# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук Кафедра теории вероятностей и кибербезопасности**

# ОТЧЕТ

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

*дисциплина: Моделирование информационных процессов*

Студент: Маслова Анастасия

Группа: НКНбд-01-21

**МОСКВА**

2024 г

Цель лабораторной работы: исследование протокола TCP и алгоритма управления очередью RED.

Постановка задачи:

1. Описание моделируемой сети:

– сеть состоит из 6 узлов;

– между всеми узлами установлено дуплексное соединение с различными пропускной способностью и задержкой 10 мс (см. рис. 2.4);

– узел r1 использует очередь с дисциплиной RED для накопления пакетов, максимальный размер которой составляет 25;

– TCP-источники на узлах s1 и s2 подключаются к TCP-приёмнику на узле s3;

– генераторы трафика FTP прикреплены к TCP-агентам.

1. – Измените в модели на узле s1 тип протокола TCP с Reno на NewReno, затем на Vegas. Сравните и поясните результаты.

– Внесите изменения при отображении окон с графиками (измените цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

Выполнение работы:

Для выполнения первого задания я использовала следующий код:

# создание объекта Simulator

set ns [new Simulator]

# Узлы сети:

set N 5

for {set i 1} {$i < $N} {incr i} {

set node\_(s$i) [$ns node]

}

set node\_(r1) [$ns node]

set node\_(r2) [$ns node]

# Соединения:

$ns duplex-link $node\_(s1) $node\_(r1) 10Mb 2ms DropTail

$ns duplex-link $node\_(s2) $node\_(r1) 10Mb 3ms DropTail

$ns duplex-link $node\_(r1) $node\_(r2) 1.5Mb 20ms RED

$ns queue-limit $node\_(r1) $node\_(r2) 25

$ns queue-limit $node\_(r2) $node\_(r1) 25

$ns duplex-link $node\_(s3) $node\_(r2) 10Mb 4ms DropTail

$ns duplex-link $node\_(s4) $node\_(r2) 10Mb 5ms DropTail

# Агенты и приложения:

set tcp1 [$ns create-connection TCP/Reno $node\_(s1) TCPSink $node\_(s3) 0]

$tcp1 set window\_ 15

set tcp2 [$ns create-connection TCP/Reno $node\_(s2) TCPSink $node\_(s3) 1]

$tcp2 set window\_ 15

set ftp1 [$tcp1 attach-source FTP]

set ftp2 [$tcp2 attach-source FTP]

# Мониторинг размера окна TCP:

set windowVsTime [open WindowVsTimeReno w]

set qmon [$ns monitor-queue $node\_(r1) $node\_(r2) [open qm.out w] 0.1];

[$ns link $node\_(r1) $node\_(r2)] queue-sample-timeout;

# Мониторинг очереди:

set redq [[$ns link $node\_(r1) $node\_(r2)] queue]

set tchan\_ [open all.q w]

$redq trace curq\_

$redq trace ave\_

$redq attach $tchan\_

# Добавление at-событий:

$ns at 0.0 "$ftp1 start"

$ns at 1.1 "plotWindow $tcp1 $windowVsTime"

$ns at 3.0 "$ftp2 start"

$ns at 10 "finish"

# Формирование файла с данными о размере окна TCP:

proc plotWindow {tcpSource file} {

global ns

set time 0.01

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file"

}

# Процедура finish:

proc finish {} {

global tchan\_

# подключение кода AWK:

set awkCode {

{

if ($1 == "Q" && NF>2) {

print $2, $3 >> "temp.q";

set end $2

}

else if ($1 == "a" && NF>2)

print $2, $3 >> "temp.a";

}

}

set f [open temp.queue w]

puts $f "TitleText: red"

puts $f "Device: Postscript"

if { [info exists tchan\_] } {

close $tchan\_

}

exec rm -f temp.q temp.a

exec touch temp.a temp.q

exec awk $awkCode all.q

puts $f \"queue

exec cat temp.q >@ $f

puts $f \n\"ave\_queue

exec cat temp.a >@ $f

close $f

# Запуск xgraph с графиками окна TCP и очереди:

exec xgraph -bb -tk -x time -t "TCPRenoCWND" WindowVsTimeReno &

exec xgraph -bb -tk -x time -y queue temp.queue &

exit 0

}

$ns run

В результате я получила данный результат:

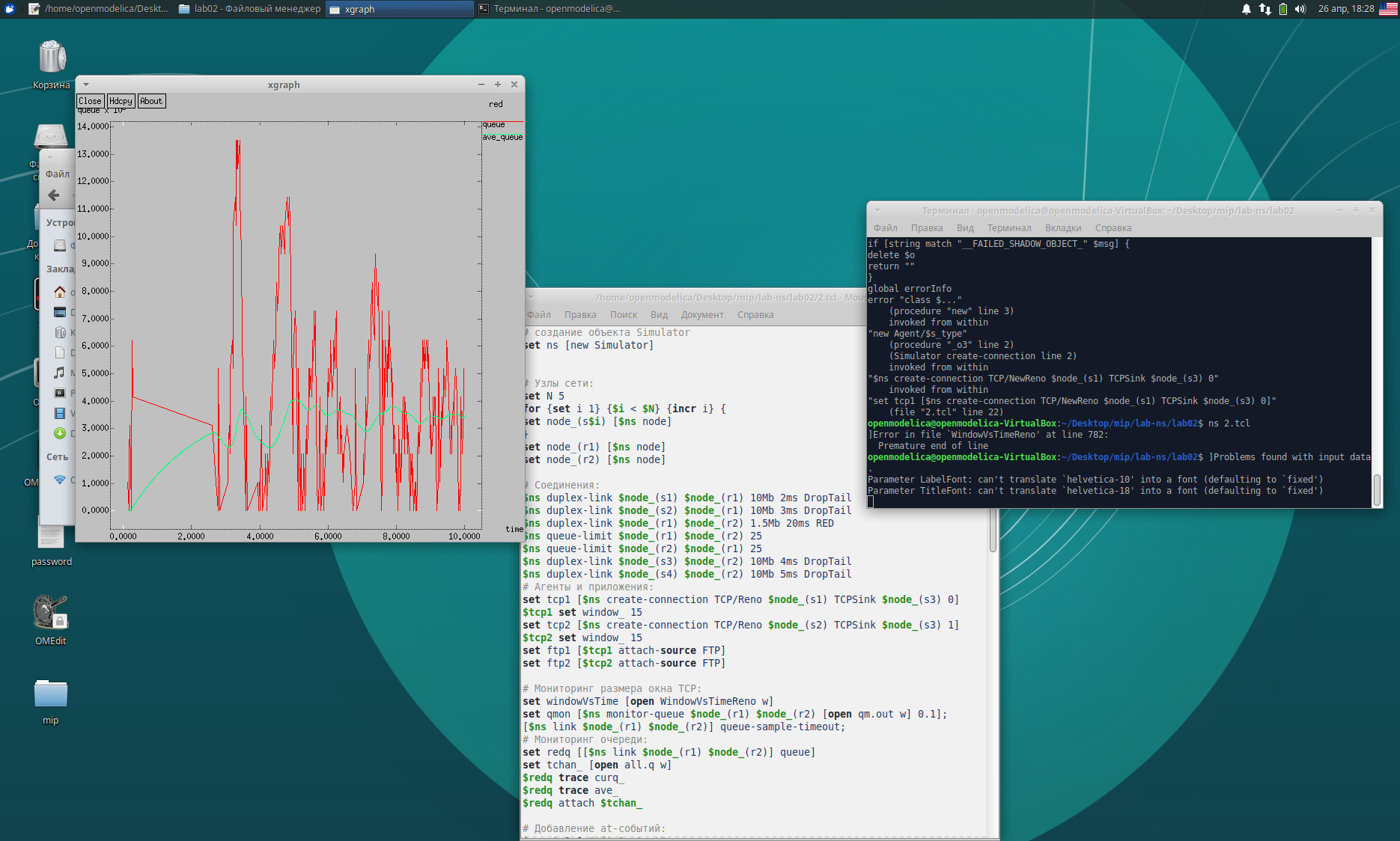


рис. Результат работы кода

Для второго задания я использовала следующий код:

# создание объекта Simulator

set ns [new Simulator]

# Узлы сети:

set N 5

for {set i 1} {$i < $N} {incr i} {

set node\_(s$i) [$ns node]

}

set node\_(r1) [$ns node]

set node\_(r2) [$ns node]

# Соединения:

$ns duplex-link $node\_(s1) $node\_(r1) 10Mb 2ms DropTail

$ns duplex-link $node\_(s2) $node\_(r1) 10Mb 3ms DropTail

$ns duplex-link $node\_(r1) $node\_(r2) 1.5Mb 20ms RED

$ns queue-limit $node\_(r1) $node\_(r2) 25

$ns queue-limit $node\_(r2) $node\_(r1) 25

$ns duplex-link $node\_(s3) $node\_(r2) 10Mb 4ms DropTail

$ns duplex-link $node\_(s4) $node\_(r2) 10Mb 5ms DropTail

# Агенты и приложения:

set tcp1 [$ns create-connection TCP/Vegas $node\_(s1) TCPSink $node\_(s3) 0]

$tcp1 set window\_ 15

set tcp2 [$ns create-connection TCP/Reno $node\_(s2) TCPSink $node\_(s3) 1]

$tcp2 set window\_ 15

set ftp1 [$tcp1 attach-source FTP]

set ftp2 [$tcp2 attach-source FTP]

