

Designing installations for verification of the model of active queue management discipline RED in the GNS3

T. R. Velieva*, A. V. Korolkova*, and D. S. Kulyabov*

*Department of Applied Informatics and Probability Theory
Peoples' Friendship University of Russia
Miklukho-Maklaya str., 6, Moscow, 117198, Russia

Abstract—The problem of RED-module mathematical model results verification, based on GNS3 experimental stand, is discussed in this article. The experimental stand consists of virtual Cisco router, traffic generator D-ITG and traffic receiver. The process of construction of such stand is presented. Also, the interaction between experimental stand and a computer of investigation in order to obtain and analyze data from stand is revised.¹

I. INTRODUCTION

A stochastic model of the traffic management RED type module was built [1], [2], [3], [4]. Verification of the model was carried out on the NS-2 basis. However, we would like to conduct verification on a real router. As a result was the task of designing an experimental stand. It was decided to verify the clean RED algorithm [5] based on Cisco router. For the construction of the stand software package GNS3 (Graphical Network Simulator) was chosen.

Thus, the purpose of the study is to build on the GNS3 basis a virtual stand consisting of a Cisco router, a traffic generator and a receiver. A traffic generator D-ITG (Distributed Internet Traffic Generator) is used as.

II. THE MODEL OF THE ACTIVE QUEUE MANAGEMENT RED MODULE

The model of the active queue management RED module is a development of fluid model [1], [2], [3], [6], [7]. In the works [8], [9], [10] for methodology unification the method of one-step process randomization was used. Based on one-step processes we construct the stochastic model of RED, and the model contains two main elements: source of traffic and receiver.

The source sends packets, the receiver processes the packets and send an acknowledge. Our model is based on the assumption that the source and receiver interact with management. Thus, we obtain two equations, one of which describes a TCP window, and the second — instant queue length. The intensity of sending packets depends on the window size.

The detailed description of RED stochastic model is given in [4] and in appendix (section VIII), where was obtained:

$$\begin{cases} dW = \frac{1}{T}dt - \frac{W}{2}dN + \sqrt{\frac{1}{T} + \frac{W}{2}\frac{dN}{dt}}dV^1, \\ dQ = \left(\frac{W}{T} - C\right)dQ + \sqrt{\frac{W}{T} - C}dV^2, \\ \frac{d\hat{Q}}{dt} = w_q C(Q - \hat{Q}). \end{cases}$$

Here W is the window size average value, Q is the average value of the instantaneous queue, \hat{Q} is exponentially weighted moving average, C is service intensity, dV^i is two dimensional Wiener random process.

III. CISCO IOS STRUCTURE

To use the IOS image one should understand, which features given delivery option support [11]. The parameter “Feature set” (fig. 1) is responsible for this:

- IP Base: initial level of functionality is included in all other sets of features. Provides a basic routing, that is static routes, RIP, OSPF, EIGRP, only IPv4. Includes VLAN (802.1Q and ISL), which previously were available only in the IP Plus set. Also includes NAT.
- IP Services (3rd level switches): dynamic routing protocols, NAT, IP SLA.
- Advanced IP Services: support IPv6 is added.
- IP Voice: functionality VoIP and VoFR is added.
- Advanced Security: IOS/Firewall, IDS, SCTP, SSH and IPsec (DES, 3DES and AES) are added.
- Service Provider Services: IPv6, Netflow, SSH, BGP, ATM and VoATM are added.
- Enterprise Base: the support for L3 protocols such as IPX and AppleTalk is added. Also IBM DLSw+, STUN/BSTUN and RSRB are added.

Since version IOS 12.3T Cisco uses a new naming scheme for images (table I). However, this method of naming does not cover all the subtleties of image acquisition, so elements of the old naming scheme (table II) are still used.

¹978-1-4799-5291-5/14/\$31.00 ©2014 IEEE

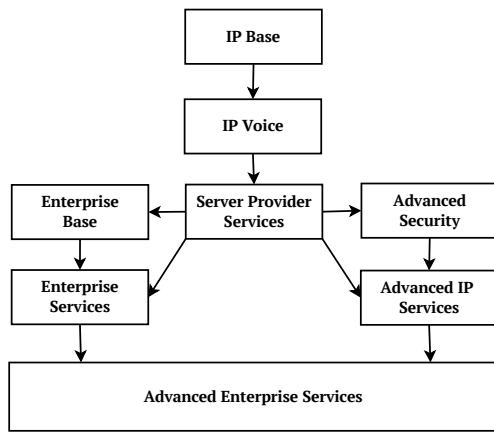


Fig. 1. Feature tree of the new switch IOS naming convention

TABLE I. NEW NAMING CONVENTION FOR CISCO ROUTER IOS

Code	Features
Base	Entry level image (IP Base, Enterprise Base)
Services	Addition of IP Telephony Service, MPLS, Voice over IP, Voice over Frame Relay and ATM (Included in SP Services, Enterprise Services)
Advanced	Addition of VPN, Cisco IOS Firewall, 3DES encryption, SSH, Cisco IOS IPSec and Intrusion Detection Systems (IDS) (Advanced Security, Advanced IP Services)
Enterprise	Addition of multi-protocols, including IBM, IPX, AppleTalk (Enterprise Base, Enterprise Services)

