ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.715

В.И. Гостев, Т.П. Довженко, А.С. Артющик

Государственный университет телекомуникаций, Киев

ИССЛЕДОВАНИЕ СЕТИ TCP/IP С ПРИМЕНЕНИЕМ ОСНОВНЫХ АЛГОРИТМОВ АКТИВНОГО УПРАВЛЕНИЯ ОЧЕРЕДЬЮ

Рассмотрены основные алгоритмы активного управления очередями (RED, PI, REM) и проведено исследование сети TCP/IP с применением данных алгоритмов.

Ключевые слова: *REM, RED,PI, AQM, алгоритм активного управления очередью, TCP/IP-протокол.*

Вступление

Неравномерный рост скоростей каналов передачи данных неизбежно приводит к возникновению «узких» мест в телекоммуникационной сети и соответственно к возникновению перегрузок, особенно при подключении сетей доступа к транспортной сети. Традиционные протоколы управления очередями и предотвращения перегрузок не справляются с управлением трафиком со сложной динамикой и нелинейностью изменения нагрузки, что приводит к возникновению перегрузок и появлению глобальной синхронизации TCP потоков. Это в свою очередь уменьшает эффективную скорость передачи данных и ухудшает параметры качества, такие как процент потерянных пакетов, задержки и вариации задержек. В данной работе производиться исследование сети TCP/IP с применением различных современных алгоритмов управления очередями таких как: RED (Random Early Detection), PI-controller (Proportional-Integral controller), REM(Random Exponential Marking). результаті обробки сигналів, що відбуваються з різних напрямків сектору огляду, формуються однакові за структурою дискретно кодовані (фазоманіпульовані) псевдовипадкові сигнали, що відрізняються одне від одного тільки часовим зсувом.

Постановка задачи

Главная цель работы заключается в исследовании работы AQM-системы с PI, RED и REM-алгоритмами в условиях изменения нагрузки трафика.

Обзор алгоритмов управления очередями

Алгоритмы управления очередью маршрутизаторов в TCP/IP сетях бывают пассивными и активными (рис. 1).

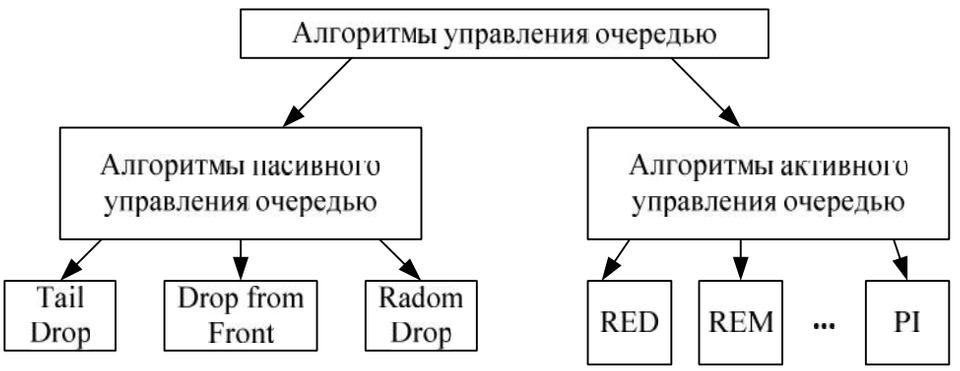

Рис. 1. Классификация алгоритмов управления очередями

Основные задачи алгоритмов управления очередями – минимизация средней длины очереди при одновременном обеспечении высокого коэффициента использования канала, а также справедливое распределение буферного пространства между различными потоками данных[1]. Схемы управления очередями различаются, в основном, критерием, по которому отбрасываются пакеты, и местом в очереди, откуда производится отбрасывание пакетов (начало или конец очереди). Наиболее простым критерием для отбрасывания пакетов является достижение

очередью определенного порога, называемого максимальной длиной очереди.

