ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.715

В.И. Гостев, Т.П. Довженко

Государственный университет телекоммуникаций, Киев

ИССЛЕДОВАНИЕ СЕТИ TCP/IP С ПРИМЕНЕНИЕМ ОСНОВНЫХ TCP-АЛГОРИТМОВ ПРЕДОТВРАЩЕНИЯ ПЕРЕГРУЗОК

Рассмотренны основные разновидности TCP-протоколов предотвращения перегрузок сети (TCP-Reno, TCP-NewReno, TCP-Vegas) и проведено исследование сети TCP/IP с применением данных протоколов.

Ключевые слова: *TCP-Reno, TCP-NewReno, TCP-Vegas, алгоритм активного предотвращения перегрузок сети, TCP-протокол.*

Введение

Пакетная коммутация позволяет повысить эффективность использования каналов, но при этом приводит к снижению надежности доставки. При перегрузке канала с пакетной коммутацией, данные на входе канала не смогут поместится во входной буфер и будут сброшены. Для обеспечения гарантированной доставки пакетов по каналам без гарантированной доставки были разработаны специальные протоколы, одним из которых является протокол TCP. Со временнем требования к передачи пакетных данных повысились: система передачи данных должна обеспечивать малое значение и вариацию задержки. Эта задача решается с помощью регулирования скорости передачи и приема сообщений. Управление скоростью передачи осуществляется с помощью алгоритма управления перегрузкой (TCP congestion control algorithm). В данной работе производиться исследование сети TCP/IP с применением различных современных TCP-алгоритмов предотвращения перегрузок таких как: Reno, New Reno, Vegas.

Постановка задачи

Главная цель работы заключается в исследовании работы системы предотвращения перегрузок с использованием основных TCP-протоколов(Reno, New Reno, Vegas) в условиях изменения нагрузки трафика.

Обзор основных алгоритмов предотвращения перегрузок. Перегрузка в сетях TCP/IP возникает в случае, когда количество передаваемых данных начинает приближаться к значению допустимой пропускной способности сети[1]. При этом ухудшаются основные показатели качества обслуживания, а также происходит рост числа потерянных пакетов и времени задержки. Основной задачей TCP-алгоритмов предотвращения перегрузок является поддержание значения потока данных, передаваемых по сети, ниже уровня, при котором пропускная способность сети начинает резко падать, ограничивая потоки входящего и исходящего трафика.

Для эффективного использования пропускной способности нужно выбрать именно тот противоперегрузочный алгоритм, который сможет обеспечить максимальное использование ресурсов сети TCP/IP при минимальных потерях.

Протокол TCP-Reno. При использовании алгоритма TCP-Reno, размер окна меняется циклически. Он увеличивается при каждом цикле до потери пакета[2]. Когда происходит потеря пакета, TCP Reno уменьшает размер окна до половины текущего размера. Это называется - *аддитивным увеличением и мультипликативным уменьшением*.

TCP Reno имеет два этапа изменения размера окна:

1) фаза медленного старта;

2) фаза избежание перегрузки.

Когда отправитель получает подтверждение доставки в момент времени $t + tA$ [с] текущее значение размер окна перегрузки *cwnd(t)* преобразуется в *cwnd(t+tA)* согласно формуле:

$$c\quad(t + t_A) = \begin{cases} c\quad(t) + 1, \text{при } c\quad(t) < ss\ h(t) \\ c\quad(t) + \frac{1}{c\quad(t)}, \text{при } c\quad(t) \geq ss\ h(t) \end{cases} \quad (1)$$

где *ssth(t)* (пакет) – значение порога, при котором TCP переходит от фазы медленного старта в фазу избежания перегрузки.

Если потери пакетов происходят в следствии таймаута, значение *cwnd(t)* и *ssth(t)* обновляются следующим образом :

$$\begin{cases} c \quad (t) = 1 \\ s \quad h(t) = \frac{c \quad (t)}{2} \end{cases} \quad (2)$$

Если же потери пакетов обнароужены согласно алгоритму быстрой передачи, *cwnd(t)* и *ssth(t)* обновляются иначе:

$$\begin{cases} c \quad (t) = s \quad h(t) \\ s \quad h(t) = \frac{c \quad (t)}{2} \end{cases} \quad (3)$$