TABLE II. OLD NAMING CONVENTION FOR CISCO ROUTER IOS

Code	Features
I	IP
Y	IP on 1700 Series Platforms
S	IP Plus
S6	IP Plus No ATM
S7	IP Plus No Voice
J	Enterprise
O	IOS Firewall/Intrusion Detection
K	Cryptography/IPSEC/SSH
K8	56Bit DES Encryption (Weak Cryptography)
K9	3DES/AES Encryption (Strong Cryptography)
X	H323
G	Services Selection Gateway (SSG)
C	Remote Access Server or Packet Data Serving Node (PDSN)
B	Apple Talk
N	Novel IP/IPX
V	Vox
R	IBM
U	Unlawful Intercept
P	Service Provider
Telco	Telecommunications Feature Set
Boot	Boot Image (Used on high end routers/switches)

For unknown reasons, not all IOS images are efficient in GNS3. Several images were tested. Here's a short list of working and non-working images.

Workable images:

- C1700-adventerprisek9-mz.124-8
- C1710-bk9no3r2sy-mz.124-23
- C1720-l2sy-mz.121-11
- C2600-adventerprisek9_sna-mz.124-25b
- C2691-adventerprisek9_sna-mz.124-23
- C3660-jsx-mz.123-4.T
- C3745-adventerprisek9_sna-mz.124-15.T14
- C7200-adventerprisek9_sna-mz.152-4.m4

Unworkable images:

- C2600-adventerprisek9-sna-mz.124-23
- C3745-adventerprisek9_sna-mz.124-15.T14
- C3745-adventerprisek9_ivs-mz.124-15.T14
- C3745-adventerprisek9_ivs-mz.124-15.T8

IV. GNS3 INSTALLATION AND CONFIGURATION

GNS3 allows you to simulate a virtual network consisting of routers and virtual machines [12]. In fact, it is a graphical interface for different virtual machines. To emulate Cisco devices emulator dynamips is used. In addition, you can use such emulators as VirtualBox and Qemu. The latter is particularly useful when we work with the KVM, allowing the hardware implementation of the processor. The graphical interface makes it easy to switch different virtual machines. Also, there is the opportunity to connect projected topology with the real network. Wireshark allows to monitor traffic within the designed topology.

To work with GNS3 we need to install Dynamips, VirtualBox and/or QEMU, xdotool, Wireshark. To install the above software for Linux operating systems (in our case, GNS3 installed on Ubuntu 14.04) the following commands are prescribed in the console:

```

sudo apt-get install dynamips
sudo apt-get install qemu
sudo apt-get install virtualbox
sudo apt-get install xdotool
sudo apt-get install wireshark

```

After that GNS3 is set with a command in a terminal:

```
sudo apt-get install gns3
```

Then GNS3 is run. GNS3 interface opens (fig. 2).

At the top the context menu, on the left-hand — the equipment to choose, at the bottom — a console window, on the right-hand — a network management menu. Before start you need to pre-configure GNS3.

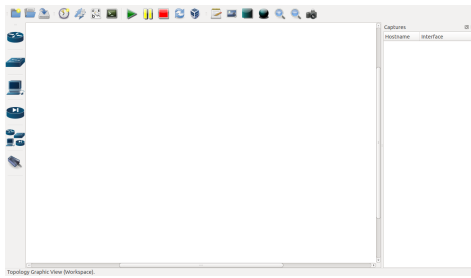


Fig. 2. GNS3 interface

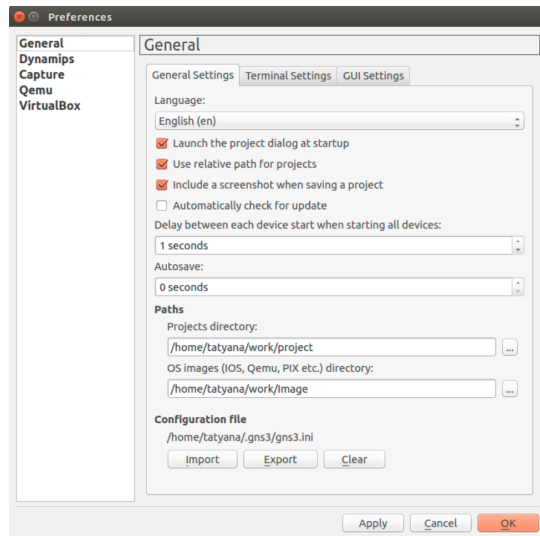


Fig. 3. General preferences of GNS3

For this the item **Preferences** is selected in the **Edit** of context menu. (fig. 3).

The language can be changed in **General** submenu. Here the path to the folder with designs and images of equipment is proscribed (fig. 4).

