УДК 004.715

Т. П. ДОВЖЕНКО, аспирант,
Государственный университет телекоммуникаций, Киев

ИССЛЕДОВАНИЕ СЕТИ *TCP/IP* С ПРИМЕНЕНИЕМ ОСНОВНЫХ АЛГОРИТМОВ АКТИВНОГО УПРАВЛЕНИЯ ОЧЕРЕДЬЮ *(RED, REM)*

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

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

Вступление

Неравномерный рост скоростей каналов передачи данных неизбежно приводит к появлению «узких» мест в телекоммуникационной сети и, соответственно, к возникновению перегрузок, особенно при подключении сетей доступа к транспортной сети. Традиционные протоколы управления очередью и предотвращения перегрузок не справляются с управлением трафиком, характеризуемым 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, таких как процент потерянных пакетов, значения задержек и вариации задержек.

В данной статье предлагается исследование сети TCP/IP с применением основных современных алгоритмов управления очередью, а именно: **RED** (*Random Early Detection*) и **REM** (*Random Exponential Marking*).

Постановка задачи

Главная цель статьи — исследование работы AQM-системы с RED- и REM-алгоритмами в условиях изменения нагрузки трафика.

Обзор алгоритмов управления очередью

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

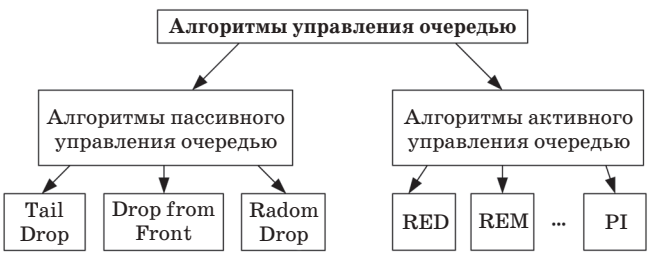

Рис. 1. Классификация алгоритмов управления очередью

Основные задачи алгоритмов управления очередью — минимизация средней длины очереди при одновременном обеспечении высокого коэффициента использования канала, а также справедливое распределение буферного пространства между различными потоками данных [1]. Схемы управления очередью различаются, в основном, критерием, по которому отбрасываются пакеты, и местом в очереди, откуда происходит 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 очередью*** **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 очередью*** **AQM**, обеспечивающие заблаговременное обнаружение перегрузки. Основные цели алгоритмов AQM:

- минимизация джиттера задержки пакетов при помощи контроля средней длины очереди;
- предотвращение эффекта глобальной синхронизации TCP трафика;
- обеспечение непредвзятого обслуживания трафика, характеризующегося кратковременными всплесками;
- строгое ограничение максимальной средней длины очереди.

Примерами алгоритмов AQM являются ранее упомянутые RED, REM, а также **WRED** (*Weighted Random Early Detection*), **AVQ** (*Adaptive Virtual Queue*), **FLC** (*Fuzzy Logic Controller*) и др. [2–4]. AQM-алгоритмы могут быть классифицированы по критериям принятия решения об отбрасывании пакетов из очереди при возникновении перегрузки. Можно выделить четыре типа алгоритмов управления очередью [2]:

1) AQM на основе средней длины очереди;
2) AQM на основе уровня потери пакетов и эффективности использования канала;
3) AQM на основе класса;
4) AQM на основе теории контроля.

Наиболее распространенными и перспективными являются AQM-алгоритмы на базе:

- средней длины очереди (RED);
- степени перегрузки («цены») канала связи (REM).

Метод произвольного раннего обнаружения RED

Алгоритм произвольного раннего обнаружения RED базируется на вычислении средней длины очереди и вероятности отбрасывания пакетов. При определении вероятности отбрасывания пакетов механизм RED вычисляет не текущую, а экспоненциально взвешенную среднюю длину очереди. Текущая средняя длина определяется на основании предыдущей средней и текущей действительной длины. Использование механизмом RED средней длины очереди обусловлено стремлением реагировать только на продолжительную перегрузку сети и «не замечать» моментальных всплесков трафика. Средняя длина очереди вычисляется по формуле [2; 5]

$$\text{avg} = (1 - W)\text{avg}_{\text{old}} + W \cdot q, \qquad (1)$$

где W — весовой коэффициент очереди, $W = 2^{\frac{1}{n}}$; n — экспоненциальный весовой коэффициент; q — текущий размер очереди; avg_{old} — средняя длина очереди на предыдущем шаге измерений.

