Математическое моделирование

УДК 004.021, 519.2, 519.6

Численное и имитационное моделирование дисциплин обслуживания очередей типа RED на маршрутизаторе

Д. С. Кулябов, М. Н. Геворкян, Х. Р. Мачука, К. Диаррассуба, Д. Т. Г. Дали

Кафедра прикладной информатики и теории вероятностей
Российский университет дружбы народов
ул. Миклухо-Маклая, д. 6, Москва, Россия, 117198

Целью данной работы является обобщение стохастической модели RED (Random Early Detection) на случай дисциплин обслуживания AURED, SARED и GRED, а также верификация результатов численного моделирования с помощью имитационного моделирования. Стохастическая модель строится на основе системы из трёх стохастических уравнений Ито. Численное решение проводится с помощью стохастических методов Рунге–Кутты со слабой сходимостью второго порядка. Комплекс программ для численного моделирования написан авторами статьи на языке Python версии 3 с использованием библиотек `NumPy` и `SciPy`. В статье достаточно подробно описываются компоненты программного комплекса. Для имитационного моделирования авторы использовали открытый пакет программного обеспечения для моделирования компьютерных сетей `NS2`. В данной статье авторы лишь кратко описывают те моменты, которые касаются дисциплин обслуживания очередей, не останавливаясь на общем описании NS2. Результаты моделирования представлены в виде графиков зависимостей средней и текущей длин очереди от времени. На основе анализа полученных графиков показано, что имитационное и численное моделирование дало качественно соответствующие друг другу результаты.

Ключевые слова: RED, маршрутизация, стохастические дифференциальные уравнения, стохастические численные методы, ns2.

1. Введение

В современных компьютерных сетях большую роль играют системы маршрутизации. Часто именно неэффективная работа маршрутизатора является причиной низкой производительности сети. К современным маршрутизаторам предъявляются строгие требования по обеспечению стабильной работы и должного качества обслуживания сетевого трафика.

Одним из инструментов по управлению трафиком на маршрутизаторе является протокол обслуживания очереди (буфера). В настоящий момент наиболее широко используются дисциплины обслуживания типа RED (Random Early Detection). RED был впервые представлен в статье [1] и в дальнейшем появилось множество усовершенствований исходного алгоритма, в частности алгоритмы ARED, AURED, GRED и SARED [2–4]. Базовые принципы работы всех модификаций алгоритма RED весьма сходны, поэтому имеет смысл говорить о семействе алгоритмов типа RED.

В данной статье представлен комплексный анализ нескольких дисциплин обслуживания типа RED, а именно: ARED [5], AURED [2], GRED [6] и SARED [4]. Для анализа используются методы численного и имитационного моделирования. В первой части статьи изложены основные сведения по изучаемым дисциплинам обслуживания очередей, описываются алгоритмы их функционирования. Далее кратко излагаются стохастическая и детерминированная модели RED, представленные в статьях [7, 8]. Данные модели распространены авторами на изучаемые алгоритмы.

Статья поступила в редакцию 21 февраля 2016 г.
Работа частично поддержана грантами РФФИ № 14-01-00628, 15-07-08795 и 16-07-00556.

Во второй части статьи описывается созданный авторами программный комплекс для численного моделирования дисциплин обслуживания типа RED [9, 10], даётся описание его архитектуры и вычислительных возможностей. Также кратко изложены основные моменты, касающиеся имитационного моделирования дисциплин RED, ARED и GRED в NS2.

В третьей части описываются численный и имитационный эксперименты. Численное моделирование проводится с помощью комплекса программ, описанных во второй части. Имитационное моделирование проводится с помощью пакета программ NS2 (Network Simulator) [11]. На основании проведённых численного и имитационного эксперимента делается вывод о качественном соответствии численной и имитационной моделей.

2. Дисциплины обслуживания очередей

2.1. Алгоритм RED

Рассмотрим алгоритм работы дисциплины Random Early Detection (RED) [1].

Алгоритм 1 Алгоритм RED

$q_{avg} \leftarrow 0$
$count \leftarrow -1$

while *packet* **do** ▷ Пока приходят пакеты
  $q_{avg} \leftarrow QueueAvg(\ )$ ▷ Вычисляем средний размер очереди
  if $q > 0$ **then**
    $q_{avg} \leftarrow (1 - w_q)q_{avg} + w_q q$
  else
    $m \leftarrow f(time - q_{time})$
    $q_{avg} \leftarrow (1 - w_q)^m q_{avg}$
  end if

  if $q_{min} < q_{avg} < q_{max}$ **then**
    $count + count + 1$
    $p \leftarrow p_{max}(q_{avg} - q_{min})/(q_{max} - q_{min})$
    $p \leftarrow p/(1 - p \cdot count)$

    $Mark(packet, p)$ ▷ Маркируем пакет в вероятностью p
    $count \leftarrow 0$
  else if $q_{avg} \geqslant q_{max}$ **then**
    $Mark(packet, 1)$
    $count \leftarrow 0$
  else
    $count \leftarrow -1$
  end if

  if $q = O$ **then** $q_{time} \leftarrow time$
  end if
end while

Переменные:
– Q и q — максимальная вместительность очереди и текущая длина очереди;
– $\hat{Q}$ или q_{avg} — средняя длина очереди (avg — average queue size);
– q_{time} — время, с которого очередь находится в пустом состоянии;
– $count$ — число пакетов, начиная с первого помеченного;

Константы:

– w_q — вес очереди;
– q_{min} — минимальный порог заполнения очереди;
– q_{max} — максимальная порог заполнения очереди;
– p_{max} — максимальное значение вероятности сброса p;

Остальные параметры:

– p — текущая вероятность того, что пакет будет помечен;
– $time$ — текущее время;
– $f(t)$ — линейная функция от времени t.