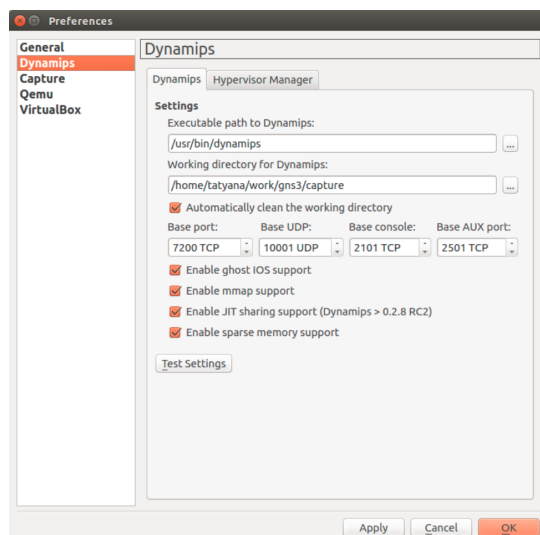


Fig. 4. Dynamips preferences

In submenu **Capture** traffic capture parameters are configured (fig. 5) (however, in this article we do not use the opportunity to capture the traffic). In line **Default Presets** the option **Wireshark Live Traffic Capture (Linux)** is chosen. In the next line, the path to the directory to store data capture is specified. And the last line the command to start Wireshark capture is proscribed:

```
tail -f -c +0b %c | wireshark -k -i
```

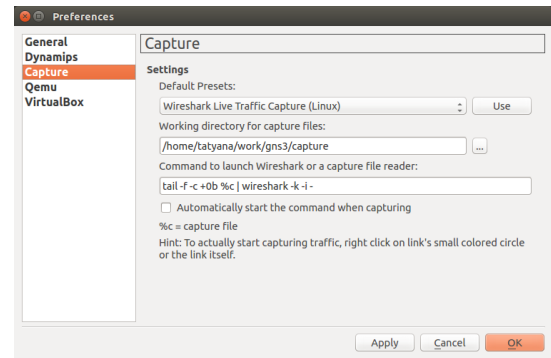


Fig. 5. Capture preferences

In the **Qemu** the tab **General Settings** sets the path to Qemuwrapper (this file is supplied with GNS3). The directory for capture is specified. In line **Path to qemu** the path to the file that emulates the processor is set. The following line specifies the path to the virtual machine. Port numbers are reserved by default. To test Qemuwrapper one need to click on the button **Test Settings**, a green label on the implementation of the test should appear (fig. 6).

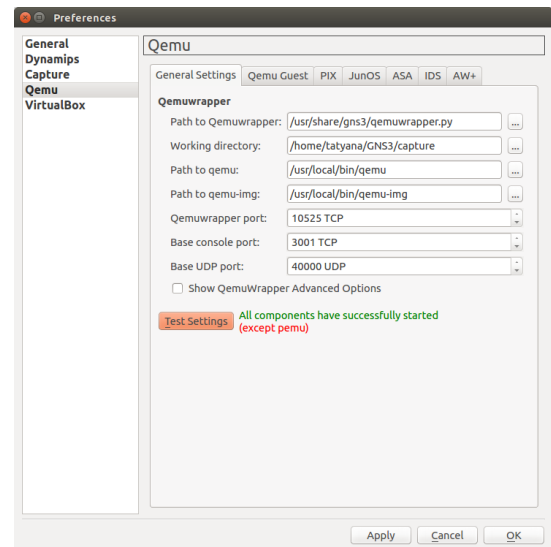


Fig. 6. Preferences GNS3 – Qemu

Next we need to install the OS image to the virtual machine. Set of images can be taken from the page <http://www.gns3.net/appliances/>. We chose Linux Core 4.7.7, because it contains a traffic generator D-ITG.

In line **Qemu flavor** we specify the model and name of processor. After that the path to the OS image is set. Push

Save and **OK**. After that, the **Qemu guest** can be selected on the toolbox (fig. 7).

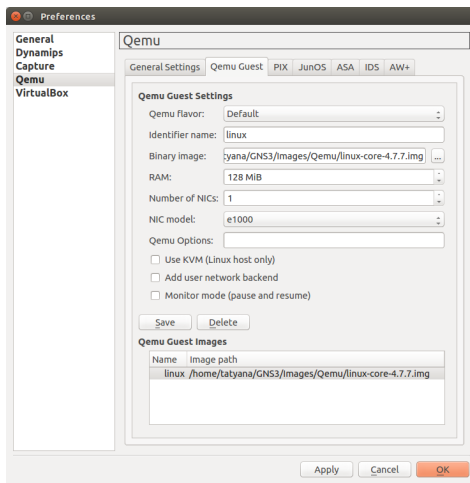


Fig. 7. Preferences GNS3

Now you can assemble the stand. For this we portable from device selection menu router and virtual machine **Qemu guest**, rename (**right mouse button** > **change the hostname**) (fig. 8).

