МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ»

Факультет физико-математических и естественных наук
Кафедра теории вероятностей и кибербезопасности

«Допустить к защите»

Заведующий кафедрой
теории вероятностей
и кибербезопасности
д. т. н., профессор
_____ К. Е. Самуйлов
«___» _____ 20__г.

Выпускная квалификационная работа бакалавра

Направление 02.03.02 «Фундаментальная информатика и информационные технологии»

Тема «Моделирование систем управления трафиком»

Выполнил студент Тагиев Байрам Алтай оглы

Группа НФИбд-02-20
Студенческий билет № 1032200531

Руководитель выпускной
квалификационной работы
профессор кафедры
теории вероятностей
и кибербезопасности
д. ф.-м. н., профессор
Д. С. Кулябов

_____

Автор _____

Москва
2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ»

Аннотация

выпускной квалификационной работы

Тагиева Байрама Алтай оглы

на тему: Моделирование систем управления трафиком

Алгоритм управления очередью, который применяется в маршрутизаторах, играет важную роль в обеспечении качества обслуживания (QoS). В этой работе представляется оценка производительности на основе моделирования и сравнения некоторых популярных методов управления очередями, такие как случайное раннее обнаружение (RED) и DropTail с точки зрения размера очереди, задержки в очереди, изменения задержки в очереди, скорости отбрасывания пакетов и использования полосы пропускания. Моделирование проходит в рамках имитационного и натурного моделирования для понимания эталонной модели и реальных показателей. Результаты моделирования показывают, что алгоритмы семейства RED показывают лучшие результаты в отношении Drop Tail с точки зрения задержки в очереди, изменения задержки в очереди.

Автор ВКР _____ Тагиев Байрам Алтай оглы

Содержание

Перечень условных обозначений и сокращений

PQM — Passive Queue Management — алгоритм пассивного управления очередью

AQM — Active Queue Management — алгоритм активного управления очередью

RED — Random Early Detection — алгоритм случайного раннего обнаружения

ARED — Adaptive RED — самонастраивающийся RED

RARED — Refinde ARED — улучшенный ARED

NS-2 — Network Simulator 2

TCP — Transmission Control Protocol — протокол управления передачей

UDP — User Datagram Protocol — протокол пользовательских датаграмм

RTT — Round-trip time (delay) — время приема-передачи

QoS — Quality of service — качество обслуживания

Введение

Данное исследование посвящено анализу и сравнению алгоритмов управления очередями на основе RED, реализованных с использованием программных средств NS2 и MiniNet. Основная цель исследования - изучить принципы работы и эффективность алгоритмов RED путем моделирования их поведения в различных сетевых условиях. В рамках проекта была предпринята попытка смоделировать поведение сетей с использованием алгоритмов RED и сравнить их производительность с целью определения оптимальных настроек для обеспечения качественной и надежной передачи данных. Результаты моделирования предназначены для определения оптимальных параметров алгоритмов управления очередями, которые могут способствовать общему повышению производительности сетевых систем.

Актуальность темы

Важность этого исследования заключается в изучении механизма активного управления очередями (AQM) в RED, который помогает оптимизировать распределение сетевых ресурсов и обеспечивает соответствие требованиям пользователей к скорости и надежности передачи данных.

Цель работы:

Целью данной выпускной квалификационной работы является исследование и анализ алгоритмов семейства RED (Random Early Detection) для активного управления очередью. Основная задача заключается в изучении принципов работы и эффективности данных алгоритмов в контексте управления потоками данных в сетевых маршутизаторах, а также сравнительный анализ различных алгоритмов в рамках натурной и имитационной модели.

Для достижения поставленных целей будут проведены исследования и анализ различных алгоритмов из семейства RED. В ходе работы будет проанализировано влияние параметров алгоритмов RED на производительность и стабильность сетевых систем, а также будет проведено сравнение эффективности различных вариантов алгоритма RED.