# Мониторинг размера окна TCP:

set windowVsTime [open WindowVsTimeVegas w]

set qmon [$ns monitor-queue $node\_(r1) $node\_(r2) [open qm.out w] 0.1];

[$ns link $node\_(r1) $node\_(r2)] queue-sample-timeout;

# Мониторинг очереди:

set redq [[$ns link $node\_(r1) $node\_(r2)] queue]

set tchan\_ [open all.q w]

$redq trace curq\_

$redq trace ave\_

$redq attach $tchan\_

# Добавление at-событий:

$ns at 0.0 "$ftp1 start"

$ns at 1.1 "plotWindow $tcp1 $windowVsTime"

$ns at 3.0 "$ftp2 start"

$ns at 10 "finish"

# Формирование файла с данными о размере окна TCP:

proc plotWindow {tcpSource file} {

global ns

set time 0.01

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file"

}

# Процедура finish:

proc finish {} {

global tchan\_

# подключение кода AWK:

set awkCode {

{

if ($1 == "Q" && NF>2) {

print $2, $3 >> "temp.q";

set end $2

}

else if ($1 == "a" && NF>2)

print $2, $3 >> "temp.a";

}

}

set f [open temp.queue w]

puts $f "TitleText: red"

puts $f "Device: Postscript"

if { [info exists tchan\_] } {

close $tchan\_

}

exec rm -f temp.q temp.a

exec touch temp.a temp.q

exec awk $awkCode all.q

puts $f \"queue

exec cat temp.q >@ $f

puts $f \n\"ave\_queue

exec cat temp.a >@ $f

close $f

# Запуск xgraph с графиками окна TCP и очереди:

exec xgraph -bb -tk -x time -t "TCPVegasCWND" WindowVsTimeVegas &

exec xgraph -bb -tk -x time -y queue temp.queue &

exit 0

}

$ns run

Меняя код в некоторых местах для NewReno и Vegas, я получила следующие результаты:

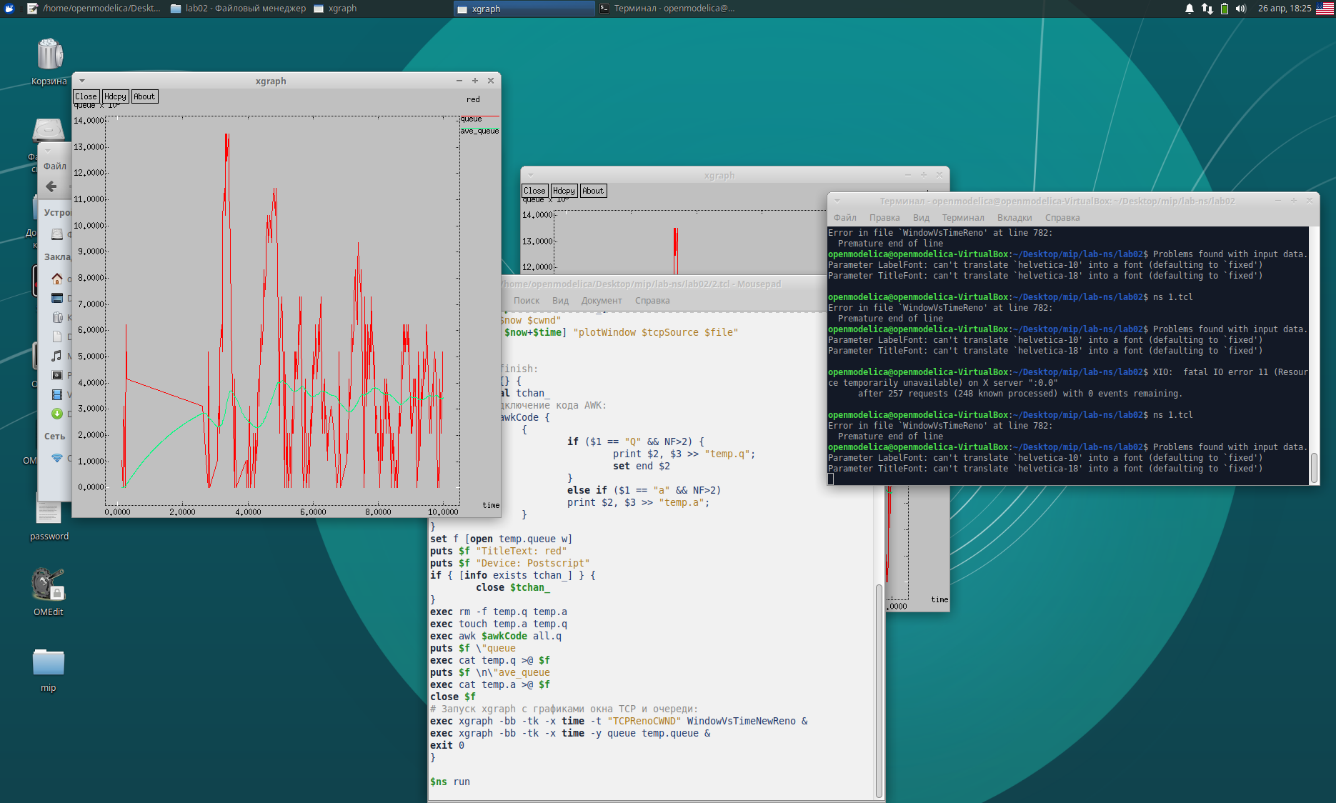


рис. Результат работы кода с типом протокола TCP NewReno

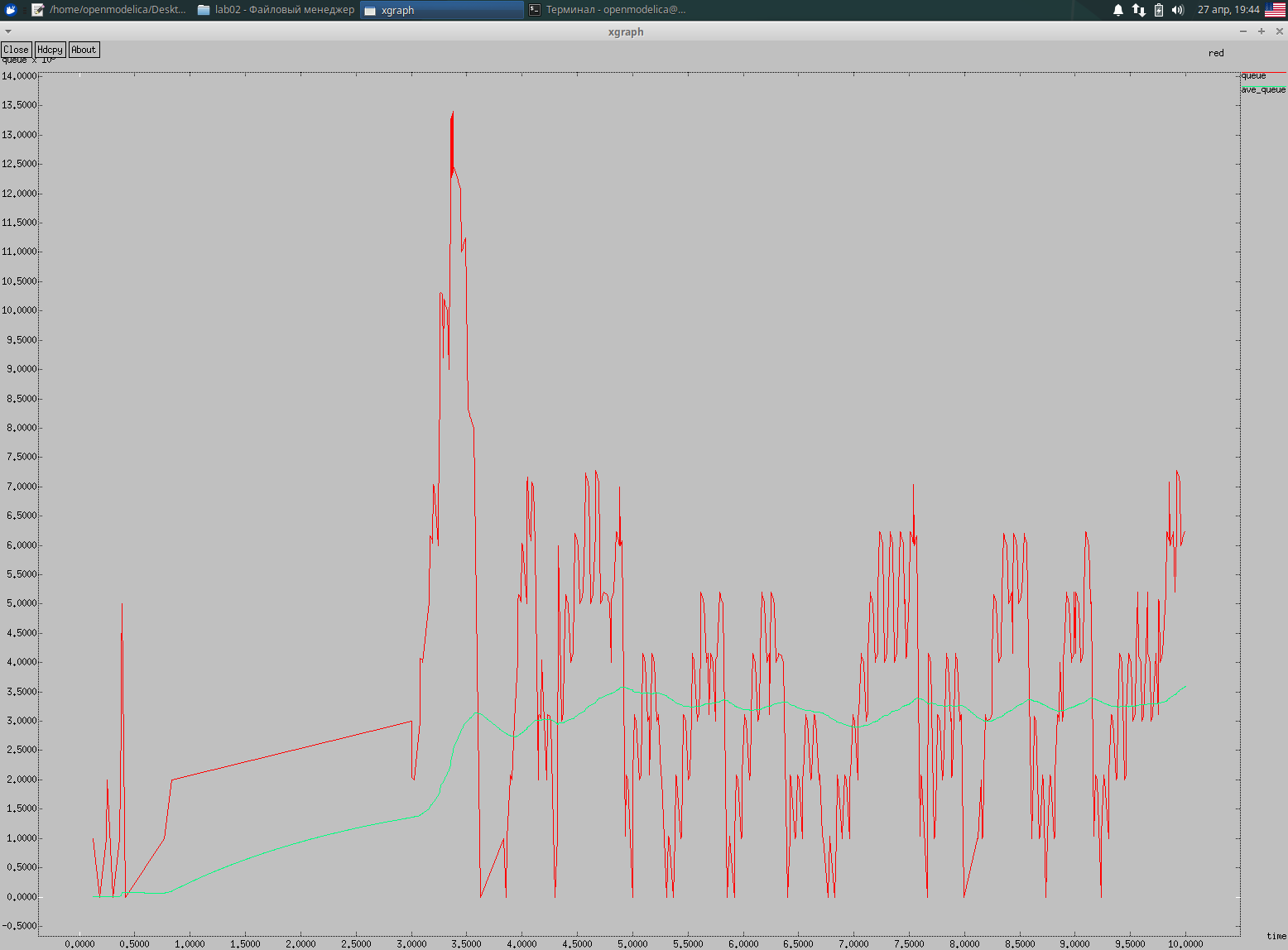


рис. Результат работы кода с типом протокола TCP Vegas

Вывод: в ходе работы я исследовала протокол TCP и алгоритм управления очередью RED.