Лабораторная работа № 2

Имитационное моделирование

Королев И.А.

Российский университет дружбы народов, Москва, Россия

Цель работы

Цель работы

Более подробно познакомится с протоколом TCP и мониторингом очередей.

Задание

Пример с дисциплиной RED

Пример с дисциплиной RED

- 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. На рис. 2.4 приведена схема моделируемой сети.

Упражнение

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).

Теоретическое введение

Протокол TCP

Протокол TCP

Протокол управления передачей (Transmission Control Protocol, **TCP**) имеет средства управления потоком и коррекции ошибок, ориентирован на установление соединения.

Мониторинг очередей

Мониторинг очередей

Объект мониторинга очереди оповещает диспетчера очереди о поступлении пакета.

Диспетчер очереди осуществляет мониторинг очереди.

Выполнение лабораторной работы

Пример с дисциплиной RED

Реализация модели

Реализация модели

Реализация модели. Описываются узлы сети, соединения, агенты и приложения, мониторинг размера окна, мониторинг очереди, добавление at-событий и формирование файла с данными о размере окна TCP.

```
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]

$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

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]

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;

set redq [[$ns link $node_(r1) $node_(r2)] queue]
set tchan_ [open all.q w]
$redq trace curq_
$redq trace ave_
$redq attach $tchan_

$ns at 0.0 "$ftp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"
$ns at 3.0 "$ftp2 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"
}
```

Реализация модели

Реализация модели

Реализация модели. Добавление процедуры finish. Подключение кода AWK. Открытие файла f на запись. Выполнение кода AWK, подпись траекторий в легенде. Запуск xgraph с графиками окна TCP и очереди.

```
proc finish {} {
        global tchan

        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

        exec xgraph -bb -tk -x time -t "TCPRenoCWND" WindowVsTimeReno &
        exec xgraph -bb -tk -x time -y queue temp.queue &
        exit 0
}


$ns run
```

Рис. 2: Реализация модели

График изменения TCP-окна. Тип TCP-агента: Reno

График изменения TCP-окна. Тип TCP-агента: Reno

В начале соединения окно перегрузки быстро растет, что характерно для фазы медленного старта. Примерно на 2 секундах происходит резкое уменьшение окна, что свидетельствует о потери пакетов. После первого спада окно перегрузки начинает увеличиваться, но с периодическими падениями, что указывает на Reno. Максимальное значение окна примерно 33, минимальное значение окна около 1.

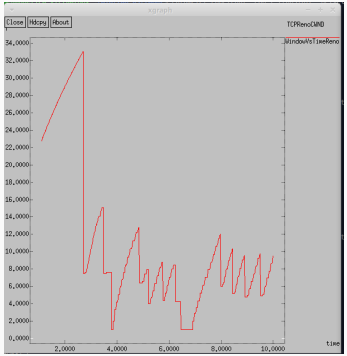


График изменения длины очереди и средней длины очереди. Тип TCP-агента: Reno

График изменения длины очереди и средней длины очереди. Тип TCP-агента: Reno

Текущий размер очереди показывает высокие колебания. Средняя очередь постепенно растет и остается на стабильном уровне, что говорит о настройке RED. Максимальное значение примерно 13.5, минимальное значение окна около 0.

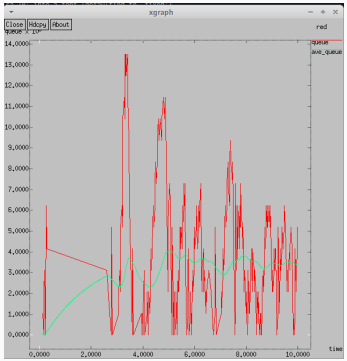


Рис. 4: График изменения длины очереди и средней длины очереди. Тип TCP-агента: Reno

Упражнение.

Изменить в модели на узле s1 тип протокола TCP с Reno на NewReno.

Изменить в модели на узле s1 тип протокола TCP с Reno на NewReno.

Изменяем тип протокола на NewReno.

```
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. 5: Модель. Изменили на другой тип протокола.

График изменения TCP-окна. Тип TCP-агента: NewReno

График изменения TCP-окна. Тип TCP-агента: NewReno

В начале соединения окно перегрузки быстро растет, что характерно для фазы медленного старта. Примерно на 3 секундах происходит резкое уменьшение окна, что свидетельствует о потери пакетов. После первого спада окно перегрузки начинает увеличиваться, но с периодическими падениями, что указывает на NewReno. Максимальное значение окна примерно 33, минимальное значение окна около 4.

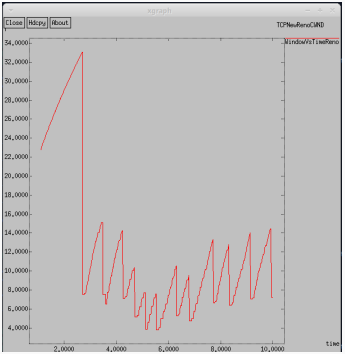


График изменения длины очереди и средней длины очереди. Тип TCP-агента: NewReno

График изменения длины очереди и средней длины очереди. Тип TCP-агента: NewReno

Текущий размер очереди показывает довольно высокие колебания, но более маленькие, чем у Reno. Средняя очередь постепенно растет и остается на стабильном уровне, что говорит о настройке RED. Максимальное значение примерно 13.5, минимальное значение окна около 0.

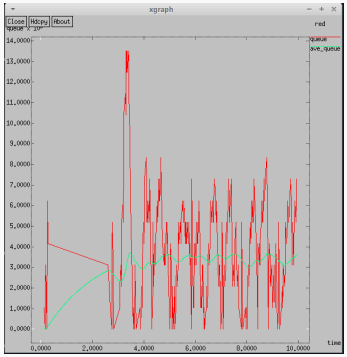


Рис. 7: График изменения длины очереди и средней длины очереди. Тип TCP-агента: NewReno

Изменить в модели на узле s1 тип протокола TCP с Reno на Vegas

Изменить в модели на узле s1 тип протокола TCP с Reno на Vegas

Изменяем тип протокола на Vegas.