Для достижения целей работы будет использовано программное обеспечение и инструменты моделирования NS2 и Mininet. Основной фокус будет сосредоточен на сравнительном анализе результатов моделирования и выборе оптимальных параметров алгоритма RED для достижения наилучшей производительности.

Основными задачами работы будут:

- Изучение принципов работы алгоритмов семейства RED.
- Сравнительный анализ эффективности различных вариантов алгоритма RED.

- Проведение экспериментов с использованием различных инструментов моделирования для оценки эффективности алгоритмов RED.

В итоге данной работы ожидается получение информации об алгоритмах семейства RED и их применении в сетевых системах, а также оценка эффективности этих алгоритмов с использованием различных инструментов моделирования. Полученные результаты могут быть использованы для оптимизации управления потоками данных в сетевых системах и повышения их производительности и стабильности.

В результате этого исследования мы стремимся получить представление об алгоритмах в рамках RED и их применении в сетевых маршрутизаторах. Мы также оценим эффективность этих алгоритмов с помощью различных инструментов моделирования. Результаты, полученные в результате этого исследования, могут быть использованы для оптимизации управления потоками данных в сетевых системах и повышения их производительности и стабильности.

Краткое содержание работы

Данная работа состоит из введения, трех основных разделов, списка литературы и приложений. В первом разделе рассматриваются инструменты сетевого моделирования и принципы их работы, а также краткий обзор проверяемых алгоритмов. Во втором разделе представлен обзор алгоритмов RED и их реализации в моделях в средствах моделирования NS-2 и Mininet. В третьем разделе подробно представлены результаты моделирования, построены необходимые графики и сделаны выводы относительно эффективности предложенных алгоритма с использованием двух инструментов моделирования, обобщены результаты работы и сделаны основные выводы.

Глава 1. Теоретическое введение

1.1. Методы моделирования

Натурным моделированием называют проведение исследования на настоящем предмете с последующей обработкой результатов опыта на основе теории подобия. Натруное моделирование разделяется на научный эксперимент, комплексные испытания и производственный эксперимент. Научный эксперимент характеризуется обширным применением средств автоматизации, использованием весьма всевозможных средств обработки информации, возможностью вмешательства человека в процесс выполнения эксперимента.

Имитационное моделирование — это способ исследования, при котором исследуемая система сменяется моделью, с достаточной точностью описывающей настоящую систему, с которой проводятся опыты с целью извлечения информации об этой системе. Такую модель можно «проиграть» во времени, как для одного испытания, так и заданного их множества.

1.2. Алгоритмы управления очередями

Алгоритмы управления и обработки очереди разделяются на два основных класса:

Алгоритмы пассивного управления очередью (PQM) [1] — класс алгоритмов, применяемых для обработки очередей, при котором при достижении порогового значения, алгоритм отбрасывает пакеты в соответствии с некоторым правилом конкретной реализации алгоритма. Данные алгоритмы просты в исполнении, однако лишены возможности адаптироваться под разные нагрузки.

Примеры алгоритмов пассивного управления очередью:

- DropTail — отбрасывает пакеты с конца очереди
- DropHead — отбрасывает пакеты с начала очереди
- RandomDrop — отбрасывает случайные пакеты из очереди

В маршрутизаторах и коммутаторах активное управление очередью (AQM) [2] относится к процессу отбрасывания пакетов внутри буфера, связанного с контроллером сетевого интерфейса (NIC), до того, как этот буфер заполнится, с целью уменьшения перегрузки сети или повышения сквозной производительности.