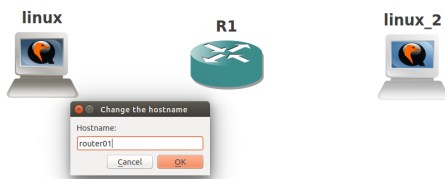


Fig. 8. Virtual Network

Host01 is a package source; host02 is a destination. Let's configure the router slots. To do this, right click on router01 and select **Configure** > **router01** > **Slots**. For slot0 we select FastEthernet (fig. 9).

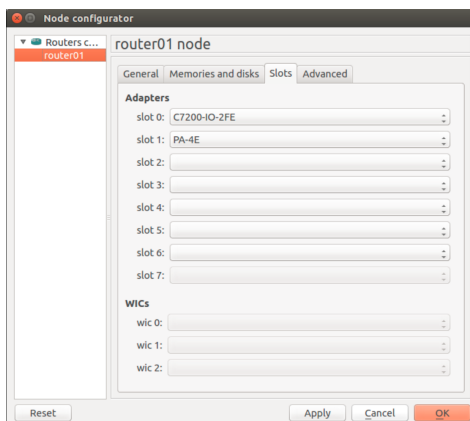


Fig. 9. Router configurator

Then we create a connection between the router and the virtual machine. We should right-click on host01, select **add link** > **FastEthernet** > **e0** and connect with router01, choosing f0/0. A connection between host02 and router01 is created similarly,

only now we choose Ethernet type of connection (speed 10 Mbps).

Now we need to connect the virtual machines with the host computer (for receiving and processing the logs). Connection will be carried through TUN/ TAP interface (TAP simulates an Ethernet device and works at the link layer model OSI, Ethernet frames and terms used to create a bridge). To do this, drag it to the workspace cloud and configure it: **Configure** > **NIO TAP**. Write the name tap0 and press **add** > **apply** > **ok** (fig. 10).

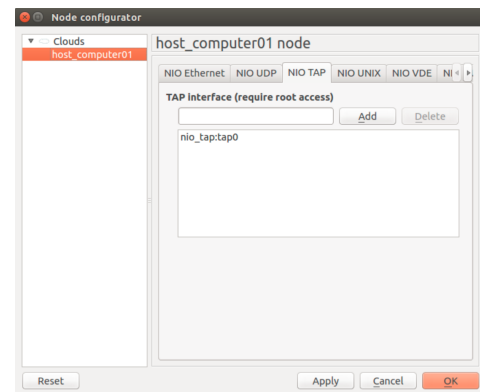


Fig. 10. Cloud interface configurator

We create a connection to router01, selecting the type of connection FastEthernet, and get topology (fig. 11).

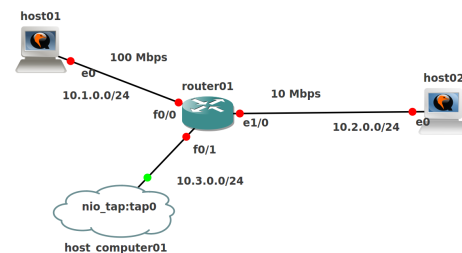


Fig. 11. Virtual Network

Next, you need to configure the devices.

Run router01 by clicking (**right button** > **start**). Open the console for it (**right button** > **Console**) and enter the following commands (fig. 12):

```
Router>enable
Router#configure terminal
router01(config)#hostname router01
router01(config)#interface f0/0
router01(config-if)#ip address 10.1.0.1
255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#interface e1/0
router01(config-if)#ip address 10.2.0.1
255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#full-duplex
router01(config-if)#exit
```

```
router01(config)#interface f0/1
router01(config-if)#ip address 10.3.0.1
255.255.255.0
router01(config-if)#no shutdown
router01(config-if)#exit
router01(config)#exit
router01#write memory
```

```

root@kali:~# netcat
root01(config-if)#exit
root01(config)#interface e1/0
root01(config-if)#ip address 10.2.0.1 255.255.255.0
root01(config-if)#no shutdown
root01(config-if)#full-duplex
root01(config-if)#exit
root01(config)#interface f0/1
root01(config-if)#ip address 10.3.0.1 255.255.255.0
root01(config-if)#no shutdown
root01(config-if)#exit
root01(config)#exit
root01#
*Jun 15 18:04:50.867: %SYS-5-CONFIG_I: Configured from console by console
root01#wr mem
Warning: Attempting to overwrite the NVRAM configuration previously written
to a different version of the system image
Overwrite the previous NVRAM configuration?[confirm]
Building configuration...
[OK]
root01#usr/bin/filetool.sh -b
% Invalid input detected at '^' marker.
root01#usr/bin/filetool.sh -b -h

```

Fig. 12. Router CLI configuration

Now we open the console for host01. As the name of the user we specify `tc`, the password is blank. Go to the superuser: `sudo su`. Edit parameters host01 (to start editing in `vi` we should enter `i`; to save the changes and exit we should press the button `esc`, then enter `:wq` and press the `Enter`). We need to edit two files: `/opt/bootsync.sh` and `/opt/bootlocal.sh`:

```
vi /opt/bootlocal.sh
```

At end of file we write the following:

```
ifconfig eth0 10.1.0.10 netmask \
    255.255.255.0 up
route add default gw 10.1.0.1
```

After saving the changes to a file we exit from it, and execute the command to save the configuration (Linux Core uses the Tinycore configuration system):

```
/usr/bin/filetool.sh -b
```

Similarly host02 is configured, but ip address (network 10.2.0.0/24) should be changed.

