

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет  
вычислительной техники

Кафедра  
Вычислительная техника

Направление подготовки 09.04.01 «Информатика и вычислительная техника»  
Магистерская программа Вычислительные машины, комплексы, системы и сети

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ  
РАБОТА МАГИСТРА**

на тему

**МОДЕЛИРОВАНИЕ И ИССЛЕДОВАНИЕ АЛГОРИТМОВ  
УПРАВЛЕНИЯ ОЧЕРЕДЯМИ В КОММУТАТОРАХ**

Студент Аладьев 06.06.18 Аладьев Юрий Юрьевич  
(подпись, дата)  
Руководитель Коннов 06.06.18 Коннов Н.Н.  
(подпись, дата)  
Нормоконтролер Кучин 9.06.18 Кучин А.В.  
(подпись, дата)  
Рецензент начальник БЦСП, Лукин 9.06.18 Лукин Д.В.  
АО «НПП «Рубин» (подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 25.05.2018г №38-09-12)

Заведующий кафедрой Пашенко (подпись) Пашенко Д.В.

Работа защищена с отметкой Пашенко (подпись) (протокол заседания ГАК от 14.06.18 № 9)

Секретарь ГАК Гурин (подпись) Гурин Е.И.

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

Вычислительная техника

«Утверждаю»

Заведующий кафедрой

Д.В. Пашенко

«  »    20   г

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ  
МАГИСТРА**

1. Студент Аладьев Юрий Юрьевич гр. 16BBm1 факультета ВТ  
<Фамилия, имя, отчество полностью>

2. Тема работы Моделирование и исследование алгоритмов управления очередями в коммутаторах

Тема утверждена приказом ПГУ № 979 а/о от "03" 10 2016

2. Руководитель работы к.т.н., профессор Коннов Н.Н.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

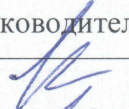
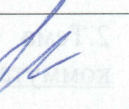
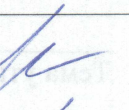
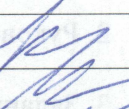
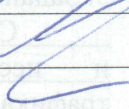
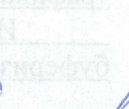
Создание имитационных моделей алгоритмов управления очередями в коммутаторах и исследование этих моделей, а также создание генератора сетевого трафика.

Исследуемые алгоритмы управления очередями: алгоритм TailDrop, RED с полной буферизацией и с обработкой трафика "на лету".

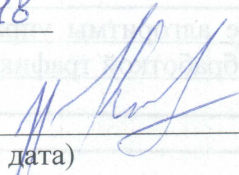
4. Перечень подлежащих разработке вопросов \_\_\_\_\_

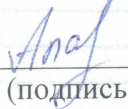
- 1) Анализ различных алгоритмов управления очередями в коммутаторах. \_\_\_\_\_
- 2) Анализ и выбор средства моделирования. \_\_\_\_\_
- 3) Разработка обобщенной структуры буфера выходного порта коммутатора на основе сетей Петри. \_\_\_\_\_
- 4) Создание имитационной модели алгоритма управления очередями TailDrop. \_\_\_\_\_
- 5) Создание имитационной модели алгоритма управления очередями RED. \_\_\_\_\_
- 6) Разработка генератора сетевого трафика для исследования созданных моделей. \_\_\_\_\_
- 7) Исследование созданных моделей алгоритмов управления очередями на различных наборах сетевого трафика. \_\_\_\_\_

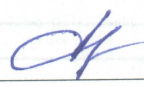
5. Календарный график выполнения работы

| № п/п | Наименование этапов работы                             | Объем работы | Срок выполнения | Подпись руководителя                                                                  |
|-------|--------------------------------------------------------|--------------|-----------------|---------------------------------------------------------------------------------------|
| 1     | Анализ предметной области                              | 10           | 21.05.18        |    |
| 2     | Выбор средства моделирования                           | 5            | 23.05.18        |   |
| 3     | Разработка обобщенной структуры буфера выходного порта | 10           | 24.05.18        |  |
| 4     | Создание имитационной модели алгоритма TailDrop        | 15           | 27.05.18        |  |
| 5     | Создание имитационной модели алгоритма RED             | 20           | 01.06.18        |  |
| 6     | Разработка генератора сетевого трафика                 | 10           | 03.06.18        |  |
| 7     | Исследование полученных моделей                        | 10           | 05.06.18        |  |
| 8     | Оформление пояснительной записки                       | 20           | 09.06.18        |  |

Дата выдачи задания "21" 05 2018

Руководитель магистерской работы 21.05.2018  Н. Кошлов  
(подпись, дата)

Задание к исполнению принял студент  21.05.2018  
(подпись, дата)

Работу к защите допустить  
Декан факультета  09.06.2018  
(подпись, дата)

## Реферат

Пояснительная записка 72 листа, 6 таблиц, 25 рисунков.

ТЕЛЕКОММУНИКАЦИОННАЯ СЕТЬ, КОММУТАТОР,  
КАЧЕСТВО ОБСЛУЖИВАНИЯ QOS, МЕХАНИЗМЫ QOS,  
УПРАВЛЕНИЕ ОЧЕРЕДЯМИ, TAILDROP, RED

*Объектом исследования* является коммутатор Ethernet с поддержкой качества обслуживания.

*Предметом исследования* являются алгоритмы управления очередями обработки сетевого трафика в коммутаторе для предотвращения перегрузок в телекоммуникационной сети.

*Цель данной работы* состояла в разработке моделей алгоритмов управления очередями в коммутаторе и исследование вероятностно-временных характеристик системы в зависимости от интенсивности поступающей нагрузки и выбранного алгоритма управления очередями для улучшения параметров качества обслуживания при передаче данных по пакетной сети на основе протоколов TCP/IP.

В результате проделанной работы были разработаны модели алгоритмов управления очередями TailDrop и RED, была усовершенствована модель генератора сетевого трафика и исследовано поведение разработанных моделей на различных наборах сетевого трафика.

*Научная и практическая значимость.* Построенные модели алгоритмов управления очередями в коммутаторах позволяют оценить вероятностно-временные характеристики подобных систем, а также оценить влияние проектируемых систем на параметры качества обслуживания в сетях с поддержкой QoS.

## Содержание

|                                                                                                  |    |
|--------------------------------------------------------------------------------------------------|----|
| Введение.....                                                                                    | 4  |
| 1 Качество обслуживания QoS и его механизмы.....                                                 | 6  |
| 1.1 Общие сведения о качестве обслуживания QoS.....                                              | 6  |
| 1.2 Механизмы реализации QoS.....                                                                | 7  |
| 1.2.1 Механизмы классификации и маркирования .....                                               | 8  |
| 1.2.2 Механизмы обслуживания очередей .....                                                      | 9  |
| 1.2.3 Механизмы управления очередью .....                                                        | 12 |
| 1.2.4 Механизмы управления полосой пропускания .....                                             | 13 |
| 1.3 Особенности механизма управления очередями .....                                             | 14 |
| 2 Моделирование алгоритмов управления очередями .....                                            | 20 |
| 2.1 Выбор средства моделирования .....                                                           | 20 |
| 2.2 Построение моделей в среде CPN Tools.....                                                    | 23 |
| 2.2.1 Описание модели алгоритма пассивного управления<br>очередью Tail Drop .....                | 31 |
| 2.2.2 Описание модели алгоритма активного управления очередью<br>RED.....                        | 33 |
| 2.2.3 Описание модели алгоритма активного управления очередью<br>RED с полной буферизацией ..... | 37 |
| 3 Генератор сетевого трафика для моделей коммутаторов в среде<br>CPN Tools .....                 | 40 |
| 3.1 Обобщенная модель генератора сетевого трафика .....                                          | 44 |
| 3.2 Генератор сетевого трафика с различными значениями<br>параметров класса QoS .....            | 45 |
| 3.3 Генератор сетевого трафика с "тяжелыми хвостами" .....                                       | 53 |
| 4 Исследование моделей алгоритмов управления очередями .....                                     | 58 |
| Заключение .....                                                                                 | 61 |
| Список использованных источников .....                                                           | 62 |
| ПРИЛОЖЕНИЕ А. ПРЕЗЕНТАЦИЯ .....                                                                  | 65 |

## **Введение**

Компьютерные сети, которые также называются вычислительными сетями, или сетями передачи данных, являются объединением двух важнейших научно-технических отраслей современной цивилизации — компьютерных и телекоммуникационных технологий. С одной стороны, сети представляют собой частный случай распределенных вычислительных систем, в которых группа компьютеров согласованно выполняет набор взаимосвязанных задач, обмениваясь данными в автоматическом режиме. С другой стороны, компьютерные сети могут рассматриваться как средство передачи информации на большие расстояния, для чего в них применяются методы кодирования и мультиплексирования данных, получившие развитие в различных телекоммуникационных системах.

В конце 90-х гг. прошлого века особую популярность приобрело семейство Ethernet среди прочих технологий локальных сетей. В данное семейство вошли: классическая технология Ethernet 10 Мбит/с, Fast Ethernet 100 Мбит/с и Gigabit Ethernet 1000 Мбит/с. Популярность данного семейства объясняется низкой стоимостью оборудования Ethernet. Широкий диапазон скоростей позволяет рационально построить локальную сеть, применяя ту технологию семейства, которая в наибольшей степени отвечает задачам предприятия и потребностям пользователей. Важно также, что все технологии Ethernet очень близки друг к другу по принципам работы, что упрощает обслуживание и интеграцию этих сетей.

В настоящее время большое внимание уделяется качеству обслуживания сети (Quality of Service, QoS). Термин «качество обслуживания» означает способность коммуникационной системы обеспечивать то или иное качество услуг в зависимости от вида

передаваемых данных. Например, при передаче голоса или видео по коммуникационной сети необходимо обеспечить приоритетное продвижение таких пакетов, причем без потерь информации, с тем чтобы на приемном конце не ощущалось "выпадения" кадров изображения или прерывания голоса собеседника. Таким образом, целью качества обслуживания состоит в разработке методов качественной передачи трафика через сеть, для того чтобы обеспечить комфортную доставку трафика до конечного пользователя.

Одной из самых важных характеристик качества обслуживания является задержка доставки пакета. Задержка доставки пакета является статистической (вероятностной) величиной. Она состоит из двух составляющих: предсказуемой и непредсказуемой. Предсказуемая составляющая задержки определяется обработкой пакета, и зависит от его размера. Данная составляющая задержки не может быть минимизирована. Интерес же представляет непредсказуемая составляющая задержки пакета – джиттер, т.к. она определяется используемыми в оборудовании механизмами обеспечения качества обслуживания. В связи с этим качество обслуживания совершенствуется за счет реализации новых эффективных алгоритмов обслуживания и управления очередями, являющихся основой для предоставления гарантированного качества услуг связи. Это, естественно, требует проведения предварительного моделирования телекоммуникационных сетей с целью создания наиболее адекватной системы, способной соответствовать заданным и подразумеваемым требованиям.

