Лабораторная работа №2.

Исследование протокола TCP и алгоритма управления очередью RED

Александр Андреевич Шуплецов

Содержание

# 1 Цель работы

Приобретение навыков моделирования сетей передачи данных с помощью средcтва имитационного моделирования NS-2 с дисцплиной RED, а также анализ полученных результатов моделирования.

# 2 Выполнение работы

1. Создадим файл для лабораторной работы 2, пример с дисциплиной RED.

#создание объекта Simulator  
set ns [new Simulator]  
  
#открытие на запись файла out.nam для визуализатора nam  
set nf [open out.nam w]  
  
#все результаты моделирования будут записаны в переменную nf  
$ns namtrace-all $nf  
  
#открытие на запись файла трассировки out.tr  
#для регистрации всех событий  
set f [open out.tr w]  
#все регистрируемые события будут записаны в переменную f  
$ns trace-all $f  
  
# 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  
 # выполнение кода AWK  
 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  
 }  
  
# 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"  
}  
  
# 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\_  
  
#at-событие для планировщика событий, которое запускает  
#процедуру finish через 5 с после начала моделирования  
# 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"  
#запуск модели  
$ns run

1. Запустим файл примера лабораторной работы.

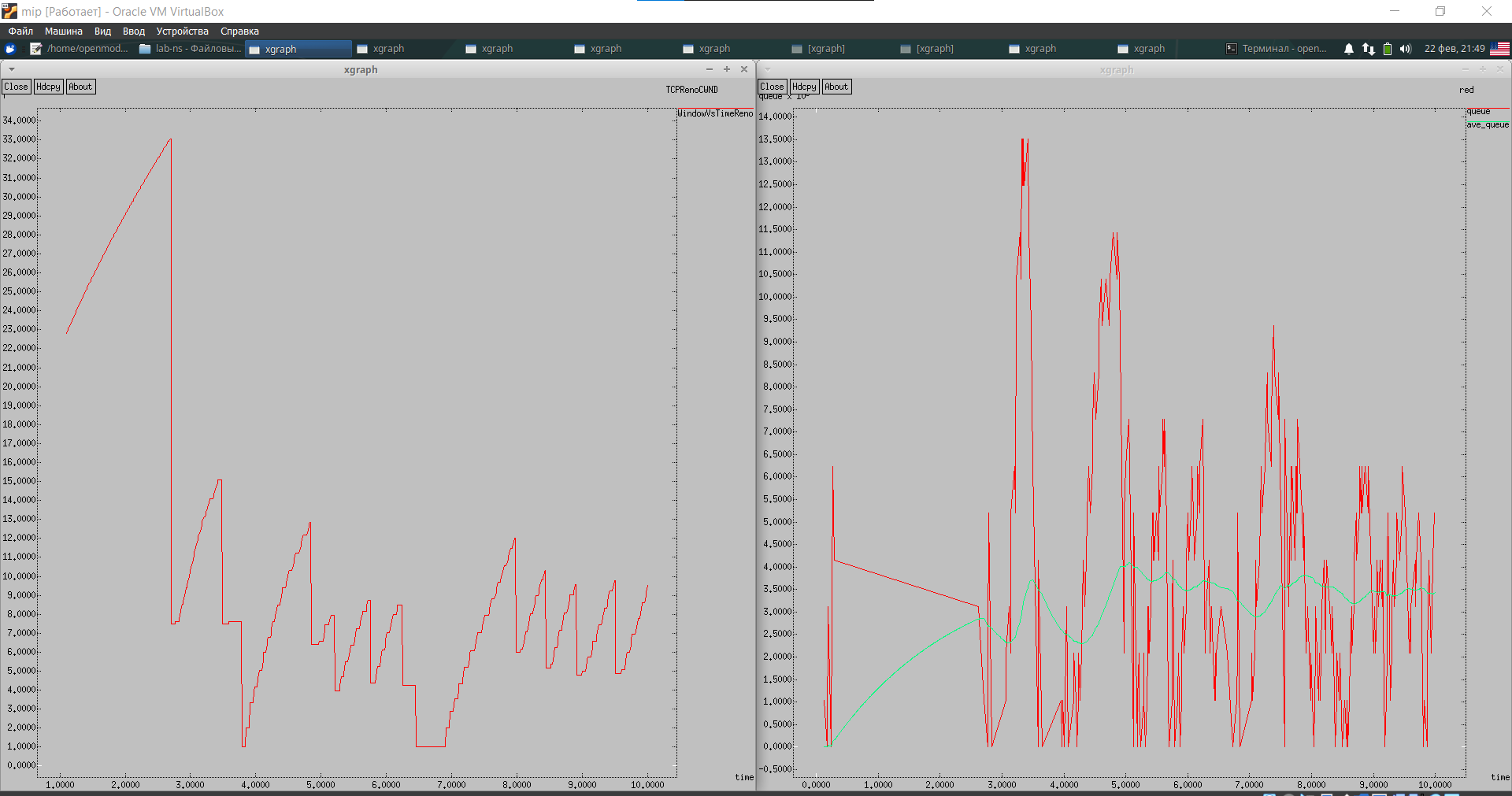


Figure 1: вывод файла лабораторной работы

1. Изменим тип протокола с Reno на Newreno.

Агенты и приложения:  
set tcp1 [$ns create-connection TCP/Newreno $node\_(s1) TCPSink $node\_(s3) 0]  
$tcp1 set window\_ 15  
set tcp2 [$ns create-connection TCP/Newreno $node\_(s2) TCPSink $node\_(s3) 1]  
$tcp2 set window\_ 15  
set ftp1 [$tcp1 attach-source FTP]  
set ftp2 [$tcp2 attach-source FTP]

1. Запустим файл с протоколом Newreno.