Примером алгоритмов пассивного управления очередями (PQM) является алгоритм «отбрасывания хвоста» (Tail Drop), где трафик не разделяется по типам пакетов и вероятность отбрасывания каждого пакета одинакова. Когда очередь заполняется до некоторого заданного максимального значения, все вновь поступающие пакеты отбрасываются, пока очередь не освободится для поступления входящего трафика. Имеются альтернативные алгоритмы отбрасывания пакетов: Random Drop (отбрасывание случайно выбранного) и Drop from Front (отбрасывание первого пакета в очереди). Алгоритмам PQM присущ ряд недостатков [2-4], для устранения которых используются алгоритмы активного управления очередями (AQM), обеспечивающие заблаговременное обнаружение перегрузки. Основными целями алгоритмов AQM являются:

- минимизация дрожания задержки пакетов путем контроля среднего размера очереди;
- предотвращение эффекта глобальной синхронизации TCP-трафика;
- обеспечение непредвзятого обслуживания трафика, характеризующегося

кратковременными всплесками;

- строгое ограничение максимального среднего размера очереди.

Примерами алгоритмов AQM являются ранее упомянутые RED, PI, REM, а также WRED(Weighted random early detection), AVQ(Adaptive Virtual Queue), FLC(Fuzzy Logic Controller) и другие [2 - 4]. AQM-алгоритмы могут быть классифицированы по критериям принятия решения об отбрасывании пакетов из очереди при возникновении перегрузки. Можно выделить 4 типа алгоритмов управления очередями [2]:

- AQM на основе средней длины очереди;
- AQM на основе уровня потери пакетов и эффективности использования канала;
- AQM на основе класса;
- AQM на основе теории контроля.

Наиболее распространенными и перспективными являются AQM-алгоритмы, основанные на:

- средней длине очереди (RED);
- степени перегрузки («цене») канала связи (REM);
- вычислении ошибки между текущей длиной очереди и заданной длиной (PI).

Метод произвольного раннего определения (RED)

Алгоритм раннего произвольного обнаружения RED базируется на вычислении среднего размера очереди и вычислении вероятности отбрасывания пакетов. При определении вероятности отбрасывания пакетов механизм RED вычисляет не текущий, а экспоненциально взвешенный средний размер очереди. Текущий средний размер определяется на основании предыдущего среднего и текущего действительного размера. Использование механизмом RED среднего размера очереди обусловлено стремлением реагировать только на продолжительную перегрузку сети и «не замечать» моментальных всплесков трафика. Средний размер очереди avg вычисляется по формуле [2, 5]:

$$a\imath \quad = (1-)a \ g_{ol} \ + W * q, \qquad (1)$$

де W - весовой коэффициент очереди, $W = 2^{\frac{1}{n}}$; n - экспоненциальный весовой коэффициент; q - текущий размер очереди; $a\imath _ol$ - значение средней длины очереди на предыдущем шаге измерений.

Вероятность отбрасывания пакетов P_a представляет собой функцию, линейно зависящую от среднего размера очереди (рис. 2).

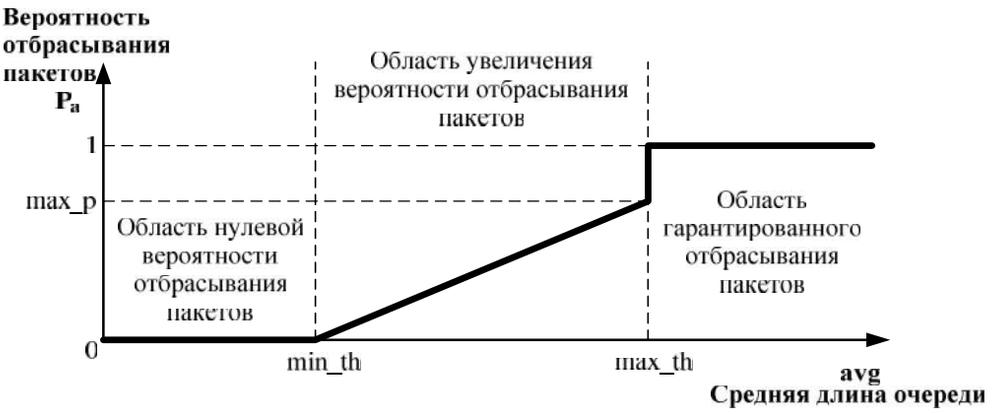

Рис. 2. Зависимость вероятности отбрасывания пакетов механизмом RED от средней длины очереди