При использовании RED маршрутизатор вычисляет средний размер очереди, используя низкоуровневый фильтр с экспоненциальным весовым коэффициентом (*экспоненциально взвешенное скользящее среднее*). Средний размер очереди сравнивается с двумя пороговыми значениями — минимальным и максимальным порогами. Если средний размер очереди меньше, чем минимальный порог, то пакеты не маркируются для сброса. Если средний размер очереди превышает максимальный порог, то все прибывающие пакеты маркируются для последующего сброса. При этом гарантируется, что при увеличении числа сброшенных пакетов, средний размер очереди не превысит максимального порога.

Когда средний размер очереди находится в пределах минимального и максимального порогов, то каждый пакет помечается для последующего сброса с вероятностью p_a, где p_a — функция от среднего размера очереди avg. Каждый раз, когда пакет помечается, вероятность того, что он пришёл от конкретного соединения, прямо пропорциональна полосе пропускания этого соединения.

2.2. Алгоритм AURED

Алгоритм AURED (Autonomus RED) [2] вместо фиксированного значения p_{max} изменяет p_{max} периодически, основываясь на оценке уровня производительности.

Алгоритм 2 Алгоритм AURED

$P_k \leftarrow q_k/Q$
if $P_k \leqslant P_{k-1}$ **then**
  $op_k \leftarrow -op_{k-1}$
else
  if $op_k > 0$ **then**
    $p_{max} \leftarrow \alpha + p_{max}$
  else
    $p_{max} \leftarrow p_{max} \cdot \beta$
  end if
end if
$P_k \leftarrow P_k$
$op_{k-1} \leftarrow op_k$

Специфические параметры алгоритма

– P_k — текущий уровень производительности, вычисляется как $P_k = q_k/Q$;
– P_{k-1} — уровень производительности на предыдущем шаге;
– op_k, op_{k-1} — оператор настройки (adjustment operator), предлагается взять $op_0 = 1$;
– α, β — параметры, позволяющие настроить изменение p_{max}. Авторы алгоритма предлагают $\alpha = 0.02$, а $\beta = 0.8$.

2.3. Алгоритм SARED

Задача алгоритма SARED [10] — удерживать очередь в стабильном состоянии и близко к заданному уровню $[q_{min}, q_{max}]$. В SARED для вычисления экспоненциального взвешенного скользящего среднего используется два весовых коэффициента w_{qn} (normal) и w_{qh} (high). На каждом шаге алгоритма корректируется величина $\hat{q}$ по следующим формулам:

$$\hat{q} = \begin{cases} (1 - w_{q\ normal})\hat{q} + w_{q\ normal}q, & \hat{q} \in I, \\ (1 - w_{q\ high})\hat{q} + w_{q\ high}q, & \hat{q} \notin I, \end{cases}$$

где I обозначает интервал от $q_{min} + 0.4 \cdot (q_{max} - q_{min})$ до $q_{min} + 0.6 \cdot (q_{max} - q_{min})$.

2.4. Алгоритм GRED

В алгоритме GRED (Gentle RED) [6] заданные границы очереди разбиваются на два интервала: $[q_{min}, q_{max}]$ и $[q_{max}, 2q_{max}]$, а также изменяется алгоритм вычисления вероятности сброса.

Алгоритм 3 Алгоритм GRED

if $q_{avg} < q_{min}$ **then**
  $p = 0$
else if $q_{min} \leqslant q_{avg} < q_{max}$ **then**
  $p \leftarrow p_{max}(q_{avg} - q_{min})/(q_{max} - q_{min})$
  $p \leftarrow p/(1 - p \cdot count)$
else if $q_{max} \leqslant q_{avg} < 2q_{max}$ **then**
  $p \leftarrow p_{max} + (1 - p_{max}) \cdot (q_{avg} - q_{max})/q_{max}$
  $p \leftarrow p/(1 - p \cdot count)$
else if $q_{avg} \geqslant$ **then**
  p = 1
end if

2.5. Стохастическая и детерминированная модели модуля управления на маршрутизаторе с дисциплиной обслуживания типа RED

Кратко опишем стохастическую модель модуля управления на маршрутизаторе с дисциплиной обслуживания типа RED.

