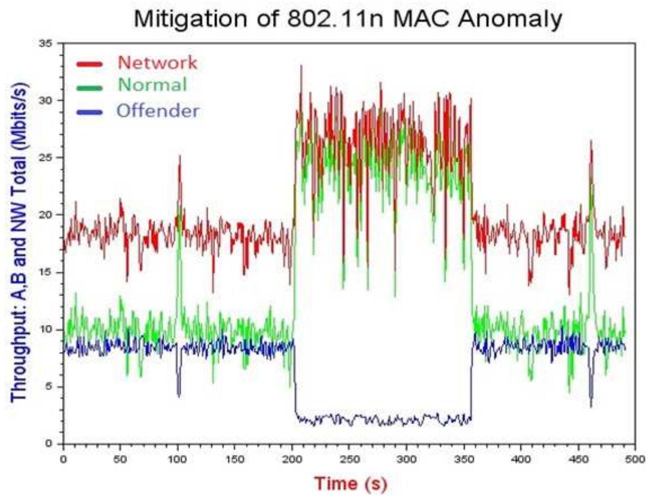


Fig. 16. Stations A, B and Network´s throughput. Traffic shaping applied at 200-360s interval

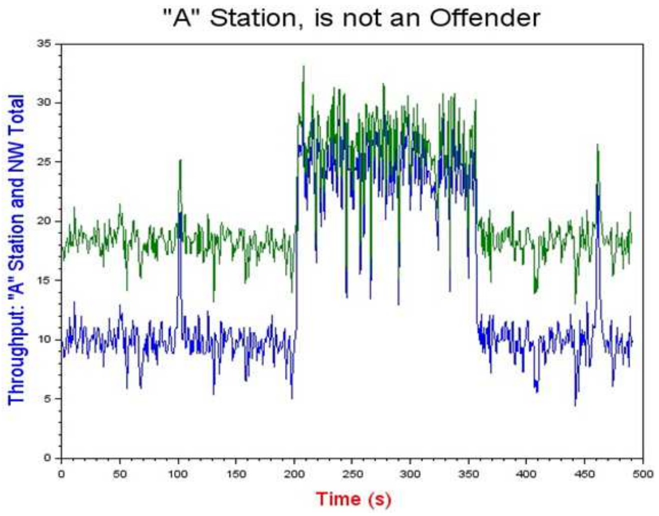


Fig. 17. Station A is a normal station. Compared with network, a similarity between curves is observed

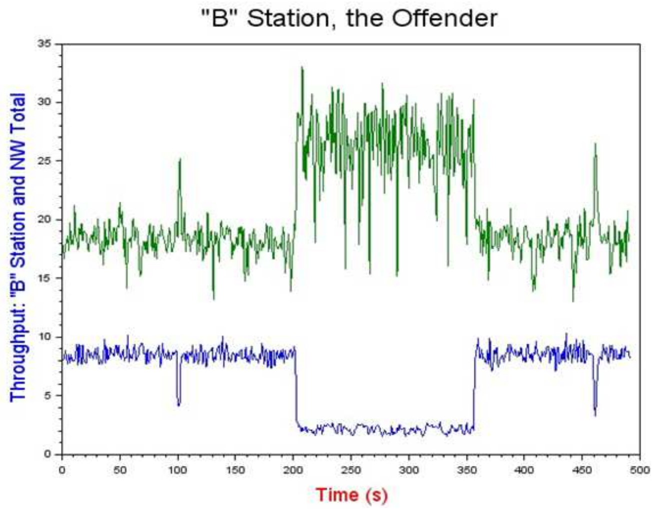


Fig. 18. Station B is an offending station. Oposition between curves is observed

Discussion and Conclusion

The goal to maintain the network throughput at its original workable level, when the MAC anomaly is present, was achieved with success. Processes to monitor, identify and mitigate MAC anomaly was used to achieve this goal.

