**Федеральное государственное автономное образовательное учреждение высшего образования**
**«Российский университет дружбы народов имени Патриса Лумумбы»**

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Направление: 09.03.03 Прикладная информатика

# ОТЧЕТ

о прохождении учебной практики
Научно-исследовательская работа
(получение первичных навыков научно-исследовательской работы)

Место прохождения практики: отдел технической поддержки пользователей (департамент технологических и информационных ресурсов) РУДН и научные центры института прикладной математики и телекоммуникаций
Сроки прохождения: с «17» апреля 2023 г. по «17» июня 2023 г.

ФИО: Саргсян Арам Грачьяевич
Курс, группа 3, НПИбд-02-20
Студенческий билет № 1032201740

Руководители практики:
от РУДН к.ф.-м.н. Е.Г. Медведева

Научный руководитель:
к.ф.-м.н., доц. А.В. Королькова

Руководитель от организации:
д.т.н., проф. К.Е. Самуйлов

Оценка _____

г. Москва
2023 г.

# Оглавление

# Введение

Согласно программе учебной практики по направлению 09.03.03 «Прикладная информатика» целями практики являются:

- формирование навыков использования современных научных методов для решения научных и практических задач;
- формирование универсальных, общепрофессиональных и профессиональных компетенций в соответствии с ОС ВО РУДН;
- формирование навыков проведения исследовательской работы;
- формирование навыков работы с источниками данных;
- знакомство с принципами функционирования и изучение методов разработки и анализа моделей функционирования сложных систем, их фрагментов и отдельных элементов;
- применение методов для анализа и расчёта показателей функционирования сложных систем, их фрагментов и отдельных элементов.

Также опредены задачи практики:

- изучение специфики функционирования и соответствующих методов анализа сложных систем;
- формирование навыков решения конкретных научно-практических задач самостоятельно или в научном коллективе;
- формирование навыков проведения исследовательской работы и получении научных и прикладных результатов;
- изучение принципов и методов построения моделей сложных систем (в том числе технических систем, сетей и систем телекоммуникаций);
- изучение принципов и методов анализа поведения параметров моделей сложных систем (в том числе программных и технических систем, сетей и систем телекоммуникаций, и т.п.);

- приобретение практических навыков в области изучения научной литературы и (или) научно-исследовательских проектов в соответсвии с будущим профилем профессиональной области.

Для достижении вышеупомянутых целей и задач в рамках учебной практики по теме «Моделирования алгоритма управления очередями RED в средстве моделирования NS-2» мною было выполнено следующее:

- рассмотрены основные методы имитационного, аналитического и натурного моделирования сетей;
- исследована специфика моделирования различных сетей с помощью программы NS-2;
- проведен сравнительный анализ результатов имитационного моделирования сети (построены и проанализированы графики размера TCP-окна, длины очереди и средней взвешенной длины очереди) при различных модификациях алгоритма RED, разных пороговых значений и типов TCP.

# Глава 1

# Методы и материалы

В этом разделе представим краткий обзор средств моделирования сетей передачи данных.

## 1.1 NS-2

NS-2 (Network simaulator 2) — это программное средство моделирования сетей, использующееся для исследования и анализа поведения компьютерных сетей. Запуск имитационной модели в данной среде позволяет анализировать различные протоколы и алгоритмы сетевой связи.

NS-2 разработан на языке программирования C++ и TCL, что обеспечивает гибкость и расширяемость средства моделирования. NS-2 содержит библиотеку классов, которые представляют различные элементы сети, такие как узлы, маршрутизаторы, каналы связи и протоколы передачи данных. Для создания модели сети определяются характеристики и параметры каждого элемента сети: пропускная способность канала, задержки, вероятность потери пакетов и другие. После завершения симуляции NS-2 предоставляет мощные инструменты анализа результатов, включая возможность визуализации данных посредством программы NAM (Network animator), статистический анализ и сравнение результатов экспериментов, что позволяет изучать и оценивать производительность различных протоколов и алгоритмов в различных сценариях сети [1, 2].

## 1.2 Mininet

Mininet — это симулятор сетевых топологий на основе виртуаилизации, который позволяет моделировать и изучать поведение сетей в контролируемой среде, основанный на использовании виртуальных машин и пространств имен Linux для создания изолированных сетевых узлов. Моделирование сетевых топологий с помощью Mininet позволяет

исследовать различные сетевые протоколы, маршрутизацию, управление трафиком и т.д. Возможности моделирования с помощью Mininet включают создание виртуальных сетевых узлов, конфигурирование топологий (связь между узлами, настраивать IP-адреса, маршрутизацию), имитировать различные условия сети, такие как задержки, потери пакетов и пропускную способность, интеграция с контроллерами для исследования новых протоколов и алгоритмов.

## 1.3 Cisco Packet Tracer

Packet Tracer — это программное средство, предоставляемое компанией Cisco Systems, позволяющей смоделировать, конфигурировать и отлаживать сетевые сценарии, широко используемое в области сетевых технологий. Данное программное обеспечение предоставляет виртуальную среду, которое позволяет создавать сетевые топологии и настраивать устройства Cisco: маршрутизаторы, коммутаторы и т.д. Графический интерфейс позволяет соединять устройства, устанавливать параметры соединений и задавать настройки протоколов. Cisco Packet Tracer позволяет имитировать передачу данных в сети. Пользователи могут выполнять различные тесты связи, проводить диагностику и мониторинг сетевых устройств, а также создавать и анализировать журналы событий.

## 1.4 GNS-3

GNS-3 — это программное средство моделирования сетей, позволяющий создавать виртуальные сети, состоящие из реальных или виртуальных устройств, и анализировать их поведение. GNS-3 разработан на языке программирования Python и основан на эмуляторе динамических узлов Dynamips, который позволяет запускать реальные образы операционных систем. В отличие от Packet Tracer, GNS-3 позволяет смоделировать не только устройства Cisco, но и другие устройства, например, Juniper, Palo, Alto и другие, что позволяет смоделировать различные типы сетей, включая центры обработки данных и облачные инфраструктуры. Одной из главных особенностей GNS-3 является интеграция с виртуальными машинами, что расширяет возможности моделирования. Появляется возможность создавать сетевые сценарии, в которых виртуальные машины выполняют реальные функции, такие как серверы, клиенты, точки доступа Wi-Fi и т.д. Это позволяет проводить натурное моделирование и получить более реалистичные результаты в рамках виртуальной среды.

