

Лабораторная работа 2

Исследование протокола TCP и алгоритма управления очередью RED

Ланцова Я. И.

Российский университет дружбы народов, Москва, Россия

Информация

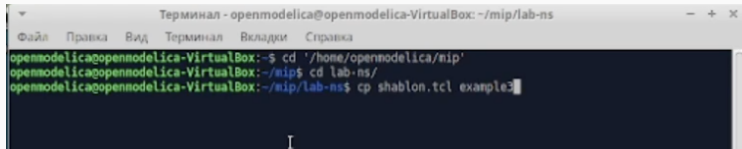
- Ланцова Яна Игоревна
- студентка
- Российский университет дружбы народов

Исследовать протокол TCP и алгоритм управления очередью RED.

1. Выполнить пример с дисциплиной RED;
2. Изменить в модели на узле s1 тип протокола TCP с Reno на NewReno, затем на Vegas.
Сравнить и пояснить результаты;
3. Внести изменения при отображении окон с графиками (изменить цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

Выполнение лабораторной работы

Выполнение лабораторной работы



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
openmodelica@openmodelica-VirtualBox:~$ cd '/home/openmodelica/mip'
openmodelica@openmodelica-VirtualBox:~/mip$ cd lab-ns/
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example3
```

Рис. 1: Перейдем в рабочую директорию и скопируем содержимое созданного шаблона в новый файл:

- сеть состоит из 6 узлов;
- между всеми узлами установлено дуплексное соединение с различными пропускной способностью и задержкой 10 мс;
- 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.

Выполнение лабораторной работы

```
~/home/openmodelica/mip/lab-ns/example6.tcl - Mousepad
Файл Правка Поиск Вид Документ Справка

set ns [new Simulator]

set of [open out.nam w]
$ns namtrace-all $nf

set f [open out.tr w]
$ns trace-all $f

# УЗЛЫ сети:
set n 5
for {set i 1} {$i < $n} {incr i} {
    set node [$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 FFD
$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 windowSize [open WindowSizeTimeReno w]
set qon [$ns monitor-queue $node.(r1) $node.(r2)
[open qn.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 avg
$redq attach $tchan
```

Рис. 2: Напишем сценарий, реализующий модель согласно описанию

Выполнение лабораторной работы

```
- /home/sergeymelnicov/mip/lab-ns/example4.tcl - Mousepad
File Edit View Window Help
$ns at 0.0 "sttp1 start"
$ns at 1.1 "plotWindow stcp1 $windowVstTime"
$ns at 3.0 "sttp2 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 == "q" && 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.q temp.a
    exec awk $awkCode all.q & # выполнение кода AWK
    puts $f "\nqueue"
    exec cat temp.q >& $f
    puts $f "\nave_queue"
    exec cat temp.a >& $f
    close $f

    # Запуск xgraph с графиком окна TCP и очереди:
    exec xgraph -bb -tk -x time -t "TCPWndCwnd" WindowVstTimeReno &
    exec xgraph -bb -tk -x time -y queue temp.queue &
    exit 0
}
```

Рис. 3: Изменим процедуру finish

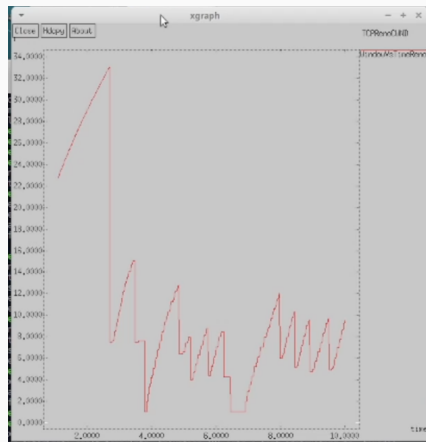


Рис. 4: График динамики размера окна TCP

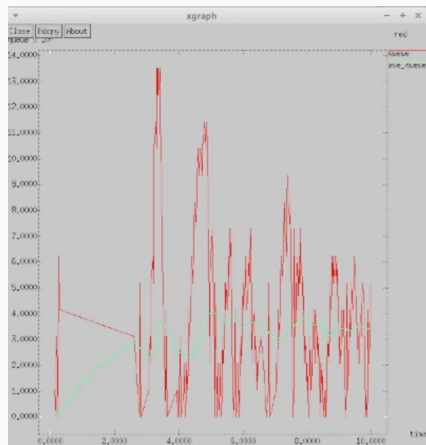


Рис. 5: График динамики длины очереди и средней длины очереди