Помимо этого, данная функция зависит также от минимального порогового значения $\min_th$, максимального порогового значения $\max_th$ и знаменателя граничной вероятности $\max_p$ (mark probability denominator), определяющего часть отбрасываемых пакетов при достижении средним размером очереди максимального порогового значения. Вероятность отбрасывания пакетов рассчитывается по формуле:

$$P_a = \begin{cases} 0, & \text{если } \arg < \min_{th}; \\ \max_p \cdot \dfrac{ai - \min_{th}}{\max_{th} - \min_{th}}, & \text{если } \min_{th} \le \arg < \max_{th}; \\ 1, & \text{если } \max_{th} \le \arg. \end{cases} \quad (2)$$

Когда средний размер очереди превышает минимальное пороговое значение, механизм RED начинает отбрасывать пакеты. Интенсивность отбрасывания пакетов возрастает прямо пропорционально возрастанию среднего размера очереди до тех пор, пока он не достигнет максимального порогового значения. Когда средний размер очереди превышает максимальное пороговое значение, механизм RED отбрасывает все пакеты, предназначенные для постановки в очередь.

Метод пропорционально-интегрального контроллера (PI)

Пропорционально-интегральный контроллер PI (Proportional-Integral) [6] является классическим регулятором с обратной связью, используемым в системах автоматического управления. Работа контроллера основывается на вычислении ошибки между текущей длиной очереди q(t) и заданной длиной q_{ref}:

$$e(t) = q(t) - q_{ref}, \quad (3)$$

Управляющий сигнал u(t), изменяющий вероятность сброса, вычисляется как сумма трех значений: первое слагаемое пропорционально текущей ошибки e(t), второе слагаемое пропорционально интегралу ошибки (для устранения постоянной составляющей) и последнее слагаемое - дифференциалу ошибки (для противодействия будущим отклонениям):

$$u(t) = K_p e(t) + Kt \int e(t)dt + K_D \frac{d}{dt} e(t), \quad (4)$$

где K_x - соответствующие коэффициенты пропорциональности.

На практике для управления очередью используется упрощенная дискретная формула вычисления вероятности сброса без дифференциального слагаемого:

$$p(k) = (a - b\} \times \times (q(k) - q_{ref} + + b(q(k) - q(k-1) + p(k-1)), \quad (5)$$

где коэффициенты пропорциональности a и b фиксированы и равны $1{,}822*10^{-5}$ и $1{,}81*10^{-5}$ соответственно [6].

Метод случайного экспоненциального маркирования (REM)

В методе случайного экспоненциального маркирования REM (Random Exponential Marking) [7] используется мера перегрузки p, называемая «ценой», и в момент времени kT вычисляемая по формуле:

$$p(k) = (a - b) \times \times (q(k) - q_{ref} + b(q(k) - q(k-1)) + p(k-1), \quad (6)$$

где c - пропускная способность канала (пакетов за интервал времени);

q(kT) - текущая длина очереди;

x(kT) - скорость поступления пакетов;

α и γ - константы больше нуля, равны 0.1 и 0.001 соответственно;

T - интервал времени измерений;

k - номер интервала.

Вероятность сброса/маркировки пакетов рассчитывается по формуле:

$$prob\,(kT) = 1 - \varphi^{-p(kT)}, \quad (7)$$

где φ — константа больше единицы.

Решение задачи

Исследование сети проводилось с использованием программного комплекса NS-2[8]. Схема сети (рис. 3) состоит из 4 FTP источников сообщения (S1, S2, S3, S4), которые с помощью маршрутизатора передают информацию на TCP-приемник.

Ниже представлена часть программного кода на языке Otcl (NS-2) отвечающая за создание узлов сети.