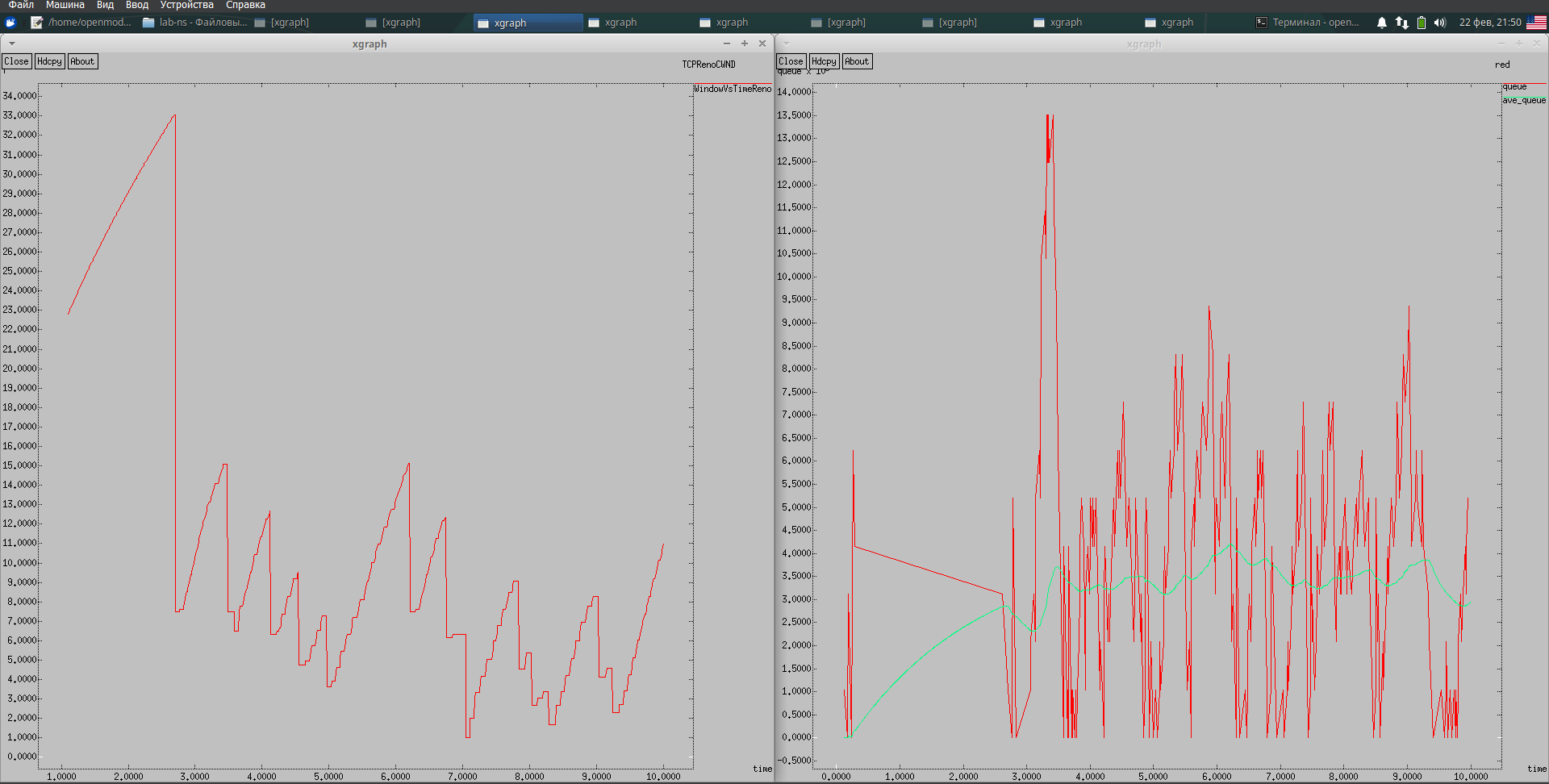


Figure 2: вывод с Newreno

1. Изменим тип протокола на Vegas.

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/Vegas $node\_(s2) TCPSink $node\_(s3) 1]  
$tcp2 set window\_ 15  
set ftp1 [$tcp1 attach-source FTP]  
set ftp2 [$tcp2 attach-source FTP]

1. Запустим файл с протоколом Vegas.