Стохастическая модель представляет собой систему из трёх стохастических дифференциальных уравнений Ито:

$$\begin{cases} \mathrm{d}W(t) = \left(\dfrac{1}{T(t)} - \dfrac{W^2(t)P(\hat{Q})}{T(t)} \right) \mathrm{d}t + \sqrt{\dfrac{1}{T(t)} + \dfrac{W^2(t)P(\hat{Q})}{T(t)}} \mathrm{d}V^1, \\ \mathrm{d}Q(t) = \left(\dfrac{W(t)}{T(t)} - C \right) \mathrm{d}t + \sqrt{\dfrac{W(t)}{T(t)} - C} \mathrm{d}V^2, \\ \mathrm{d}\hat{Q}(t) = w_q C(Q(t) - \hat{Q}(t)). \end{cases}$$

Здесь $W(t)$ — размер TCP окна, $Q(t)$ — мгновенная длина очереди, $\hat{Q}(t)$ — экспоненциально взвешенное скользящее среднее длины очереди, $T(t)$ — время двойного оборота (за время двойного оборота приходят все подтверждения на отправленное окно TCP), $P(\hat{Q})$ — функция вычисления вероятности сброса пакета,

C — интенсивность обслуживания, $\mathrm{d}V^1, \mathrm{d}V^2$ — винеровские процессы, соответствующие случайным процессам $W(t)$ и $Q(t)$. Вывод и обоснование модели см. в работе [8].

Детерминированная модель, подробно рассмотренная в работе [7], получается из стохастической системы путём отбрасывания стохастического члена.

3. Программное обеспечение для моделирования алгоритмов обслуживания очереди типа RED

3.1. Численное моделирования алгоритмов типа RED

Программный комплекс для численного моделирования дисциплин обслуживания очереди на маршрутизаторе типа RED был написан на языке программирования Python версии 3 с использованием следующих библиотек: `numpy`, `matplotlib` и `scipy`. Весь программный код открыт и доступен по ссылке `https://bitbucket.org/mngev/red-modeling-public`.

3.1.1. Структура программного комплекса

Модуль `red` состоит из нескольких файлов.

- `red.py` — основной файл, в котором находятся все классы, реализующие дисциплины обслуживания очереди типа RED;
- `rungekutta.py` — несколько методов, реализующих детерминированные численные схемы Рунге-Кутты 2,3,4,5 и 6 порядков (используются в `red.py` для решения системы обыкновенных дифференциальных уравнений);
- `sde.py` — набор функций, реализующих стохастические численные схемы типа Рунге-Кутты (также используются в `red.py` для решения системы стохастических дифференциальных уравнений);
- `plot.py` — набор функций для графического представления результатов вычисления;
- `main.py` — сценарии стандартных вычислений (вычислить и нарисовать графики стохастического и детерминированного фазового портрета, решений уравнений и параметров автоколебаний и т.д.);
- `settings.py` — в данном файле несколько функций для обработки конфигурационных ini файлов, на основе которых создаются словари python с настройками вычислений.

3.1.2. Описание классов дисциплин обслуживания

При разработке программы использовался объектно-ориентированный подход. В качестве объектов были выбраны дисциплины обслуживания очередей. Так как все они основаны на дисциплине обслуживания RED и отличаются лишь в деталях, то естественным способом их программной реализации было выделение объекта RED в родительский класс и наследование его всеми остальными объектами (ARED, AURED, DSRED, EFRED, POWARED, RARED, SARED, WRED, GRED). При этом подклассы наследуют все методы и атрибуты родительского класса и переопределяют лишь те методы, которые отличают конкретную дисциплину обслуживания от RED. Ни в одном подклассе не вводится новых методов, поэтому достаточно дать *описание родительского класса RED*.

Метод `__init__(self, q_min=0.2, q_max=0.4)` — играет роль конструктора класса RED и инициирует следующие атрибуты:

- `name` — имя дисциплины обслуживания,
- `Tp` — время прохождения пакета от источника до узла и обратно,
- `wq` — вес очереди,
- `w_max` — максимальный размер TCP-окна,
- `q_min` — минимальное пороговое значение пакетов для алгоритма RED,

– `q_max` — максимальное пороговое значение пакетов для алгоритма RED,
– `c_small` — количество обслуживаемых за 1 секунду пакетов,
– `R` — размер буфера,
– `p_max` — максимальная вероятность сброса,
– `N` — количество узлов,
– `delta` — вычисляется на основе `c_small`,
– `time_interval` — временной интервал интегрирования, кортеж (t_0, T),
– `with_pto` — включить/выключить сброс по тайм-ауту.