# Глава 2

# RED

## 2.1 Classic RED

Random Early Detection (RED) — это механизм предотвращения перегрузки на шлюзе. Он основан на общих принципах, полезен для управления средним размером очереди в сети, где не доверяют взаимодействию между протоколами передачи данных. В отличие от Droptail, который работает таким образом, что когда очередь заполняется, новые пакеты, поступающие в очередь, начинают теряться, алгоритм RED учитывает потоки трафика в сети и стремится предоставить равную пропускную способность для каждого соединения, что позволяет избежать перегрузки сети и улучшить качество обслуживания. В оригинальном RED маршрутизатор вычисляет усредненный по времени средний размер очереди с использованием фильтра нижних частот (экспоненциально взвешенное скользящее среднее) или сглаживания по длине выборки очередей, средний размер очереди сравнивается с двумя пороговыми значениями: минимальным порогом и максимальным. Когда средний размер очереди меньше минимального порога, пакеты не отбрасываются, когда средний размер очереди превышает максимальный порог, отбрасывается все поступающие пакеты. Если размер средней очереди находится между минимальным и максимальным порогом, пакеты отбрасываются с вероятностью $p$, которая линейно увеличивается до тех пор, пока средняя очередь не достигнет максимального порога. Подробно алгоритм описан в [3, 4].

Вероятность $p_b$ маркировки на отбрасывание пакетов представляет собой функцию, линейно зависящую от $\hat{q}$ (средневзвешенное скользящее среднее), минимального $q_{\min}$ и максимального $q_{\max}$ пороговых значений и параметра $p_{\max}$, определяющего часть отбрасываемых пакетов при достижении средним размером очереди значения $q_{\max}$ и

вычисляется следующим образом(2.1):

$$p_b = \begin{cases} 0, & 0 < \hat{q} \leqslant q_{\min}, \\ \frac{\hat{q}-q_{\min}}{q_{\max}-q_{\min}} p_{\max}, & q_{\min} < \hat{q} \leqslant q_{\max}, \\ 1, & \hat{q} > q_{\max}. \end{cases} \tag{2.1}$$

График функции вероятности потери пакета в зависимости от среднего размера очереди представлен на рис. 2.1.

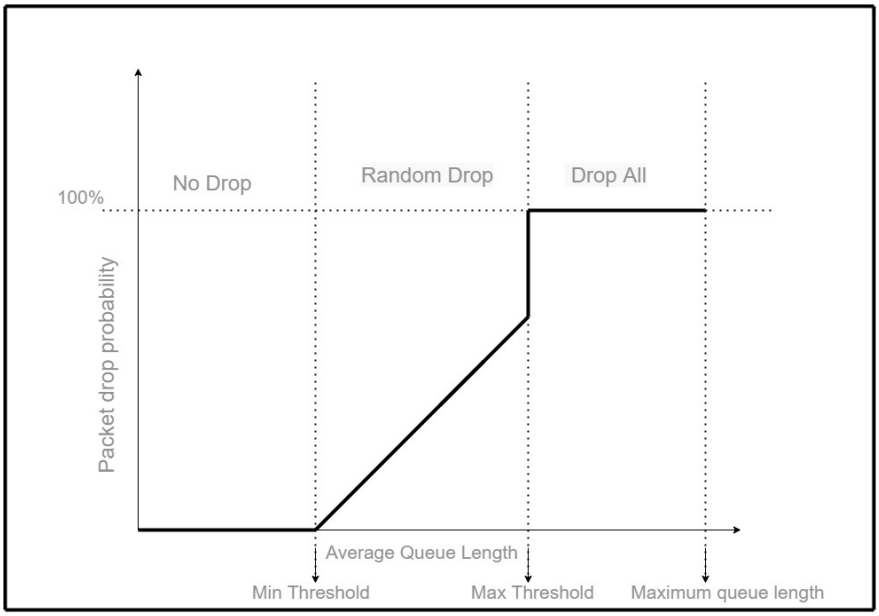


Рис. 2.1: Классический RED

В NS-2 файлы, связанные с RED, прописаны в каталоге `ns-2.35/queue`, там представлены также другие реализации очередей (среди них DropTail, BLUE и т.д.). Следует уделить внимание двум файлам: `red.cc` (исходники), и `red.h` (заголовочный файл). Вероятность отбрасывания пакета прописана в функции double

`REDQueue::calculate_p_ne файла red.cc.`

```
1 double REDQueue::calculate_p_new(double v_ave, double th_max, int gentle,
  ↪  double v_a, double v_b, double v_c, double v_d, double max_p)
2 {
3         double p;
4         if (gentle && v_ave >= th_max) { //для модификации GRED
5                 p = v_c * v_ave + v_d;
6         } else if (!gentle && v_ave >= th_max) { // Превысили пороговое
  ↪  значение в классическом RED
7                 p = 2.0;
```

```
8           } else { // p в промежутке от 0 до max_p, тогда средний размер
    ↪  очереди в промежутке th_min до th_max
9                   p = v_a * v_ave + v_b;
10                  // p = (v_ave - th_min) / (th_max - th_min)
11                  p *= max_p;
12          }
13          if (p > 2.0) и // пакеты отбрасываются
14                  p = 2.0;
15          return p;
16  }
```

## 2.2 GRED

GRED (Gentle Random Early Detection, мягкое/аккуратное произвольное раннее обнаружение) — алгоритм активного управления очередью, является расширением RED. Стандартный алгоритм увеличивает вероятность отбрасывания с 0.05 до 0.5, когда средняя длина очереди увеличивается от минимального до максимального порогового значения, но при превышении максимального порога вероятность возрастает напрямую с 0.5 до 2. Этот внезапный скачок нормализуется модификацией Gentle RED, который расширяет RED тем, что добавляет дополнительное максимальное пороговое значние, которое равно $2q_{\max}$, тем самым «сглаживая» кривую [5]. Для реализации модификации в NS-2 при описании очереди нужно задать переменную `set gentle_ true`.