Примеры алгоритмов активного управления очередью (а также статьи с подробным описанием каждого из алгоритмов, так как покрыть каждый из этих алгоритмов в рамках данной работы не получится):

- RED [3; 4]
- ARED [5—8]
- BLUE [9]
- GRED [10]

В данной работе будут изучены лишь 3 алгоритма, в связи с тем, что они уже используются в сетевом стеке ядра Linux, а также средстве имитационного моделирования NS-2. А именно:

- DropTail — как базовый алгоритм пассивного управления очередью;
- RED — как классический алгоритм активного управления очередью;
- ARED — в качестве сравнения динамического [1] алгоритма с классическим;

1.3. Алгоритмы активного управления очередью семейства RED

1.3.1. RED

RED [11] (Random Early Detection, произвольное раннее обнаружение) — алгоритм активного управления очередью (AQM) для управления переполнением очередей маршрутизаторов с возможность предотвращения перегрузок.

Вероятность p_b маркировки на отбрасывание пакетов представляет собой функцию, линейно зависящую от $\hat{q}$, 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:

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

1.3.2. Разбор алгоритма RED

Пакет при поступлении в систему попадает в модуль сброса. Решение о сбросе пакета принимается на основе значения вероятности p, получаемого от управляющего модуля.

[1] Под динамическим подразумевается алгоритм, который в процессе своей работы подстраивает параметры системы. Подробнее об этом в разделе 1.3.4

Вероятность p сброса пакетов зависит от экспоненциально взвешенного скользящего среднего размера длины очереди $\hat{q}$, также вычисляемого управляющим модулем, основываясь на текущем значении длины очереди q (см. рис. 1.1).

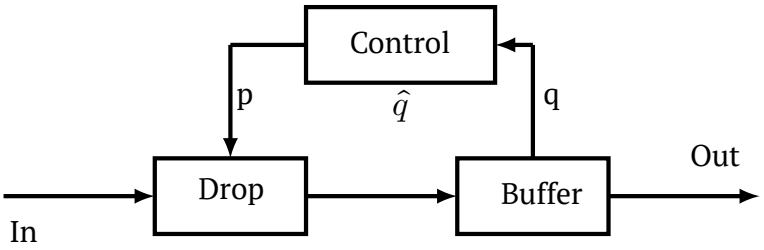

Рис. 1.1.. Модуль RED

График вероятности потери пакета в зависимости от среднего размера очереди приведен на графике 1.2.

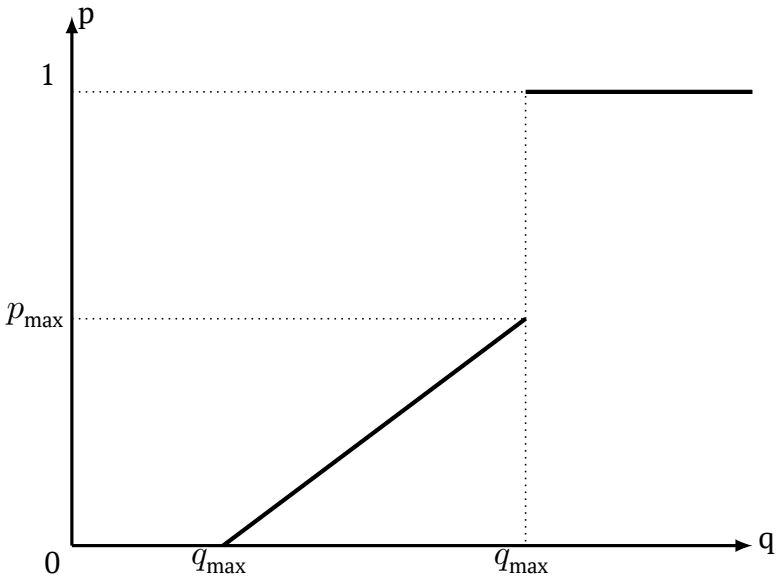

Рис. 1.2.. Вероятность потери пакета в RED

Реализация функции просчета вероятности на отбрасывание пакета в алгоритме RED выглядит следующим образом:

Algorithm 1 Функция просчета вероятности на отбрасывание

if $\hat{q} \geq q_{\text{max}}$ **then**
    $p = 1$
else
    $p = k * \hat{q} + b$ ▷ график прямой от q_{min} до q_{max}
    $p* = q_{\text{max}}$
end if

1.3.3. Проблемы RED

Одна из фундаментальных проблем с дизайном RED заключается в том, что он зависит от размера очереди как показателя рабочей нагрузки. В то время как наличие постоянной очереди указывает на состояние перегрузки, длина очереди предоставляет очень мало информации о серьезности состояния перегрузки.