Let's configure the host computer with TAP-interface. In the command window as a root we create an interface for user:

```
tunctl -u user_name -t tap0
```

Now we set the address for tap0 interface (fig. 13):

```
ifconfig tap0 10.3.0.10 netmask \
    255.255.255.0 up
```

Then you can configure filtering and routing (fig. 14):

```
iptables -I INPUT 1 -i tap0 -j ACCEPT
route add -net 10.1.0.0/24 dev tap0
route add -net 10.2.0.0/24 dev tap0
```

```
root@tatyana:~#  
tatyana@tatyana:~$ sudo -i  
[sudo] password for tatyana:  
root@tatyana:~# tuncctl -u tatyana -t tap0  
Set 'tap0' persistent and owned by uid 1000  
root@tatyana:~# ifconfig tap0 10.3.0.10 netmask 255.255.255.0 up  
root@tatyana:~#
```

Fig. 13. Address configuration on tap0 interface

```
root@tatyana:~#
tatyana@tatyana:~$ sudo -i
[sudo] password for tatyana:
root@tatyana:~# tunctl -u tatyana -t tap0
Set 'tap0' persistent and owned by uid 1000
root@tatyana:~# ifconfig tap0 10.3.0.10 netmask 255.255.255.0 up
root@tatyana:~# iptables -I INPUT 1 -i tap0 -j ACCEPT
root@tatyana:~# route add -net 10.1.0.0/24 dev tap0
root@tatyana:~# route add -net 10.2.0.0/24 dev tap0
root@tatyana:~#
```

Fig. 14. Routing configuration on tap0 interface

V. TRAFFIC GENERATOR D-ITG

Now we can generate traffic and take readings. To generate traffic we use D-ITG.

D-ITG provides estimates for key indicators of quality of service (the average packet delay, delay variation (jitter), packet loss ratio, throughput) with a high degree of certainty. Depending on the requirements of the experiment, we can change the following parameters:

- number of streams transmitted between end stations;
- duration of traffic generation;
- the intensity of each separate stream (pack/s or bit/s);
- packet traffic length or distribution law;
- type and parameters of the distribution law of the time interval between adjacent packets (for example, packet length and time interval between packets can be distributed in a uniform, exponential, normal, gamma-law or Pareto, Cauchy, Poisson);
- type of transport layer protocols: TCP, UDP, DCCP, SCTP;
- type of traffic (simulation of a flow generated by specific application protocol): VoIP, IPTV, Telnet, DNS, game traffic (Counter Strike, Quake3).

Within the D-ITG the control of experiment is performed by using the command line, and the required set of parameters to generate traffic is given by calling the program `ITGSend` using keys.

Let's consider the example of generating traffic for UDP and TCP. In host02 console we run the command of channel audition:

ITGRecv

In host01 console we run host01 traffic generation:

```
ITGSend -a 10.2.0.10 -T UDP -C 10000 \
-c 500 -t 20000 -x recv_log_file
```

Here we transfer the files to the receiver address by 10.2.0.10 protocol UDP, the transmission rate is 10,000 packets per second, the size of package is 500 bytes, the connection time is 20000 ms. All the information about

send data is recorded on the receiving end in a file called `recv_log_files`. Once data transfer has been performed, we stop listening to `host02` console (pressing `Ctrl`+`C`), then run the command in `host02` console to decode the log file:

```
ITGDec recv_log_file
```

By this command a table with the values of the received stream is displayed (fig. 15).

```
ITGDec version 2.8.1 (r1023)
Compile-time options: dccp bursty multiport
|-----|
Flow number: 1
From 10.1.0.10:57573
To 10.2.0.10:8999
|-----|
Total time           = 19.998072 s
Total packets        = 9257
Minimum delay        = 0.141002 s
Maximum delay        = 0.153761 s
Average delay        = 0.146050 s
Average jitter        = 0.003195 s
Delay standard deviation = 0.002774 s
Bytes received       = 4739584
Average bitrate      = 1896.016376 Kbit/s
Average packet rate   = 462.894623 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
|-----|

***** TOTAL RESULTS *****
Number of flows      = 1
Total time           = 19.998072 s
Total packets        = 9257
Minimum delay        = 0.141002 s
Maximum delay        = 0.153761 s
Average delay        = 0.146050 s
Average jitter        = 0.003195 s
Delay standard deviation = 0.002774 s
Bytes received       = 4739584
Average bitrate      = 1896.016376 Kbit/s
Average packet rate   = 462.894623 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
Error lines          = 0
tc@host02:~$
```