Вероятность сброса определяется следующим образом (2.2):

$$p_b = \begin{cases} 0, & 0 < \hat{q} \leqslant q_{\min}, \\ \frac{\hat{q}-q_{\min}}{q_{\max}-q_{\min}} p_{\max}, & q_{\min} \leqslant \hat{q} < q_{\max}, \\ \frac{\hat{q}-q_{\min}}{q_{\max}}(1-p_{\max}) - p_{\max}, & q_{\max} \leqslant \hat{q} < 2q_{\max}, \\ 1, & \hat{q} \geqslant q_{\max}. \end{cases} \tag{2.2}$$

График функции вероятности потери пакета в зависимости от среднего размера очереди выглядит следующим образом (рис. 2.2):

## 2.3 WRED

WRED (Weighted random early detection — взвешенное произвольное раннее обнаружение) — это алгоритм активного управления очередью, является расширением RED [6].

Взвешенный алгоритм произвольного раннего обнаружения предоставляет различные уровни обслуживания пакетов в зависимости от вероятности их отбрасывания и

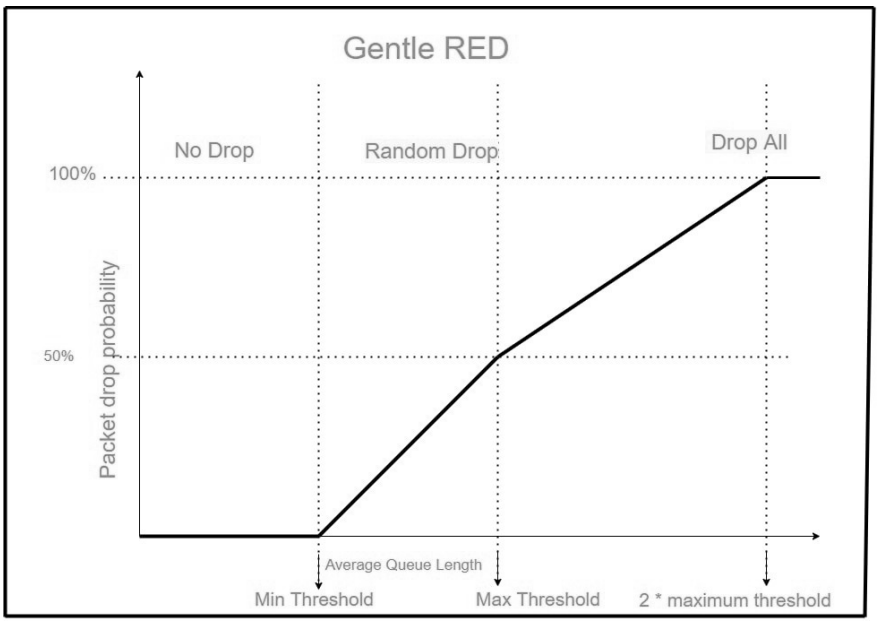


Рис. 2.2: Gentle RED

обеспечивает избирательную установку параметров механизма RED на основании типа трафика (рис. 2.3).

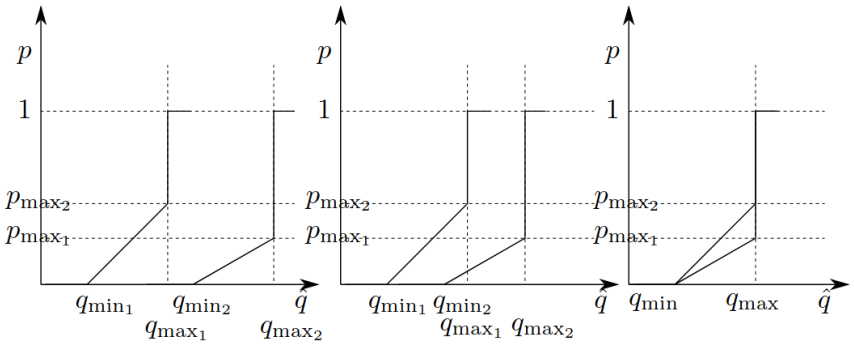


Рис. 2.3: Weighted RED

Алгоритм WRED работает с единой очередью пакетов, для которой, как и в RED, по формуле рассчитывается экспоненциально взвешенное скользящее среднее. Для каждого типа трафика задаются собственные параметры (пороговые значения, максимальный уровень сброса) и вычисляется вероятность сброса.

Например, очереди могут иметь более низкие пороговые значения для более низких приоритетов пакета. Это приведет к отбрасыванию пакетов с низким приоритетом, а следовательно, к защите пакетов с более высоким приоритетом в той же очереди.

## 2.4 NLRED

Nonlinear RED — это модификация классического алгоритма RED, в отличие от которого использует нелинейную функцию для определения вероятности отбрасывания

пакетов, что позволяет более гибко регулировать процесс управления перегрузками, учитывая различные характеристики трафика и динамику сети [7, 8]. Вероятность $p_b$ маркировки на отбрасывание пакетов вычисляется следующим способом (2.3):

$$p_b = \begin{cases} 0, & 0 < \hat{q} \leqslant q_{\min}, \\ 1, & \hat{q} > q_{\max}, \\ \frac{\hat{q} - q_{\min}}{q_{\max} - q_{\min}} 2.5 p_{\max}^2, & q_{\min} < \hat{q} \leqslant q_{\max}. \end{cases} \tag{2.3}$$

График функции вероятности потери пакета в зависимости от среднего размера очереди представлен на рис. 2.4.

