Лабораторная работа 2

Исследование протокола TCP и алгоритма управления очередью RED

Извекова Мария Петровна

Содержание

Список иллюстраций

Список таблиц

Цель работы

Ознакомиться с протоколом TCP и очередью RED, построить сценарий на симмуляторе и изобразить результате в Xgraph

Задание

Требуется разработать сценарий, реализующий модель из 6 узлов с дуплексным соединением и очередью с дисциплиной RED, построить в Xgraph график изменения TCP-окна, график изменения длины очереди и средней длины очереди

Теоретическое введение

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

Флаг Указатель срочности (Urgent Pointer, URG) устанавливается в 1 в случае использования поля Указатель на срочные данные. Флаг Подтверждение (Acknowledgment, ACK) устанавливается в 1 в случае, если поле Номер

подтверждения (Acknowledgement Number) содержит данные. В противном случае это поле игнорируется. Флаг Выталкивание (Push, PSH) означает, что принимающий стек TCP должен немедленно информировать приложение о поступивших данных, а не ждать, пока буфер заполниться. Флаг Сброс (Reset, RST) используется для отмены соединения из-за ошибки приложения, отказа от неверного сегмента, попытки создать соединение при отсутствии затребованного сервиса. Флаг Синхронизация (Synchronize, SYN) устанавливается при инициировании соединения и синхронизации порядкового номера. Флаг Завершение (Finished, FIN) используется для разрыва соединения. Он указывает, что отправитель закончил передачу данных. Управление потоком в протоколе TCP осуществляется при помощи скользящего окна переменного размера: – поле Размер окна (Window) (длина 16 бит) содержит количество байт, которое может быть послано после байта, получение которого уже подтверждено; – если значение этого поля равно нулю, это означает, что все байты, вплоть до байта с номером Номер подтверждения - 1, получены, но получатель отказывается принимать дальнейшие данные; – разрешение на дальнейшую передачу может быть выдано отправкой сегмента с таким же значением поля Номер подтверждения и ненулевым значением поля Размер окна. Регулирование трафика в TCP: – контроль доставки — отслеживает заполнение входного буфера получателя с помощью параметра Размер окна (Window); – контроль перегрузки — регистрирует перегрузку канала и связанные с этим потери, а также понижает интенсивность трафика с помощью Окна перегрузки (Congestion Window, CWnd) и Порога медленного старта (Slow Start Threshold, SSThreth)

Выполнение лабораторной работы

1. Создаем скрипт где прописываем все соединения 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.:

```
# 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]
```

Создание сети

2. В скрипте прописываем мониторинг размера окна и очереди – то, что исследуем

```
# 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"
```

Мониторинг окна

3. Прописываем часть, отвечающая за вывод результата в Xgraph. результаты их временных таблиц переносим в 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"
}