Методы класса:

– `C(x)` — вычисление интенсивности обслуженной нагрузки;
– `P(x)` — функция вероятности сброса RED применяемая при решении дифференциальных уравнений;
– `P2(x)` — функция вероятности сброса RED применяемая при решении уравнений автоколебания;
– `Q(x)` — правая часть уравнения для вычисления мгновенного размера очереди;
– `Qe(x)` — правая часть третьего уравнения системы, вычисляющая экспоненциально взвешенное скользящее среднее значение мгновенной длины очереди;
– `Ttot(x)` — время потери по тайм-ауту T_{TO};
– `W(x)` — правая часть первого уравнения системы ОДУ. Функции для вычисления размера TCP-окна;
– `W_sde(x)` — правая часть первого уравнения системы СДУ. Функции для вычисления размера TCP-окна. Отличие от `W` в отсутствии знака минус у выражения `x[0]/2.0`;
– `ode(t, x, p)` — правая часть ОДУ. Три аргумента добавлены для совместимости и имеют следующий смысл:
  – `t` — время,
  – `x` — переменная в виде массива из трёх элементов `[W(t), Q(t), Qe(t)]`,
  – `p` — параметры.
– `p_to(x)` — вероятность сброса пр тайм-ауту;
– `sde_G(x)` — матрица диффузии (в данном случае она диагональна);
– `sde_f(x)` — вектор сноса;
– `self_oscillation_equation(x)` — система из двух уравнений для вычисления параметров автоколебания;
– `solve_ode(step=0.01, x_0=[1.0, 0.0, 0.0])` — метод решает систему ОДУ, ассоциированную с дисциплиной обслуживания. Для решения используется метод Рунге-Кутты из файла `rungekutta.py`; аргументы:
  – `step` — шаг,
  – `x_0` — начальные значения в виде списка из трёх элементов;
– `solve_sde(step=0.01, x_0=[1.0, 0.0, 0.0])` — метод решает систему СДУ, ассоциированную с дисциплиной обслуживания. Для решения используется метод из файла `sde.py`; аргументы те же что и у предыдущего метода;
– `solve_self_oscillation_equation_2d(equation_type='ode')` — решает систему алгебраических уравнений для параметров автоколебания в случае изменения только `q_max`;
– `solve_self_oscillation_equation_3d()` — решает систему алгебраических уравнений для параметров автоколебания в случае изменения и `q_max` и `q_min`.

3.1.3. Описание конфигурационных файлов

Конфигурационные файлы позволяют задавать параметры численного моделирования для различных дисциплин обслуживания очереди. Также имеется возможность задать различные параметры для одной и той же дисциплины обслуживания в случае необходимости изучения данной дисциплины при различных параметрах.

Все конфигурационные файлы должны иметь формат `ini` и находиться в директории `settings`. Файл `параметры вычислений.ini` обязателен, он состоит из трёх разделов.

В разделе `[Параметры метода]` заданы следующие параметры:
- `h` — шаг сетки численного метода,
- `W_0` — начальный размер TCP окна (начальное значение переменной W),
- `Q_0` — начальный размер очереди (начальное значение переменной Q),
- `Qe_0` — начальный средневзвешенный размер очереди (начальное значение переменной $\hat{Q}$).

В разделе `[Прочие настройки]` на данный момент указывается формат файла изображения графиков, который необходимо создать. Параметр `формат файла` может принимать значения `pdf` и `png`.

В разделе `[Типы вычислений]` можно включить или отключить тот или иной тип вычисления:
- `Решение детерминированное`
- `Решение стохастическое`
- `Спектр детерминированный`
- `Спектр стохастический`
- `Автоколебания детерминированные`
- `Автоколебания стохастические`

Также в директории `settings` можно создать файлы настройки параметров дисциплин, для которых предполагается произвести расчёты. Название этого файла может быть произвольным, но его следует обязательно добавить в список `qd_settings`, заданный в файле `main.py`. Все перечисленные в этом списке файлы будут последовательно использованы для вычислений. Если файл в список не включён, то он будет проигнорирован.

Файлы с настройками параметров дисциплин должны обязательно содержать раздел `[DEFAULT]`, в котором перечислены все возможные параметры и им присвоены значения по умолчанию. Все дальнейшие разделы должны начинаться с названия поддерживаемых расчётной программой дисциплин обслуживания (в `red.py` должен присутствовать соответствующий класс). После названия дисциплины должен стоять пробел, а далее может идти любой поясняющий текст.

Если в том или ином разделе не указать вообще ни одного параметра, то будут использованы значения по умолчанию. Название секции при этом убирать из файла не следует, так как в противном случае вычислений для этой дисциплины обслуживания проведено не будет.

Есть возможность для одной и той же дисциплины задать несколько разных наборов параметров, для этого в названии секции надо *через пробел* добавить любое дополнительное слово или число. Например, `[RED вариант 1]` и `[RED вариант 2]`. Оно будет использовано в названии файлов с результатами вычислений.

3.2. Имитационное моделирования дисциплины RED, ARED и GRED с помощью NS2

Вся необходимая информация по работе с ns2 изложена в официальной документации [11] и в примерах, доступных в дистрибутиве. Поэтому здесь мы лишь кратко опишем те моменты, которые касаются дисциплин обслуживания очередей.

