Error Generator for Communication Lines in Local Network Simulation Tool

Andrey V. Shabaldin, Dmitry M. Timchenko, *Tomsk State University, Tomsk*

***Abstract*—The paper presents a software tool for local network simulation. The tool includes GUI for constructing a network consisting of routers, application servers and switches connected to each other by communication lines. The simulator supports a number of widely-used protocols of TCP/IP stack. The network simulation tool also features a generator for errors and delays in communication lines that makes a data transmission process closer to real network environments. The error and delay generation algorithms are based on the Poisson distribution, that is proven to be adequate for wireless networks. The simulator is used for telecommunication system labs.**

***Index Terms*—Computer networks, error analysis, simulation software**

I. Introduction

THERE are a number of network simulators for computer networks. Some of them support specific hardware and provide an opportunity for a user to gain the experience working with high-cost devices without necessity of buying them [1]. Others are generic simulators that are intended for studying data flows and network protocols behavior in an environment of a modeled network before applying the network configuration into real infrastructure [2].

In the paper, we present a simulator which was rather developed for educational purposes. It contains a simple graphical user interface (GUI) and supports several basic protocols of TCP/IP stack. With GUI a user may construct network topology by placing device icons into the workspace and by connecting them with communication lines. Then a user can set a network configuration for each device: host name, IP address and network mask. More complex configuration can be performed from a console using command line that is available for every device. When the network configuration stage is completed a user may initialize the simulator. There are several generators for network packet (ping, traceroute, etc) that allow to execute data flows in the network. The results can be observed either through output of a packet generator or through Wireshark – the network packet analyzer (formerly Ethereal) [3].

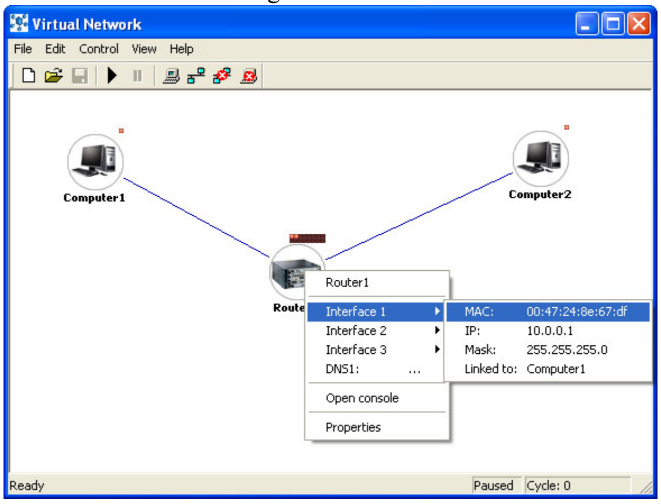
To make the network simulator more realistic, we add noise to communication lines. The reason is: given the environment where data losses and delays occur, it is much more interesting to observe communication protocols behavior, especially those which can detect and correct errors. If a student can control the error rate, he or she can learn how stable the data flow is for different noise levels and for different communication protocols.

For noise simulation we implemented error and delay generators. The noise generation algorithms are based on generating of pseudo-random Poisson-distributed values. Poisson distribution is known to adequately describe errors in wireless networks. Error rates in wired networks are lower compared to wireless networks; however, we assume that the error distribution in wired networks is close to the error distribution in wireless networks. We also use Poisson distribution to generate delays between adjacent packets, however, the nature of delay is not limited to noise in communication line, but also may appear due to network overload and other reasons.

The rest of the paper is organized as follows. We briefly overview the Network simulation tool in Section II. In Section III, we remind necessary definitions and formulate the hypothesis of error probability distribution in communication lines, while in Section IV we describe experimental results which confirm the selected hypothesis. In Section V the implementation details of error and delay generation algorithms are considered. The conclusion is outlined in Section VI.

II. Network Simulation Tool Overview

Network simulator is a software tool that allows a user to construct a network diagram which consists of network devices linked to each other with communication lines; to configure devices using GUI and console and to simulate packet data transmission. The screenshot of the main window of our tool is shown in Figure 1.