```
# 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/Reno $node_(s2) TCPSink $node_(s3) 1]  
$tcp2 set window_ 15  
set ftp1 [$tcp1 attach-source FTP]  
set ftp2 [$tcp2 attach-source FTP]
```

Рис. 6: Сначала требуется изменить тип протокола TCP Reno на NewReno. Для этого изменим код:

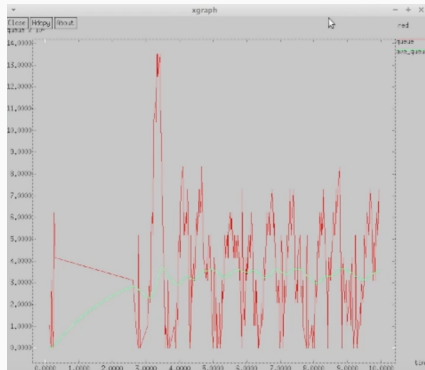


Рис. 7: График динамики длины очереди и средней длины очереди

```
# 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]
```

Рис. 8: Теперь требуется изменить тип протокола TCP NewReno на Vegas. Для этого изменим код:

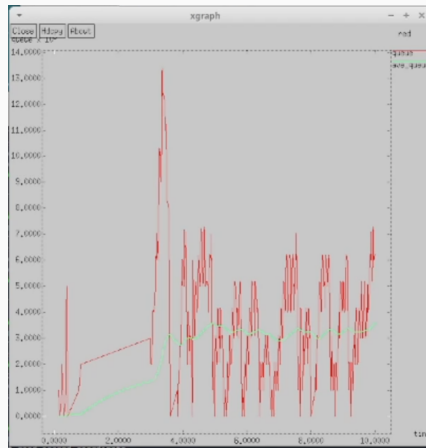


Рис. 9: График динамики длины очереди и средней длины очереди

Выполнение лабораторной работы

```
# 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: Pink"
    puts $f "Device: Postscript"
    puts $f "0.Color: Pink"
    puts $f "1.Color: Blue"
    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 \[AAA\]
    exec cat temp.q >@ $f
    puts $f \n\BBB\
    exec cat temp.a >@ $f
    close $f

    # Запуск xgraph с графиками окна TCP и очереди:
    exec xgraph -fg white -bg black -bb -tk -x time -t "TCP RenoCwnd" WindowVstimeReno &
    exec xgraph -fg white -bg black -bb -tk -x time -y queue temp.queue &
    exit 0
}
```

Рис. 10: Внесем изменения при отображении окон с графиками, изменим цвет фона, и т.д.

```
# Мониторинг размера окна TCP:
set windowVsTime [open WindowVsTimeReno w]
puts $windowVsTime "0.Color: Pink"
puts $windowVsTime \ "Window Size"
set qmon [$ns monitor-queue $node_(r1) $node_(r2) [open qm.out w] 0.1];
[$ns link $node_(r1) $node_(r2)] queue-sample-timeout;
```

Рис. 11: В разделе мониторинга размера окна TCP также изменим цвет траектории и подпись легенды.

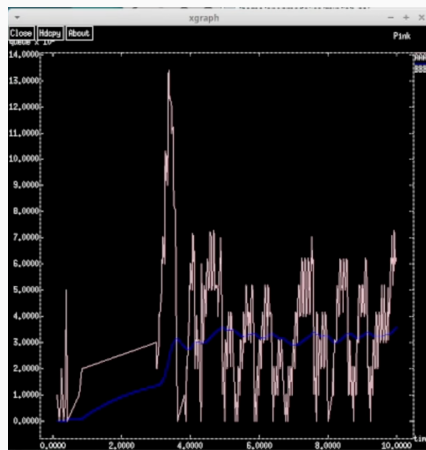


Рис. 12: График динамики длины очереди и средней длины очереди с изменением отображения

В процессе выполнения данной лабораторной работы я исследовала протокол TCP и алгоритм управления очередью RED.