Из-за зависимости алгоритма RED от размера очереди ему присуща проблема точного определения степени перегруженности. Как следствие, RED требует правильных параметров для правильной работы в различных сценариях перегрузки. В то время как RED может достичь идеального рабочего состояния, это может быть достигнуто только при наличии достаточной буферной емкости и соотминимального порога q_{min}ветствующих настроек параметров.

1.3.4. ARED

В алгоритме Adaptive RED (ARED), разработанном Фенгом [5; 6] и усовершенствованная С. Флойд [7], функция сброса модифицируется посредством изменения по принципу AIMD (принцип AIMD заключается в том, что увеличение некоторой величины производится путём сложения с некоторым параметром, у уменьшение — путём умножения на параметр).

Алгоритм ARED функционирует следующим образом. Для каждого интервала interval (параметр) в секундах, если $\hat{q}$ больше целевого (желаемого) значения $\hat{q_t}$ и $p_{\text{max}} \leqslant 0,5$, то p_{max} увеличивается на некоторую величину α; в противном случае, если $\hat{q}$ меньше целевого значения $\hat{q_t}$ и $p_{\text{max}} \geqslant 0,01$, то p_{max} уменьшается в β раз:

$$p_{\text{max}} = \begin{cases} p_{\text{max}} + \alpha, & \hat{q} > \hat{q_t}, \quad p_{\text{max}} \leqslant 0,5, \\ \beta * p_{\text{max}}, & \hat{q} < \hat{q_t}, \quad p_{\text{max}} \geqslant 0,01, \end{cases}$$

где

$$q_{\text{min}} + 0,4\left(q_{\text{max}} - q_{\text{min}}\right) < \hat{q_t} < q_{\text{min}} + 0,6\left(q_{\text{max}} - q_{\text{min}}\right).$$

У алгоритма ARED есть следующие преимущества, по сравнению с классическим RED:

- автоматическая установка $\widehat{q}_t$ на основе пропускной способности соединения
- медленное адаптирование $p_{\max}$, для того, чтобы средний размер очереди находился в пределах $q_{\min}$ и $q_{\max}$, а также сама вероятность находилась в промежутке [0.01, 0.5] (что эквивалентно [1%, 50%])

Данные изменения существенно помогают решить основную проблмеу классического RED (которые описаны в 1.3.3).

1.4. Средства моделирования

1.4.1. Средство имитационного моделирования ns-2

NS-2 (Network Simulator 2) — это симулятор дискретных событий, предназначенный для исследования компьютерных сетей. NS-2 предоставляет существенную поддержку для моделирования протоколов TCP, маршрутизации и многоадресной рассылки по проводным и беспроводным (локальным и спутниковым) сетям.

1.4.2. Средство натурного моделирования Mininet

Mininet — это симулятор сетевых топологий, который позволяет моделировать и анализировать поведение сети в имитируемой среде. Симулятор основан на виртуальных машинах Linux и технологиях пространства имен, которые используются для создания изолированных сетевых компонентов. Используя Mininet, можно изучать различные сетевые протоколы, алгоритмы маршрутизации и методы управления трафиком. Возможности моделирования Mininet включают создание виртуальных сетевых узлов, настройку топологий (включая связь между узлами и конфигурацию IP-адресов), моделирование различных сетевых условий (таких как задержки, потеря пакетов и пропускная способность) и интеграцию с контроллерами для изучения новых протоколов и алгоритмов.