Вероятность P_{avg} отбрасывания пакетов представляет собой функцию, линейно зависящую от средней длины очереди (рис. 2). Эта функция зависит также от минимального порогового значения $\min_{\text{th}}$, максимального порогового значения $\max_{\text{th}}$ и знаменателя граничной вероятности $\max_{\text{p}}$ (*mark probability denominator*), определяющего часть отбрасываемых пакетов при достижении средней длины очереди максимального порогового значения. Вероятность отбрасывания пакетов рассчитывается по формуле

$$P_{\text{avg}} = \begin{cases} 0, & \text{avg} < \min_{\text{th}}; \\ \max_{\text{p}} \cdot \dfrac{\text{avg} - \min_{\text{th}}}{\max_{\text{th}} - \min_{\text{th}}}, & \min_{\text{th}} \le \text{avg} < \max_{\text{th}}; \\ 1, & \max_{\text{th}} \le \text{avg}. \end{cases} \qquad (2)$$

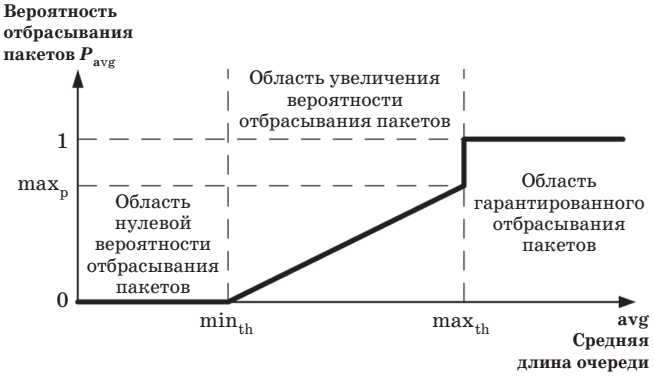

Рис. 2. Зависимость вероятности отбрасывания пакетов механизмом RED от средней длины очереди

Когда средняя длина очереди превышает минимальное пороговое значение, механизм RED начинает отбрасывать пакеты. Интенсивность их отбрасывания возрастает прямо пропорционально увеличению средней длины очереди до тех пор, пока она не достигнет максимального порогового значения. Когда средняя длина очереди превышает максимальное пороговое значение, механизм RED отбрасывает все пакеты, предназначенные для постановки в очередь.

Метод случайного экспоненциального маркирования REM

Основное отличие метода REM от RED-алгоритма заключается в использовании меры определения перегрузки сети и функции вероятностного сброса/маркировки пакетов. В методе случайного экспоненциального маркирования REM [6] мера p перегрузки, называемая *ценой*, в момент времени kT вычисляется по формуле:

$$p(k) = \max(0, p((k-1)T) + \gamma(\alpha(q(kT) - q_{\text{ref}}) + x(kT) - c)), \qquad (3)$$

где c — пропускная способность канала (пакетов за интервал времени); $q(kT)$ — текущая длина очереди; q_{ref} — заданая длина очереди; $x(kT)$ — скорость поступления пакетов; α и γ — положительные константы, равные соответственно 0,1 и 0,001; T — интервал времени измерений; k — номер интервала.

Основной идеей является стабилизация пропускной способности линии связи и уменьшение длины очереди маршрутизатора.

Каждый выходной затор при REM-методе содержит переменную, в которой хранится информация о перегрузки сети. Эта переменная используется для определения вероятности сброса/маркировки пакетов, которая рассчитывается по формуле

$$\text{prob}(kT) = 1 - \varphi^{-p(kT)}, \qquad (4)$$

где φ — некоторая константа, $\varphi > 1$.