Data throughput measurements were implemented, tested and used with success. This enabled analysis of the network conditions and identification of MAC anomaly by visual inspections the graphics.

Mitigation of MAC anomaly was implemented with success, using traffic shaping technique by means of a script running in embedded Linux.

Considering the current state of the art, it was verified the possibility to deploy a hardware and software platform of reduced cost and reduced hardware, using embedded Linux to identify and mitigate the IEEE 802.11 MAC Anomaly. It was successfully demonstrated the effect of traffic shaping applied to mitigate the MAC anomaly, in order to restore the normal network operation. A mathematic model that uses measured throughput data to identify the offender station by the similarity or opposition of its throughput curve to the network´s total throughput is being studied. Afterwards, a shell script to automate the process of identification will be written. A computer program to be run in OpenWRT, integrating all the steps: Throughput data measurement, MAC anomaly identification and mitigation will be implemented. Figure 16 displays the total throughput in red, by taking a closer look, it was possible to assess a gain while the mitigation through traffic shaping took place of 42.5216% or 7.781465 Mbits/s. Given that the average throughput, in the periods before, from 0 to 199s and after, from 361s to 491s the total average throughput was of 18.300012Mbits/s, whereas the interval using the referred mitigation the average throughput was of 26.081476Mbits/s.

Funding Information

The authors have no support or funding to report.

Author's Contributions

Argemiro Bevilacqua and Vitor Chaves de Oliveira: Conception and design, write computer programs, data acquisition, implement/test the traffic shaping method, analysis and interpretation of data. Configure and test the embedded Linux setup. Write the draft article. Studies about the subject and experiments development.

Gunnar Bedicks Jr.: Conception and design of data modeling, analysis and interpretation of data. General guidance on the concepts of radio systems. Orientation in statistical graphics procedures. Contributed to the writing of the manuscript. Contributed to the establishment of experiments schema. Important review contributions.

Alexandre de Assis Mota and Lia Toledo Moreira Mota: Conception and design of data modeling, analysis and interpretation of data. General guidance on the concepts of MAC Anomaly, traffic shaping method, analysis and interpretation of data. Orientation in statistical graphics procedures. Contributed to the writing of the manuscript. Establishment of good programming practice and program design issues. Important review contributions.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Biazotto, L.H., A.D.A. Mota and L.T.M. Mota, 2012. Low cost test bench to assess energy efficiency of communications network equipment. J. Comput. Sci., 8: 1849-1853. DOI: 10.3844/jcssp.2012.1849.1853

Branquinho, O.C., N. Reggiani and D.M. Ferreira, 2006. Mitigating 802.11 MAC anomaly using SNR to control backoff contention window. Proceedings of the International Conference on in Wireless and Mobile Communications, Jul. 29-31, IEEE Xplore Press, Bucharest, pp: 55-55. DOI: 10.1109/ICWMC.2006.61

Cano, E. and D.K. Francesco, 2015. System level traffic shaping in disk servers with heterogeneous protocols. Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics (HEP' 13).

Cheng, Y.H., L. Zhi-shu, J.C. Xing and L. Zhu, 2007. A novel MAC mechanism to resolve 802.11 performance anomaly. J. Zhejiang Univ. Sci. A, 8: 1573-1583. DOI: 10.1631/jzus.2007.A1573

DD-WRT Project, 2015. A Linux based alternative OpenSource firmware suitable for a great variety of WLAN routers. Open Source Software.

Guirardello, M., 2008. QoS police with media access prioritization for IEEE 802.11 networks. MSc Thesis, Pontifical Catholic University of Campinas.

Hallinan, C., 2006. Embedded Linux Primer: A Practical, Real-World Approach. 1st Edn., Pearson Education India, ISBN-10: 8131764079, pp: 537.