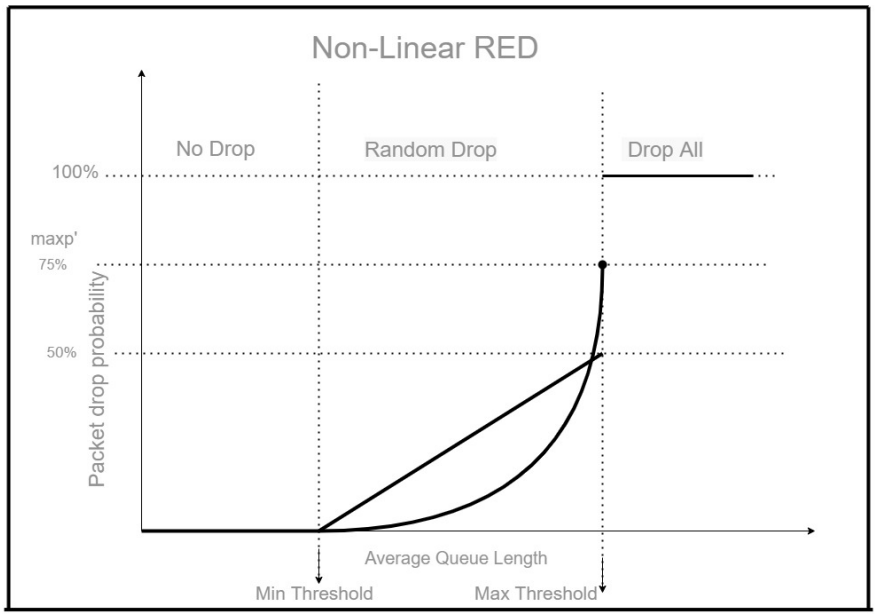


Рис. 2.4: NonLinear RED

По умолчанию NLRED не реализован в NS-2. Для её добавления я проделал следующие шаги:

1. Установил к себе на машину патч `NLRED.patch` от Mohit P. Tahiliani для версии 2.34, совместимой также для версии 2.35.

2. Отредактировал файл, заменив везде номер версии на 2.35 и переместил в каталог `ns-allinone`.

3. В терминале запустил команду `patch -p1 < NLRED.patch`, а затем запустил `./install`.

4. В настройке очереди сети указал значение переменной `nonlinear 2`.

## 2.5 ARED и RARED

В алгоритме Adaptive RED (ARED) функция сброса модифицируется посредством изменения по принципу AIMD, заключающейся в том, что увеличение некоторой величины производится путём сложения с некоторым параметром, у уменьшение — путём умножения на параметр [9]. Для её реализации в NS-2 необходимо указать в настройке очереди `set adaptive 2`.

Алгоритм ARED функционирует следующим образом (2.4), (2.5). Для каждого интервала `interval` (параметр) в секундах, если $\hat{q}$ больше целевой (желаемой) $\hat{q}_t$ и $p_{\max} \leqslant 0,5$, то $p_{\max}$ увеличивается на некоторую величину $\alpha$; в противном случае, если $\hat{q}$ меньше целевой $\hat{q}_t$ и $p_{\max} \geqslant 0,01$, то $p_{\max}$ уменьшается в $\beta$ раз:

$$p_{\max} = \begin{cases} p_{\max} + \alpha, \ \hat{q} > \hat{q}_t, \ p_{\max} \leqslant 0,5, \\ \beta * p_{\max}, \ \hat{q} < \hat{q}_t, \ p_{\max} \geqslant 0,01, \end{cases} \tag{2.4}$$

$$q_{\min} + 0,4(q_{\max} - q_{\min}) < \hat{q}_t < q_{\min} + 0,6\,(q_{\max} - q_{\min})\,. \tag{2.5}$$

Основные особенности:

- автоматическая установка минимального порога $q_{\min}$. Он устанавливается в зависимости от пропускной способности канала $C$ и задержки целевой очереди;
- автоматическая установка максимального порога $q_{\max}$. Он устанавливается в зависимости от значения месяца;
- автоматическая настройка $w_q$. Он устанавливается в зависимости от пропускной способности канала $C$;
- адаптивная настройка $p_{\max}$. Он адаптирован в соответствии с текущей средней длиной очереди.

Алгоритм Refined ARED (2.6), (2.7) является модификацией ARED и предлагает более активно изменять вероятность сброса $p_{\max}$, чтобы иметь возможность быстрой адаптации к изменяющейся экспоненциально взвешенной скользящей средней длине очереди $\hat{q}$:

$$p_b = \begin{cases} p_{\max} + \alpha, \quad \hat{q} > \hat{q}_t, \quad p_{\max} \leqslant 0,5, \\ \beta p_{\max}, \quad \hat{q} \leqslant \hat{q}_t, \quad p_{\max} \geqslant 0,5, \end{cases} \tag{2.6}$$

$$\begin{cases} q_{\min} + 0,48\,(q_{\max} - q_{\min}) < \hat{q}_t < q_{\min} + 0,52\,(q_{\max} - q_{\min}), \\ \alpha = \left(0,25\dfrac{\hat{q} - \hat{q}_t}{\hat{q}_t}\right) p_{\max}, \\ \beta = 1 - \left(0,17\dfrac{\hat{q} - \hat{q}_t}{\hat{q}_t - q_{\min}}\right). \end{cases} \tag{2.7}$$

По умолчанию Refined ARED не реализован в NS-2. Для его добавления я проделал следующие шаги:

1. Установил к себе на машину патч `Re-ARED.patch` от Mohit P. Tahiliani для версии 2.34, совместимой также для версии 2.35.

2. Отредактировал файл, заменим везде номер версии на 2.35 и переместил в каталог `ns-allinone`.

3. В терминале запустил команду patch `-p1 < Re_ARED.patch`, а затем `./install`.

4. В настройке очереди сети указал значение переменной `adaptive 1` и для RARED дополнительно `refined adaptive 2`.

# Глава 3

# Результаты

Мною был написана программа, реализующая имитационную модель сети со следующей топологией:

- $N = 20$ TCP-источников, $N$ TCP-приёмников, двух маршрутизаторов $R1$ и $R2$ между источниками и приёмниками ($N$ — не менее 20);
- между TCP-источниками и первым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail;
- между TCP-приёмниками и вторым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail;
- между маршрутизаторами установлено симплексное соединение ($R1$–$R2$) с пропускной способностью 20 Мбит/с и задержкой 15 мс очередью типа RED, размером буфера 300 пакетов; в обратную сторону — симплексное соединение ($R2$–$R1$) с пропускной способностью 15 Мбит/с и задержкой 20 мс очередью типа DropTail;
- данные передаются по протоколу FTP поверх TCPReno;
- параметры алгоритма RED: $q_{\min} = 75$, $q_{\max} = 150$, $q_w = 0,002$, $p_{\max} = 0.1$;
- максимальный размер TCP-окна 32; размер передаваемого пакета 500 байт; время моделирования — не менее 20 единиц модельного времени.

