Лабораторная работа №2

Исследование протокола TCP и алгоритма управления очередью RED

Ли Тимофей Александрович

Содержание

Список иллюстраций

Цель работы

Исследовать протокол TCP и алгоритм управления очередью RED с помощью средства имитационного моделирования NS-2, а также с использованием приложения для построения графиков XGRAPH.

Выполнение лабораторной работы

2.2 Пример с дисциплиной RED

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.

Требуется разработать сценарий, реализующий модель, построить в Xgraph график изменения TCP-окна, график изменения длины очереди и средней длины очереди.

Для решения задачи создал файл example01.tcl и написал следующий код: (рис. @fig:001):

```tcl
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
        # выполнение кода AWK
        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
```

Рис. 0.1: example01.tcl

Далее, внес изменения в код, чтобы он правильно работал в моей версии Xgraph (рис. @fig:002)

```tcl
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 "title = 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
        # выполнение кода AWK
        exec cat temp.q >@ $f
        exec cat temp.a >@ $f
        close $f
        # Запуск xgraph с графиками окна TCP и очереди:
        exec xgraph WindowVsTimeReno &
        exec xgraph temp.queue &
        exit 0
}
$ns run
```

Рис. 0.2: example01.tcl (измененное)

Получились такие графики размера окна TCP и размера и очереди соответственно: (рис. @fig:003)

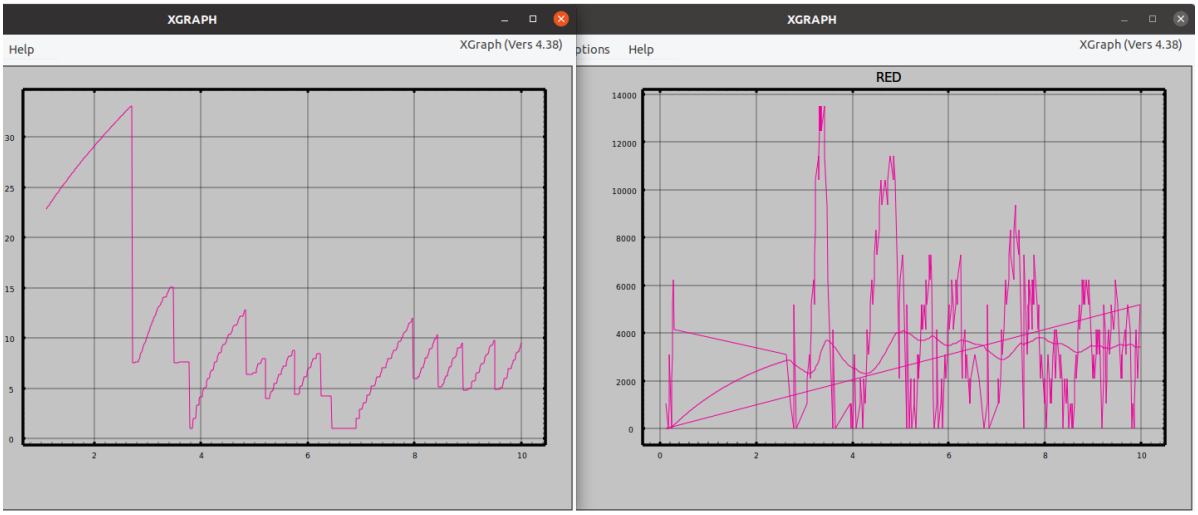

Рис. 0.3: графики для TCP/Reno

Упражнение

- 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).

Для начала я решил сделать вторую часть упражнения, чтобы упростить восприятие графиков. Для этого я внес следующие изменения в процедуру finish: (рис. @fig:004)

```
50 proc finish {} {
51         global tchan_
52         # подключение кода AWK:
53         set awkCode {
54         {
55                 if ($1 == "Q" && NF>2) {
56                 print $2, $3 >> "temp.q";
57                 set end $2
58                 }
59                 else if ($1 == "a" && NF>2)
60                 print $2, $3 >> "temp.a";
61         }
62         }
63
64         set f [open temp.queue w]
65         puts $f "title = RED"
66         #puts $f "device = Postscript"
67         if { [info exists tchan_] } {
68                 close $tchan_
69         }
70         exec rm -f temp.q temp.a
71         exec touch temp.a temp.q
72         exec awk $awkCode all.q
73         # выполнение кода AWK
74         puts $f color=3
75         exec cat temp.q >@ $f
76         puts $f next\ncolor=4
77         exec cat temp.a >@ $f
78         close $f
79         # Запуск xgraph с графиками окна TCP и очереди:
80         exec xgraph -wbgr -title_x TIME -title "TCPRenoCWND" WindowVsTimeReno &
81         exec xgraph -wbgr -title_x TIME -title_y QUEUE temp.queue &
82         exit 0
83 }
```