## **1 Качество обслуживания QoS и его механизмы**

### **1.1 Общие сведения о качестве обслуживания QoS**

Quality of Service (QoS) — технология предоставления различным классам трафика различных приоритетов в обслуживании — качества обслуживания. Другими словами это технология, обеспечивающая выделение предпочтений высокоприоритетному сетевому трафику, устройству или критичному приложению. Под качеством обслуживания понимается способность сетевой инфраструктуры предоставлять улучшенное обслуживание определенному виду передаваемого трафика (пакетов) при помощи различных технологий [1].

Для большинства случаев качество связи определяется четырьмя параметрами:

- Скорость передачи информации, описывает номинальную пропускную способность среды передачи информации. Зависит от ширины полосы пропускания канала связи и отношения сигнал/шум.
- Задержка при передаче пакета, измеряется в миллисекундах.
- Колебания (дрожание) задержки при передаче пакетов — джиттер.
- Потеря пакетов. Определяет количество пакетов, потерянных в сети во время передачи [2, 3].

Существует две основные архитектуры QoS:

- интегрированное обслуживание;
- дифференцированные услуги.

Интегрированное обслуживание обеспечивает сквозное качество обслуживания, т.е. происходит резервирование ресурсов на всем пути

прохождения трафика. Для резервирования ресурсов используется протокол RSVP, гарантируя необходимую пропускную способность. Протокол резервирования ресурсов (RSVP) компания Microsoft предложила для QoS в Windows 2000. Существенным недостатком является постоянное резервирование ресурса, даже в том случае, если он не используется или используется не полностью.

Дифференциальный вид услуг предполагает наличие определенного набора средств классификации и механизмов организации очередей, обеспечивающих работу с приоритетами и выполняет две функции. Формирование трафика на границах сети – функции классификации, маркировки пакетов и управление интенсивностью. Политика пошагового обслуживания PNB (Per-Hop Behavior) включает функции распределения ресурсов и отбрасывания пакетов.

## 1.2 Механизмы реализации QoS

Существуют следующие механизмы, используемые для обеспечения заданного качества сервиса в сетях передачи данных:

- механизмы классификации и маркирования;
- механизмы обслуживания очередей;
- механизмы управления очередями;
- механизмы управления полосой пропускания.

Для обеспечения заданного уровня QoS в сетевом оборудовании одновременно используются несколько механизмов, из списка указанного выше [4]. В зависимости от трафика необходимо совмещать различные механизмы для обеспечения наиболее высокой производительности телекоммуникационного оборудования. В связи с этим любой показатель характеризующий производительность, зависит от используемых механизмов.

На рисунке 1 представлена структура коммутатора с поддержкой качества обслуживания [2].

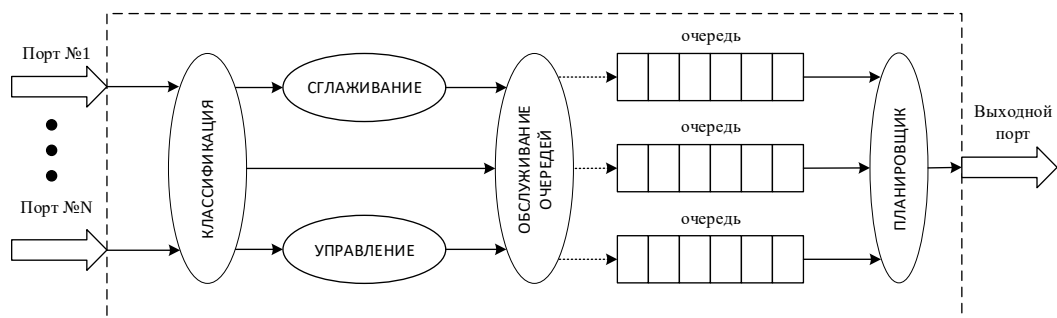


Рисунок 1 — Структура коммутатора с поддержкой качества обслуживания

Кадры входящего трафика поступают на входные порты коммутатора, далее проходят классификацию, обычно либо на основе типа класса которому соответствует кадр. Затем, в соответствии с присвоенным классом, кадры Ethernet могут проходить через функцию сглаживания или через функцию управления нагрузкой, которая определяет какое количество кадров сетевого трафика может быть пропущено для дальнейшей передачи для каждого класса. После пакеты перемещаются в очереди, которые могут быть определены на каждый поток или на каждый класс. Каждой очереди соответствует свой механизм управления очередями, например TailDrop или RED, при помощи которого определяется, поместить ли текущий кадр в очередь или же отбросить. Механизмы управления очередями могут использоваться также для заблаговременного сброса или маркировки кадров, чтобы заранее предупредить момент времени повышения нагрузки и не допустить переполнения очереди.

### 1.2.1 Механизмы классификации и маркирования

Классификация трафика может производиться на основании различных признаков: адресов источника/приемника, порт источника/приемника, различной комбинации этих и других признаков,

содержащихся в заголовках пакетов. Правила классификации определяются на основе политики администрирования сети. Она может производиться на каждом коммуникационном устройстве, в том числе и на узле-отправителе. Если классификация используется только в пределах одного коммуникационного узла (коммутатора или маршрутизатора), то помечать пакеты каким-либо образом не требуется. В случае использования информации о классе другими сетевыми устройствами, заголовки многих протоколов имеют отдельное поле для этих целей. Если специального поля для хранения класса отсутствует, разрабатывается дополнительный протокол, содержащий такое поле в своем заголовке (как это вышло с протоколом Ethernet – разработаны стандарты IEEE 802.1Q и 802.1p). Также необходимо учитывать, имеется ли централизованная политика отнесения пакета к определенному классу, так как одно устройство может не согласиться с классом, назначенным другим устройством. В этом случае оно будет классифицировать пакет в соответствии с политикой, принятой непосредственно на данном устройстве.

### 1.2.2 Механизмы обслуживания очередей

После классификации пакетов и помещения их в очередь, соответствующую классу, они ждут своей обработки устройством обслуживания очередей (планировщиком). Существует несколько основных алгоритмов обслуживания очередей:

- Приоритетного обслуживания
- Взвешенного обслуживания
- Комбинированного обслуживания

Особняком стоит алгоритм FIFO (First In – First Out). Все поступившие пакеты в случае перегрузки ставятся в одну очередь и обслуживаются в порядке поступления. В чрезмерной простоте и есть главный недостаток, так как пакеты чувствительного к задержкам трафика могут быть надолго задержаны пульсациями эластичного трафика.

Алгоритм приоритетного обслуживания применяется для преимущественной обработки одного класса трафика по сравнению с другими. Механизм приоритетного обслуживания основан на разделении всего сетевого трафика на небольшое количество классов и назначение каждому из них приоритета.

В сетевом устройстве, поддерживающем приоритетное обслуживание, имеется несколько очередей (буферов), по одной для каждого класса. В период перегрузок пакет помещается в очередь, которая соответствует его классу. Планировщик (обслуживающее устройство) выбирает из очередей пакеты по следующей схеме: если в более приоритетной очереди появляются пакеты, то планировщик в первую очередь передает их в выходную очередь до тех пор пока в более приоритетной очереди не закончатся все пакеты. Далее планировщик переходит к обслуживанию менее приоритетной очереди. Поэтому пакеты с низким приоритетом обрабатываются только в случае, когда пусты все более приоритетные очереди.

Приоритетное обслуживание очередей обеспечивает высокое качество обслуживания для пакетов из самой приоритетной очереди. Пакеты высоко приоритетного потока всегда получают, ту пропускную способность, которая им необходима. Уровень задержек таких пакетов минимален, но не нулевой, так как зависит от интенсивности потока и размера пульсации. Чем выше эти параметры, тем вероятнее возникновение очереди, образованной пакетами этого потока. Трафик всех

остальных приоритетных классов почти прозрачен для трафика самого приоритетного класса. Качество их обслуживания будет ниже, при этом уровень снижения может быть довольно существенным и зависит от характеристик высоко приоритетного трафика. Например, в период высокой интенсивности трафика высшего приоритета, обслуживание остального может приостановиться, поэтому приоритетное обслуживание обычно применяется для класса трафика, чувствительного к задержкам, но имеющего небольшую интенсивность (например телефонии).

Алгоритм взвешенного обслуживания разработан с целью предоставления всем классам трафика определенного минимума пропускной способности. Под весом класса понимается процент предоставляемой ему пропускной способности от полной пропускной способности ресурса, предоставляемый в случае перегрузок (для входного потока таким ресурсом является процессор, а для потока после выполнения коммутации -выходной интерфейс). Трафик делится на несколько классов и для каждого класса выделяется отдельная очередь для пакетов. Очереди обслуживаются последовательно и циклически, и в каждом цикле обслуживания из каждой очереди выбирается такой объем байтов, который соответствует весу данной очереди. Допустим, у нас имеются четыре очереди, которым выделено 10%, 20%, 30% и 40% от пропускной способности интерфейса соответственно. Если цикл просмотра очередей равен одной секунде, а скорость выходного интерфейса составляет 100 Мбит/с, то в случае перегрузки первой очереди выделяется 100 мс времени и выбирается 10 Мбит данных, из второй — 20 Мбит, из третьей — 30 Мбит и из четвертой — 40 Мбит.

В результате каждому классу предоставляется гарантированный минимум пропускной способности, что является более желательным результатом по сравнению с подавлением трафика низших классов в алгоритме приоритетного обслуживания. Так как данные выбираются

пакетами, а не байтами, то реальное распределение пропускной способности между классами будет несколько отличаться от заданного. Чем больше время цикла, тем точнее соблюдаются соотношения между классами трафика, так как влияние отдельных пакетов уменьшается. С другой стороны, длительный цикл приводит к большим задержкам при передаче, потому что планировщик возвращается к каждой очереди не чаще чем один раз за цикл (в нашем примере — не чаще одного раза в секунду). Поэтому при выборе времени цикла не следует забывать про задержки, которые могут возникнуть в этом случае.

На уровень задержек и вариаций задержек пакетов для каждого отдельного класса трафика при взвешенном обслуживании в большой степени влияет коэффициент использования. В этом случае коэффициент использования подсчитывается как отношение интенсивности входного трафика конкретного класса к пропускной способности, выделенной для этого же класса.

### 1.2.3 Механизмы управления очередью

Механизмы управления очередью выполняют следующие функции:

- выборка очередного кадра из очереди для последующей обработки;
- выборка кадров, которые будут отброшены при переполнении очереди.

Механизмы управления очередями взаимосвязаны с механизмами обслуживания очередей, из-за этого иногда алгоритмы управления очередями включают как подмножество механизмов обслуживания очередей. Но такая классификация не является корректной, так как механизмы обслуживания очередей отвечают за распределение обслуживания нескольких очередей, а механизмы управления очередями

работают с одной очередью. Совмещение различных алгоритмов этих двух механизмов дают различный результат при одном и том же входном трафике.

#### 1.2.4 Механизмы управления полосой пропускания

Механизмы управления полосой пропускания выполняют резервирование определенного процента сетевых ресурсов, необходимого информационному потоку обмена данными. Применение механизма управления полосой пропускания наиболее критично, при передаче трафика реального времени.