1.5. Дополнительные инструменты

1.5.1. TC — Traffic Control

Linux предлагает инструменты для управления передачей пакетов и манипулирования ими. Подсистема Linux Traffic Control (TC) помогает контролировать, классифицировать, формировать и планировать сетевой трафик. TC также изменяет содержимое пакетов во время классификации с помощью фильтров и действий. Подсистема TC достигла этого

за счет использования дисциплин массового обслуживания (qdisc), фундаментальных элементов архитектуры TC.

Механизм планирования упорядочивает ячейки перед тем, как они войдут в разные очереди или выйдут из них. Наиболее распространенным планировщиком является планировщик First-In-First-Out (FIFO). Можно выполнять операции с qdiscs временно с помощью утилиты tc или постоянно с помощью NetworkManager.

1.5.2. Iperf3 — генератор TCP, UDP и SCTP трафика

Iperf - это инструмент, используемый для активного измерения максимально достижимой пропускной способности IP-сетей. Он позволяет пользователям настраивать различные параметры, связанные с синхронизацией, протоколами и размерами буфера.

При каждом тестировании создается отчет, содержащий информацию об измеренной пропускной способности, скорости передачи данных, потере пакетов и других соответствующих параметрах. Программа разделена на клиентскую и серверную части, поэтому для ее работы понадобится как минимум 2 устройства, подключенных к сети.

Глава 2. Моделирование алгоритмов

2.1. Эталонная модель

В рамках анализа алгоритмов была созданна эталонная модель, в рамках которой будут проводиться тестирование. Описание моделируемой сети:

- Сеть состоит из 10 TCP-источников и TCP-приемников, двух маршрутизаторов R1 и R2 между приемниками и источниками.
- Между TCP-источниками и первым маршрутизатором установлена задежрка в 20 мс.
- Между TCP-приемниками и вторым маршрутизатором также установлена задежрка в 20 мс.
- Между маршрутизаторами установлена задежрка в 15 мс. и очередью типа DropTail / RED / ARED (именно здесь и будет происходить сравнение изучаемых алгоритмов).
- Максимальный размер TCP-окна 32; размер передаваемого пакета 1000 байт; время моделирования — 20 секунд.

В связи с тем, что только два основных алгоритма активного управления очередью (RED, ARED) реализованы в сетевом стеке ядра Linux — они же и будут рассматриваться для сравнения. Иные алгоритмы требуют внесения изменения в модуль ядра.

Данную модель реализовать как в Mininet, так и в NS-2. Сравнение в разных средствах моделирования представлено для понимания того, как модель себя будет вести в идеальной среде (в рамках имитационного моделирования) и в реальной среде (в рамках натурного моделирования).

2.2. Реализация в NS-2

Для реализации модели на NS-2 необходимо описать ее на языке TCL. Структура модели описана связными моулями, которые реализую отдельные этапы. Этапы моделирования:

- инициализация узлов, соединение и установка настроек.
- настройка мониторинга и параметры для самого алгоритма (RED/ARED)
- составлние файлов для построения графиков графиков
- построение графиков.

Ключевым отличием между настройками парамтеров RED и ARED лишь в одной строчке, которая включает сам ARED, (где параметр redq — соединение, на котором работаем AQM).

```
$redq set adaptive_ 1
```

Также, для чистоты эксперимента будут использоваться TCP-агент Linux — имплементацию сетевого стека, применяемую в ядре Linux. Так как это лишь имитационная модель, то и сам TCP-агент лишь копирует поведение, а не полноценно использует модуль ядра.

Для отрисовки графиков используется GNUPlot, при помощи которого мы извлекаем данные из файлов. Для этого был написан скрипт, который отрисовываает изменение размера длины очереди, средней длины очереди и размера окна, как основные показатели для сравнения.

В файле C импортируются основные модули, которые реализиуют необходимые для моделирования функционал.

В файле D инициализируются основные узлы нашей системы, описание которых представлено в разделе 2.1. В этом же файле мы и указываем какой алгоритм управления очередью мы используем: либо RED/ARED, либо DropTail (данный алгоритм используется лишь как показатель того, что алгоримы активного управления очередью показывают гораздо более оптимальную обработку очереди и задрежки в ней).