Heusse, M., F. Rousseau, G. Berger-Sabbatel and A. Duda, 2003. Performance anomaly of 802.11b. Proceedings of the IEEE Societies 22nd Annual Joint Conference of the IEEE Computer and Communications, Mar. 30-Apr. 3, IEEE Xplore Press, San Francisco, CA, pp: 836-843. DOI: 10.1109/INFCOM.2003.1208921

Hubert, B., 2015. Linux advanced routing and traffic control.

Ierusalimschy, R., 2015. Lua the programming language 20 years.

Ingle, C.P. and R. Daryapurkar, 2013. Performance analysis of embedded Linux in embedded system. Int. J. Innovative Technol. Explor. Eng.

Lin, K. and A.Q. Hu, 2003. Bandwidth control for wireless access networks. Proceedings of the IEEE International Conference on Neural Networks and Signal Processing, Dec. 14-17, IEEE Xplore Press, Nanjing, pp: 1650-1653. DOI: 10.1109/ICNNSP.2003.1281199

Lopez-Aguilera, E., J. Casademont and E.G. Villegas, 2010. A study on the influence of transmission errors on WLAN IEEE 802.11 MAC performance. Wireless Commun. Mob. Comput., 11: 1376-1391. DOI: 10.1002/wcm.934

Marques, C.P.C., 2013. Offenders identification via analysis of stations´s sensitivity in IEEE 802.11 networks throughput. MSc. Thesis, Pontifical Catholic University of Campinas.

Oliveira, V., I. Yano, A. Mota and L. Mota, 2013a. Feasibility of desktop virtualization per software services and local hardware based on the network throughput. J. Comput. Sci., 9: 827-837. DOI: 10.3844/jcssp.2013.827.837

Oliveira, V., A. Mota and L. Mota, 2013b. Impacts of application usage and local hardware on the throughput of computer networks with desktop virtualization. Am. J. Applied Sci., 10: 117-122. DOI: 10.3844/ajassp.2013.117.122

Oliveira, V.C., G. Bedicks Jr. and C. Akamine, 2015. Methodology to asses IP connection availability: A prerequisite for feasible video stream through CDNs. J. Comput. Sci., 11: 426-437. DOI: 10.3844/jcssp.2015.426.437

OpenWRT Project, 2015. Wireless freedom.

Palazzi, C.E., M. Brunati and M. Roccetti, 2010. An OpenWRT solution for future wireless homes. Proceedings of the IEEE International Conference on Multimedia and Expo, Jul. 19-23, IEEE Xplore Press, Suntec City, pp: 1701-1706. DOI: 10.1109/ICME.2010.5583223

Peris, A.J.F., 2012. Throughput in IEEE 802.11 networks with offenders present. MSc Thesis, Pontifical Catholic University of Campinas.

Shrivastava, V., S. Rayanchu, J. Yoon and S. Banerjee, 2008. 802.11n under the microscope. Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement. Oct. 20-22, Vouliagmeni, Greece, pp: 105-110. DOI: 10.1145/1452520.1452533

Singh, G. and B.S. Sohi, 2008. Experimental evaluation of medium access schemes in 802.11 wireless networks. J. Comput. Sci., 4: 888-896. DOI: 10.3844/jcssp.2008.888.896

Stallings, W., 1999. SNMP, SNMPv2, SNMPv3 and RMON 1 and 2. 1st Edn., Pearson Education India, ISBN-10: 8131702308, pp: 619.

Vila-Carbo, J., J. Tur-Masanet and E. Hernandez-Orallo, 2008. An evaluation of switched ethernet and linux traffic control for real-time transmission. Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation ETFA, Sept. 15-18, IEEE Xplore Press, Hamburg, pp: 400-407. DOI: 10.1109/ETFA.2008.4638424

Zhang, L., P. Senac, E. Lochin, J. Lacan and M. Diaz, 2008. Cross-layer based erasure code to reduce the 802.11 performance anomaly: When FEC meets ARF. Proceedings of the 6th ACM International Symposium on Mobility Management and Wireless Access. Oct. 27-31, Vancouver, Canada, pp: 119-124. DOI: 10.1145/1454659.1454679