Протокол TCP-NewReno. На данный момент, TCP-NewReno(см. RFC-6582), являеться одним из наиболее распостраненных TCP-алгоритмов. Он основан на алгоритме Fast Retransmit & Fast Recovery (быстрая повторная пересылка и быстрое восстановление). В случае, когда доступно выборочное подтверждения (SACK), отправитель знает, какие пакеты следует переслать повторно на фазе быстрого восстановления (Fast Recovery). В отсутствии опции SACK нет достаточной информации относительно тех пакетов, которые нужно повторно переслать. При получении трех дублированных подтверждений (DUPACK) отправитель считает пакет потерянным и посылает его повторно. После этого отправитель может получить дополнительные дублированные подтверждения, так как получатель осуществляет подтверждение пакетов, которые находятся в пути, в момент когда отправитель перешел в режим Fast Retransmit. В случае потери нескольких пакетов из одного окна отправитель получает новые данные, в момент когда придет подтверждение для повторно посланных пакетов. Если потерян один пакет и не было смены порядка пакетов, тогда подтверждение этого пакета будет означать успешную доставку всех предыдущих пакетов до перехода в режим Fast Retransmit. Однако, если потеряно несколько пакетов, тогда потверждение повторно посланного пакета подверждает доставку некоторых но не всех пакетов, посланных до перехода в режим быстрой повторной пересылки (Fast Retransmit). Такие подтверждения называются частичными.

Протокол TCP-Vegas. TCP Vegas использует разницу между ожидаемой и фактической скоростью потоков для оценки пропускной способности сети[3]. Основная идея состоит в том, что когда сеть не перегружена, фактическая скорость потока будет близка к ожидаемой. В противном случае фактическая скорость потока будет меньше, чем ожидаемая скорость потока. С помощью разницы скоростей, TCP Vegas оценивает уровень заторов в сети и соответствующим образом обновляет размер окна. Источник, используя ожидаемую и фактическую скорость потока вычисляет оценку отставания в очереди по формуле [4]:

$$D \quad = (E \quad - A \quad)B \quad , \quad (4)$$

где *Expected* – ожидаемая скорость; *Actual* – фактическая скорость; *BaseRTT* – минимальное значение RTT при первом подключении.

Детали алгоритма:

1. Источник вычисляет ожидаемую скорость потока по формуле

$$E. \quad = \frac{c \quad (t)}{B} \quad , \quad (5)$$

где *cwnd(t)*– текущий размер окна.

2. Источник оценивает текущую скорость потока с помощью фактического RTT:

$$A \quad = \frac{c \quad (t)}{R} , \quad (6)$$

где *RTT* – время прохождения сигнала туда и обратно.

Источник обновляет размер окон следующим образом:

$$c \quad (t) = \begin{cases} c \quad (t) + 1, \text{если } D \quad < \alpha \\ c \quad (t) - 1, \text{если } D \quad > \beta \\ c \quad (t), \text{ в другом случае} \end{cases} , \quad (7)$$

где α и β - некоторые константы[5], t - текущий момент времени. Эта модификация TCP требует высокого разрешения таймера отправителя.

Решение задачи

Исследование сети проводилось с использованием программного комплекса NS-2. Схема сети (см. рис. 1) состоит из 3 FTP источников сообщения(S1, S2, S3), которые с помощью двух маршрутизаторов передают информацию на TCP-приемник (S4). Ниже представлена часть программного кода на языке Otcl (NS-2) отвечающая за создание узлов сети.

Рис. 1. Схема сети для имитационного моделирования

```
set ns [new Simulator]

set node_(s1) [$ns node]#FTP-источник
set node_(s2) [$ns node]#FTP- источник
set node_(s3) [$ns node]#FTP- источник
set node_(r1) [$ns node]#Маршрутизатор №1
set node_(r2) [$ns node]#Маршрутизатор №2
set node_(s4) [$ns node]#TCP-приемник
```

Скорость канала между конечными узлами сообщения и маршрутизаторами составляет 100 Мб/с, задержка для каждого конечного узла разная: S1 = = 2 мс, S2 = 3 мс, S3 = 4 мс. Скорость канала между двумя транзитными маршрутизаторами составляет 5 Мб/с (канал с перегрузкой), а задержка – 20 мс.