Появление данных механизмов связано с тем, что традиционно трафик передавался по методу «best effort» – «с максимальными усилиями». То есть сеть старается обработать поступающий трафик как можно быстрее, но при этом никаких гарантий относительно результата своих усилий не дает. Не гарантируется ни проверка сети обеспечить поток данных сетевыми ресурсами, ни приоритезация. Другими словами, безотносительно к какому типу трафика относятся информационные пакеты, они обрабатываются по принципу «первый пришел - первый получил обслуживание». Очевидно, что эта модель не подходит для передачи трафика со специфическими требованиями к задержке, производительности или надежности передачи данных. Для перехода к новым приложениям на сетях с коммутацией пакетов нужен свой механизм обеспечения качества передачи. В такой структуре сети пакеты, чувствительные к задержке, должны отличаться от пакетов данных, не требующих срочной обработки, и это отличие должно фиксироваться узлами сети.

### 1.3 Особенности механизма управления очередями

Работа механизма управления очередями оценивается, как его способность эффективно контролировать сетевой трафик во время возникновения перегрузок сети. В настоящее время существуют несколько таких алгоритмов. Все они подразделяются на два типа: пассивные и активные алгоритмы управления очередями.

На рисунке 2 представлена классификация механизмов управления очередями.



Рисунок 2 — Классификация механизмов управления очередями

Наиболее простой пассивный алгоритм управления очередями коммутаторов — это алгоритм TailDrop (Отбрасывание конца очереди). Алгоритм TailDrop может обслуживать только сетевой трафик, неразделенный по типам классов пакетов.

Принцип его работы заключается в следующем. Определяется максимальная длина очереди (в пакетах или в байтах), все вновь прибывшие кадры трафика определяются в конец очереди, и так пока очередь не заполнится до максимального заданного размера, все последующие кадры, поступившие в коммутатор отбрасываться, пока

очередь не освободится, и не будет хватать места, для поступления новых кадров сетевого трафика.

Главный недостаток данного алгоритма заключается в следующем, что если использовать протокол ТСР и возникнет ситуация переполнения очереди, то кадры трафика начнут теряться. При этом на стороне источников трафика блоки ТСР в рабочих станциях поймут, что в сети возникла перегрузка и необходимо замедлить передачу кадров. При переполнении очереди возможны ситуации, когда несколько идущих подряд через коммутатор кадров будут отброшены — в результате несколько источников решат замедлить передачу. Затем как только сеть немного разгрузится и коммутатор снова начнет пропускать кадры Ethernet источники снова возобновят передачу данных с прежней скоростью, тем самым снова возникнет ситуация переполнения очереди и отбрасывания кадров сети.

Также данный алгоритм может вызвать так называемый эффект «локаута». Это когда коммутатор начинает обрабатывать либо один поток данных, либо несколько потоков (например, видео-потоков), тем самым прекращая обработку кадров других потоков.

Алгоритм Tail Drop неразумно использует память коммутатора: очереди на длительные периоды времени оказываются заполненными. Это происходит из-за того, что алгоритм сообщает только о том, что очередь заполнена. Увеличение размера очереди приведет к увеличению времени доставки данных от одного источника до другого, что не является выходом из данной ситуации. Поэтому разумно, чтобы размер очереди в коммутаторах был небольшим. Однако, известно, что трафик в сети неравномерен, и поэтому коммутатор должен иметь такой размер буфер, чтобы «амортизировать» неравномерность трафика.

Для противодействия подобным ситуациям был разработан метод случайного раннего обнаружения перегрузок RED, отбрасывающий пакеты с вероятностью, линейно увеличивающейся с ростом средней длины очереди.

Алгоритм RED [5] отслеживает средний размер очереди буфера и отбрасываемых пакетов, основываясь на статистической вероятности. Если размер занимаемой очереди находится ниже определенного порога, то все кадры сети пропускаются в обычном режиме. Когда очередь превышает данный порог и начинает постепенно расти, то вероятность отбрасывания кадров соответственно тоже начинает расти. И когда очередь полностью заполняется, то все поступающие на вход коммутатора пакеты отбрасываются. Другими словами, когда наполненность очереди коммутатора превышает какое-либо пороговое значение, вероятность отбрасывания входящего кадра зависит от степени превышения этого порогового значения.

RED становится намного эффективнее других алгоритмов в случае малых размеров очередей, а также при «взрывном» характере трафика.

Метод RED оценивает среднее значение очереди путём вычисления экспоненциально взвешенного скользящего среднего *avg* по рекуррентной формуле (1).

$$avg = (1 - w)avg + w * que, \quad (1)$$

где *que* – текущее значение очереди, а *w* – весовой коэффициент с рекомендованным значением 0,002 [6].

Если очередь пуста, то делается оценка среднего значения очереди *avg* в период отсутствия пакетов по следующей формуле:

$$avg = avg * (1 - n)^n, \quad (2)$$

где  $n$  — предположительное количество пакетов поступивших в очередь. Без такого предположения оценка  $avg$  будет неправильная. Если пакет поступает в момент, когда очередь пуста, то  $n$  рассчитывается по формуле:

$$n = \frac{t_{\text{тек}} - t_0}{i}, \quad (3)$$

где  $t_{\text{тек}}$  — текущее время,  $t_0$  — момент времени, когда очередь стала пустой, а  $i$  — средний размер пакета.

Вероятность сброса пакетов линейно изменяется в интервале от 0 до заданного значения максимальной вероятности по формуле:

$$P_{\text{отб}} = \frac{\max_{\text{вер}} * (avg - \min_{\text{que}})}{(\max_{\text{que}} - \min_{\text{que}})}, \quad (4)$$

где  $\min_{\text{que}}$  — минимальный порог среднего значения очереди до которого не происходит сброс,  $\max_{\text{que}}$  — максимальный порог, после которого сбрасываются все пакеты, а  $\max_{\text{вер}}$  — максимальное значение вероятности сброса. График вероятности сброса пакета приведён на рисунке 3.

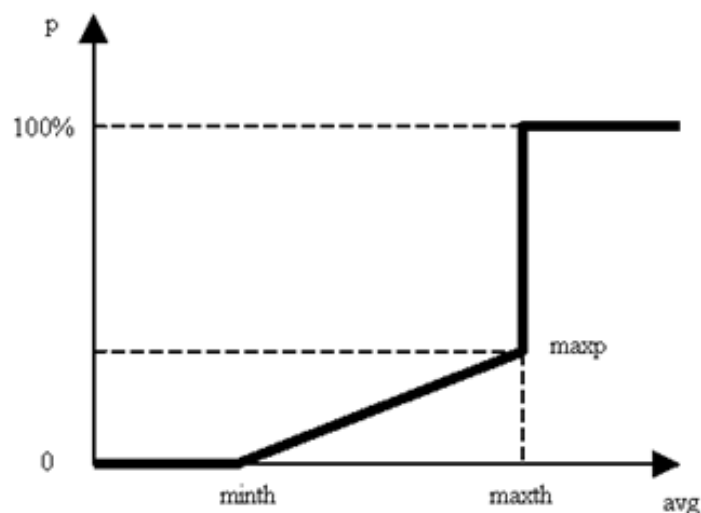


Рисунок 3 — Зависимость вероятности сброса от усреднённой длины очереди для метода RED

В современных коммутаторах применяется модификация WRED [7] (Weighted RED) — взвешенный RED. Оно является расширением

алгоритма Random early detection (RED), где одна очередь может иметь несколько различных пороговых размеров очереди. Пороговый размер каждой очереди связан с конкретной реализацией QoS: IP precedence или DSCP (для DiffServ) и тому подобное.

Для адаптивной подстройки параметров был предложен механизм Adaptive RED [8], способный динамически изменять  $max_{вер}$  в зависимости от загрузки очереди и автоматически рассчитывающий значение  $max_{вер}$  и весовой коэффициент усреднения  $w$  по формуле:

$$w = 1 - e^{1/v}, \quad (5)$$

где  $v$  — скорость в канале в пакетах в секунду.

Для динамического регулирования параметра  $max_{que}$  вводится аддитивный коэффициент  $\alpha = \min(0.01, max_p/4)$  и мультипликативный коэффициент  $\beta = 0.9$ . Через каждый заданный интервал измерения среднее значение очереди  $avg$  сравнивается с некоторым уровнем  $l$ :

$$l = [min_{que} + 0.4 * (max_{que} - min_{que}), min_{que} + 0.6 * (max_{que} - min_{que})], \quad (6)$$

и если выполняется условие  $(avg > l) \text{ and } (max_{вер} \leq 0.5)$ , то  $max_{вер}$  увеличивается по формуле:

$$max_{вер} = max_{вер} + \alpha \quad (7)$$

В противном случае, если  $(avg < l) \text{ and } (max_{вер} \geq 0.01)$ , то  $max_p$  уменьшается по формуле

$$max_{вер} = max_{вер} + \beta \quad (8)$$

Таким образом, параметры метода RED динамически адаптируются в зависимости от уровня интенсивности нагрузки сетевого трафика.

## **2 Моделирование алгоритмов управления очередями**

### **2.1 Выбор средства моделирования**

В настоящее время существует множество программ, предназначенных для создания и исследования моделей. Основную часть систем моделирования занимают программы, использующие математический аппарат и предназначенные для сбора статистических данных. Как правило, такие системы не являются узконаправленными, а возможность моделирования является лишь одной из множества доступных функций. К таким системам можно отнести программы MatLab и GPSS.

Так же существуют специальные системы, предназначенные только для моделирования определенных параметров исследуемых систем. К таким системам можно отнести аппарат сетей Петри, предназначенный для моделирования поведенческих свойств исследуемой системы.

Рассмотрим более подробно различные инструментальные средства моделирования.

#### **2.1.1 MatLab**

MATLAB (сокращение от англ. «Matrix Laboratory») — пакет прикладных программ для решения задач технических вычислений и одноимённый язык программирования, используемый в этом пакете[9].

MATLAB представляет собой основу всего семейства продуктов MathWorks и является главным инструментом для решения широкого спектра научных и прикладных задач, в таких областях как: моделирование объектов и разработка систем управления, проектирование коммуникационных систем, обработка сигналов и изображений, измерение

сигналов и тестирование, финансовое моделирование, вычислительная биология и др.

### 2.1.2 GPSS

Система GPSS World, разработанная компанией Minuteman Software (США), – это мощная среда компьютерного моделирования общего назначения, разработанная для профессионалов в области моделирования. Это комплексный моделирующий инструмент, охватывающий области как дискретного, так и непрерывного компьютерного моделирования, обладающий высочайшим уровнем интерактивности и визуального представления информации.

Использование GPSS World дает возможность оценить эффект конструкторских решений в чрезвычайно сложных системах реального мира.

Недостатками является то, что в программе на языке GPSS достаточно сложно представить непосредственно процессы обработки данных на уровне алгоритмов. Кроме того, модель представляет собой программу, а значит не имеет графической интерпретации, что затрудняет процесс разработки модели и снижает наглядность модели в целом.

### 2.1.3 Сети Петри

Сети Петри — математический аппарат для моделирования динамических дискретных систем. Впервые были описаны Карлом Петри в 1962 году в докторской диссертации «Связь автоматов».

