

Исследование протокола TCP и алгоритма управления очередью RED

Лабораторная работа №2.

Рогожина Н.А.

22 февраля 2025

Российский университет дружбы народов, Москва, Россия

Информация

- Рогожина Надежда Александровна
- студентка 3 курса НФИбд-02-22
- Российский университет дружбы народов
- <https://mikogreen.github.io/>

Цель работы

Проанализировать разницу между 3-мя протоколами передачи данных:

1. TCP Reno
2. TCP NewReno
3. TCP Vegas

Выполнение лабораторной работы

Первым делом мы копируем шаблон в новый файл и открываем его на редактирование.

[illegible]

Рис. 1: Создание нового файла

В тексте лабораторной работы был дан код алгоритма, который было необходимо реализовать. Мы его переписали в новый созданный файл.


```
~/home/openmodelica/otp/lib-ns/example6.ncl - Moxyerpad
Файл  Правка  Поиск  Вид  Документ  Справка

# 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

# Узыли сети:
set N 5
for {set i 1} {$i < $N} {incr i} {
    set node_($i) [$ns node]
}
set node_r(1) [$ns node]
set node_r(2) [$ns node]

# Соединения:
$ns duplex-link $node_($i1) $node_r(1) 100b 2ms DropTail
$ns duplex-link $node_($i2) $node_r(1) 100b 2ms DropTail
$ns duplex-link $node_r(1) $node_r(2) 1.5mb 20ms RED
$ns queue-limit $node_r(1) $node_r(2) 25
$ns queue-limit $node_r(2) $node_r(1) 25
$ns duplex-link $node_($i3) $node_r(2) 100b 4ms DropTail
$ns duplex-link $node_($i4) $node_r(2) 100b 5ms DropTail

# Агенты и приложения:
set tcp1 [$ns create-connection TCP/Reno $node_($i1) TCPSink $node_($i3) 0]
$tcp1 set window 15
set tcp2 [$ns create-connection TCP/Reno $node_($i2) TCPSink $node_($i3) 1]
$tcp2 set window 15
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]

# Мониторинг размера окна TCP:
set windowVsTime [open WindowVsTime.ho w]
set upon [$ns monitor-queue $node_r(1) $node_r(2) [open on.out w] 0.1]
$ns link $node_r(1) $node_r(2) queue-samp10-timer;

# Мониторинг очереди:
set reqd [[$ns link $node_r(1) $node_r(2)] queue]
set tchan [open all.q w]
$reqd trace curq
$reqd trace ave_
$reqd attach $tchan_
```

Рис. 2: Реализация модели

Реализация модели

```

File      Правка      Поиск      Вид      Документ      Справка
$rmq trace ave
$rmq attach $tchan_

# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

# at-событие для планирования событий, которое запускает
# процедуру finish через 5 с после начала моделирования
# Планирование at-событий:
$ns at 0.0 "$fcp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"
$ns at 3.0 "$fcp2 start"
$ns at 10 "finish"

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_
}

```

Рис. 3: Реализация модели

```
File  Пресса  Поиск  Вид  Документ  Справка
/home/openmodelica/tmp/fab-ns/example6.csl - Moxsepad
$ns at 10 "finish"

proc plotWindow (tcpSource file) {
  global ns
  set time 0.01
  set now [$ns now]
  set cwnd [$tcpSource set cwnd_]
  puts $file "show source"
  $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 == "n" && 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 "\nqueue"
  exec cat temp.q >& $f
  puts $f "\n\nawk_queue"
  exec cat temp.a >& $f
  close $f

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

# Запуск модели
$ns run
```

Рис. 4: Реализация модели

Результаты первого запуска

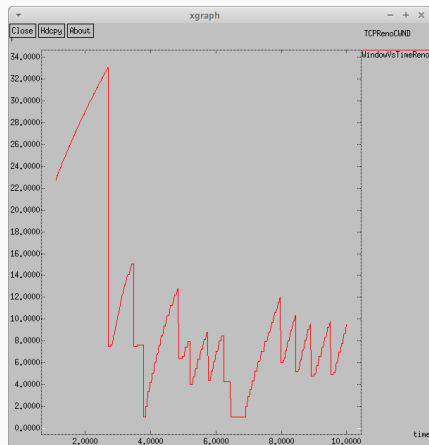


Рис. 5: График изменения размера окна Reno

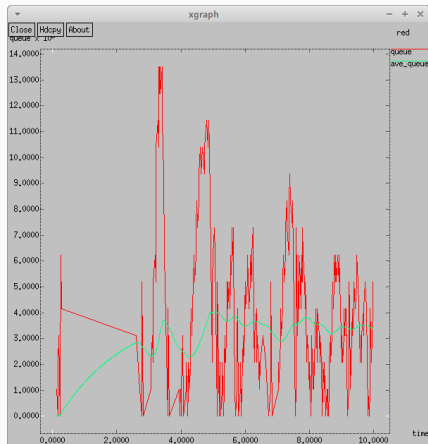


Рис. 6: График фактической и средней длины очереди Reno

Для более приятной и понятной визуализации, я изменила график:

1. `-bg white` для белого фона
2. `0.Color=purple` для цвета первой рисуемой линии
3. `1.Color=orange` для цвета второй рисуемой линии
4. Третью линию оставила как есть, красной.