В файле E уже описываются основные параметры алгоритмов урпавления очередью, а именно:

- `thresh_` — минимальная граница для среднего размера очереди в пакетах;
- `maxthresh_` — максимальная граница для среднего размера очереди в пакетах;
- `q_weight_` — вес очереди, используется для вычисления среднего размера очереди;
- `queue_in_bytes_` — принимает значение true, если используется режим измерения среднего размера очереди в байтах, а не пакетах;
- `adaptive_` — для переключения между режимами RED и ARED;

Файл F преставляет из себя одну функцию — задание моделируемого времени. В нем мы указываем начало и конец моделирования, а также запуск потока данных. В данном случае в качестве передаваемых данных используется FTP-трафик.

Последний файл G отвечает за формировнае файлов трассировки, а также графиков размера очереди и средней длины очереди, а также размера окна, как основные показатели, необходимые для сравнения алгоритмов управления очередью.

2.3. Реализация в Mininet

FIXME: тут описать структуру файлов, которые будут использоваться. Тк сама модель готова, но пока не готова для того, чтобы ее в файл, чтобы потом не забыть ее обновить.

2.3.1. Данная часть уже в работе, но пока что не готова

Заключение

Текст.

В работе было рассмотрено:

1. Принципы работы алгоритмов пассивного и активного управления очередями.
2. Примеры алгоритмов семеймтва RED, преимущества и недостатки.
3. Произведен сравнительный анализ алгоритмов в рамках тестового сценария нагрузки сети, сделан вывод в пользу FIXME: указать алгоритм, который покажет себя лучше.

   FIXME: добавить еще пункты

Итог: FIXME: Добавить сюда результаты

Список литературы

1. Reformed passive queue management algorithm / W. Jiang [и др.]. — 2011. — Сент. — DOI: 10.1109/ICECENG.2011.6057005.
2. *Dhumane A.* Active Queue Management. — 2014. — Нояб.
3. *Floyd S.*, *Jacobson V.* Random early detection gateways for congestion avoidance // IEEE/ACM Transactions on Networking. — 1993. — Т. 1, вып. 4. — С. 397—413. — DOI: 10.1109/90.251892.
4. A self-configuring RED gateway / W.-C. Feng [и др.] // IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320) (New York, NY, USA). — IEEE, 03.1999. — DOI: 10.1109/infcom.1999.752150.
5. Techniques for Eliminating Packet Loss in Congested TCP/IP Networks / W.-c. Feng [и др.]. — 1999. — Окт.
6. Self-configuring RED gateway / S.-c. Gateway [и др.] // Proceedings - IEEE INFOCOM. — 2000. — Нояб. — Т. 3.
7. *Floyd S.*, *Gummadi R.*, *Shenker S.* Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. — 2001. — Сент.
8. *La R.*, *Ranjan P.*, *Abed E.* Analysis of adaptive random early detection (ARED) //. — Elsevier, 2003. — DOI: 10.1016/s1388-3437(03)80276-7.
9. *Feng W.-c.*, *Kandlur D.*, *Saha D.* BLUE: An Alternative Approach to Active Queue Management. — 2000. — Сент. — DOI: 10.1145/378344.378350.
10. Revisiting the Gentle Parameter of the Random Early Detection (RED) for TCP Congestion Control / N. Hamadneh [и др.] // Journal of Communications. — 2019. — С. 229—235. — DOI: 10.12720/jcm.14.3.229-235.
11. *Floyd S.*, *Jacobson V.* Random Early Detection Gateways for Congestion Avoidance // IEEE/ACM Transactions on Networking. — 1993. — Сент. — Т. 1. — С. 397—413. — DOI: 10.1109/90.251892.

Приложение А. Название первого приложения

А.1. Название секции

Текст.

А.2. Название секции

Текст.

Приложение В. Название второго приложения

В.1. Название секции

Текст.