Сеть Петри представляет собой двудольный ориентированный граф, состоящий из вершин двух типов — переходов и позиций, соединённых между собой дугами. Вершины одного типа не могут быть соединены

непосредственно. В позициях могут размещаться маркеры, способные перемещаться по сети [10].

Событием называют срабатывание перехода, при котором маркеры из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно, либо разновременнo, при выполнении некоторых условий.

Сети Петри в основном используются для моделирования систем с параллельными взаимодействующими компонентами.

При выборе системы моделирования для выполнения данной выпускной работы преимущество было отдано программам, использующим аппарат сетей Петри. Это обусловлено тем, что сети Петри – это универсальная алгоритмическая система, позволяющая описывать произвольные объекты.

Как было сказано ранее, существует множество различных видов сетей Петри. Каждый вид сети Петри в отдельности может быть использован для разработки простых моделей. Для описания сложных моделей были разработаны специальные средства моделирования, которые соединяют в себе несколько видов сетей Петри. В качестве основного вида сети Петри в данных пакетах моделирования используются цветные сети. Это связано с тем, что цветные сети Петри являются мощным аппаратом, позволяющим создавать довольно сложные модели.

Для выполнения моделирования и исследования полученных моделей в рамках данной выпускной квалификационной работы был выбран свободно распространяемый пакет моделирования CPN Tools, использующий математический аппарат иерархических временных цветных сетей Петри. Цветные сети Петри позволяют описывать алгоритмы, зависящие от значения жетона, передаваемого в сети.

Иерархические сети Петри позволяют описывать объекты любой сложности. А временные сети Петри делают возможным моделирование динамических характеристик объектов.

Пакет CPN Tools совмещает в себе сети Петри и функциональный язык моделирования CPN ML, который является расширением языка Standard ML, и позволяет подробно исследовать различные вопросы поведения сложных компьютерных сетей [11, 12].

## 2.2 Построение моделей в среде CPN Tools

При построении сети Петри, моделирующей работу коммутатора Ethernet, следует учитывать, что модель, построенная с использованием аппарата сетей Петри, не является моделью устройства, а описывает термины и протокол работы устройства.

Структура коммутатора представлена на рисунке 4 [13]. Кадр поступает во входной порт. После этого кадр проходит через таблицу коммутации и направляется в буфер выходного порта и затем в сам выходной порт коммутатора.

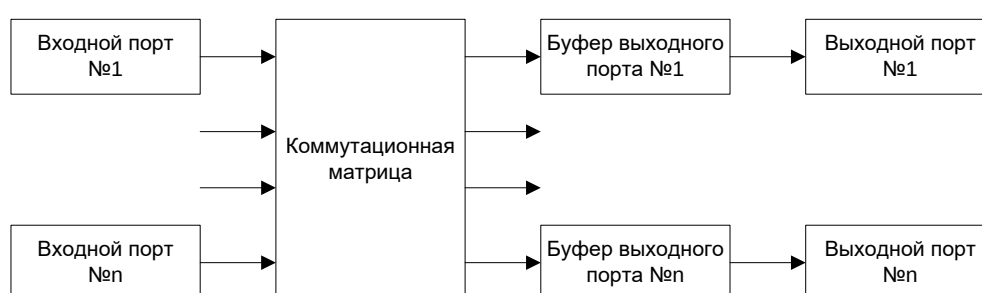


Рисунок 4 — Структура коммутатора

Известны работы [14, 15], описывающие построение функциональной модели коммутационной матрицы, входных и выходных портов. Таким образом, можно определить конкретную часть коммутатора,

непосредственно отвечающую за управление очередями — буфер выходного порта.

За основу при разработке модели буфера выходного порта были взяты работы [16, 17].

Обобщенная структура разработанной модели буфера выходного порта представлена на рисунке 5.

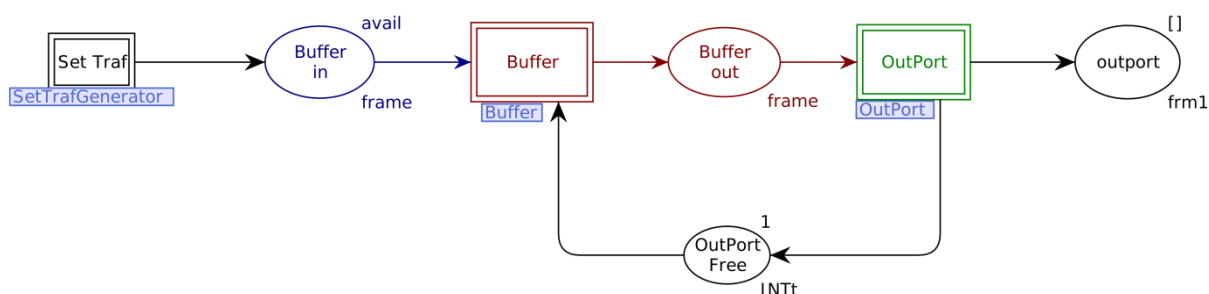


Рисунок 5 — Обобщенная структура модели буфера выходного порта

Подсеть **SetTrafGenerator** описывает функцию считывания сетевого трафика из файла.

Подсеть **Buffer** реализует функции работы буфера выходного порта.

Позиции "Buffer\_in" и "Buffer\_out" являются соответственно входом и выходом буфера выходного порта.

Подсеть **OutPort** реализует функции выходного порта коммутатора.

Для описания модели буфера выходного порта коммутатора использовались следующие множества цветов.

Составной цвет *frm1* описывает структура кадра Ethernet и содержит в себе следующие поля: порядковый номер кадра, MAC-адреса источника и приемника, приоритет кадра, размер передаваемых данных, два поля временных задержек. Более подробно цвет, описывающий структуру кадра, рассмотрен в разделе 3.1.

*colset frm1 = product INT\*MAC\*MAC\*QoS \*INT\*INTt\*INTt declare input\_ms*

Дополнительно был создан составной цвет *frm* аналогичный цвету *frm1* за исключением отсутствия одного поля задержки.

*colset frm = product INT\*MAC\*MAC\*QoS \*INT\*INTt*

Цвет *frame* описывает нахождение в позиции фишки *f* цвета *frm*, описывающей Ethernet кадр, либо фишки цвета *avail*, показывающей, что позиция пуста.

*colset frame = union f1:frm + avail timed*

Составной цвет *que* был создан для отслеживания размера буфера коммутатора:

*colset que = product INT\*INT\*INT\*INTt\*INTt timed*

Цвет *LBuffer* описывает буфер коммутатора:

*colset LBuffer = list frm*

Цвет *Bitsize* описывает размер данных в битах и может принимать значения от нуля до тринадцати тысяч. Была создана переменная этого цвета *szfrm*, описывающая размер данных кадра сетевого трафика.

*colset BitSize = int with 0..13000*

Далее рассмотрим реализацию каждой подсети подробнее.

Подсеть *SetTrafGenerator* представлена на рисунке 6.

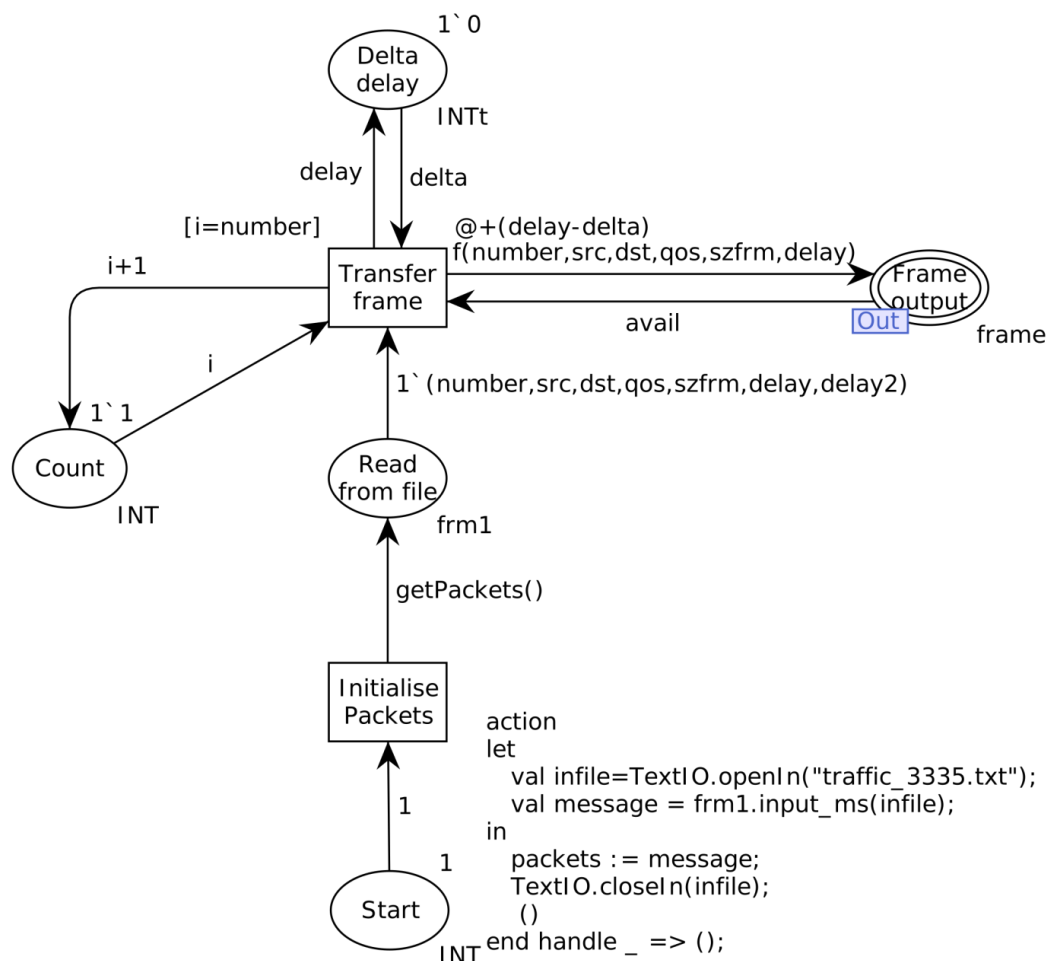


Рисунок 6 — Подсеть SetTrafGenerator

В таблице 1 представлено описание всех позиции и переходов подсети SetTrafGenerator.