Полная реализация программы приведена в разделе **Приложения**, для вывода графиков была использована программа GNUPLOT.

Смоделировав сеть с указанными параметрами и запустив gnuplot-скрипт, я получил следующий график (рис. 3.1).

Как мы видим из первого графика, в момент времени $t = 1$ с достигается максимальные длины очереди 140000 пакетов и средней длины очереди 120000 пакетов, а

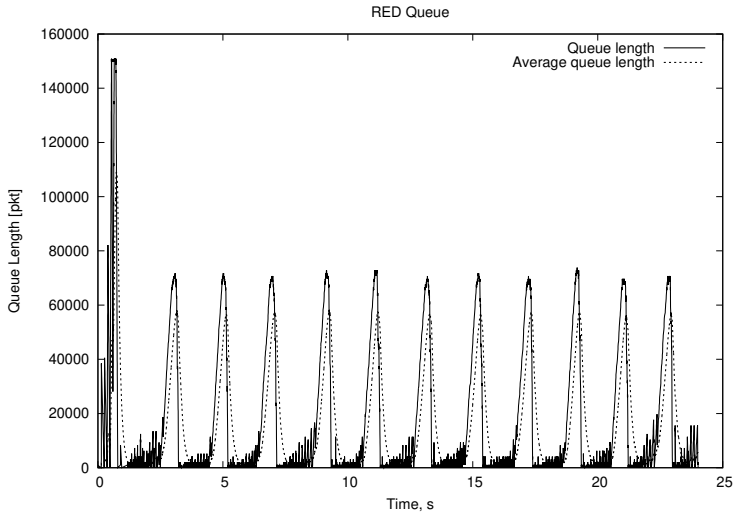


Рис. 3.1: График длины очереди и средней длины очереди на линке (R0–R1) ($q_{\min} = 75$, $q_{\max} = 150$, $q_w = 0.002$, $p_{\max} = 0.1$, TCP типа TCP/Reno)

при дальнейшем моделировании длина очереди варьируются от 0 до 70000 пакетов, а средняя очередь от 0 до 60000, наступает стационарное состояние.

Для выявления влияния пороговых значений на длину очереди смоделировал сеть и вывел на одном графике траектории средней длины очереди при разных пороговых значениях и при одинаковых остальных параметрах (рис. 3.2).

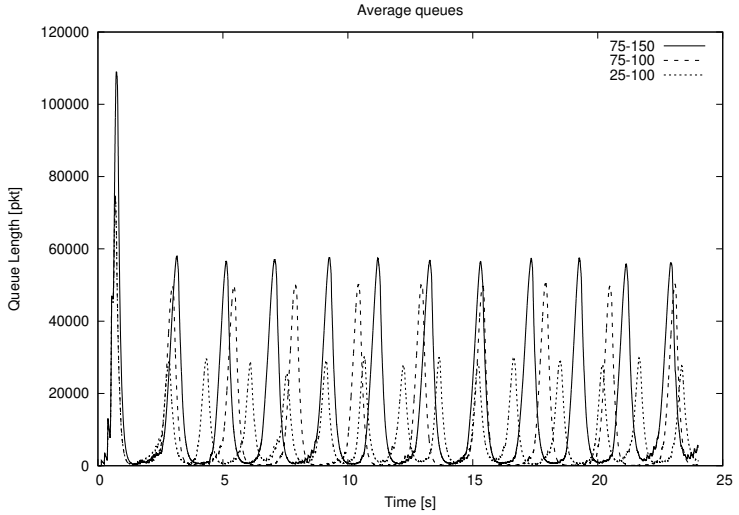


Рис. 3.2: График средней длины очереди для разных пороговых значений

Как видно из графика, увеличение диапазона между $q_{\min}$ и $q_{\max}$ способствует увеличению длины очереди на линке.

Для сравнений различных модификаций смоделировали сеть, задав в качестве очереди RED, GRED, NLRED, RARED (рис. 3.3, 3.4, 3.5, 3.6).

Различие между RARED с остальными модификациями легко объяснить его автоматическим выбором минимального и максимального пороговых величин. Средняя