Обновление меры перегрузки происходит периодически или асинхронно, в зависимости от степени несоответствия (т. е. разности между скоростью вводящего потока и пропускной способностью канала) и очереди несоответствия (т. е. разности между длиной очереди и цели). Цена перегрузки увеличивается, если взвешенная сумма этого несоответствия является положительной, и уменьшается в противном случае. Взвешенная сумма имеет положительное значение, когда либо входной поток превышает пропускную способность канала, либо в буфере имеется избыток непереданных данных, который должен быть устранен. В противном случае цена будет отрицательной. Когда число пользователей увеличивается, несоответствия в скорости и очереди растут, увеличивая меру перегрузки и, следовательно, вероятность маркировки/сброса пакетов. При этом к источникам

сообщения посылается сигнал о перегруженности сети, после чего источники сокращают соответственные цены перегрузки. При низких входных скоростях от источников произойдет снижение цены перегрузки и вероятности маркировки, что приведет к росту коэффициента использования сети, незначительным потерям данных и задержке.

Графики зависимости вероятности сброса пакетов от цены перегрузки для RED и REM приведены на рис. 3.

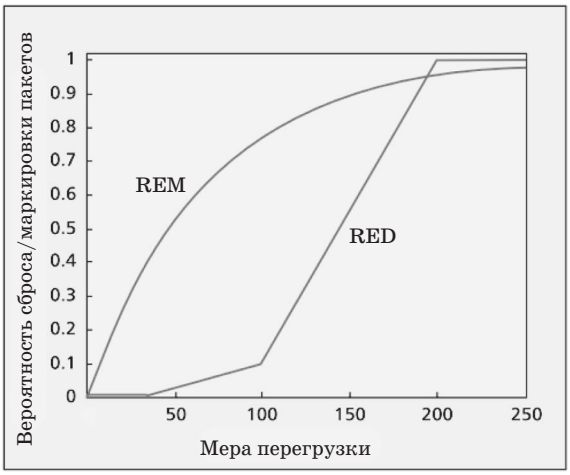

Рис. 3. Функциональная зависимость вероятности сброса пакетов от меры перегрузки для RED- и REM-методов

Решение задачи

Исследование сети проводилось с использованием программного комплекса NS-2 [7]. Схема сети (рис. 4) состоит из 4 FTP источников сообщения (S1, S2, S3, S4), которые с помощью маршрутизатора передают информацию на TCP-приемник. Далее представлена часть программного кода на языке Otcl (NS-2), отвечающая за создание узлов сети.

```
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 Мбит/с, задержка для каждого источника своя: 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 Мбит/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 с — S2. На 8-й секунде включится S3, а в момент времени 12 с — S4. Программный код, отвечающий за установку параметров и соединение между узлами, представлен далее.

```
$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) queue-
Pos 0
$ns duplex-link-op $node_(r2) $node_(r1) queue-
Pos 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
```

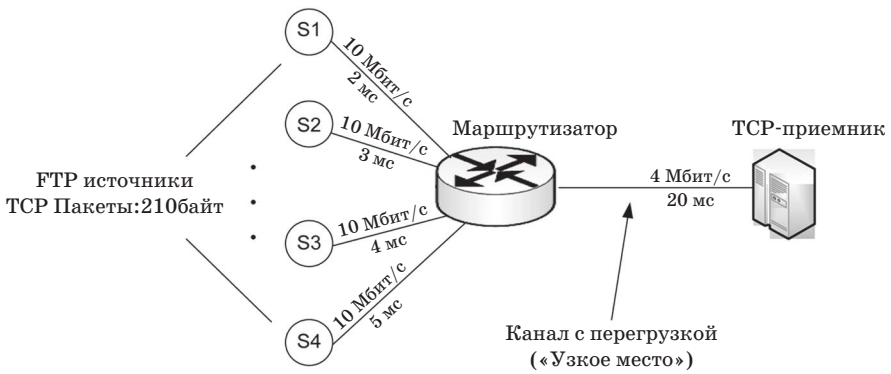

Рис. 4. Схема сети для имитационного моделирования