Fig. 15. Input flow statistics. UDP

We take similarly steps for TCP and get the following results (fig. 16).

```
ITGDec version 2.8.1 (r1023)
Compile-time options: dccp bursty multiport
|-----|
Flow number: 1
From 10.1.0.10:34185
To 10.2.0.10:8999
|-----|
Total time           = 20.015369 s
Total packets        = 4742
Minimum delay        = 0.132785 s
Maximum delay        = 1.180837 s
Average delay        = 0.190162 s
Average jitter        = 0.005034 s
Delay standard deviation = 0.137616 s
Bytes received       = 2371000
Average bitrate      = 947.671762 Kbit/s
Average packet rate   = 236.917940 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
|-----|

***** TOTAL RESULTS *****
Number of flows      = 1
Total time           = 20.015369 s
Total packets        = 4742
Minimum delay        = 0.132785 s
Maximum delay        = 1.180837 s
Average delay        = 0.190162 s
Average jitter        = 0.005034 s
Delay standard deviation = 0.137616 s
Bytes received       = 2371000
Average bitrate      = 947.671762 Kbit/s
Average packet rate   = 236.917940 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
Error lines          = 0
root@host02:~#
```

Fig. 16. Input flow statistics. TCP

For multicast traffic in `host01` command window for file `send` we describe each flow of traffic. For example, as follows:

```
cat > send <<EOF
-a 10.2.0.10 -C 1000 -c 512 -T UDP
-a 10.2.0.10 -C 2000 -c 512 -T UDP
-a 10.2.0.10 -C 3000 -c 512 -T UDP
-a 10.2.0.10 -C 4000 -c 512 -T UDP
-a 10.2.0.10 -C 5000 -c 512 -T UDP
EOF
```

Now let's run the traffic flow from `host01`. The transmission traffic data from `host01` to `host02` is written in the files `send.log` and `recv.log`:

```
ITGSend send -l send.log -x recv.log
```

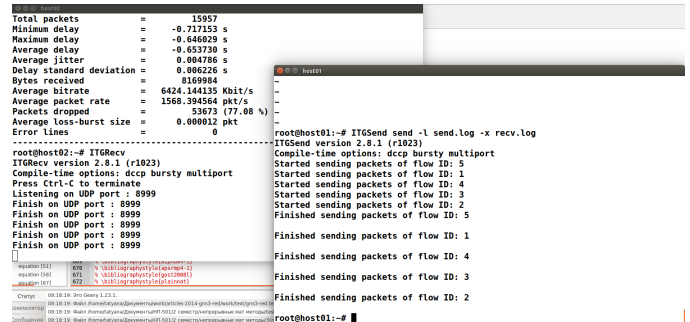
Let's display the report of flow transmission. To do this, we write the command console `host02`

```
ITGDec send.log
```

to decrypt the sent traffic (fig. 17) and command

```
ITGDec recv.log
```

to decrypt the received traffic.



```
ITGDec version 2.8.1 (r1023)
Compile-time options: dccp bursty multiport
|-----|
Flow number: 1
From 10.1.0.10:34185
To 10.2.0.10:8999
|-----|
Total time           = 20.015369 s
Total packets        = 4742
Minimum delay        = 0.132785 s
Maximum delay        = 1.180837 s
Average delay        = 0.190162 s
Average jitter        = 0.005034 s
Delay standard deviation = 0.137616 s
Bytes received       = 2371000
Average bitrate      = 947.671762 Kbit/s
Average packet rate   = 236.917940 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
|-----|

***** TOTAL RESULTS *****
Number of flows      = 1
Total time           = 20.015369 s
Total packets        = 4742
Minimum delay        = 0.132785 s
Maximum delay        = 1.180837 s
Average delay        = 0.190162 s
Average jitter        = 0.005034 s
Delay standard deviation = 0.137616 s
Bytes received       = 2371000
Average bitrate      = 947.671762 Kbit/s
Average packet rate   = 236.917940 pkt/s
Packets dropped       = 0 (0.00 %)
Average loss-burst size = 0.000000 pkt
Error lines          = 0
root@host02:~#
```

Fig. 17. Log decryption

Thus, we obtain data about each flow, as well as the value of the outgoing and incoming flows parameters.

VI. VISUALIZATION OF RESULTS

Charts can be constructed for the following parameters: delay, jitter, bitrate, packet loss.

Assume that we have 5 streams, which are transmitted over TCP. Flow rate 1 is 1000 pps, flow 2 is 2000 pps, stream 3 is 3000 pps, stream 4 is 4000 pps, stream 5 is 5000 pps. Packet sizes are 512 bytes and the same for all streams, and Connection Time is 20000 ms. For packet loss demonstration we will use UDP protocol (fig. 18).