В ns2 существуют реализации трёх разновидностей дисциплины RED: оригинальный алгоритм RED, адаптивный алгоритм ARED и Gentle RED. Выбор конкретного алгоритма осуществляется путём изменения параметров скрипта. Рассмотрим эти параметры подробнее.
- `bytes_`: включает (`true`) или выключает (`false`) режим «byte mode», в котором размер пакетов влияет на вероятность их пометки на сброс;
- `Queue-in-bytes_`: если значение параметра установлено как `true`, то средняя длина очереди будет измеряться в битах. Также при этом параметры `thresh_`

и `maxthres_` будут измеряться по вычисленному среднему размеру пакетов — - `mean_pktsize_`. По умолчанию устанавливается значение `false`;
- `thres_`: минимальный порог длины очереди $q_{\min}$;
- `maxthres_`: максимальный порог длины очереди $q_{\max}$;
- `mean_pktsize_`: приблизительная оценка размера пакета в битах. Значение по умолчанию — 500;
- `q_weight_`: весовой фактор w_q, используемый при вычислении средней длины очереди;
- `wate_`: этот параметр позволяет выдерживать интервал между отбрасываемыми пакетами, если установить его значение как `true`;
- `linterm_`: обратное значение параметра $p_{\max}$. По умолчанию 10;
- `setbit_`: принимает значение `false`, если RED отбрасывает помеченные пакеты. В случае установки значения `true`, в помеченные пакеты добавляется бит перегрузки (congestion bit) — некоторые реализации TCP реагируют на этот бит;
- `drop-tail_`: если значение истина, то при переполнении буфера или при превышении числа пакетов в очереди значения $q_{\max}$ переключатся на алгоритм Drop Tail.

Значения, установленные по умолчанию для параметров `q_weight_`, `maxthresh_` и `thres_`, равны соответственно 0,002, 15 и 5. В более поздних реализациях ns2 они вычисляются автоматически.

Параметры ARED и GRED. Чтобы задействовать ARED или GRED необходимо дополнительно задать следующие параметры:
- `adaptive_` — включение или отключение адаптивного алгоритма ARED;
- `alpha_` и `beta_` — значение параметров α и β дисциплины ARED;
- `gentle_` — включение или отключение Gentle RED.

Мониторинг очереди

Одним из наиболее важных объектов в ns-2 является монитор очереди. Он позволяет собирать информацию о длине очереди, о прибывших, уходящих и отброшенных пакетах. Для внедрения монитора между двумя узлами, необходимо добавить следующие строки:

```
#Включаем монитор очереди
set qmon [$ns monitor-queue $R1 $R2 [open qm.tr w] 0.01]
[$ns link $R1 $R2] queue-sample-timeout
```

Объект `monitor-queue` имеет 4 аргумента: первые два определяют соединение, на котором находится очередь, третий — выходной файл, куда будут записываться данные, а четвёртый — частота занесения данных в файл.

Выходной текстовой файл с результатами мониторинга состоит из 11 колонок: время, узел источник, узел приёмник (2 и 3 соответственно узлы, которые определяют очередь), размер очереди в битах, размер очереди в пакетах, число прибывших пакетов, число пакетов, покинувших очередь, число пакетов, отброшенных очередью, число прибывших битов, число битов вышедших из очереди, число отброшенных битов.

Мониторинг очереди RED

Для мониторинга параметров очереди RED (например, между узлами $n2$ и $n3$) необходимо добавить следующие строки кода:

```
set redq [[$ns link $n2 $n3] queue]
set traceq [open red-queue.tr w]
$redq trace curq_
$redq trace ave_
$redq attach $traceq
```

Здесь `curq_` — текущий размер очереди, а `ave_` — средний размер очереди. В результате получим выходной файл, состоящий из трёх колонок. Первая колонка содержит флаг `Q` (текущий размер очереди) или `a` (средний размер очереди). Далее следуют время и значение наблюдаемого параметра.

Мониторинг потоков

Файл «монитор потоков» включает более детальную информацию о типе отбрасывания. Он учитывает разницу между ранними отбрасываниями (Early Drops,

ED), т.е. такими, которые произошли благодаря работе алгоритма RED, и отбрасываниями, произошедшими в результате переполнения буфера. Файл имеет следующий формат.

- Колонка 1: время записи информации (данной строки) в файл.
- Колонка 2 и 5: обе колонки дают `id` потока.
- Колонка 3: `null` (нулевой показатель).
- Колонка 4: вид потока.
- Колонка 6 и 7: источник и пункт назначение потока.
- Колонка 8 и 9: полное число прибывших данных в конкретном потоке в пакетах и в битах.
- Колонка 10 и 11: число ранних отбрасываний из конкретного потока в пакетах и битах.
- Колонка 12 и 13: полное число прибывших данных во всех потоках в пакетах и битах.
- колонка 14 и 15: число ранних отбрасываний во всех потоках в пакетах и байтах.
- Колонки 16 и 17: число отбрасываний обоих типов во всех потоках в пакетах и в битах.
- Колонки 18 и 19: число отбрасываний обоих типов в конкретном потоке в пакетах и битах.

Добавление модулей AURED и SARED

В ns2 отсутствует реализация как AURED, так и SARED. Соответствующие модули были добавлены авторами путём модификации модуля `red.c`. Процесс добавления собственного модуля в ns2 сравнительно прост. Опишем его по шагам (используется дистрибутив ns-allinone-2.35):

- в каталоге `ns-allinone-2.35/ns-2.35/queue` создаём файлы с описанием алгоритмов: `aured.cc`, `aured.h`, `sared.cc`, `sared.h` (в нашем случае это были модифицированные копии файлов `red.cc` и `red.h`);
- после этого добавленные модули вносим в соответствующий список в файле `Makefile.in`, находящемся в каталоге `ns-allinone-2.35/ns-2.35`;
- запускаем компиляцию ns путём последовательного запуска двух скриптов `./configure` и `make` из директории `ns-allinone-2.35/ns-2.35`.