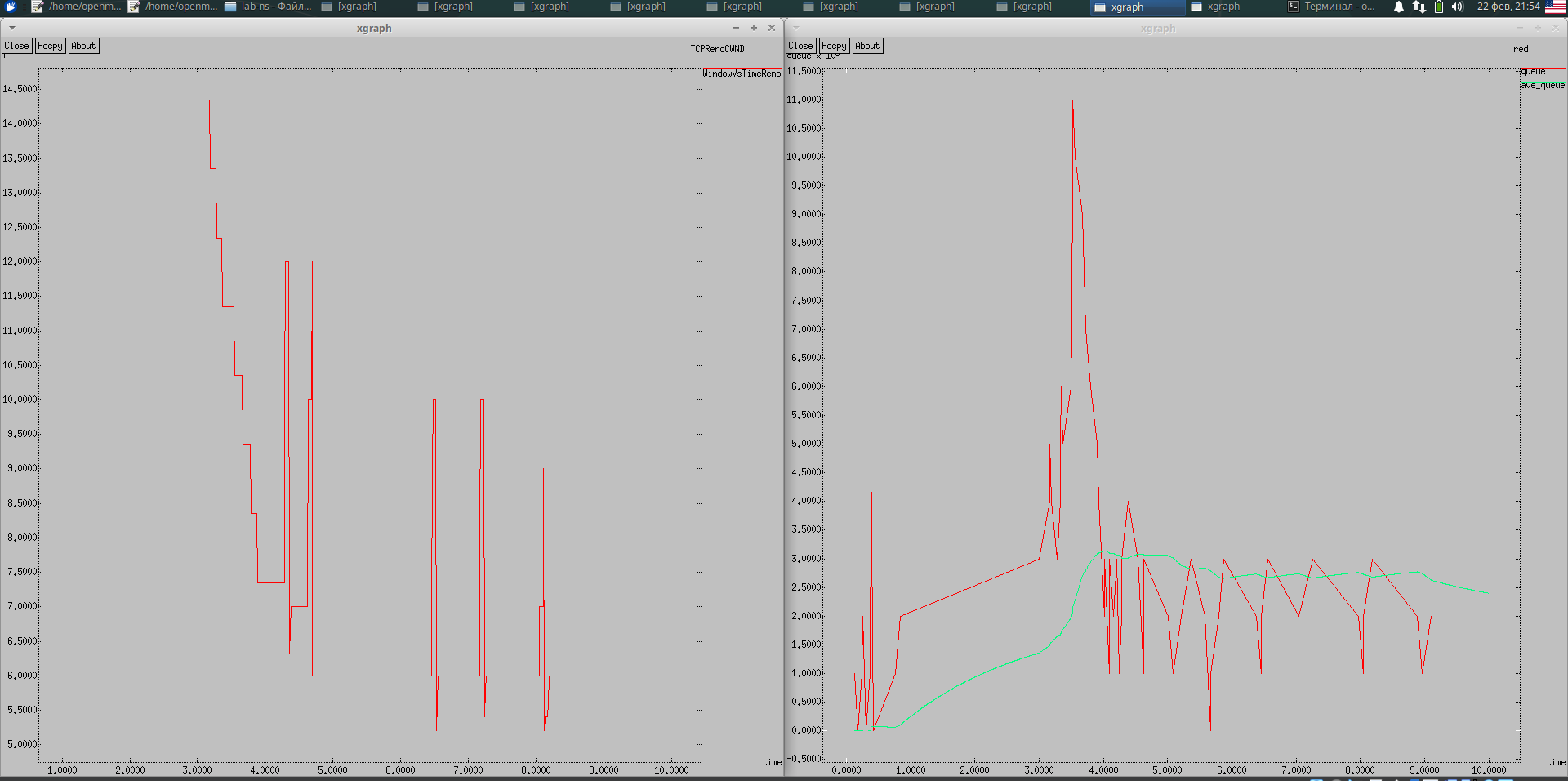


Figure 3: вывод с Vegas

1. Изменим цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде.

#создание объекта Simulator  
set ns [new Simulator]  
  
#открытие на запись файла out.nam для визуализатора nam  
set nf [open out.nam w]  
  
#все результаты моделирования будут записаны в переменную nf  
$ns namtrace-all $nf  
  
#открытие на запись файла трассировки out.tr  
#для регистрации всех событий  
set f [open out.tr w]  
#все регистрируемые события будут записаны в переменную f  
$ns trace-all $f  
  
# 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"  
puts $f "0.Color: Red"  
puts $f "1.Color: Blue"  
if { [info exists tchan\_] } {  
 close $tchan\_  
}  
exec rm -f temp.q temp.a  
exec touch temp.a temp.q  
 # выполнение кода AWK  
exec awk $awkCode all.q  
puts $f \"queue"  
exec cat temp.q >@ $f  
puts $f \n\"avg\_queue"  
exec cat temp.a >@ $f  
close $f  
# Запуск xgraph с графиками окна TCP и очереди:  
exec xgraph -fg white -bg black -bb -tk -x vremya -t "TCPRenoCWND" WindowVsTimeReno &  
exec xgraph -fg white -bg black -bb -tk -x vremya -y ochered temp.queue &  
}  
# Формирование файла с данными о размере окна TCP:  
proc plotWindow {tcpSource file} {  
 global ns  
 set time 0.01  
 set now [$ns now]  
 set cwnd [$tcpSource set cwnd\_]  
 puts $file "$now $cwnd"  
 $ns at [expr $now+$time] "plotWindow $tcpSource $file"  
}  
  
# 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]  
puts $windowVsTime "0.Color: Yellow"  
puts $windowVsTime \"window\_shape\_dinamic  
# Мониторинг очереди:  
set redq [[$ns link $node\_(r1) $node\_(r2)] queue]  
set tchan\_ [open all.q w]  
$redq trace curq\_  
$redq trace ave\_  
$redq attach $tchan\_  
  
#at-событие для планировщика событий, которое запускает  
#процедуру finish через 5 с после начала моделирования  
# 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"  
#запуск модели  
$ns run

1. Выведем на экран результаты измененного файла.