Таблица 1 — Описание позиций и переход подсети SetTrafGenerator

| Обозначение        | Тип     | Описание                                                           |
|--------------------|---------|--------------------------------------------------------------------|
| Start              | позиция | стартовая позиция                                                  |
| Read from file     | позиция | хранит фишки считанных кадров из файла в формате цвета frm1        |
| Count              | позиция | текущий номер преобразованного кадра                               |
| Delta delay        | позиция | задержка между поступлениями кадров в сеть                         |
| Frame output       | позиция | выходная позиция подсети                                           |
| Initialise Packets | переход | считывание данных из файла                                         |
| Transfer frame     | переход | преобразование фишки цвета frm1 в цвет frame и добавление задержки |

Данные в файле хранятся в определенном виде, соответствующем структуре цвета frm1. Ниже представлен фрагмент такого файла.

```
1`(1,3,5,7,512, 296,0)++
1`(2,2,9,7,2464, 7389,0)++
1`(3,1,3,7,512, 20178,0)++
```

При срабатывании перехода "Initialise Packets" данные считываются из файла и помещаются в позицию "Read from file" в формате цвета frm1. Далее при срабатывании перехода "Transfer frame" в позиции "Count" увеличивается счетчик преобразованных фишек на единицу, в позицию "Delta delay" помещается фишка со значением поля delay текущего кадра, а берется фишка delta с предыдущим значением позиции. Срабатывание перехода происходит с задержкой равной (delay – delta). Это сделано для того, чтобы далее в сеть кадры поступали в порядке очереди своих номеров и со своей временной задержкой. На выходе подсети SetTrafGenerator получают фишки цвета frame, обозначающие кадры сетевого трафика.

Далее на рисунке 7 представлена подсеть OutPort.

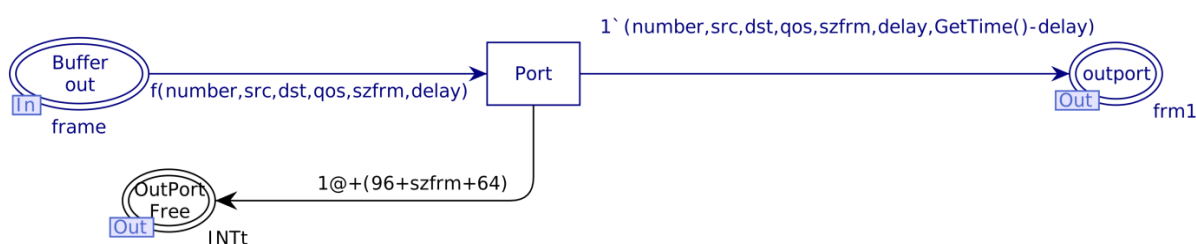


Рисунок 7 — Подсеть Outport

В таблице 2 представлено описание всех позиции и переходов подсети Outport.

Таблица 2 — Описание позиций и переход подсети OutPort

| Обозначение | Тип     | Описание                                                    |
|-------------|---------|-------------------------------------------------------------|
| Buffer Out  | позиция | входная позиция подсети цвета frame                         |
| outport     | позиция | выходная позиция подсети цвета frm1                         |
| OutPortFree | позиция | формирование фишки, означающей освобождения выходного порта |
| Port        | переход | преобразование фишек из цвета frame в цвет frm1             |

В данной подсети фишка цвета frame преобразуется в фишку цвета frm1, т.е. в структуру кадра добавляется еще одно поле временной задержки, которое рассчитывается, как "Текущее время – Время поступления кадра в сеть" и является временем пребывания кадра в буфере выходного порта коммутатора. При срабатывании перехода "Port" происходит преобразование цветов фишек и в позиции "OutPortFree" формируется фишка с задержкой равной  $96 + \text{размер текущего кадра в битах} + 64$ ". Данная фишка является сигналом освобождения выходного порта коммутатора для следующего кадра в очереди буфера выходного порта.

Подсеть Buffer представлена на рисунке 8.

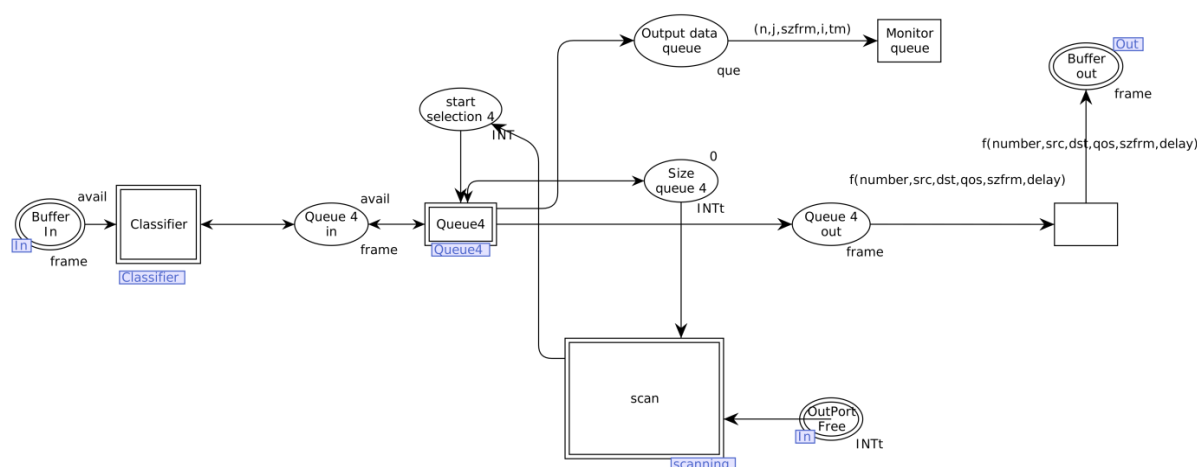


Рисунок 8 — Подсеть Buffer

Данная подсеть включает в себя еще несколько подсетей. Ниже представлено их описание

Подсеть Classifier классифицирует поступающие на вход буфера коммутатора кадры и в зависимости от класса кадра распределяет по каналам. В данной работе исследовались механизмы управления очередями, поэтому было принято решение оставить один канал и все кадры пускать через него.

Подсеть Scan формирует фишки выборки кадра из очереди буфера выходного порта в зависимости от наличия в очереди кадров и состояния позиции "OutPortFree".

В подсети Queue4 реализован алгоритм управления очередью коммутатора. Так как в рамках данной выпускной квалификационной работы было реализовано несколько алгоритмов управления очередью, основная структура модели буфера выходного порта использовалась одна, а изменялась лишь эта подсеть в зависимости от выбранного алгоритма. Реализация каждого алгоритма будет описана в следующих подразделах.

В позиции "Size queue 4" хранится значение размера очереди буфера коммутатора.

Также был создан монитор, который отслеживает значение очереди в буфере коммутатора и записывает эту информацию в файл в формате "4, знак операции (1 – если пакет поступил в очередь, 0 – если пакет удален из очереди), размер текущего кадра, размер текущей очереди, время изменения очереди".

Позиции "Buffer In" и "Buffer Out" являются соответственно входом и выходом буфера выходного порта.

Позиции "Queue 4 in" и "Queue 4 out" являются соответственно входом и выходом очереди буфера выходного порта.

Позиция "Start selection 4" является сигналом начала выборки кадров из очереди буфера выходного порта.

Подсеть Classifier представлена на рисунке 9.

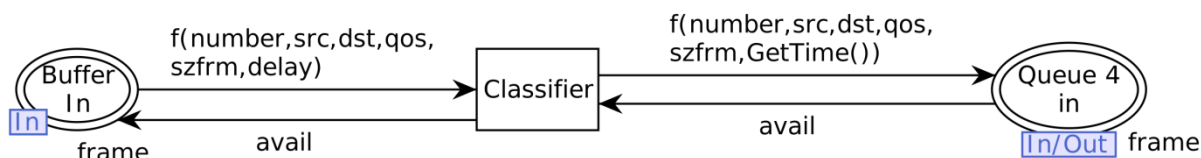


Рисунок 9 — Подсеть Classifier

В таблице 3 представлено описание всех позиции и переходов подсети Classifier.

Таблица 3 — Описание позиций и переход подсети Classifier

| Обозначение | Тип     | Описание                                                         |
|-------------|---------|------------------------------------------------------------------|
| Buffer In   | позиция | входная позиция подсети цвета frame, вход буфера выходного порта |
| Queue 4 in  | позиция | выходная позиция подсети цвета frm1, вход очереди буфера         |
| Classifier  | переход | классификация кадров                                             |

Как уже было сказано ранее, в модели буфера коммутатора использовался один канал, поэтому в этой подсети фишка кадра просто передается на вход очереди буфера коммутатора. Так как в модели возможен не один канал, а несколько, была выделана данная подсеть для расширяемости системы.

Подсеть Scan представлена на рисунке 10.

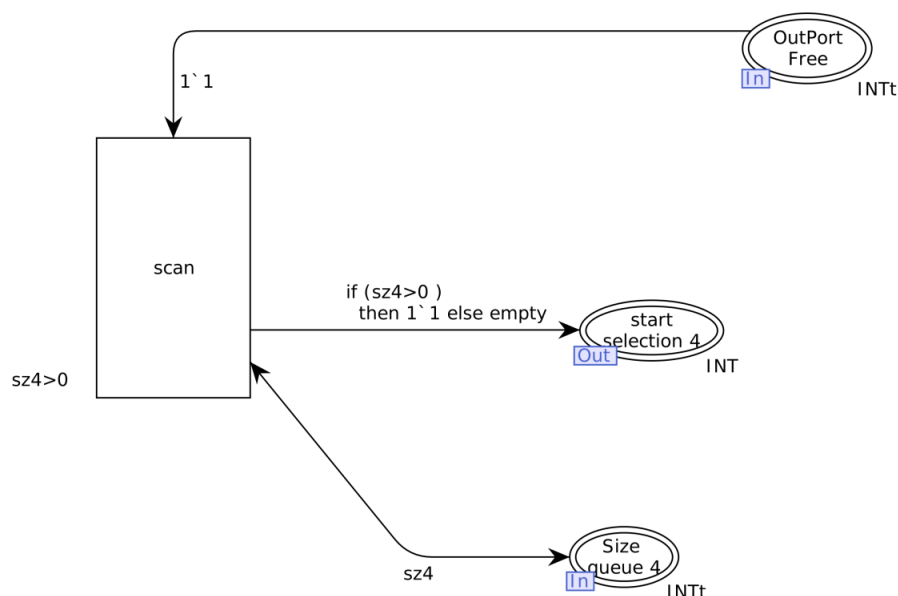


Рисунок 10 — Подсеть Scan

При наличии в очереди буфера коммутатора кадров срабатывает переход "scan" и с задержкой, сформированной в позиции "OutPort Free", формируется фишка выборки кадра из очереди.

### 2.2.1 Описание модели алгоритма управления очередью Tail Drop

На рисунке 11 представлена подсеть, реализующая алгоритм пассивного управления очередью Tail Drop.