4. Численное и имитационное моделирование алгоритмов RED, AURED, SARED и GRED

Для численного моделирования алгоритмов AURED, SARED и GRED был использован вышеописанный программный комплекс. Стохастическая и детерминированная численные модели были распространены на случай алгоритмов AURED, SARED и GRED. Был написан модуль (подкласс), расширяющий программный комплекс и вводящий в него модели вышеперечисленных алгоритмов [9, 10].

Был проведён численный эксперимент с помощью этого комплекса программ, реализующих решение системы стохастических дифференциальных уравнений стохастическим методом Рунге-Кутты слабого порядка 2.0 и сильного порядка 1.0. Полученные результаты были представлены в графическом виде (см. рис. 1, 3, 5, 7, 9).

Также было проведено имитационное моделирование с использованием симулятора NS2. Полученные результаты были представлены в виде графиков зависимости средней и мгновенной длины очереди от времени (см. рис. 2, 4, 6, 8, 10).

- Анализ графиков позволяет утверждать, что численная и имитационная модели дают сходный качественный результат. Средняя длина очереди в начале работы алгоритма резко повышается, а затем убывает и стабилизируется, осциллируя около фиксированного значения.
- Протоколы ARED, AURED и SARED отличаются от RED существенно меньшим колебанием мгновенной длины очереди, что соответствует данным из литературы, посвящённой этим протоколам. Протокол GRED отличается от