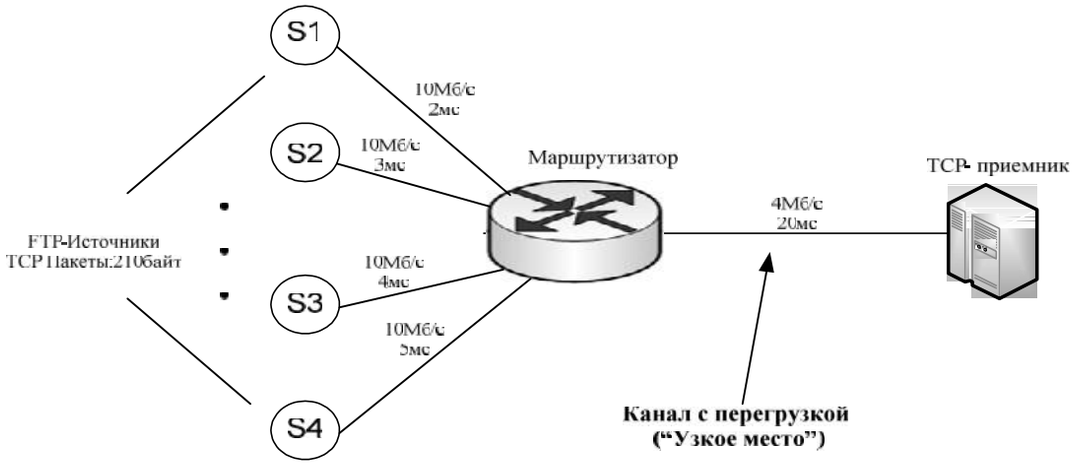

Рис. 3. Схема сети для имитационного моделирования

```
set ns [new Simulator]
set node_(s1) [$ns node]
set node_(s2) [$ns node]
set node_(r1) [$ns node]#Маршрутизатор
set node_(r2) [$ns node] # TCP-приемник
set node_(s3) [$ns node]
set node_(s4) [$ns node]
```

Скорость канала между источниками сообщения и маршрутизатором составляет 10 Мб/с, задержка для каждого источника разная: S1= 2 мс, S2= 3 мс, S3= 4 мс, S4= 5 мс; объем каждого пакета – 210 байт. Скорость канала между маршрутизатором и TCP-приемником составляет 4 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 4 сек - S2. На 8-й секунде включиться S3, а в момент времени 12 сек - S4. Программный код, отвечающий за установку параметров и соединение между узлами представлен ниже.

```
$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) 4Mb 20ms *#Вместо знака "звездочки" прописываем нужный алгоритм
$ns duplex-link $node_(s3) $node_(r1) 10Mb 4ms DropTail
$ns duplex-link $node_(s4) $node_(r1) 10Mb 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 left-down
$ns duplex-link-op $node_(s4) $node_(r1) orient left-up

set tcp1 [$ns create-connection TCP $node_(s1) TCPSink $node_(r2) 0]
$tcp1 set window_ 15
set tcp2 [$ns create-connection TCP $node_(s2) TCPSink $node_(r2) 1]
$tcp2 set window_ 15
set tcp3 [$ns create-connection TCP $node_(s3) TCPSink $node_(r2) 2]
$tcp1 set window_ 15
set tcp4 [$ns create-connection TCP $node_(s4) TCPSink $node_(r2) 3]
$tcp2 set window_ 15
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]
set ftp3 [$tcp3 attach-source FTP]
set ftp4 [$tcp4 attach-source FTP]
```

Продолжительность процесса моделирования составляет 25 сек. Приведем код, отвечающий за включение источников сообщения в сеть TCP/IP.

```
$ns at 0.0 "$ftp1 start"
$ns at 4.0 "$ftp2 start"
$ns at 8.0 "$ftp3 start"
$ns at 12.0 "$ftp4 start"
$ns at 25.0 "finish"
```

После выполнения программы получим графики зависимости длины очереди(Мб) от времени работы сети для каждого алгоритма.