# 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 color=1"   ;# Очередь (по умолчанию, обычно красный)
exec cat temp.q >@ $f
puts $f "\nave_queue color=2" ;# Средняя длина очереди (зеленый)
exec cat temp.a >@ $f
close $f
```

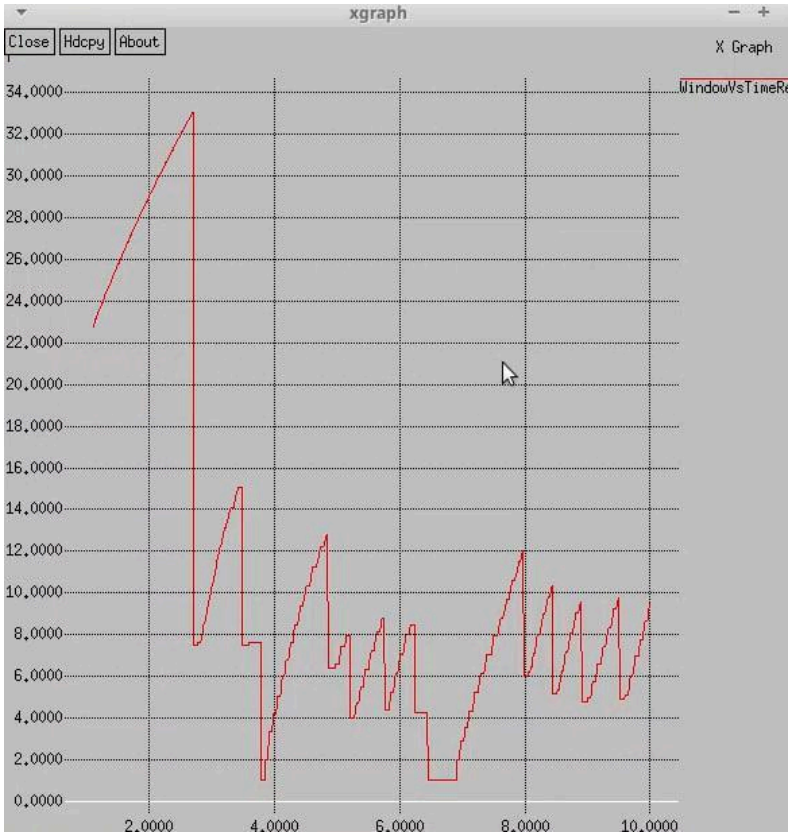
Скрипт Xgraph

4. Сначала рассматриваем протокол TCP Reno

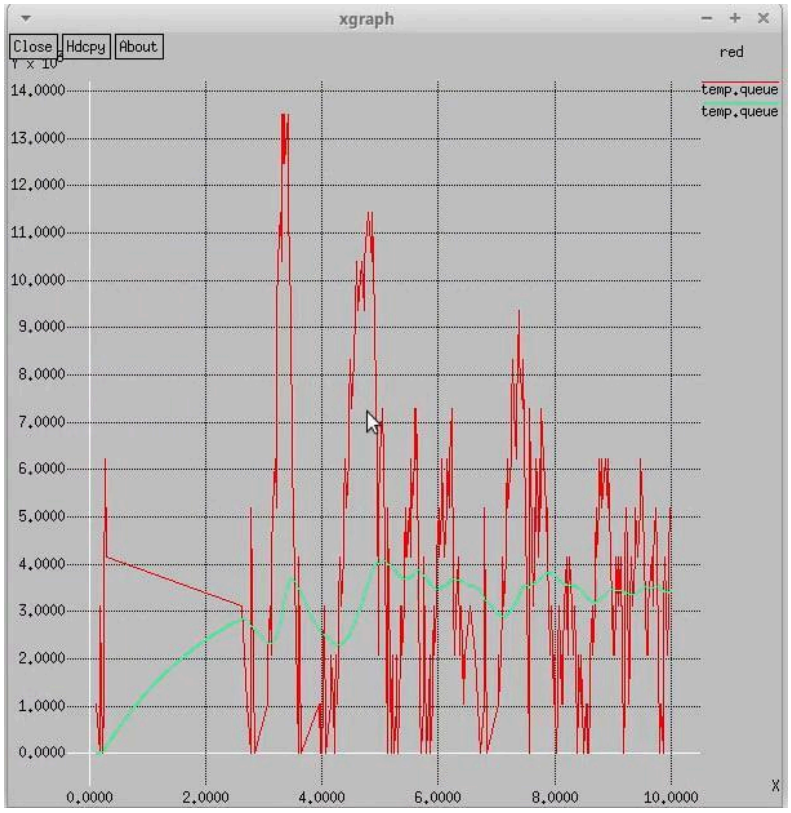
```
# Агенты и приложения:
set tcp1 [$ns create-connection TCP/Reno $node_(s1) TCPSink $node_(s3) 0]
$tcp1 set window  15
```

Протоколы

5. Результаты размеров окна и очереди