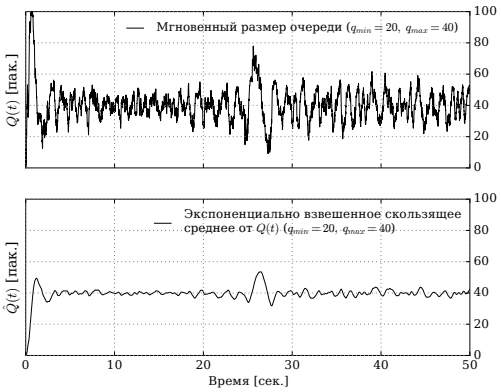

Рис. 1. Результаты численного моделирования алгоритма RED

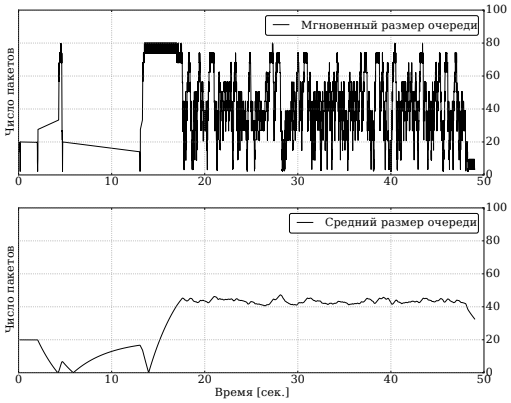

Рис. 2. Результаты имитационного моделирования алгоритма RED

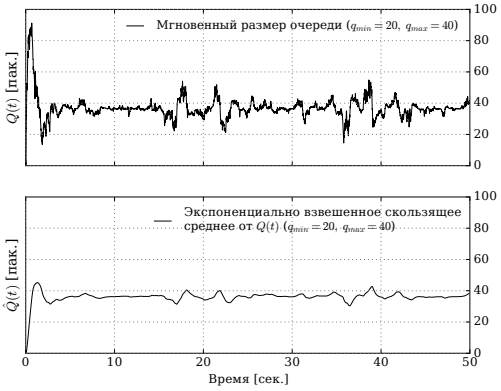

Рис. 3. Результаты численного моделирования алгоритма ARED

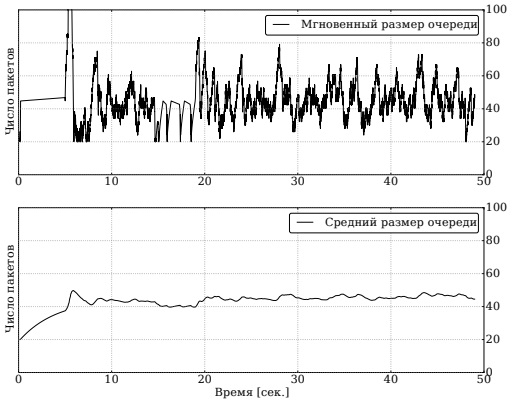

Рис. 4. Результаты имитационного моделирования алгоритма ARED

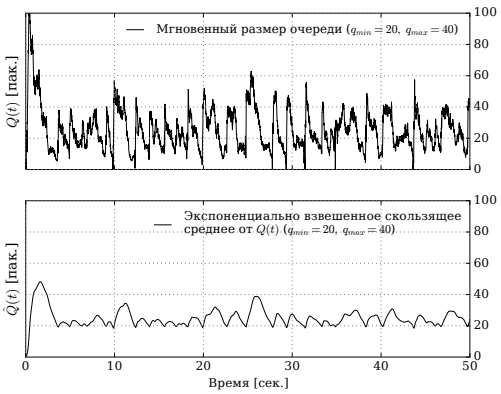

Рис. 5. Результаты численного моделирования алгоритма GRED

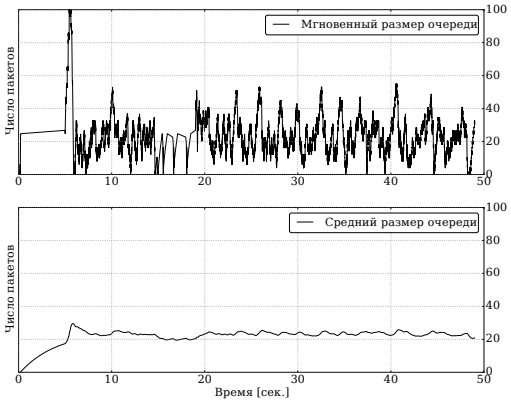

Рис. 6. Результаты имитационного моделирования алгоритма GRED

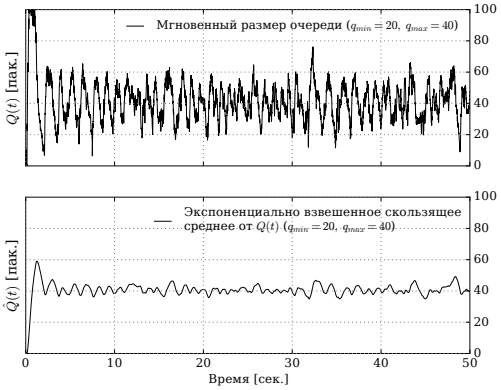

Рис. 7. Результаты численного моделирования алгоритма AURED

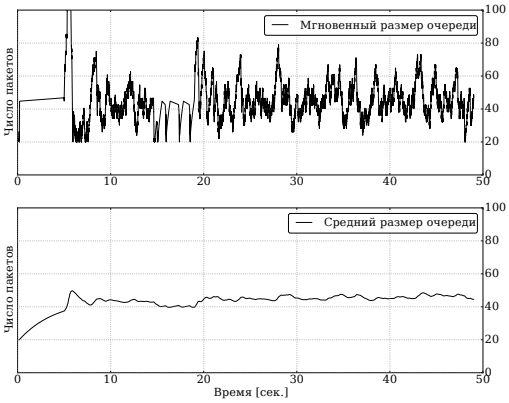

Рис. 8. Результаты имитационного моделирования алгоритма AURED

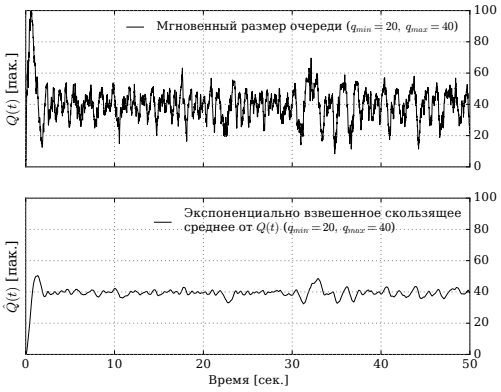

Рис. 9. Результаты численного моделирования алгоритма SARED

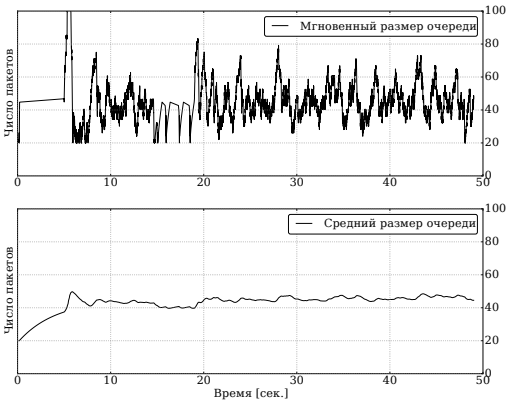

Рис. 10. Результаты имитационного моделирования алгоритма SARED

RED меньшей амплитудой колебания текущей длины очереди в начале своей работы. Меньшие колебания мгновенной длины очереди положительно сказываются на производительности маршрутизатора, так как уменьшается вероятность спонтанного переполнения буфера и переключения маршрутизатора в режим Drop Tail.

Литература

1. *Floyd S., Jacobson V.* Random Early Detection Gateways for Congestion Avoidance // IEEE/ACM Transactions on Networking. — 1993. — No 1. — Pp. 397–413. — `http://www.icir.org/floyd/papers/red/red.html`.
2. *Ho H.-J., Lin W.-M.* AURED — Autonomous Random Early Detection for TCP Congestion Control // Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on / IEEE. — Sliema: IEEE, 2008. — Pp. 79–84.
3. Performance Investigations of Some Active Queue Management Techniques Using Simulation / A.-J. Hussein, T. Fadi, D. A. M. et al. // International Journal of New Computer Architectures their Appl. — 2012. — Vol. 2, No 1. — Pp. 286–301.
4. *Javam H., Analoui M.* SARED: Stabilized ARED // Communication Technology, 2006. ICCT '06. International Conference on. — 2006. — Pp. 1–4.