```
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: Модель. Изменили на другой тип протокола.

График изменения TCP-окна. Тип TCP-агента: Vegas

График изменения TCP-окна. Тип TCP-агента: Vegas

В начале соединения окно перегрузки сохраняет значение.При Vegas максимальный размер окна составляет 20, а не 34, как в NewReno. Vegas обнаруживает перегрузки сети до того, как произойдет потеря, поэтому можно сказать он не теряет пакеты.

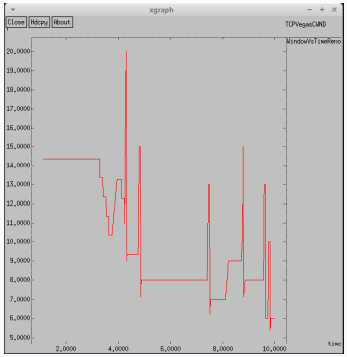
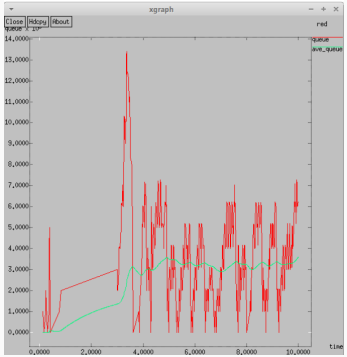


Рис. 9: График изменения TCP-окна. Тип TCP-агента: Vegas

График изменения длины очереди и средней длины очереди. Тип TCP-агента: Vegas

График изменения длины очереди и средней длины очереди. Тип TCP-агента: Vegas

С нуля до примерно двух секунд, происходит скачок и плавное изменение значения в текущем размере очереди. Текущий размер очереди показывает довольно высокие колебания. Средняя очередь постепенно растет и остается на стабильном уровне, что говорит о настройке RED. Максимальное значение примерно 13.5, минимальное значение окна около 5.



Сравнение Newreno и Vegas

Сравнение Newreno и Vegas

- Newreno лучше для сетей с высокой потерей пакетов.
- Vegas эффективнее в условиях высокой задержки.

Если сеть имеет переменную задержку и много конкурирующих соединений, то лучше выбрать Newreno. Если же сеть стабильна и перегруженность предсказуема, то Vegas.

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

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

Сначала изменяю цвет фона и подписи к осям. После, добавляю новые подписи траекторий в легенде и цвет траекторий.

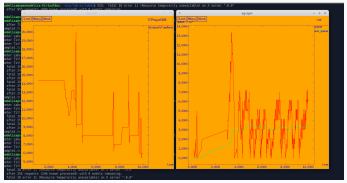


Рис. 11: Цвет фона и подписи к осям

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

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

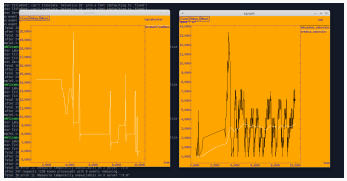


Рис. 12: Цвет траектории и подписи траектории в легенде

Измененный код реализации модели.

Измененный код реализации модели.

```
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]

$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

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]

set windowVsTime [open WindowVsTimeReno w]
puts $windowVsTime "0.Color: White"
set qmon [$ns monitor-queue $node_(r1) $node_(r2) [open qm.out w] 0.1];
[$ns link $node_(r1) $node_(r2)] queue-sample-timeout;

set redq [[$ns link $node_(r1) $node_(r2)] queue]
set tchan_ [open all.q w]
$redq trace curq_
$redq trace ave
$redq attach $tchan_

$ns at 0.0 "$ftp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"
$ns at 3.0 "$ftp2 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"
```

Рис. 13: Реализация

Измененный код реализации модели.

Измененный код реализации модели.

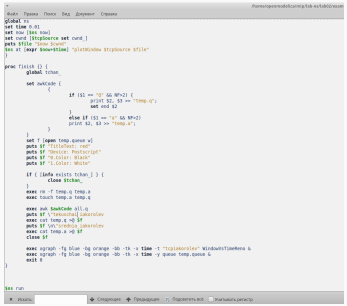


Рис. 14: Реализация

Выводы

Выводы

Более подробно познакомился с протоколом TCP и мониторингом очередей.