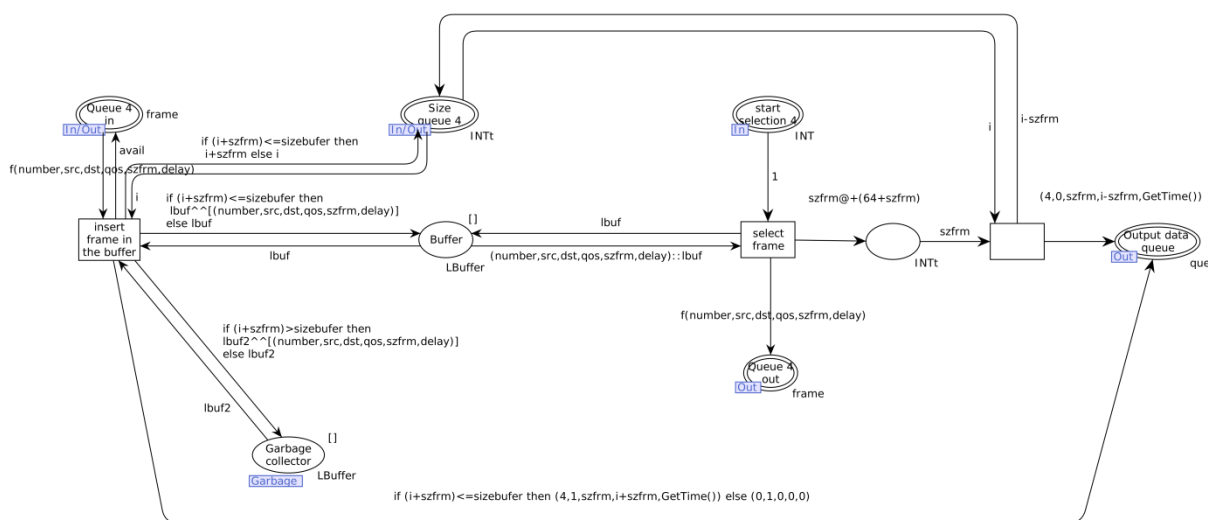


Рисунок 11 — Подсеть, реализующая алгоритм пассивного управления очередью Tail

Drop

В таблице 4 представлено описание всех позиции и переходов подсети TailDrop.

Таблица 4 — Описание позиций и переход подсети TailDrop

| Обозначение                | Тип     | Описание                                                                   |
|----------------------------|---------|----------------------------------------------------------------------------|
| Queue 4 in                 | позиция | входная позиция подсети цвета frame, вход очереди буфера выходного порта   |
| Queue 4 out                | позиция | выходная позиция подсети цвета frame, выход очереди буфера выходного порта |
| Buffer                     | позиция | очередь буфера                                                             |
| Garbage Collector          | позиция | сборщик мусора, хранит отброшенные кадра                                   |
| Size queue 4               | позиция | размер очереди буфера                                                      |
| Start selection 4          | позиция | сигнал выборки кадра из очереди                                            |
| Output data queue          | позиция | выходные данные о состоянии очереди буфера                                 |
| insert frame in the buffer | переход | запись кадра в очередь буфера                                              |
| select frame               | переход | выборка кадра из очереди буфера                                            |

На вход очереди буфера поступают фишки кадров Ethernet. При срабатывании перехода "insert frame in the buffer" размер очереди в буфере плюс размер данных текущего кадра сравнивается с максимальным размером очереди. Если размер текущей очереди и текущего кадра больше максимального значения размера очереди буфера, то кадр отбрасывается и помещается в позицию "Garbage collector", так называемый сборщик мусора. Иначе кадр помещается в очередь буфера, а также в позицию "Size queue 4" записывается новое значение размера очереди буфера. При выборке кадра из очереди буфера, в позицию "Size queue 4" записывается размер очереди буфера с учетом удаленного кадра.

Как только в буфере появляется кадр и выходной порт свободен, то по сигналу из позиции "Start selection 4" срабатывает переход "select frame". Кадр извлекается из буфера и передается на выход очереди буфера.

Для отслеживания размера очереди буфера был создан монитор "Monitor queue", который учитывает и сохраняет данные о размере очереди в файл.

Монитор написан на языке моделирования CPN ML. Ниже представлен фрагмент монитора.

```
fun obs (bindelem) =  
  let  
    fun      obsBindElem      (Buffer'Monitor_queue      (1,  
{i,j,szfrm,n,tm})) =  
      Int.toString(n) ^ ", " ^ Int.toString(j) ^ ", " ^ Int.toString(szfrm  
) ^ ", " ^ Int.toString(i) ^ ", " ^ Int.toString(tm) ^ "\n"  
      | obsBindElem _ = ""  
  in  
    obsBindElem bindelem  
  end
```

Если кадр помещается в очередь буфера, то в файл записывается строка "(4,1,szfrm,i+szfrm,GetTime())", где переменная szfrm - это размер текущего кадра, i+szfrm - это размер очереди плюс размер текущего кадра, а GetTime() функция записи текущего времени в файл. Если же кадр выбирается из очереди буфера, то строка "(4,0,szfrm,i-szfrm,GetTime())". При отбрасывании кадра на входе в очередь буфера, в файл записывается строка "(0,1,0,0,0)".

### 2.2.2 Описание модели активного алгоритма управления очередью RED

На рисунке 12 представлена подсеть, реализующая алгоритм управления очередью RED с так называемой обработкой кадров "на лету".

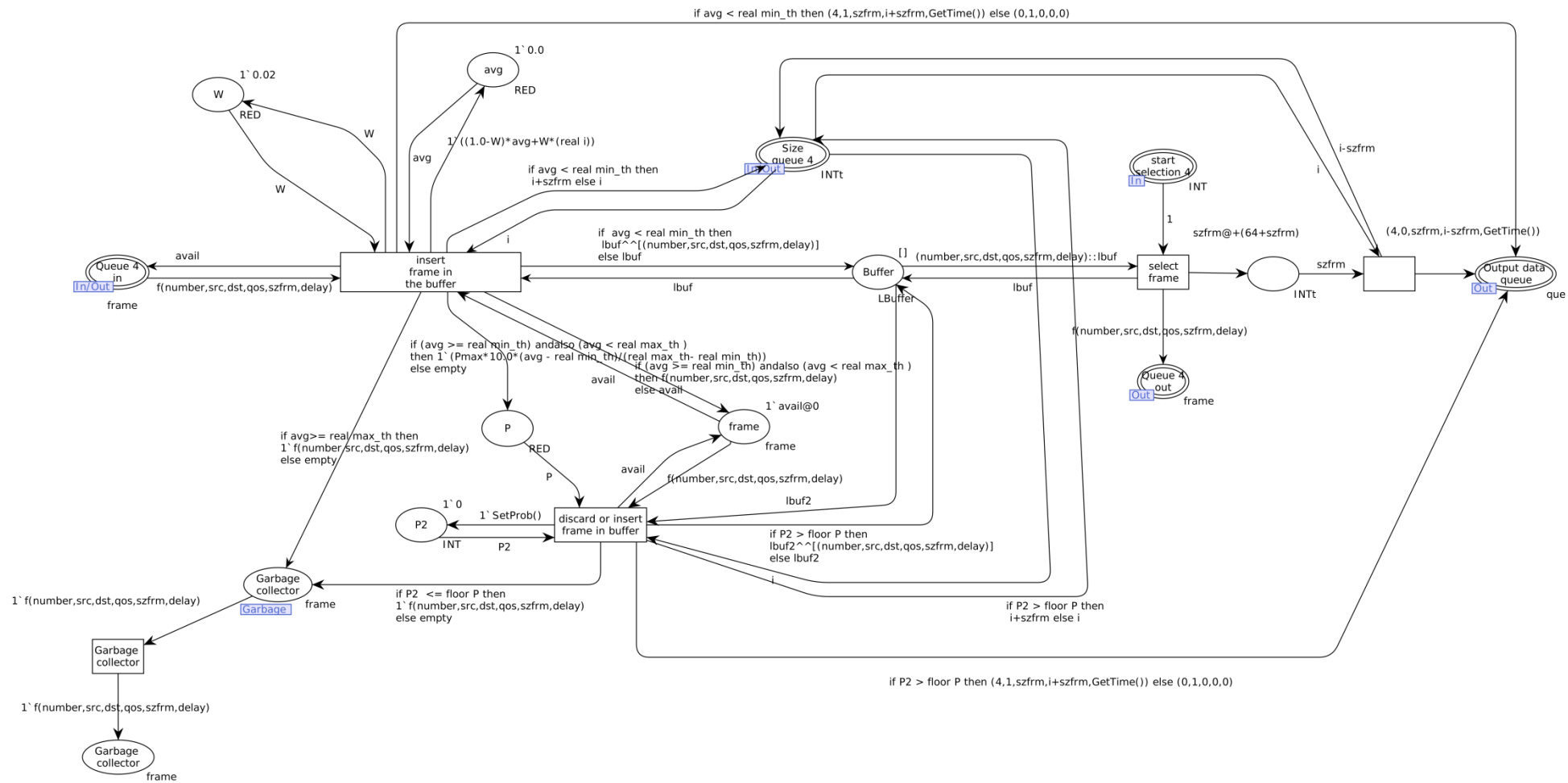


Рисунок 12 — Подсеть, реализующая алгоритм управления очередью RED

Главным отличием и преимуществом алгоритма RED от алгоритма Tail Drop является заблаговременное отслеживание и предотвращение перегрузок в сети до переполнения очереди буфера коммутатора. Кадры начинают отбрасываться из буфера с определенной вероятностью, как только размер очереди переступит через определенный барьер, задаваемый пользователем.

Дополнительно для реализации модели алгоритма RED был создан цвет RED, который принимает значения чисел с плавающей точкой.

*colset RED = real;*

Вероятность отбрасывания кадра вычисляется по формуле:

$$P_a = \begin{cases} 0, & \text{если } avg < min_{th} \\ max_p * \frac{avg - min_{th}}{max_{th} - min_{th}}, & \text{если } min_{th} \leq avg < max_{th}, \\ 1 & \text{если } avg \geq max_{th} \end{cases} \quad (9)$$

где  $max_p$  - это граничная вероятность, определяющая часть отбрасываемых кадров при достижении среднего значения очереди максимального порога,  $avg$  - это среднее значение длины очереди буфера, а  $max_{th}$  и  $min_{th}$  - это соответственно максимальное и минимальное пороговые значения.

В таблице 5 представлено описание всех позиции и переходов подсети RED.

Таблица 5 — Описание позиций и переход подсети RED

| Обозначение       | Тип     | Описание                                                                   |
|-------------------|---------|----------------------------------------------------------------------------|
| Queue 4 in        | позиция | входная позиция подсети цвета frame, вход очереди буфера выходного порта   |
| Queue 4 out       | позиция | выходная позиция подсети цвета frame, выход очереди буфера выходного порта |
| Buffer            | позиция | очередь буфера                                                             |
| Garbage Collector | позиция | сборщик мусора, хранит отброшенные кадры                                   |

Продолжение таблицы 5

|                            |         |                                            |
|----------------------------|---------|--------------------------------------------|
| Size queue 4               | позиция | размер очереди буфера                      |
| Start selection 4          | позиция | сигнал выборки кадра из очереди            |
| Output data queue          | позиция | выходные данные о состоянии очереди буфера |
| W                          | позиция | весовой коэффициент очереди                |
| avg                        | позиция | среднее значение очереди буфера            |
| P                          | позиция | вероятность отбрасывания пакетов           |
| insert frame in the buffer | переход | запись кадра в очередь буфера              |
| select frame               | переход | выборка кадра из очереди буфера            |

Каждый раз при срабатывании перехода "insert frame in the buffer" вычисляется среднее значение очереди буфера по формуле (1).