При исследовании моделируемой сети, нагрузка на маршрутизатор будет постепенно увеличиваться. Первым начнет работу источник S1. Затем, через 30 сек - S2. На 60-й секунде включиться S3. Программный код, отвечающий за установку параметров и соединение между узлами представлен ниже.

```
$ns duplex-link $node_(s1) $node_(r1) 100Mb 2ms DropTail
$ns duplex-link $node_(s2) $node_(r1) 100Mb 3ms DropTail
$ns duplex-link $node_(r1) $node_(r2) 5Mb 20ms RED
$ns duplex-link $node_(s3) $node_(r1) 100Mb 4ms DropTail
$ns duplex-link $node_(s4) $node_(r2) 100Mb 5ms DropTail

$ns duplex-link-op $node_(s1) $node_(r1) orient right-down
$ns duplex-link-op $node_(s2) $node_(r1) orient right-up
$ns duplex-link-op $node_(r1) $node_(r2) orient right
$ns duplex-link-op $node_(r1) $node_(r2) queuePos 0
$ns duplex-link-op $node_(r2) $node_(r1) queuePos 0
$ns duplex-link-op $node_(s3) $node_(r1) orient right-down
$ns duplex-link-op $node_(s4) $node_(r2) orient left-up

set tcp1 [$ns create-connection TCP/** $node_(s1) TCPSink
$node_(s4) 0]
#Вместо знака "**" прописываем нужный алгоритм
set tcp2 [$ns create-connection TCP/** $node_(s2) TCPSink
$node_(s4) 1]
#Вместо знака "**" прописываем нужный алгоритм
set tcp3 [$ns create-connection TCP/** $node_(s3) TCPSink
$node_(s4) 2]
# Вместо знака "**" прописываем нужный алгоритм
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]
set ftp3 [$tcp3 attach-source FTP]
```

Продолжительность процесса моделирования составляет 100сек. Приведем код, отвечающий за включение источников сообщения в сеть TCP/IP.

```
$ns at 0.0 "$ftp1 start"
$ns at 30.0 "$ftp2 start"
$ns at 60.0 "$ftp3 start"
$ns at 100 "finish"
```

После выполнения программы получим графики зависимости длины очереди от времени работы сети для каждого алгоритма (рис. 2 – 7).