With the help of `ITGplot` we make the graphs of bitrate, delay and jitter. For this we need to record the values obtained in individual files using the following commands:

```
ITGDec recv.log -b 1000
```

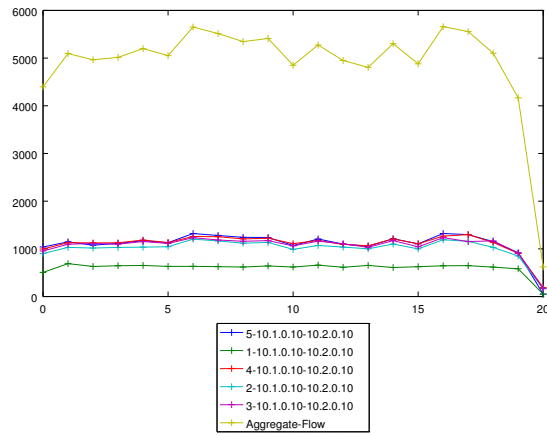


Fig. 18. UDP packets loss

```
ITGDec recv.log -d 1000
ITGDec recv.log -j 1000
```

Every 1000 ms in files bitrate.dat, jitter.dat, delay.dat the respective values of the parameters are recorded for the transmitted flows (fig. 19, 20, 21). We construct graphs using the following commands:

```
./ITGplot bitrate.dat
./ITGplot delay.dat
./ITGplot jitter.dat
```

On obtained graphs the upper curve shows the general behavior of all flows.

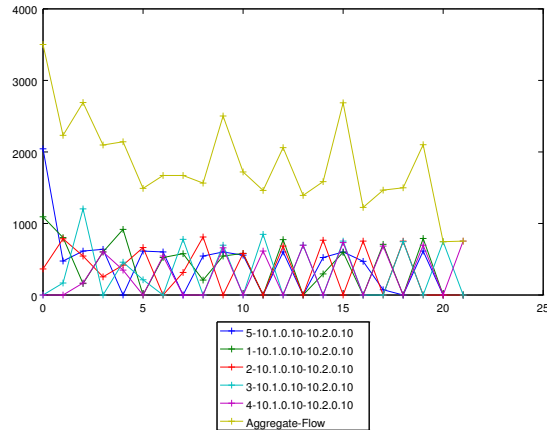


Fig. 19. Incoming bitrate

VII. CONCLUSION

Thus, we have built the foundation of the stand for verification model of traffic management module RED. However, in this paper is not considered However, the task of RED module configuration on given router is not considered as well as a problem of real time router reading.

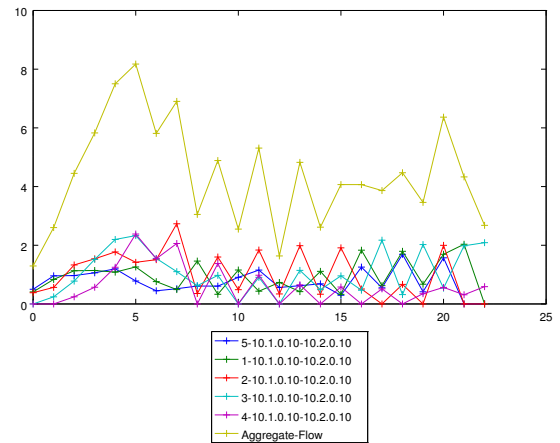


Fig. 20. Incoming delay

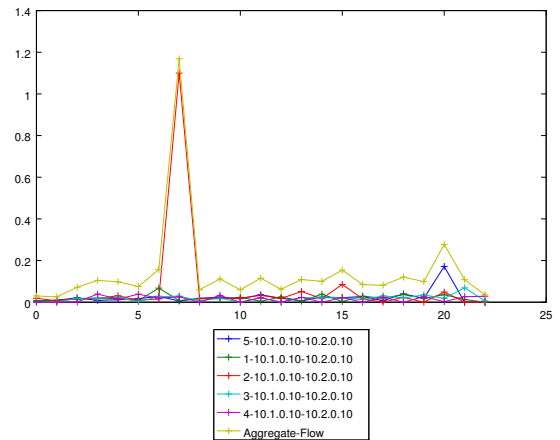


Fig. 21. Incoming jitter

VIII. APPENDIX. ACTIVE QUEUE MANAGEMENT RED MODULE STOCHASTIC MODEL CONSTRUCTION

The window change for each acknowledgement:

$$W_{n+1} = W_n + \frac{1}{W_n}.$$

For one-step processes stochastization we need to use the continuous time model. In round-trip time T there will be W_n acknowledgments:

$$\dot{W} = \frac{1}{T}.$$

The discrete equation of queue instantaneous length:

$$Q_{n+1} = Q_n + W_n - C_n.$$

Based on the discrete equation of queue instantaneous length the following equation is derived:

$$\dot{Q} = \frac{W(t)}{T(t)} - C(t).$$

Next, the behaviour of the exponentially weighted moving average queue length (the equation of connection between the source and the recipient) will be described.