В.2. Название секции

Текст.

В данном приложении представлен исходный код программы для моделирования алгоритмов RED в среде имитационного моделирования NS-2.

Приложение C. Файл ns2-red.tcl

```
# Bayram Tagiev
#
# main file

set ns [new Simulator]

set nf [open out.nam w]
$ns namtrace-all $nf



source "nodes.tcl"
source "queue.tcl"
source "plotWindow.tcl"
source "modeling.tcl"
source "finish.tcl"



$ns run
```

Приложение D. Файл nodes.tcl

```
# Bayram Tagiev
#
# Setting up nodes

set node_(r0) [$ns node]
set node_(r1) [$ns node]
$node_(r0) color "red"
$node_(r1) color "red"
$node_(r0) label "red"

set n 20

for {set i 0} {$i < $n} {incr i} {
        set node_(s$i) [$ns node]
        $node_(s$i) color "blue"
        $node_(s$i) label "ftp"
        $ns duplex-link $node_(s$i) $node_(r0) 100Mb 20ms DropTail

        set node_(s[expr $n + $i]) [$ns node]
        $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

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



for {set t 0} {$t < $n} {incr t} {
        $ns color $t green
        set tcp($t) [$ns create-connection TCP/Linux $node_(s$t)
   ↪ TCPSink $node_(s[expr $n + $t]) $t]
        $tcp($t) set window_ 32
        $tcp($t) set maxcwnd_ 32
```

```
        $tcp($t) set packetsize_ 1000
        set ftp($t) [$tcp($t) attach-source FTP]
}

$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
}
```

Приложение E. Файл queue.tcl

```
# Bayram Tagiev
#
# Adding monitors to simulation

set windowvstime [open wvst w]
set qmon [$ns monitor-queue $node_(r0) $node_(r1) [open qm.out w]]
[$ns link $node_(r0) $node_(r1)] queue-sample-timeout

set redq [[$ns link $node_(r0) $node_(r1)] queue]
$redq set qlim_ 75 150
$redq set thresh_ 75
$redq set maxthresh_ 150
$redq set q_weight_ 0.002
$redq set linterm_ 10
$redq set queue_in_bytes_ false
$redq set gentle_ false
$redq set adaptive_ 1

set tchan_ [open all.q w]
$redq trace curq_
$redq trace ave_
$redq attach $tchan_
```

Приложение F. Файл plotWindow.tcl

```
# Bayram Tagiev
#
# Plotting procedure
# makes a file with required data

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"
}
```

Приложение G. Файл modeling.tcl

```
# Bayram Tagiev
#
# Starting simulation

for {set r 0} {$r < $n} {incr r} {
        $ns at 0.0 "$ftp($r) start"
        $ns at 1.0 "plotwindow $tcp(0) $windowvstime"
        $ns at 20.0 "$ftp($r) stop"
}

$ns at 21.0 "finish"
```

Приложение Н. Файл finish.tcl

```
# finish procedure
#
# Generates necesery files
# Loads xgraph to render graphs


proc finish {} {
   global ns nf
   $ns flush-trace
   close $nf
   global tchan_
   set awkCode {
      {
         if ($1 == "Q" && NF>2) {
            print $2, $3 >> "temp.q";
            set end $2
         }
         else if ($1 == "a" && NF>2)
            print $2, $3 >> "temp.a";
      }
   }

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

   if { [info exists tchan_] } {
      close $tchan_
   }

   exec rm -f temp.q temp.a
   exec touch temp.a temp.q

   exec awk $awkCode all.q

   puts $f \"queue
```

```
    exec cat temp.q >@ $f
    puts $f \n\"ave_queue
    exec cat temp.a >@ $f
    close $f

    exec xgraph -bb -tk -x time -t "TCPRenoCWND" wvst &
    exec xgraph -bb -tk -x time -y queue temp.queue &
    exec nam out.nam &
    exit 0
}
```

Список иллюстраций

Список таблиц