MAC: 00:47:24:8e:67:df
IP: 10.0.0.1
Mask: 255.255.255.0
Linked to: Computer1

Fig. 1. The main window screenshot.

Andrey V. Shabaldin, e-mail: psi@sah.tomsk.ru.
Dmitry M. Timchenko, e-mail: timchenko_d@mail.ru.
Tomsk State University, 36 Lenin Prospekt, Tomsk, 634050, Russia

There are 3 basic types of devices: a *computer*, a *router* and a *switch*. A *computer* may host network services, such as: DNS server, DHCP server. It has exactly one external network interface, so it cannot route or switch packets.

A *router*, a device that has up to 16 external network interfaces, is used to route packets between subnets attached to its interfaces.

A *switch* works as a "dumb" switch with the automatic content-addressable memory filling. Number of ports can vary from 5 to 32.

A user of the network simulator also can take an advantage of using helper devices. There are two types of them: an *external connection helper* and an *ethereal file writer*. The network simulator allows a user to construct shared network infrastructure with network segments on different physical PCs. To connect the segments to each other *external connection helper* is used. It redirects every received packet to other physical computer over UDP socket. In another PC this packet is received and is directed into a network segment attached to the corresponding *external connection helper*. Consequently we obtain two network segments on different physical computers connected to each other by two external connection helper devices (Figure 2).

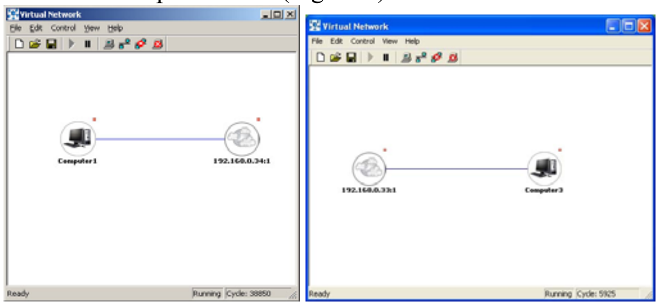


Fig. 2. Two network segments on different physical computers connected with external connection helpers.

When the network configuration is completed, the simulator could be initialized by pressing the "play" button at the top toolbar. This turns on the network traffic between network devices.

The device configuration can be performed in two ways: by using GUI or through console tools. With GUI a user can set IP address, subnet mask, specify DNS servers and default gateway for computers. All other settings may be configured using console tools. Console opens when corresponding device is clicked with the left mouse button. Just now there are three configuration tools available:

-- route – to set-up routing tables for a computer or for a router;

-- arp – to query or to modify ARP tables; each row maps an internet address into Ethernet MAC address;

-- iptables – a tool by which a user creates rules for packet filtering.

Packets flow synchronously, one packet per single communication line for each cycle. Each device has its own buffers for incoming and outgoing packets. After a packet arrives to the incoming buffer, it is being processed by protocol modules and network services. The order in which packets are processed is determined by protocol stack which is specific for each device type. Actually there are two protocol stacks: input stack for incoming packets and output stack for outgoing packets. They may differ, for example, routing module on a computer does not process incoming packets from other devices, though it forwards outgoing packets to local interfaces according to their destination addresses. Figure 3 shows the input protocol stack for a computer that hosts DNS server.