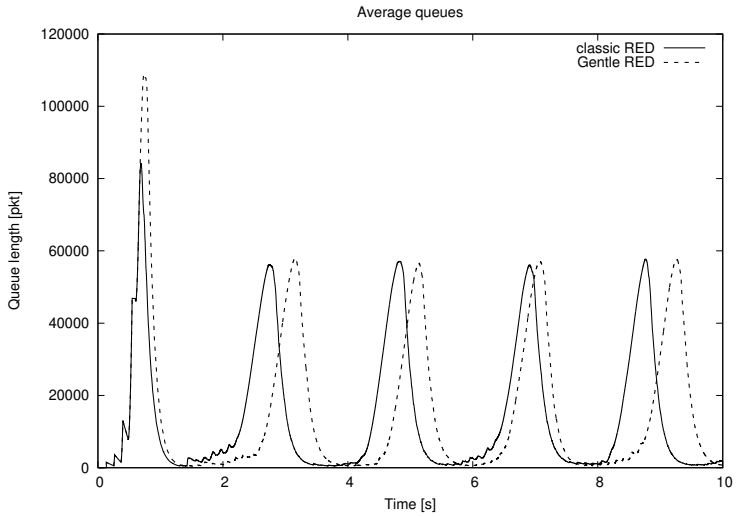


Рис. 3.3: Средняя длина очереди RED и GRED при одинаковых условиях

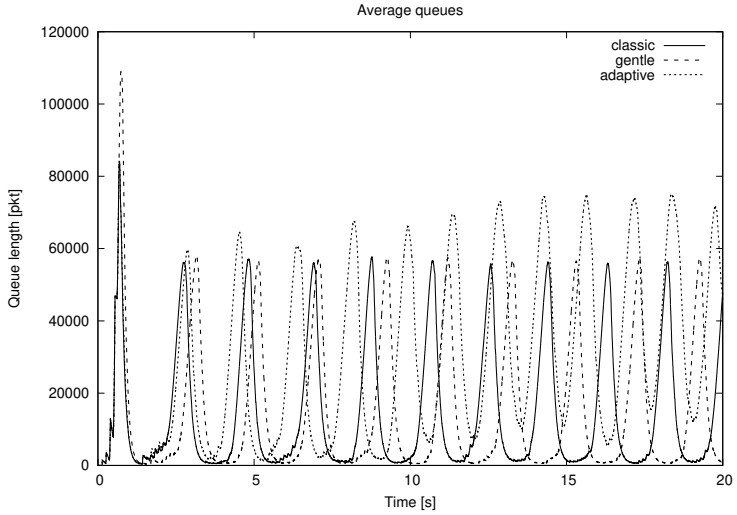


Рис. 3.4: Средняя длина очереди RED, GRED, ARED при одинаковых условиях

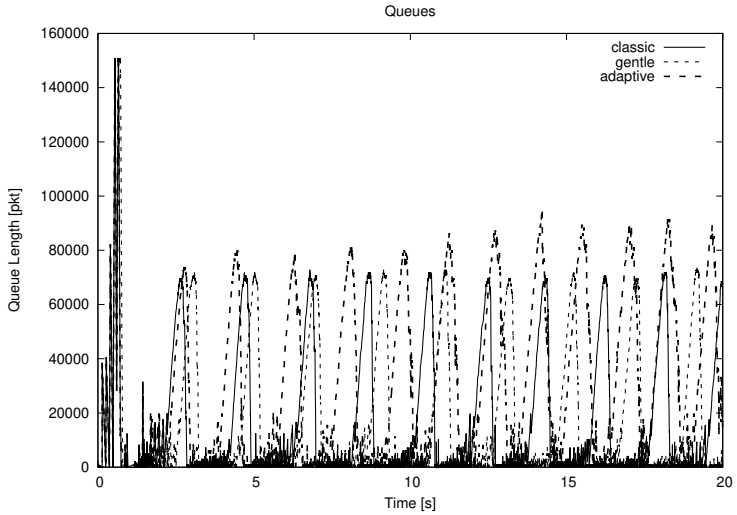


Рис. 3.5: Длина очереди RED, GRED, ARED при одинаковых условиях

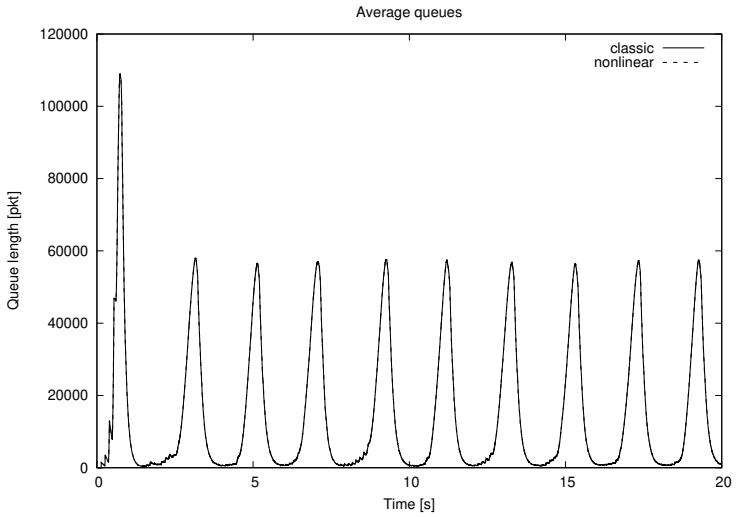


Рис. 3.6: Средняя длина очереди RED, NLRED при одинаковых условиях

очередь при Gentle RED до стационарного состояния гораздо выше, что обьяснется его более мягким алгоритмом. Моделирование NLRED не показало никаких различий с классическим RED, что указывет на неработаспособность патча.

Смоделировали сеть при разных TCP(Reno, Vegas и Newreno) (рис. 3.7, 3.8, 3.9).

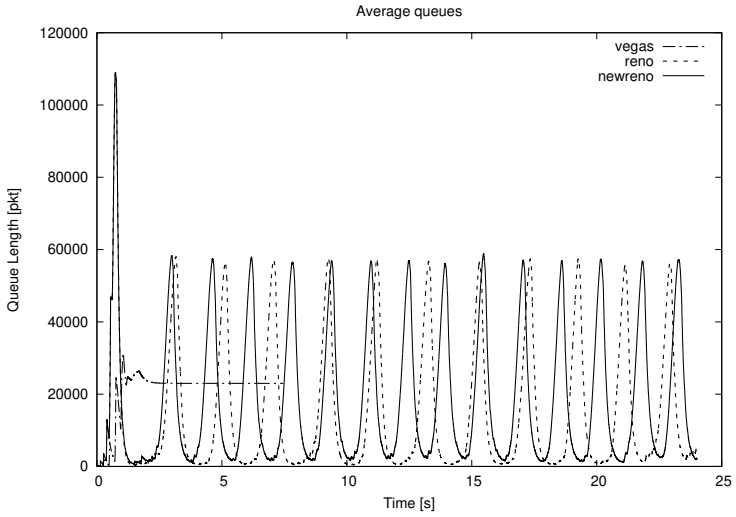


Рис. 3.7: График длины очередей на линке (R0-R1) при разных TCP

Такое заметное отличие TCP/Vegas от TCP/Reno и TCP/Newreno объясняется тем, что он использует альтернативный подход к управлению перегрузкой. Вместо использования потери пакетов в качестве индикатора перегрузки сети он анализирует задержку пакетов в сети и использует данную информацию для определения, происходит ли перегрузка в сети.