The discrete recurrent equation of average queue length:

$$\hat{Q}(t_k + \delta) = (1 - w_q)\hat{Q}(t_k) + w_q Q(t_k).$$

From above mentioned discrete equation the continuous equation of average queue length is derived:

$$\dot{\hat{Q}} = -\frac{w_q}{\delta}\hat{Q} + \frac{w_q}{\delta}Q.$$

In order to consider the packets behavior in the system the kinetic equation, or as it is called the equation of interaction, will be presented. The number of packets is specified by the TCP window.

$$\begin{cases} 0 \xrightarrow{k_1} W, \\ W \xrightarrow{k_2} 0. \end{cases}$$

The first relation describes the appearance of packages in the system, the second describe departure of the packages from the system.

Based on the written equations and with the help of the method of constructing one-step processes the Fokker–Planck equation is derived:

$$\begin{aligned} \frac{\partial w}{\partial t} = & -\frac{\partial}{\partial W} \left[\left(\frac{1}{W} - \frac{W}{2} \frac{dN}{dt} \right) w \right] + \\ & + \frac{1}{2} \frac{\partial^2}{\partial W^2} \left[\left(\frac{1}{W} + \frac{W}{2} \frac{dN}{dt} \right) w \right]. \end{aligned}$$

From the Fokker–Planck equation the corresponding Langevin equation is obtained.

$$dW = \frac{1}{W}dt - \frac{W}{2}dN + \sqrt{\frac{1}{W} + \frac{W}{2} \frac{dN}{dt}}dV^1,$$

where dV^1 is the Wiener process corresponding to the random process $W(t)$.

Similarly, the behaviour of the queue length is described and interaction equations for the instantaneous queue length are given:

$$\begin{cases} 0 \xrightarrow{k_1^2} Q, \\ 0 \xrightarrow{k_2^2} Q. \end{cases}$$

The Fokker–Planck equation for the instantaneous queue length:

$$\frac{\partial q}{\partial t} = -\frac{\partial}{\partial Q} \left[\left(\frac{W}{T} - C \right) q \right] + \frac{1}{2} \frac{\partial^2}{\partial Q^2} \left[\left(\frac{W}{T} - C \right) q \right].$$

The Langevin equation for the instantaneous queue length:

$$dQ = \left(\frac{W}{T} - C \right) dt + \sqrt{\frac{W}{T} - C} dV^2,$$

where dV^2 is the Wiener process which corresponds to the random process Q .

According to the obtained equations the resulting system of the equations is given:

$$\begin{cases} dW = \frac{1}{T}dt - \frac{W}{2}dN + \sqrt{\frac{1}{T} + \frac{W}{2} \frac{dN}{dt}}dV^1, \\ dQ = \left(\frac{W}{T} - C \right) dQ + \sqrt{\frac{W}{T} - C}dV^2, \\ \frac{d\hat{Q}}{dt} = w_q C(Q - \hat{Q}). \end{cases}$$

The detailed stochastic model of the router RED-like control module is described in [4].

REFERENCES

- [1] A. V. Korolkova, D. S. Kulyabov, and A. I. Tchernovianov, "On the Classification of RED Algorithms," *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*, no. 3, pp. 34–46, 2009, in Russian.
- [2] A. V. Korolkova and D. S. Kulyabov, "Mathematical Model of the Dynamic Behavior of RED-Like System Parameters," *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*, no. 1, pp. 68–76, 2010, in Russian.
- [3] A. V. Korolkova and I. S. Zaryadov, "The mathematical model of the traffic transfer process with a rate adjustable by red," in *2010 International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, IEEE, Moscow, Russia: IEEE, October 18-20 2010, pp. 1046 – 1050.
- [4] T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, and B. A. dos Santos, "Model Queue Management on Routers," *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*, vol. 2, pp. 81–92, 2014, in Russian.
- [5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.
- [6] V. Misra, W.-B. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP-window-size Behavior," *Proceedings of IFIP WG 7.3 Performance*, vol. 99, 1999. [Online]. Available: <http://dna-pubs.cs.columbia.edu/citation/paperfile/24/Misra99-TCP-Stochastic.pdf>
- [7] —, "Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 151–160, 2000.
- [8] A. V. Demidova and D. S. Kulyabov, "Introduction of Self-Consistent Term in Stochastic Population Model Equation," *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*, no. 3, pp. 69–78, 2012, in Russian.
- [9] A. V. Demidova, M. N. Gevorkyan, A. D. Egorov, D. S. Kulyabov, A. V. Korolkova, and L. A. Sevastyanov, "Influence of Stochasticization on One-Step Models," *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*, no. 1, pp. 71–85, 2014, in Russian.
- [10] E. G. Eferina, A. V. Korolkova, M. N. Gevorkyan, D. S. Kulyabov, and L. A. Sevastyanov, "One-Step Stochastic Processes Simulation Software Package," *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"*, no. 3, pp. 46–59, 2014.
- [11] "Cisco Feature Navigator." [Online]. Available: www.cisco.com/go/cfn
- [12] C. Welsh, *GNS3 network simulation guide*. PACKT Publisher, 2013. [Online]. Available: <http://cds.cern.ch/record/1633716>