Рис. 0.4: новая процедура finish

Теперь в 74 и 76 строках я задаю цвет линий, обозначающих моментальный и средний размер очереди соответственно. В 80 и 81 строках теги -wbgr означают цвет фона (по идее, белый, но по факту у меня отображается как сиреневый), теги -title_x, -title_y и -title обозначают подписи осей x и y и всего графика соответственно. Я не нашел, как добавить легенду к графику в моей версии xgraph, поэтому подписи к траекториям добавил вручную на графики с помощью меню edit:Attach Text-note to Data. Результат: (рис. @fig:005)

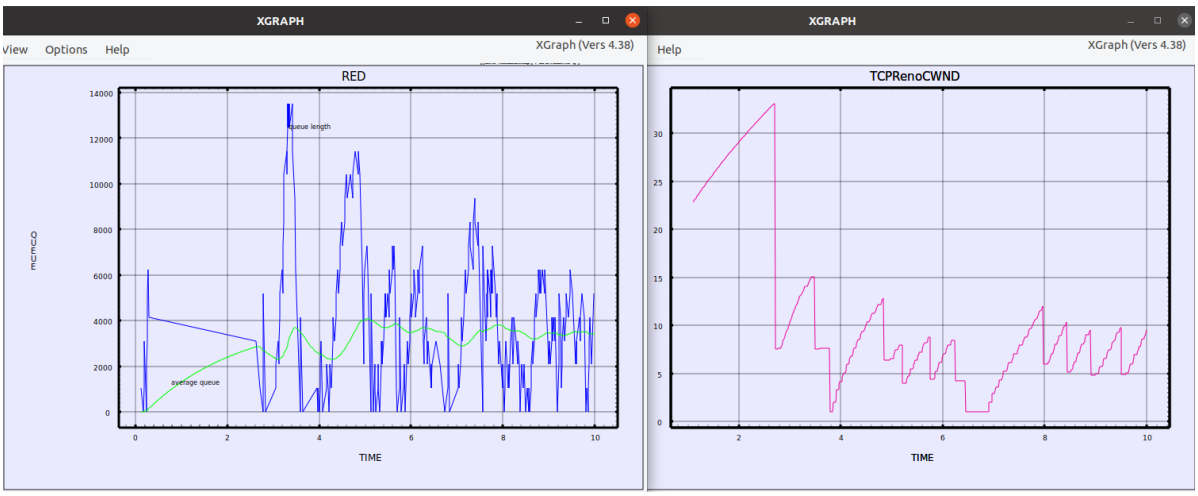

Рис. 0.5: полученные графики

Для выполнения первого пункта упражнения заменил в 19 строке кода TCP/Reno на TCP/Newreno, а затем на TCP/Vegas. Получились следующие графики для TCP/Newreno: (рис. @fig:006)