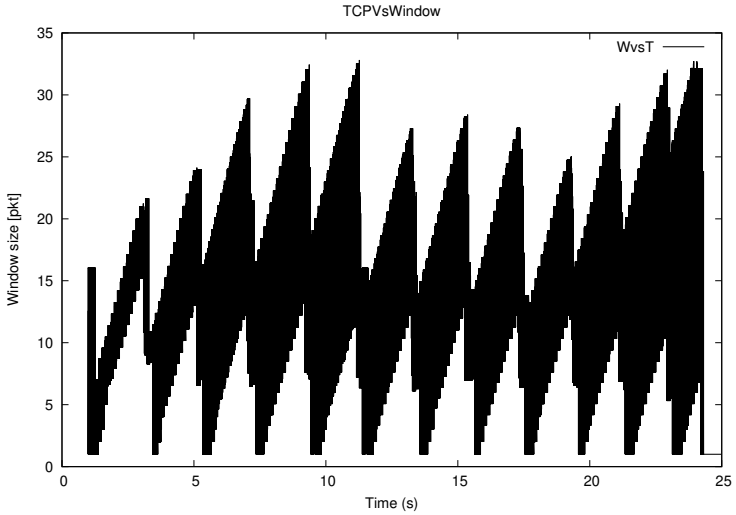


Рис. 3.8: График размера TCP-окна для всех источников (TCP типа TCP/Reno)

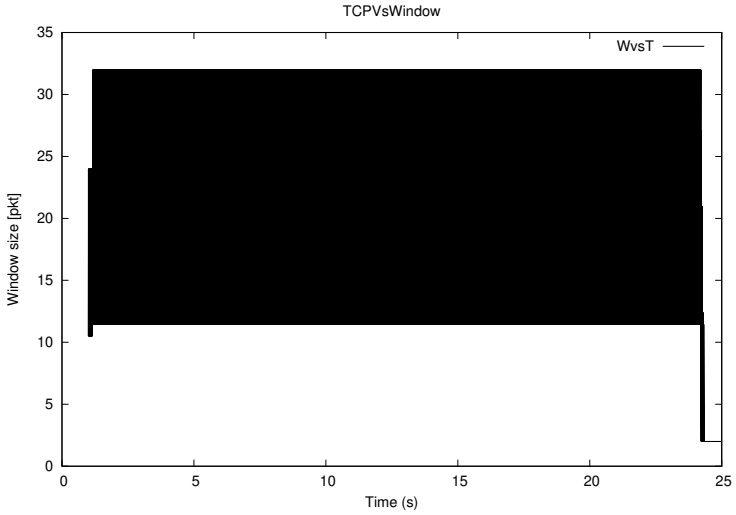


Рис. 3.9: График размера TCP-окна для всех источников (TCP типа TCP/Vegas)

# Заключение

За период практики в отделе технической поддержки пользователей (департамент технологических и информационных ресурсов) РУДН и научных центрах института прикладной математики и телекоммуникаций. были достигнуты все цели и решены все задачи, определенные в программе научной практики направления подготовки 09.03.03 «Прикладная информатика» программы «Прикладная информатика» (см. введение отчёта по практике). В процессе прохождения практики я работал с научной терминологией области исследований; научился собирать и обрабатывать данные, необходимые для формирования соответствующих выводов исследований; осуществлять целенаправленный поиск информации на русском и английском языках о новейших научных достижениях в Интернете и из других источников; строить и анализировать имитационные модели обьекта исследований.

В результате прохождения данной практики я приобрел следующие практические навыки, умения, универсальные и профессиональные компетенции:

- способность управлять проектом на всех этапах его жизненного цикла (постановка задачи, планирование, реализация);
- способность составлять естесвенно-научные отчеты с IMRAD структурой (введение, методы и материалы, результаты и дискуссия);
- способность разрабатывать имитационные модели и проводить их анализ при решении задач в профессиональной области (составлена имитационная модель сети с алгоритмом управления очередью на маршрутизаторе типа RED);
- способность проведения работ по обработке и анализу научно-технической информации и результатов исследований (изучение необходимой литературы по теме исследования на русском и английском языках, подготовка литературного обзора по теме исследований).

Таким образом, в рамках практики я рассмотрел моделирование модуля RED с помощью программного средства NS-2 версии 2.35. Также представлена программная реализация имитационной модели сети модулем RED и проведен сравнительный анализ

результатов при моделировании сети с разными входными параметрами, модификаций RED и типов TCP.

# Список литературы


[1] loyd, Sally and Fall, Kevin Ns Simulator Tests for Random Early Detection (RED) Queue Management — 1997. — April.

[2] The VINT and Project The ns Manual (formerly ns Notes and Documentation) — 2011. — November.

[3] Floyd S., Jacobson V. Random Early Detection Gateways for Congestion // IEEE/ACM Transactions on Networking. — IEEE. — 1993. — November. — Т. 1, № 4. — С. 397–413. — doi:10.1109/90.251892.

[4] Abouzeid A., Roy S. Modeling random early detection in a differentiated services network q // Computer Networks ed. by Roberts J. — IEEE. — 2002. — November — Т. 40, № 4. — С. 537–556. — doi:10.1016/S1389-1286(02)00295-5.

[5] Hamadneh N. и др. Revisiting the Gentle Parameter of the Random Early Detection (RED) for TCP Congestion Control // Journal of Communications. Engineering — Engineering and Technology Publishing. — 2019. — Т. 40, № 4. — С. 229–235 — doi:10.12720/jcm.14.3.

[6] Hamadneh, Nabhan and Murray, David and Dixon, Michael and Cole, Peter Weighted RED (WTRED) Strategy for TCP Congestion Control // ICIEIS 2011 — Springer. — 2011. — Т. 2. — С. 421–434 — doi:110.1007/978-3-642-25453-6-37.

[7] Lingyun Lu and Yang Xiao and Seok Woo and Kiseon Kim Nonlinear AQM for Multiple RED Routers // 2008 Third International Conference on Convergence and Hybrid Information Technology. — IEEE. — 2008. — November — doi:10.1109/iccit.2008.68.