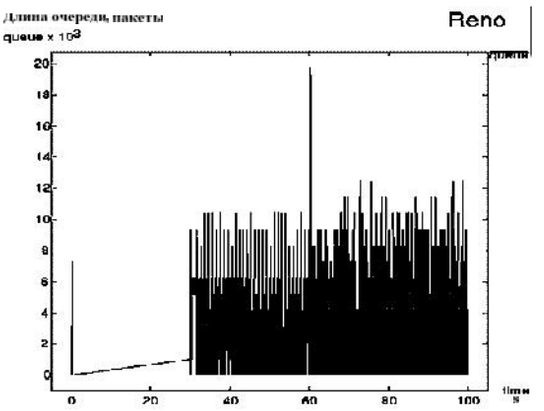

Рис. 2. Длина очереди при использовании Reno-алгоритма

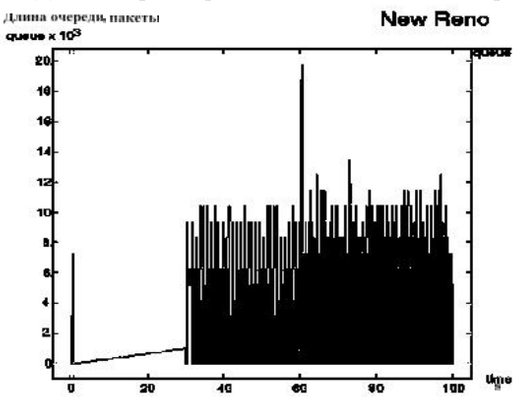

Рис. 3. Длина очереди при использовании New Reno-алгоритма

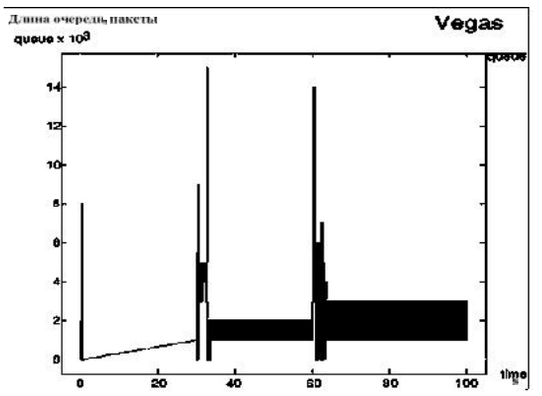

Рис. 4. Длина очереди при использовании Vegas-алгоритма

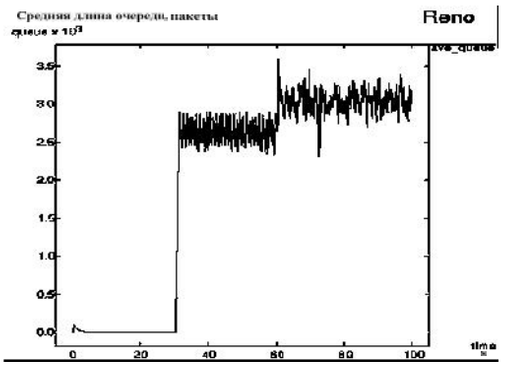

Рис. 5. Средняя длина очереди при использовании Reno-алгоритма

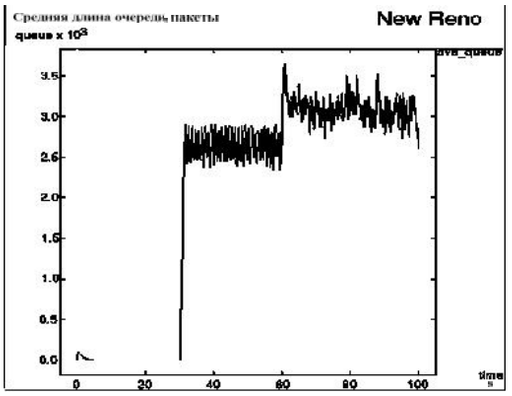

Рис. 6. Средняя длина при использовании New Reno-алгоритма

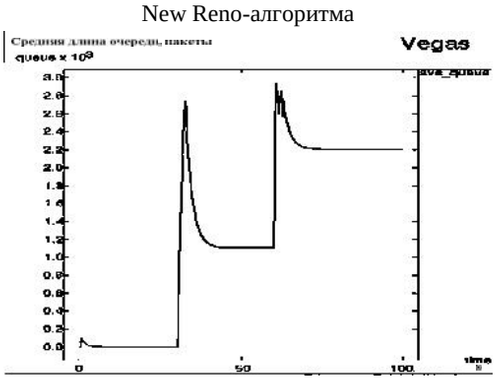

Рис. 7. Средняя длина при использовании Vegas-алгоритма

Выводы

Проанализировав полученные данные, можно сделать вывод о том, что Vegas -алгоритм является наиболее, подходящим для использования в сетях TCP/IP, по сравнению с алгоритмами Reno и New Reno, поскольку имеет наиболее минимальное и стабильное значение очереди при постепенном увеличении нагрузки сети.

Список литературы

1. КоваленкоА.А., Завизиступ Ю.Ю., Партыка С.А., Проблемы производительности протокола TCP в гетерогенных сетях и методы ее улучшения//Вестник ХНТУ.–2005.–№1(21).–С.305-311.

2. Analysis and comparison of TCP Reno and Vegas/Jeonghoon Mo, Richard J. La, Venkat Anantharam, Jean Walrand//INFOCOM – 1999 [Электронный ресурс]// http://www.eecs.berkeley.edu/~ananth/1999-2001/Richard/MoLaInfocom1999.pdf.

3. Семенов Ю.А. Протокол TCP [Электронный ресурс] // http://book.itep.ru/4/44/tcp_443.htm

4. Коваленко А.А. Динамічне керування параметрами протоколу TCP Vegas//Системи озброєння і військова техніка.–2007.– 2(10).–С.81-86.

5. Kenji Kurata, Go Hasegawa, Masayuki Murata, "Fairness Comparisons Between TCP Reno and TCP Vegas for Future Deployment of TCP Vegas", Университет Осаки, Япония[Электронный ресурс] // http://www.isoc.org/inet2000/cdproceedings/2d/ 2d_2.htm.

Поступила в редакцію 22.04.2014

Рецензент: д-р техн. наук, проф. С.В. Козелков, Государственный университет телекомуникаций, Киев.

ДОСЛІДЖЕННЯ МЕРЕЖІ TCP/IP З ВИКОРИСТАННЯМ ОСНОВНИХ АЛГОРИТМІВ ЗАПОБІГАННЯ ПЕРЕНАВАНТАЖЕННЯ

В.І. Гостєв, Т.П. Довженко

Розглянуто основні різновиди TCP-протоколів запобігання перевантажень мережі (TCP-Reno, TCP-NewReno, TCP-Vegas) і проведено дослідження мережі TCP / IP із застосуванням даних протоколів.

Ключові слова: *TCP-Reno, TCP-NewReno, TCP-Vegas, алгоритм запобігання перенавантаження мережі, TCP-протокол.*

RESEARCH OF TCP / IP NETWORK USING THE BASIC ALGORITHM OF CONGESTION AVOIDANCE

V. Gostev, T. Dovzhenko

Consideration of the basic TCP-protocols to prevent network congestion (TCP-Reno, TCP-NewReno, TCP-Vegas) and a study TCP / IP network using these protocols.

Keywords: *TCP-Reno, TCP-NewReno, TCP-Vegas, algorithm actively preventing network congestion, TCP-protocol.*