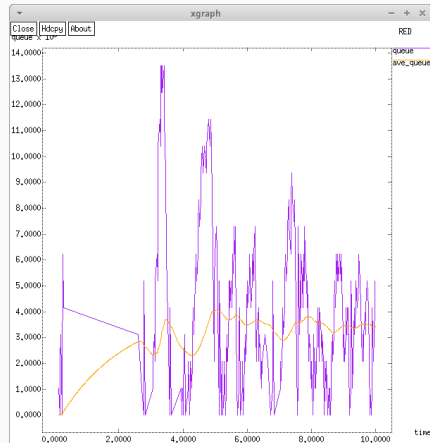


Рис. 7: Изменение визуализации графика фактической и средней длины очереди Reno

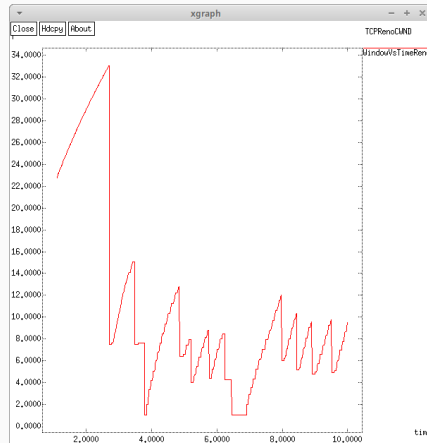


Рис. 8: Изменение визуализации графика изменения размера окна Reno

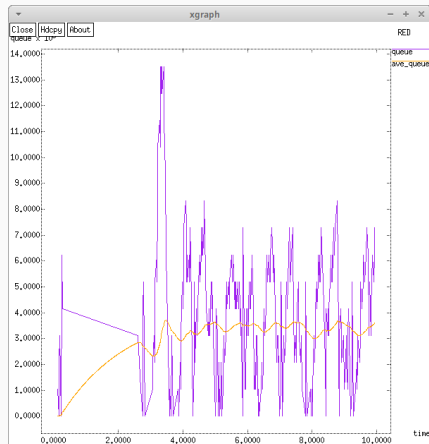


Рис. 9: График фактической и средней длины очереди NewReno

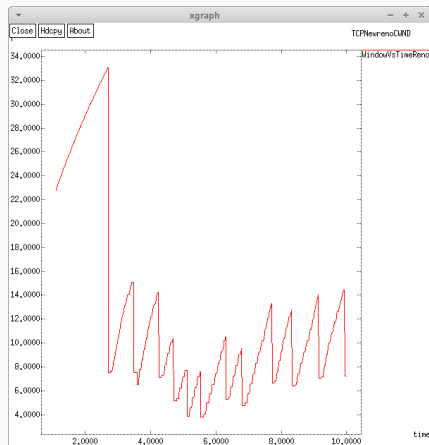


Рис. 10: График изменения размера окна NewReno

Повторная смена протокола

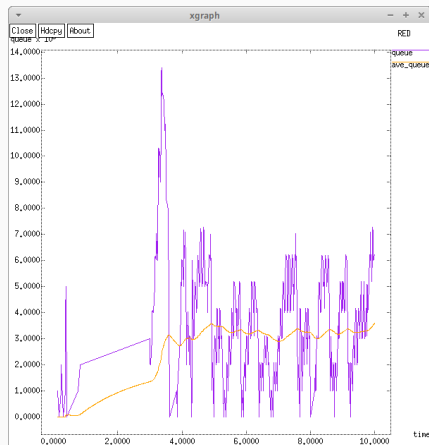


Рис. 11: График фактической и средней длины очереди Vegas

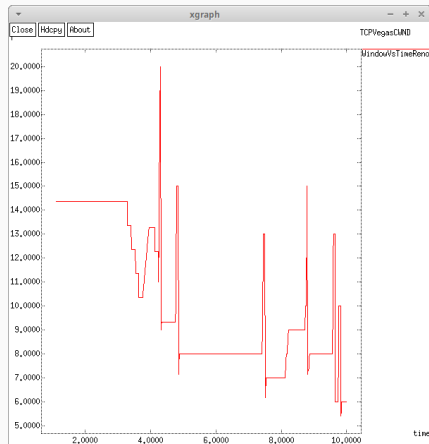


Рис. 12: График изменения размера окна Vegas

Т.к. у нас Reno и NewReno похожи по подходу, графики тоже похожи, но у NewReno стабильнее график и меньше разброс значений, т.к. у Reno увеличение окна происходит линейно и регулируется только при потере пакетов, а у NewReno есть регуляризация принятия пакетов после потери (увеличенный быстрый старт), а у TCP Vegas совершенно иной подход к регуляризации трафика - управление на основе показателя задержки, в следствие чего среднее число пакетов в очереди выглядит еще более стабильно. А также пик размера окна у него сильно меньше, чем у двух предыдущих алгоритмов.

Выводы

В ходе лабораторной работы мы определили различия между 3-мя протоколами TCP и приобрели базовые навыки работы со средством визуализации **xgraph**.