[8] Kaiyu Zhou and Kwan L. Yeung and Victor O.K. Li Nonlinear RED: A simple yet efficient active queue management scheme // Computer Networks ed. by Roberts J. — Elsevier BV. — 2006. — December. — Т. 50, № 18. — С. 3784-3794. — doi:10.1016/j.comnet.2006.04.007.

[9] Kim', Tae-Hoon and Lee', Kee-Hyun Refined Adaptive RED in TCP/IP Networks // SICE-ICASE International Joint Conference 2006 Oct. 18-2 1, 2006 in Bexco, Busan, Koreas— IEEE. — 2006. — October. — doi:10.1109/SICE.2006.314633.

# Приложения

- main.tcl

```
#Создать новый экземпляр объекта Symulator
set ns [new Simulator]
#Открыть трейс файл для nam
set nf [open output/out.nam w]
$ns namtrace-all $nf
set N 20
source "nodes.tcl"
source "links.tcl"
source "queue.tcl"
source "connections.tcl"
source "timing.tcl"
source "nam.tcl"
source "finish.tcl"
#Запуск программы
$ns run
```

- nodes.tcl

```
set node_(r0) [$ns node]
set node_(r1) [$ns node]

for {set i 0} {$i < $N} {incr i} {
        set node_(s$i) [$ns node]
        set node_(s[expr $N + $i]) [$ns node]
        }
```

- links.tcl

```
```

```
for {set i 0} {$i < $N} {incr i} {
        $ns duplex-link $node_(s$i) $node_(r0) 100Mb 20ms DropTail
        $ns duplex-link $node_(s[expr $N + $i]) $node_(r1) 100Mb 20ms
        ↪ DropTail
}

$ns simplex-link $node_(r0) $node_(r1) 20Mb 15ms RED
$ns simplex-link $node_(r1) $node_(r0) 15Mb 20ms DropTail
```
- nam.tcl
```
$node_(r0) color "red"
$node_(r1) color "red"
$node_(r0) label "RED"
$node_(r1) shape "square"
$node_(r0) label "square"

$ns simplex-link-op $node_(r0) $node_(r1) orient right
$ns simplex-link-op $node_(r1) $node_(r0) orient left
$ns simplex-link-op $node_(r0) $node_(r1) queuePos 0
$ns simplex-link-op $node_(r1) $node_(r0) queuePos 0

for {set m 0} {$m < $N} {incr m} {
        $ns duplex-link-op $node_(s$m) $node_(r0) orient right
        $ns duplex-link-op $node_(s[expr $N + $m]) $node_(r1) orient left
}
for {set i 0} {$i < $N} {incr i} {
        $node_(s$i) color "blue"
        $node_(s$i) label "ftp"
}
```
- connections.tcl
```
for {set t 0} {$t < $N} {incr t} {
        $ns color $t green
        set tcp($t) [$ns create-connection TCP/Reno $node_(s$t) TCPSink
        ↪ $node_(s[expr $N + $t]) $t]
```

```
        $tcp($t) set window_ 32
        $tcp($t) set maxcwnd_ 32
        set ftp($t) [$tcp($t) attach-source FTP]
}

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"
}
```

- queue.tcl

```
$ns queue-limit $node_(r0) $node_(r1) 300
$ns queue-limit $node_(r1) $node_(r0) 300

set windowVsTime [open output/WvsT w]
set qmon [$ns monitor-queue $node_(r0) $node_(r1) [open output/qm.out w]]
[$ns link $node_(r0) $node_(r1)] queue-sample-timeout

set redq [[$ns link $node_(r0) $node_(r1)] queue]
$redq set thresh_ 75 #q_min
$redq set maxthresh_ 150  #q_max
$redq set q_weight_ 0.002 # q_weight
$redq set linterm_ 10 # 1/p_max
$redq set gentle_ false  #
$redq set drop-tail_ true
$redq set queue-in-bytes false #очередь в пакетах
$redq set mean_pktsize_ 500
set tchan_ [open output/all.q w]
$redq trace curq_
$redq trace ave_
$redq attach $tchan_
```

```
- timing.tcl
```
```
for {set r 0} {$r < $N} {incr r} {
        $ns at 0.0 "$ftp($r) start"
        $ns at 1.0 "plotWindow $tcp($r) $windowVsTime"
        $ns at 24.0 "$ftp($r) stop"
}

$ns at 25.0 "finish"
```
```
- finish.tcl
```
```
#Finish procedure
proc finish {} {
   global ns nf
   $ns flush-trace
   close $nf
   global tchan_
   set awkCode {
      {#запись данных в файлы очереди и средней очереди
         if ($1 == "Q" && NF>2) {
            print $2, $3 >> "output/temp.q";
            set end $2
         }
         else if ($1 == "a" && NF>2)
         print $2, $3 >> "output/temp.a";
      }
   }

   set f [open output/temp.queue w]
   puts $f "TitleText: RED"
   puts $f "Device: Postscript"

   if { [info exists tchan_] } {
      close $tchan_
   }
   #обновление данных
```

```
    exec rm -f output/temp.q output/temp.a
    exec touch output/temp.a output/temp.q

    exec awk $awkCode output/all.q

    puts $f \"queue
    exec cat output/temp.q >@ $f
    puts $f \n\"ave_queue
    exec cat output/temp.a >@ $f
    close $f
    # вывод в xgraph
    exec xgraph -bb -tk -x time -t "TCPRenoCWND" output/WvsT &
    exec xgraph -bb -tk -x time -y queue output/temp.queue &
    exit 0
}
```
- out.gp
```
#!/usr/bin/gnuplot -persist

#вывод графиков
set terminal postscript eps
set output "images/queues.eps"
set xlabel "Time, s"
set ylabel "Queue Length [pkt]"
set title "RED Queue"
plot "output/temp.q" with lines linestyle 1 lt 1 lw 2 title "Queue length",
    ↪ "output/temp.a" with lines linestyle 2 lt 3 lw 2 title "Average queue
    ↪ length"

set terminal postscript eps
set output "images/TCP.eps"
set xlabel "Time (s)"
set ylabel "Window size [pkt]"
set title "TCPVsWindow"
plot "output/WvsT" with lines linestyle 1 lt 1 lw 2 title "WvsT"
```