Весовой коэффициент очереди  $W$  задается в позиции "W". Размер очереди берется из позиции "Size queue 4". Текущее среднее значение длины очереди буфера хранится в позиции avg. Максимальное и минимальное пороговые значения задаются константами:

*val max\_th=70000*

*val min\_th=30000*

Граничная вероятность, определяющая часть отбрасываемых кадров при достижении среднего значения очереди максимального порога  $max_p$  также задается константой:

*val Pmax = 0.9*

При поступлении кадра на вход очереди буфера коммутатора вычисляется текущее значение средней длины очереди и в зависимости от этого значения кадр, как показано на рисунке 11, либо отбрасывается в позицию "Garbage Collector", либо перемещается в очередь буфера коммутатора в позицию "Buffer". Есть еще третий вариант перемещения кадра, когда среднее значение очереди буфера находится в промежутке

между минимальным и максимальным порогами очереди буфера. В таком случае при срабатывании перехода "insert frame in the buffer" вычисляется вероятность отбрасывания пакета по формуле (4), значение помещается в позицию P. Текущий кадр помещается в позицию "RED". Затем в позиции "P2" с помощью функции  $fun SetProb() = discrete(0,9)$  вычисляется вероятность P2 по равномерному закону от 0 до 9. Данное значение P2 сравнивается со значением вероятности отбрасывания кадра P. И если  $P2 \leq P$ , то кадр отбрасывается в позицию "Garbage Collector", иначе кадр помещается в очередь буфера коммутатора в позицию "Buffer".

Реализация выборки кадров из очереди буфера и монитора очереди буфера схожа с описанной в подразделе 2.2.1.

### 2.2.3 Описание модели активного алгоритма управления очередью RED с полной буферизацией

Главным отличием от обычной реализации алгоритма RED, описанной в предыдущем подразделе является то, что в алгоритме с полной буферизацией учитывается время передачи кадра со входа в очередь буфера. Пока текущий кадр полностью не переместился в очередь буфера, следующий кадр останется ждать на входе.

Подсеть, реализующая алгоритм управления очередью RED с полной буферизацией представлена на рисунке 13.

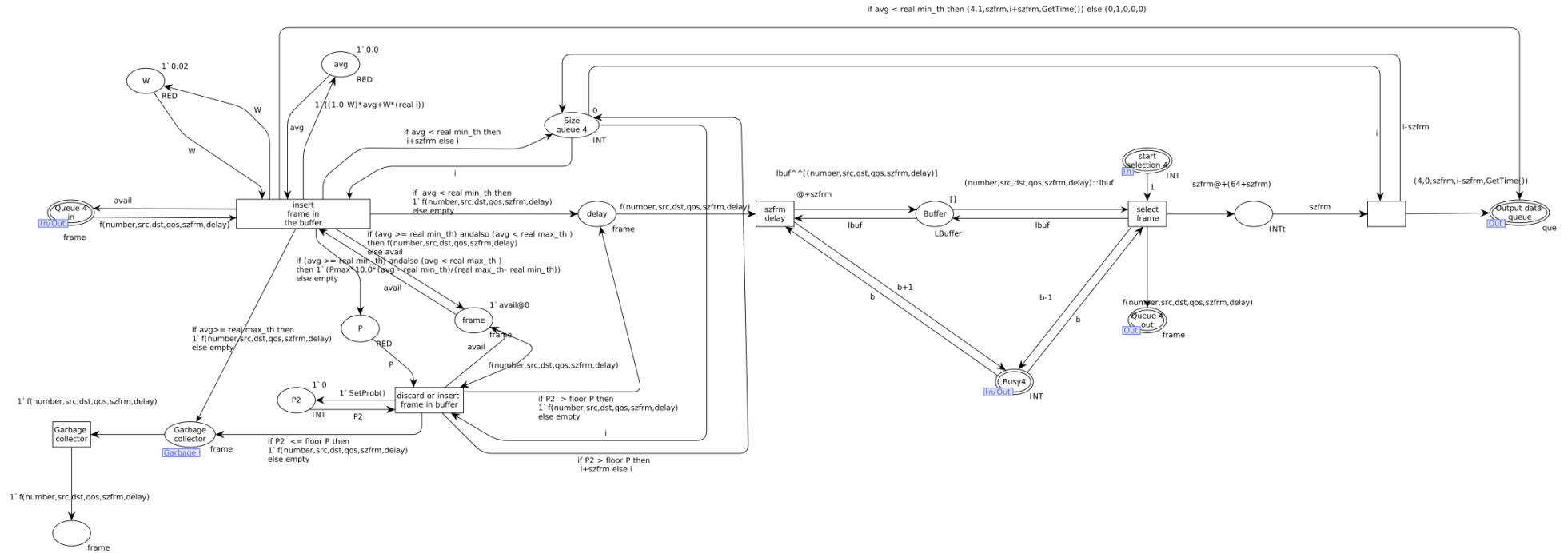


Рисунок 13 — Подсеть, реализующая алгоритм управления очередью RED с полной буферизацией

Помимо позиций и переходов описанных в таблице 5 были добавлены переход "szfrm delay", вносящий задержку на перемещение кадра в очередь буфера, и добавлена позиция "Busy4", в которой формируется сигнал занятости выхода буфера.

При срабатывании перехода "insert frame in the buffer" при условии, что кадр не отбрасывается, фишка  $f$  кадра Ethernet помещается в промежуточную позицию "delay". Далее срабатывает переход "szfrm delay" с задержкой равной размеру текущего кадра, тем самым кадр перемещается в очередь буфера не сразу, как описано в модели алгоритма RED с обработкой трафика "на лету", а по истечению определенного времени, требуемого на передачу кадра.

Такая модель буфера коммутатора наиболее приближена к работе реального коммутатора, так как на каждое перемещение кадра в коммутаторе требуется определенное время, зависящее от размера этого кадра.

### **3 Генератор сетевого трафика для моделей коммутаторов в среде CPN Tools**

Для оценки вероятностно-временных характеристик механизмов управления очередями в интеллектуальном коммутаторе Ethernet появилась необходимость в построении сети Петри, моделирующей источник потока кадров, характеристики которого были бы адекватны свойствам реального трафика Ethernet. Для построения такой модели также использовался пакет моделирования CPN Tools.

Генерация сетевого трафика с заданными характеристиками является задачей моделирования компьютерных сетей определенного назначения в частности, с поддержкой качества обслуживания (QoS). Кадры сетевого трафика анализируются в коммутаторе Ethernet, вследствие чего можно оценить эффективность обработки трафика и джиттер таких кадров.

Известные работы [18, 19] исследуют невозможность изменять законы распределения длительности интервалов формирования кадров и правила следования пакетов с различными значениями приоритетов.

Была предложена модель универсального генератора сетевого трафика со всеми настраиваемыми параметрами кадра, формируемыми по любому известному распределению или закону, описанная в следующих подразделах.

В настоящее время Ethernet работает в полнодуплексном режиме. Это обеспечивает одновременные прием и передачу кадров.

Разрабатываемые системы моделируют работу телекоммуникационной аппаратуры. Основным блоком данных в таких системах является цвет, который соответствует передаваемому по сети кадру. Так как модели жестко не привязаны к конкретному протоколу

передачи данных, то для удобства был принят формат кадра Ethernet, соответствующего стандарту IEEE 802.1Q. Форматы всех этих четырех типов кадров Ethernet приведены на рисунке 14.

| Кадр 802.3/LLC |    |   |      |               |         |                |     |
|----------------|----|---|------|---------------|---------|----------------|-----|
| 6              | 6  | 2 | 1    | 1             | 1(2)    | 46–1497 (1496) | 4   |
| DA             | SA | L | DSAP | SSAP          | Control | Data           | FCS |
|                |    |   |      | Заголовок LLC |         |                |     |

| Кадр Raw 802.3/Novell 802.3 |    |   |         |  |
|-----------------------------|----|---|---------|--|
| 6                           | 6  | 2 | 46–1500 |  |
| DA                          | SA | L | Data    |  |
|                             |    |   | FCS     |  |

| Кадр Ethernet DIX (II) |    |   |         |  |
|------------------------|----|---|---------|--|
| 6                      | 6  | 2 | 46–1500 |  |
| DA                     | SA | T | Data    |  |
|                        |    |   | FCS     |  |

| Кадр Ethernet SNAP |    |   |      |               |         |                |   |         |     |
|--------------------|----|---|------|---------------|---------|----------------|---|---------|-----|
| 6                  | 6  | 2 | 1    | 1             | 1       | 3              | 2 | 46–1492 | 4   |
| DA                 | SA | L | DSAP | SSAP          | Control | OUI            | T | Data    | FCS |
|                    |    |   | AA   | AA            | 03      | 000000         |   |         |     |
|                    |    |   |      | Заголовок LLC |         | Заголовок SNAP |   |         |     |

Рисунок 14 — Форматы кадров Ethernet

На рисунке 14 представлены следующие поля заголовка (на рисунке поле преамбулы и начальный ограничитель кадра не показаны):

- поле преамбулы состоит из семи синхронизирующих байтов — 10101010;
- начальный ограничитель кадра (Start-of-Frame-Delimiter, SFD) состоит из одного байта 10101011. Появление этой комбинации битов является указанием на то, что следующий байт — это первый байт заголовка кадра;
- адрес назначения (Destination Address, DA) может быть длиной 2 или 6 байт. На практике всегда используются MAC-адреса из 6 байт;

- адрес источника (Source Address, SA) — это 2- или 6-байтное поле, содержащее MAC-адрес узла — отправителя кадра;
- длина (Length, L) — 2-байтное поле, которое определяет длину поля данных в кадре;
- поле данных может содержать от 0 до 1500 байт. Но если длина поля меньше 46 байт, то используется следующее поле — поле заполнения, дополняющее кадр до минимально допустимого значения в 46 байт;
- поле заполнения состоит из такого количества байтов заполнителей, которое обеспечивает минимальную длину поля данных в 46 байт. Это обеспечивает корректную работу механизма обнаружения коллизий. Если длина поля данных больше или равна минимальной, то поле заполнения в кадре отсутствует;
- поле контрольной последовательности кадра (Frame Check Sequence, FCS) состоит из 4 байт контрольной суммы. Это значение вычисляется по алгоритму CRC-32.

Стандарт IEEE 802.1 Q определяет 12 возможных форматов инкапсуляции дополнительного поля в кадры Ethernet. Одним из возможных вариантов является введение тега, который передает информацию о принадлежности трафика к Ethernet II. Размер тега — 4 байта (рисунок 15).