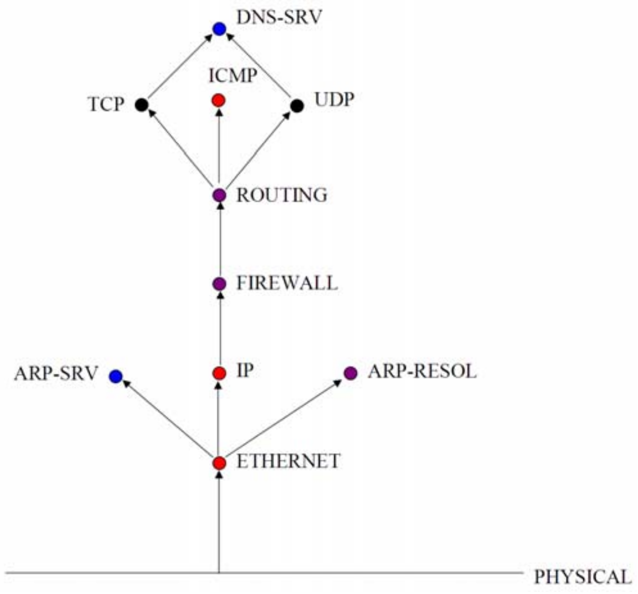


Fig. 3. The input protocol stack for DNS-server.

To simulate packet transmission several console tools can be used:

-- ping – sends ICMP echo requests to target device and listens for ICMP echo response replies. It also estimates round-trip-time in cycles, records any packet losses, and prints a statistical summary when finished.

-- traceroute – is the tool to find a path taken by a packet from a source to a destination device.

-- nslookup – is used to query DNS servers to find various details relating to DNS: IP addresses, MX records for a domain, name servers of a domain, etc.

The network simulator tool has plug-in architecture. The core consists of GUI and application programming interface (API) to interact with external plug-in modules. Plug-in modules are responsible for all other functionality. Consequently, it is possible to implement new devices, protocols and network services without rebuilding the tool.

In our paper, we describe the implementation of a new device that adds errors and delays to communication lines. To develop an algorithm for error and delay generation, first we should select the proper error distribution function and estimate its parameters.

III. Selection of Error and Delay Distribution

According to [4] possible sources of errors in wireless networks are the following:

-- signal attenuation;

-- front end overload: if a very powerful transmitter of one frequency band is near a receiver of another band, the transmitter may overwhelm filters in the receiver;

-- wave interference;

-- wave dispersion;

-- motion: if two communicating objects are moving with respect to each other, the frequency of the electromagnetic energy is being changed according to the Doppler effect.

First, we should choose the function that corresponds to the distribution of errors appearing from sources listed above. Then the choice should be proven experimentally.

We assume that:

-- t_i i=1, 2, …, N – time instances when errors occur are statistically independent;

-- the probability of an error in Δt time lapse is proportional to Δt;

-- the probability of two or more errors in Δt time lapse is a higher order quantity than Δt.

According to these assumptions, the Poisson distribution function could be chosen:

$$P(N,T)=\frac{(\gamma T)^N}{N!}e^{-\gamma T},$$

where $P(N, T)$ is the probability of the occurrences of N errors for the time lapse T;
γ is a distribution parameter that describes the average error number per a time lapse.

Given the frequency distribution function $f(t)$, the probability of the error absence in the time lapse T can be determined as

$$P(0,T)=P\{t \geq T\}=\int_T^\infty f(t)dt \;=e^{-\gamma T}.$$

After the differentiation we obtain the frequency distribution function

$$f(t)=\gamma\, e^{-\gamma t} \qquad (1)$$

where the average time lapse between error occurrences equals to

$$<t>=\int_0^\infty tf(t)dt=\frac{1}{\gamma}$$

IV. Experimental Proof of Poisson Distribution Adequacy for Describing Errors in Communication Lines

The experiments were performed in the following way. Each second we sent 32-byte ICMP echo request to remote host over WiFi network, and tracked ICMP echo replies. Overall experiment time was 6-8 hours. After that we constructed the distribution of time lapses between neighboring errors. By error we meant the expiration of maximum waiting time (5 seconds) for an echo reply. The resulting histograms are shown in Figure 4.

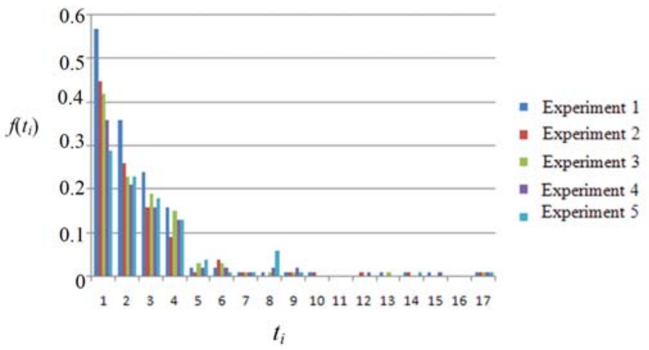


Fig. 4. The error frequency distribution of time lapses between errors in a wireless network.