5. *Xu Y.-D., Wang Z.-Y., Wang H.* ARED: A Novel Adaptive Congestion Controller / IEEE. — Guangzhou: IEEE, 2005. — Pp. 708–714.
6. Digital Enterprise and Information Systems: International Conference / H. Abdeljaber, J. Ababneh, F. Thabtah et al. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. — Pp. 592–603.
7. *Королькова А. В., Кулябов Д. С.* Математическая модель динамики поведения параметров систем типа RED // Вестник РУДН, серия «Математика. Информатика. Физика». — 2010. — № 2. — С. 54–64.
8. Модель управления очередями на маршрутизаторах / Т. Р. Велиева, А. В. Королькова, Д. С. Кулябов, Б. А. Сантуш // Вестник РУДН, серия «Математика. Информатика. Физика». — 2014. — № 2. — С. 81–92.
9. *Тьерри Ги Д. Д., Королькова А. В., Геворкян М. Н.* Расчёт параметров функционирования модуля активного управления трафиком Adaptive Virtual Queue Random Early Detection. — 2015.
10. *Калилу Д., Королькова А. В., Геворкян М. Н.* Расчёт параметров функционирования модуля активного управления трафиком Stabilized Adaptiv Random Early Detection. — 2015.
11. Advances in Network Simulation / L. Breslau, D. Estrin, K. Fall et al. // IEEE Computer. — 2000. — Vol. 33, No 5. — Pp. 59–67. — `http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html`.

UDC 004.021, 519.2, 519.6

Numerical Modeling and Simulation of RED Active Queue Management Algorithms

D. S. Kulyabov, M. N. Gevorkyan, J. R. Machuca, K. Diarrassuba, D. T. G. Dali

Department of Applied Probability and Informatics
Peoples' Friendship University of Russia
6, Miklukho-Maklaya str., Moscow, Russian Federation, 117198

The aim of this paper is to expand the stochastic model of RED (Random Early Detection) for the case of AURED, SARED and GRED queue service disciplines, as well as the verification of numerical simulation results with NS2 software. A stochastic model is based on the a system of three Ito stochastic equations. The numerical solution is carried out using stochastic Runge–Kutta methods with weak convergence of the second order. Software package for the numerical simulation written by authors in Python version 3 using libraries `NumPy` and `SciPy`. The article describes in detail the components of software package. For simulation we use open source software package for modeling Computer Networks `NS2`. In this article, the authors briefly describe those moments that touch queuing disciplines, not dwelling on the general description of NS2. The simulation results are presented as a plot of the average queue length and the current queue length as functions of time. Based on the analysis of the graphs, it is shown that the simulation and Numerical simulation gave qualitatively related to each other results.

Key words and phrases: RED, routing, stochastic differential equations, stochastic numerical methods, ns2.

References

1. S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking (1) (1993) 397–413.
2. H.-J. Ho, W.-M. Lin, AURED — Autonomous Random Early Detection for TCP Congestion Control, in: Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on, IEEE, IEEE, Sliema, 2008, pp. 79–84.
3. A.-J. Hussein, T. Fadi, D. A. M., A. Jafar, B. Mahmoud, Performance Investigations of Some Active Queue Management Techniques Using Simulation,

International Journal of New Computer Architectures their Appl 2 (1) (2012) 286–301.
4. H. Javam, M. Analoui, SARED: Stabilized ARED, in: Communication Technology, 2006. ICCT '06. International Conference on, 2006, pp. 1–4. doi:10.1109/ICCT.2006.341669.
5. Y.-D. Xu, Z.-Y. Wang, H. Wang, Ared: A novel adaptive congestion controller, IEEE, IEEE, Guangzhou, 2005, pp. 708–714.
6. H. Abdel-jaber, J. Ababneh, F. Thabtah, A. M. Daoud, M. Baklizi, Digital Enterprise and Information Systems: International Conference, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 592–603.
7. A. V. Korolkova, D. S. Kulyabov, Mathematical Model of the Dynamic Behavior of RED-Like System Parameters, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics-(2) (2010) 54–64.
8. T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, B. A. dos Santos, Model Queue Management on Routers, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics-(2) (2014) 81–92.
9. D. D. G. Thierry, A. V. Korolkova, M. N. Gevorkyan, The Calculation of Operating Parameters of the Module Active Traffic Management Adaptive Virtual Queue Random Early Detection, in russian (2015).
10. K. Diarrassuba, A. V. Korolkova, M. N. Gevorkyan, The Calculation of Operating Parameters of the Module Active Traffic Management Stabilized Adaptiv Random Early Detection, in russian (2015).
11. L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu, Advances in Network Simulation, IEEE Computer 33 (5) (2000) 59–67.
URL `http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html`