Динамика размеры окна



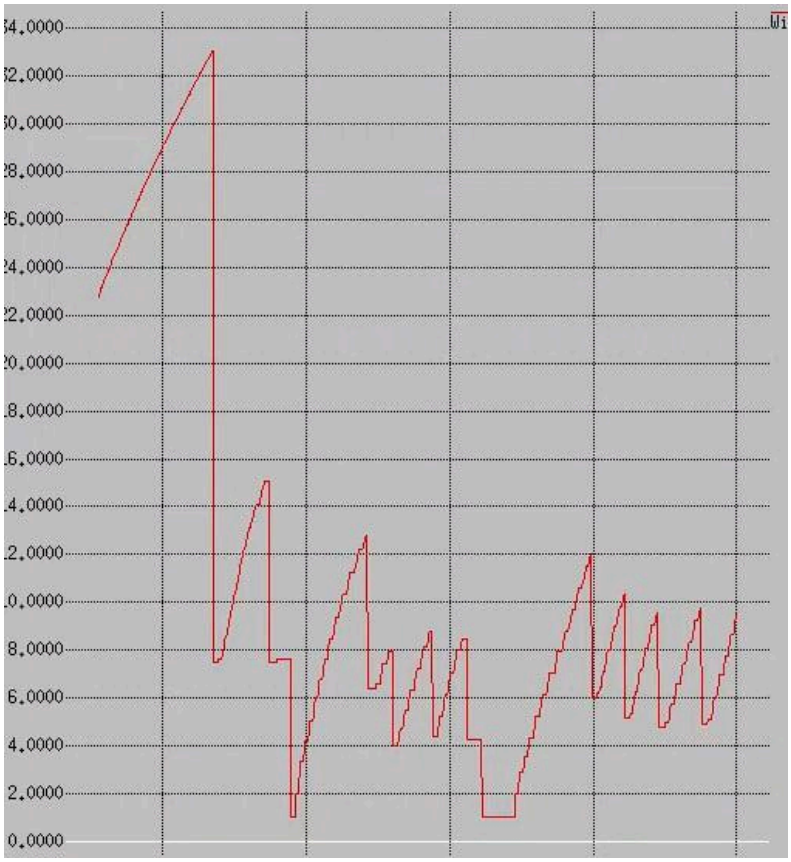
Мониторинг очереди

6. Рассматриваем протокол TCP NewReno

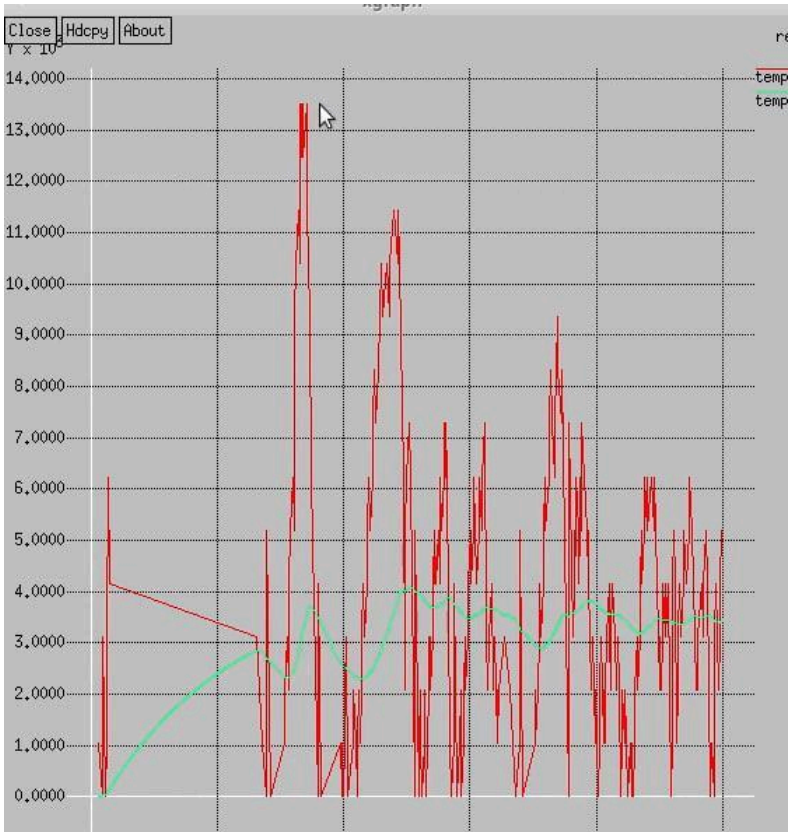
```
28 # Агенты и приложения:
29 set tcp1 [$ns create-connection TCP/NewReno $node_(s1) TCPSink $node_(s3) 0]
30 $tcp1 set window_ 15
```

Протоколы NewReno

7. Результаты размеров окна и очереди



Динамика размеры окна



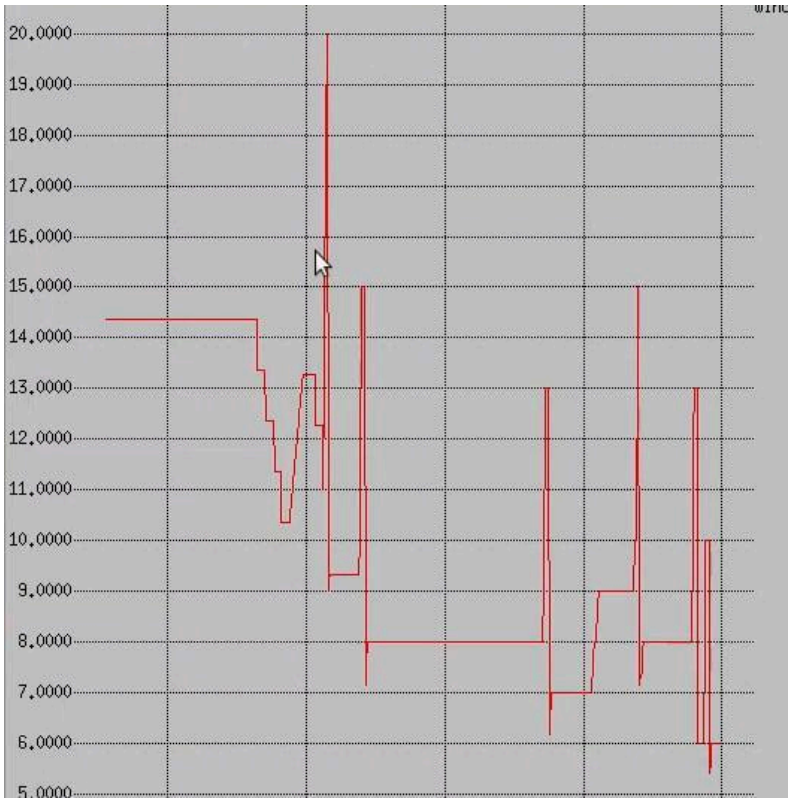
Мониторинг очереди

8. Рассматриваем протокол TCP Vegas

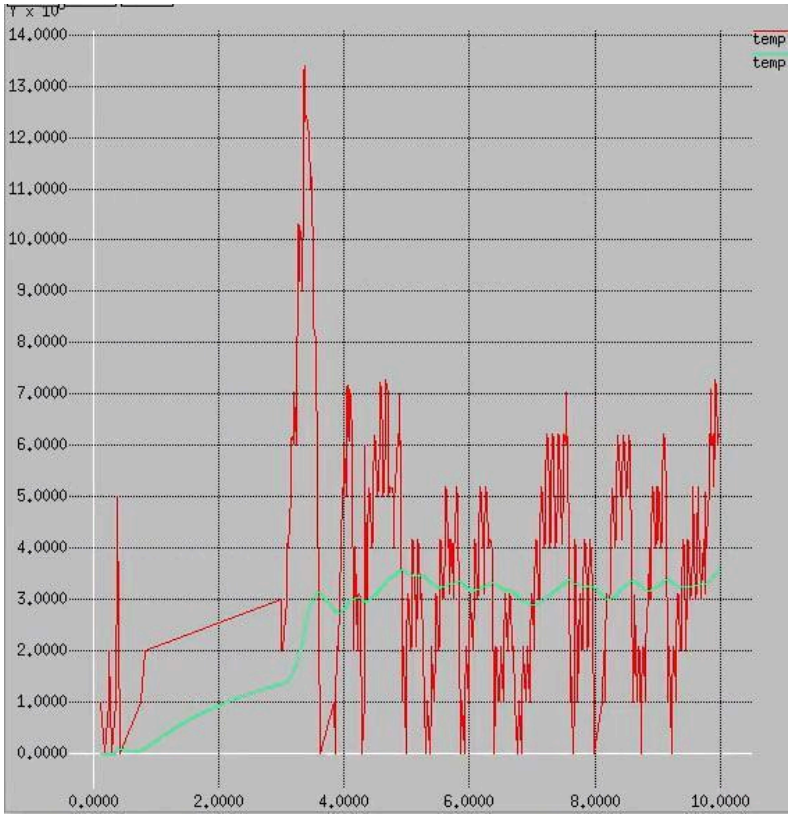
```
28 # Агенты и приложения:
29 set tcp1 [$ns create-connection TCP/Vegas $node_(s1) TCPSink $node_(s3) 0]
30 $tcp1 set window_ 15
```

Протоколы Vegas

9. Результаты размеров окна и очереди



Динамика размеры окна



Мониторинг очереди

Выводы

Ознакомиться с протоколом TCP и очередью RED, построить сценарий на симмуляторе и изобразить результате в Xgraph. Так же мы рассмотрели, что протоколы TCP Reno и NewReno реагируют на изменения, только после потерей пакетов, как протокол Vegas замечает это до, из-за чего его линии моментами не меняют направление.

Список литературы