According to the results shown in Figure 4 it can be assumed that the frequencies of time lapses between errors are distributed by decreasing exponential function. Since Poisson distribution is decreasing exponential function as well, in order to prove that the function in Figure 4 is a Poisson distribution we have calculated $<t>$ – the average time lapse between error occurrences. Then we have constructed a graph of the Poisson frequency distribution function (1). For each point in this graph dispersion were estimated. As the summary, root mean square derivations (RMS) between theoretical (function (1)) and experimental results were calculated. The results are shown in Table 1.

TABLE I
Experimental and Theoretical Results Comparison

Experiment number	<t>	*RMS*
1	38.46	4.6
2	15.87	2.3
3	33.33	3.3
4	20.40	5.0
5	58.82	6.2

V. Error and Delay Generation Algorithms

There are a number of methods of stochastic process simulation using pseudo-random generator on PC. Pseudo-random generator outputs the set of uniformly-distributed random values. So the challenge is to convert this set to arbitrary-distributed values. This can be done with non-linear transformation [5].

Let R be the probability of the error absence for the time lapse T and R is a value generated using pseudo-random generator, $0 < R < 1$. According to (1): R=$P(0,T)=e^{-\gamma T}$, or $R_i=e^{-\gamma Ti}$

Consequently, the time lapse between errors t_i could be derived as:

$$t_i=[\frac{\ln(\frac{1}{R_i})}{\gamma}] \quad (2)$$

where square brackets denote the integer part of a value.

Error generation algorithm is as follows:

1) t_0:=0, i:=1.

2) A user assigns γ – a distribution parameter that describes the average error number per time lapse.

3) $R_i \in (0, 1)$ is a random value.

4) Calculate the offset t_i using (2).

5) Invert the bit with the offset t_i.

6) i:=i+1.

7) Repeat steps 3–6 until the communication session is finished.

Delay generation algorithm is as follows:

1) t_0:=0, i:=1.

2) A user assigns γ – a distribution parameter that describes the average error number per time lapse and *max* – the maximum delay for a cycle.

3) $R_i \in (0, 1)$, $D_i \in (0, max)$ are random values.

4) Calculate the offset t_i using (2).

5) Delay the t_i-th packet for D_i cycles.

6) i:=i+1;

7) Repeat steps 3–6 until the communication session is finished.

VI. Conclusion

We have currently developed the network simulation tool that features scalable plug-in architecture. It allows us to extend the functionality of our tool by adding new protocol, device or network service modules. Currently the work is going on with implementing the device that adds a noise to communication lines. It will be able to randomly generate errors and delays with the Poisson distribution. A user will have an opportunity to configure the noise level by changing average error number per time lapse. In the meantime, we already have derived the proper error distribution, have developed the algorithm for error and delay generation. Now we are implementing GUI and core for error-generating plug-in module. When plug-in is finished it will be used with the network simulation tool to make the data flow simulation more realistic.

References

[1] Boson NetSim :: Cisco Network/Router Simulation Software [Online]. Available: http://www.boson.com/AboutNetSim.html.

[2] Network Emulator 3.0 [Online]. Available: http://lionet.info/ne/ne3/.

[3] Wireshark – free packet sniffer computer application [Online]. Available: http://www.wireshark.org.

[4] D. Eckhardt, P. Steenkiste, “Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network,” in proc. of ACM SIGCOMM '96 Conference, pp 243--254.

[5] Пономарев Г.А., Пономарева В.Н., Якубов В.П. Статистические методы в радиофизике: практикум с применением диалого-вычислительных комплексов. –Томск 1989.