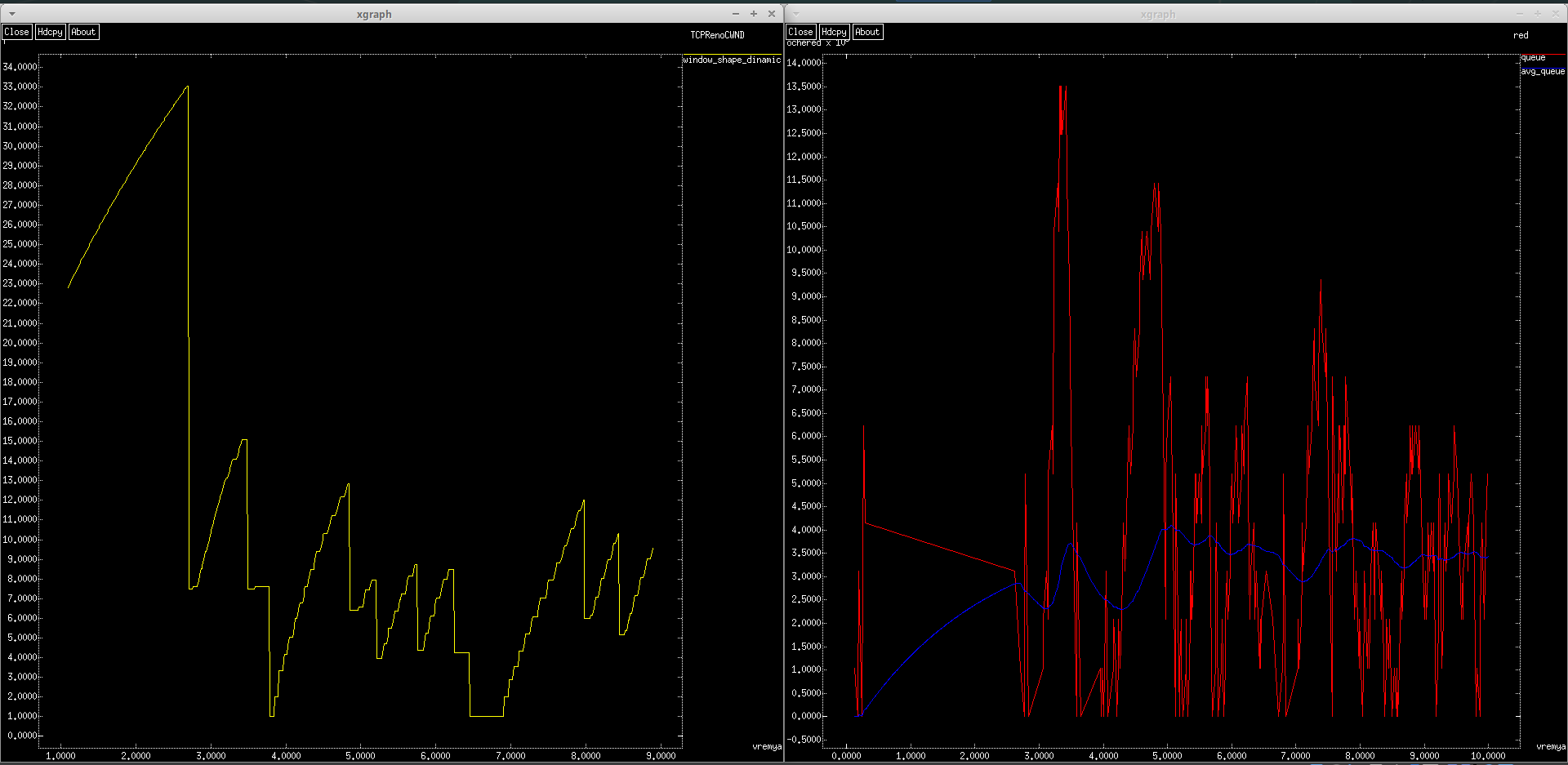


Figure 4: вывод с новыми параметрами

# 3 Выводы

Я приобрел навыки моделирования сетей передачи данных с помощью средcтва имитационного моделирования NS-2 и примера с дисциплиной RED, а также сделал анализ полученных результатов моделирования.

# Список литературы

Королькова А. В., Кулябов Д.С. “Материалы к лабораторным работам”