```
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 с. Код, отвечающий за включение источников сообщения в сеть TCP/IP, имеет следующий вид:

```
$ns at 0.0 "$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 получаем графики зависимости длины очереди, Мбит, от времени работы сети для каждого алгоритма (рис. 5 и 6).

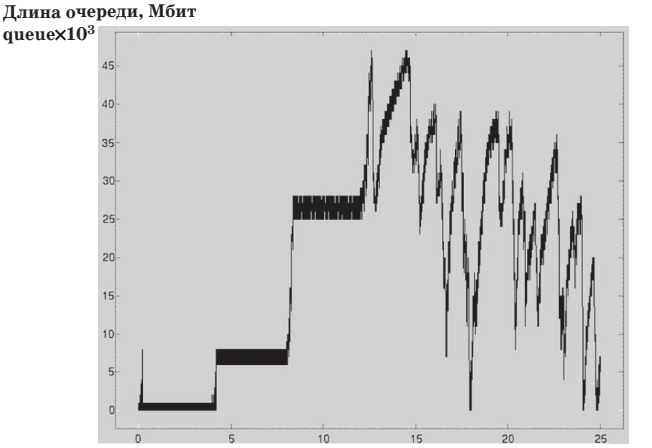

Рис. 5. Длина очереди с использованием RED-алгоритма

Вывод

Учитывая среднее значение длины очереди (см. рис. 5 и 6), приходим к выводу о том, что REM-алгоритм является наиболее подходящим

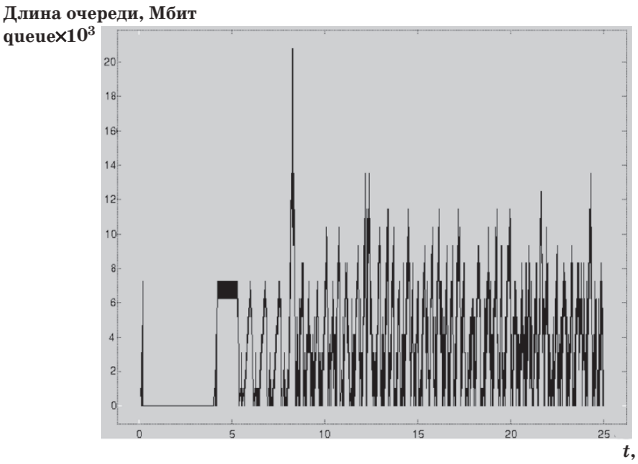

Рис. 6. Длина очереди с использованием REM-алгоритма

для использования в сетевых маршрутизаторах, поскольку с увеличением информационных потоков внутри TCP/IP сети длина очереди положительно остается стабильной, что влияет на качество работы сети.

Литература

1. ***Коваленко, Т. Н.*** *Модель активного управления очередями в распределенных инфокоммуникационных системах, представленная сетью Петри / Т. Н. Коваленко // Проблемы телекоммуникаций.— 2012.— № 2 (7).— С. 58–67.*

2. ***Багатоканальний*** *електрозв'язок та телекомунікаційні технології: підручник: 2 ч. Ч.1 / [О. В. Лемешко, В. А. Лошаков, В. В. Поповський та ін.; за заг. ред. проф. В. В. Поповського].— Харк.: ТОВ «Компанія СМІТ», 2010.— 470 с.*

3. ***Вегешна, Ш.*** *Качество обслуживания в сетях IP / Ш. Вегешна; пер. с англ.— М. : Вильямс, 2003.— 368 с.*

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

5. ***Su, L.*** *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. ***REM:*** *active queue management / [S. Athuraliya, S. H. Low, V. H. Li, Yin Qinghe] // IEEE Networking Magazine.— 2001.— 15(3).— P. 48–53.*

7. ***The Network Simulator*** *NS-2 [Електронний ресурс].— Режим доступу:*
http://www.isi.edu/nsnam/ns/

Т. П. Довженко

ДОСЛІДЖЕННЯ МЕРЕЖІ *TCP/IP* ІЗ ВИКОРИСТАННЯМ ОСНОВНИХ АЛГОРИТМІВ АКТИВНОГО КЕРУВАННЯ ЧЕРГОЮ (*RED, REM*)

Розглянуто основні алгоритми активного керування чергою (RED, REM) і виконано дослідження мережі TCP/IP із застосуванням цих алгоритмів.

Ключові слова: PEM; RED; AQM; алгоритм активного керування чергою; TCP/IP-протокол.

T. Dovzhenko

RESEACH OF *TCP/IP* NETWORK USING THE BASIC ALGORITHM OF ACTIVE QUEUE MANAGEMENT (*RED, REM*)

In this article are considered basic algorithms of active queue management (RED, REM) and research TCP/IP network with using these algorithms.

Keywords: REM; RED; AQM; active queue management algorithm; TCP/IP-protocol.