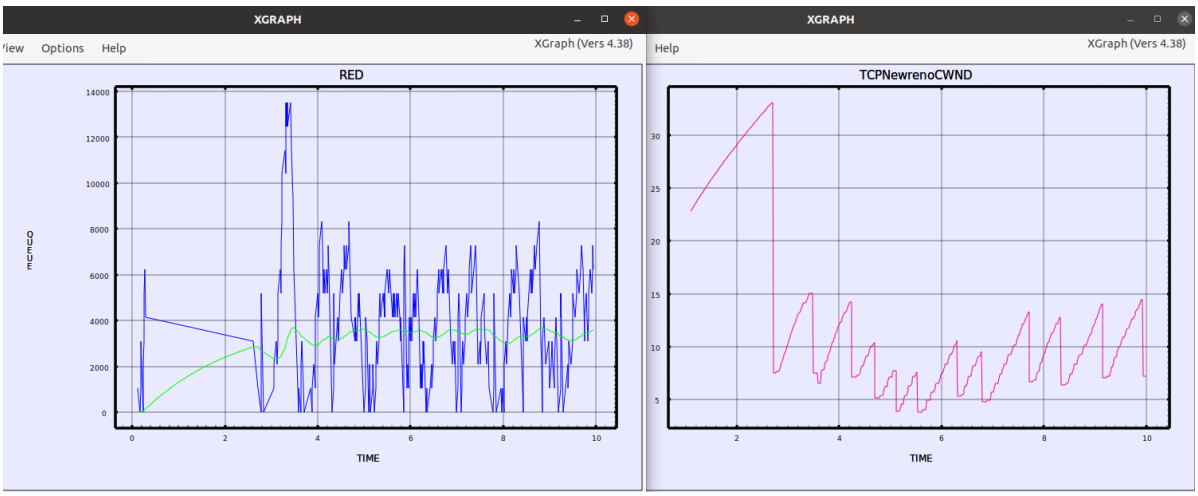

Рис. 0.6: графики для Newreno

и для TCP/Vegas: (рис. @fig:007)

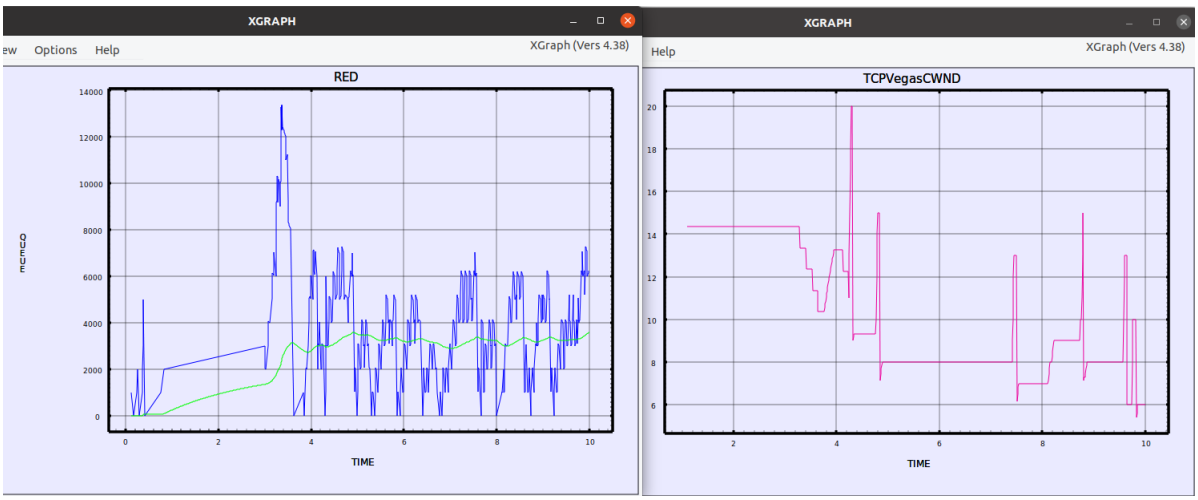

Рис. 0.7: графики для Vegas

График размера окна TCP для Reno обусловлен алгоритмом Reno, суть которого заключается в том, что при пропадании пакета по тайм-ауту размер окна уменьшается на единицу, а при получении сообщения о недоставке пакета значение окна уменьшается в два раза. Изменение графиков в случае использования типа NewReno обусловлено тем, что эта модель использует алгоритм Fast Retransmit и Fast Recovery (быстрая повторная пересылка и быстрое восстановление). TCP/Vegas контролирует размер окна путем мониторирования отправителем времени доставки для пакетов, посланных ранее. Если обнаруживается увеличение этого времени, система узнает, что сеть приближается к перегрузке и сокращает ширину окна. Если же время уменьшается, то отправитель определит, что сеть преодолела перегрузку, и увеличит размер окна. Так, размер окна будет стремиться к требуемому значению.

Выводы

Исследовал протокол TCP и алгоритм управления очередью RED с помощью NS-2 и Xgraph.