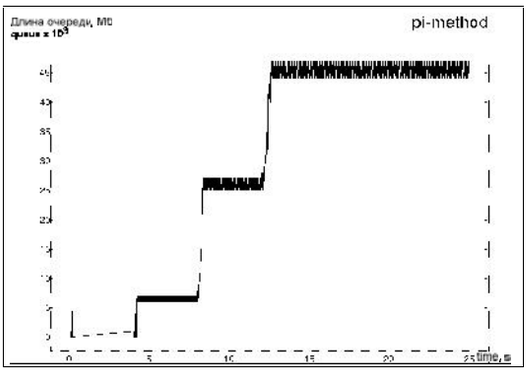

Рис. 4. Длина очереди с использованием PI-алгоритма

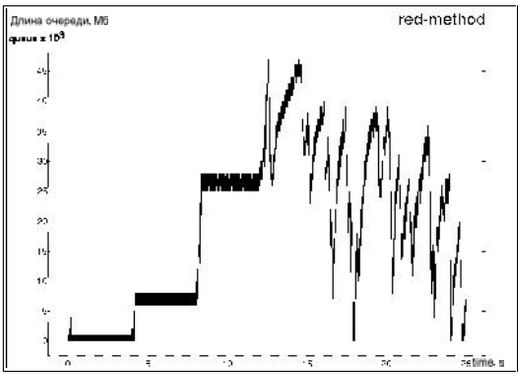

Рис. 5 Длина очереди с использованием RED-алгоритма

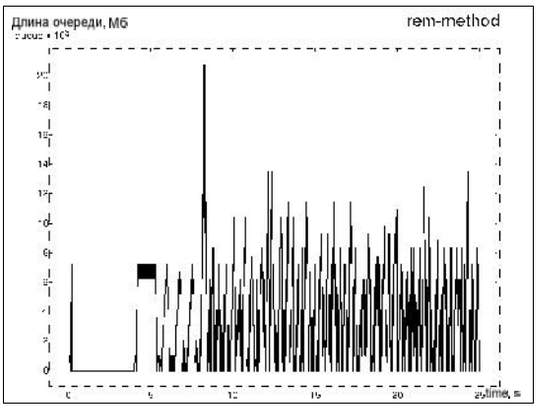

Рис. 6. Длина очереди с использованием REM-алгоритма

Выводы

Учитывая среднее значение длины очереди (рис. 4 - 6), можно сделать вывод о том, что REM-алгоритм является наиболее подходящим, среди всех выше представленных активных алгоритмов управления очередью (RED, PI), для использования в сетевых маршрутизаторах, поскольку с увеличением информационных потоков внутри TCP/IP сети, длина очереди(Мб) остается стабильной, что в свою очередь позитивно влияет на качество работы сети.

Список литературы

1. Коваленко Т.Н. Модель активного управления очередями в расределенных инфокоммуникационных системах, представленная сетью Петри / Т.Н. Коваленко // журнал «Проблемы телекоммуникаций». – 2012. – № 2(7). – С.58-67.

2. Лемешко О.В. Багатоканальний електрозв'язок та телекомунікаційні технології: Підручник. У 2-х ч. Ч.1 / О.В. Лемешко, В.А. Лошаков, В.В. Поповський та ін.; за заг. ред. проф. В.В. Поповського. – Х.: ТОВ «Компанія СМІТ», 2010. – 470 с.

3. Вегешна Ш. Качество обслуживания в сетях IP: пер. с англ. – М. : Вильямс, 2003. – 368с.

4. Кучерявый Е. А. Управление трафиком и качество обслуживания в сети Интернет / Е. А. Кучерявый. – СПб. : Наука и техника, 2004. – 336 с.

5. An active queue management scheme for Internet congestion control and its application to differentiated services / L. Su and J.C. Hou, The Ohio State University, Columbus, OH, January 2000. – 25 p.

6. Hollot C.V., Misra V., Towsley D. and Gong W.B., Analysis and Design of Controllers for AQM Routers Supporting TCP Flows, IEEE Transactions on Automatic Control, June 2002, pp. 945-959.

7. Athuraliya, S., Low, S.H., Li, V.H., Qinghe Yin, REM: active queue management, IEEE Networking Magazine 15 (3) (2001) p. 48-53.

8. The Network Simulator NS-2, http://www.isi.edu/nsnam/ns/

9. Масленников А.Г. Методы активного управления очередями маршрутизаторов / А.Г. Масленников// Московский технический университет связи и информатики, BC/NW 2011; №2 (19). – 1-12с.

Поступила в редакцію 1.04.2014

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

ДОСЛІДЖЕННЯ МЕРЕЖІ TCP/IP ІЗ ВИКОРИСТАННЯМ ОСНОВНИХ АЛГОРИТМІВ АКТИВНОГО УПРАВЛІННЯ ЧЕРГОЮ

В.І. Гостєв, Т.П. Довженко, О.С. Артющик

У статті розглянуто основні алгоритми активного управління чергами (RED, PI, REM) та проведено дослідження TCP/IP мережі із застосуванням цих алгоритмів.

Ключові слова: *PEM, RED, PI, AQM, алгоритм активного управління чергою, TCP/IP-протокол.*

RESEARCH OF TCP / IP NETWORK USING THE BASIC ALGORITHM OF ACTIVE QUEUE MANAGEMENT

V.I. Gostev, T.P. Dovzhenko, A.S. Artyuschyk

In this article consideration of the basic algorithms of active queue management (RED, PI, REM) and research TCP/IP network using these algorithms.

Keywords: *REM, RED, PI, AQM, active queue management algorithm, TCP/IP-protocol.*