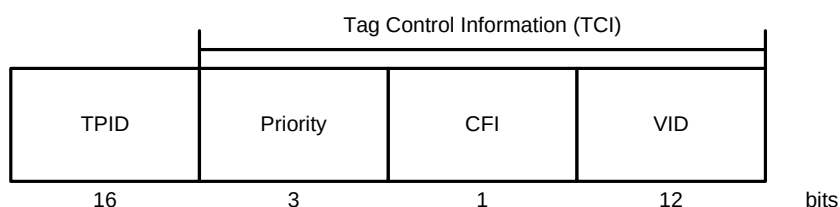


Рисунок 15 — Тег стандарта IEEE 802.1 Q

Он состоит из таких полей:

- тег идентификатора протокола TPID (Tag Protocol Identifier).  
Размер поля — 16 бит. Указывает, какой протокол используется для тегирования. Для 802.1q используется значение 0x8100;
- тег контроля информации TCI (Tag Control Information) - поле, инкапсулирующее в себе поля приоритета, канонического формата и идентификатора VLAN;
- приоритет (Priority) - размер поля составляет 3 бита. Используется стандартом IEEE 802.1p для задания приоритета передаваемого трафика;
- индикатор канонического формата CFI (Canonical Format Indicator) - размер поля составляет 1 бит. Указывает на формат MAC-адреса. 0 — канонический (кадр Ethernet), 1 — не канонический (кадр Token Ring, FDDI);
- идентификатор (VID) VLAN'a VLAN Identifier - размер поля составляет 12 бит. Указывает к какому VLAN'у принадлежит фрейм. Диапазон возможных значений VID от 0 до 4094.

Таким образом, конечный формат кадра Ethernet представлен на рисунке 16.



Рисунок 16 — Формат кадра Ethernet с добавлением тега

### 3.1 Обобщенная модель генератора сетевого трафика

Обобщенная модель генератора сетевого трафика представлена на рисунке 17. Она состоит из генераторов регулярного и стохастического трафиков, которые вырабатывают фишки, соответствующие кадрам сетевого трафика. Смеситель обеспечивает соединение этих двух потоков кадров в единый поток. Полученный сетевой трафик записывается в файл. Это обеспечивает возможность повторять исследования моделей при их изменении или исправлении, тем самым позволяет анализировать и выбирать наилучшую модель в конкретной ситуации.

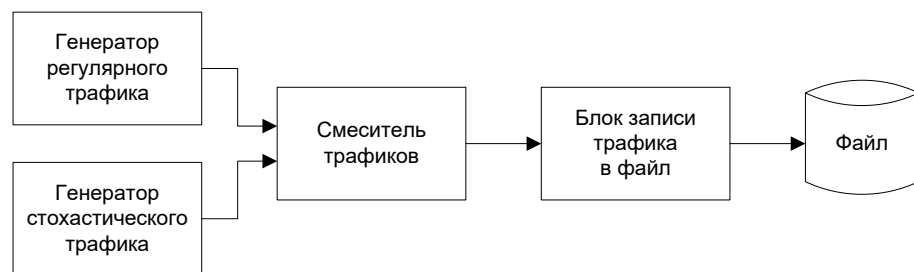


Рисунок 17 — Обобщенная модель генератора сетевого трафика

Исходя из структуры Ethernet кадра, была выбрана структура описывающая кадр, которая содержала следующие параметры: два MAC адреса, приоритет качества обслуживания QoS, размер данных, содержащихся в кадре, время прихода кадра в сеть и время ухода кадра из сети.

В соответствии со структурой были созданы следующие цвета:

- цвет *colset MAC = int with 1..10* описывает MAC-адрес источника и приемника и может принимать целочисленные значения от единицы до десяти;
- цвет *colset QoS = int with 0..9* описывает приоритет кадра и может принимать целочисленные значения от нуля до девяти;
- стандартный цвет *INT* описывает поле размера данных пакета;

- цвет  $colset\ INTT = int\ timed$  описывает временные задержки.

Был определен составной цвет  $frm$ , который описывает целиком структуру кадра и состоит из цветов описанных выше:

$$colset\ frm = product\ MAC*MAC*QoS*INT*INTT*INTT$$

Дополнительно был создан составной цвет  $frm1$  аналогичный цвету  $frm$  с добавлением в структуру цвета  $INT$  для описания порядкового номера для записи кадров сетевого трафика в файл:

$$colset\ frm1 = product\ MAC*MAC*QoS*INT*INTT*INTT*INT$$

В процессе моделирования возникла проблема описания состояния входных и выходных портов буфера, так как применяемые в пакете *CPN Tools* сети Петри не являются ингибиторными, т.е. отсутствуют дуги запрещающие срабатывание перехода, если в определенной позиции есть фишка. Таким образом, возникали ситуации, когда кадр еще не был записан в очередь, а уже пришел новый кадр на запись в очередь. Выходом из данной проблемы стало введение еще одного цвета:

$$colset\ frame1 = union\ f1:frm + avail\ timed$$

Данная строка означает, что в позиции цвета  $frame$  может находиться либо фишка  $f1$  цвета  $frm$ , описывающая Ethernet кадр, либо фишка цвета  $avail$ , показывающая, что позиция пуста.

### 3.2 Генератор сетевого трафика с различными значениями параметров класса QoS

Модель генератора сетевого трафика с различными значениями параметров класса QoS на основе сетей Петри в пакете *CPN Tools* представлена на рисунке 18.



В таблице 6 представлено описание всех позиций и переход генератора сетевого трафика.

| Обозначение               | Тип     | Описание                                                                    |
|---------------------------|---------|-----------------------------------------------------------------------------|
| Min Max veroytnost        | позиция | вероятности кадров с максимальным и минимальным значением длины кадра       |
| Created frame             | позиция | создание фишек кадра Ethernet                                               |
| Poisson traffic collector | позиция | создание фишек кадра Ethernet стохастического трафика с временной задержкой |
| Period                    | позиция | начальная позиция генератора регулярного трафика                            |
| Regular traffic collector | позиция | создание фишек кадра Ethernet регулярного трафика с временной задержкой     |
| Limit packets             | позиция | предел кадров регулярного трафика                                           |
| Count                     | позиция | количество сгенерированных кадров стохастического и регулярного трафиков    |
| Summa                     | позиция | общий размер всех передаваемых данных                                       |
| Out                       | позиция | формирование итогового кадра Ethernet                                       |
| Frames                    | позиция | выходная позиция                                                            |
| Set frame parameters      | переход | установка параметров кадра Ethernet                                         |
| Set QoS                   | переход | установка значения класса кадра                                             |
| Set delay 1               | переход | добавление задержки стохастического трафика                                 |
| Set delay 2               | переход | добавление задержки регулярного трафика                                     |
| Mixer block 1             | переход | смеситель стохастического трафика                                           |
| Mixer block 2             | переход | смеситель регулярного трафика                                               |
| form traffic              | переход | формирование итогового кадра Ethernet                                       |
| write in file             | переход | запись кадров в файл                                                        |

MAC-адреса источника или приемника в модели определяются с помощью функции  $fun SetMAC() = MAC.ran()$  в диапазоне значений от 1 до 10.

Была поставлена задача создать трафик с различными значениями параметров класса QoS в процентном соотношении 40:30:20:10. В Ethernet коммутаторе классы можно разделить на несколько частей, в зависимости от важности кадра телекоммуникационной сети. Чаще всего приоритеты разбиваются на два вида, но их может быть и больше. В данной модели использовался частный случай с четырьмя видами приоритетов: от высокого приоритета к самому низкому в процентном соотношении 40:30:20:10. Высокий приоритет имеет всегда значение 7, низкий имеет значение 1. Приоритет регулярного трафика всегда высокий. Приоритет стохастического трафика генерируется в два этапа. Сначала формируются значение приоритета по равномерному распределению в диапазоне от 0 до 9 с помощью функции *fun SetQoS() = discrete(0,9)*. На втором этапе в зависимости от интервала попадания формируется итоговый приоритет кадра. Ниже приведен фрагмент определения приоритета кадра.

```
if qos3 = 0 then f1(src1,dst1,1,szfrm1,delay11,delay21)
else if qos3 < 3 then f1(src1,dst1,3,szfrm1,delay11,delay21)
else if qos3 < 6 then f1(src1,dst1,5,szfrm1,delay11,delay21)
else f1(src1,dst1,7,szfrm1,delay11,delay21)
```

В результате получается, что классы кадров стохастического трафика распределены в процентном соотношении 40:30:20:10 от наиболее высокого значения приоритета к самому низкому.

Распределение длины кадра данных имеет ярко выраженный бимодальный характер: до 50 % всех пакетов имеют максимальный или минимальный размер данных (64 и 1500 байт, соответственно), а длину остальных можно считать равномерно распределенной в оставшемся диапазоне. Распределение длины кадра стохастического трафика определялось следующим образом. В начальной позиции "Min Max veroytnost" задавались вероятности кадров с максимальным и

минимальным значением длины кадра. Далее функция *SetSizeInterval* формировала дискретное значение в диапазоне от 0 до 99. Данный диапазон можно подстроить под любые требования различных настроек системы: *fun SetSizeInterval() = discrete(0,99)*. Конечную длину кадра считала функция *SetSizeFul*:

```
fun SetSizeFull(x,vmin,vmax) =  
if x<vmin then 64*8 else if x>(99-vmax) then (22+1500)*8  
else discrete(65,21+1500)*8;
```

В нее передавалось дискретное значение, полученное ранее, это значение необходимо для определения вероятности попадания длины кадра в определенную область. Весь интервал значений был поделен на три области: минимальную, максимальную и среднюю области. Далее после определения области генерировалась конечная длина кадра в битах. Размеры длины кадра являются также гибкими параметрами и могут быть перенастроены. Рассмотрим алгоритм функции *SetSizeFull* более подробно:

- если полученная вероятность меньше переменной *vermin*, то длина кадра равна минимальному значению (минимальное значение равно  $64 \cdot 8$  бит);

- если полученная вероятность больше  $100 - vermax$ , то длина кадра равна максимальному значению (максимальное значение равно  $1522 \cdot 8$  бит);

- если полученная вероятность попадала в среднюю область, то размер длины кадра выбирался случайным образом по равномерному распределению, диапазон которого равен от минимального значения до максимального.

Трафики модели являются гибкими и могут изменяться в зависимости от различных требований, применяемых к модели. Регулярный трафик вырабатывался с определенным периодом. Одновременно формировались четыре кадра регулярного трафика. Для того чтобы устранить одновременность прихода кадров в коммутатор, использовалось добавление небольшой задержки. Методы устранения одновременного прихода кадров в коммутатор применяются и в реальных системах управления коммутатором Ethernet. Для того чтобы кадры каждый раз шли не в одинаковой последовательности, была дополнительно создана функция  $fun\ RandDelay() = discrete(1,4)$ , умноженная на 10. В итоге задержка между этими кадрами была равна  $Задержка = discrete(1,4) * 10 + \text{Номер\_кадра}$ .

Стохастический трафик формировался через интервалы времени, распределенные по экспоненциальному закону с помощью функции:

```
fun SetTrafInterval() =
floor(exponential(1.0/10000.0)+0.5)
```

После формирования пакетов стохастического трафика с различными сгенерированными параметрами, полученный кадр передавался в смеситель трафика. Пакеты регулярного трафика были заданы жестко. Смеситель трафика обеспечивал соединение двух видов трафика (стохастического и регулярного). В модели в позиции "Limit packets" задавалось ограничение на количество генерируемых кадров регулярного трафика. Это ограничение является настраиваемым и может быть уменьшено либо увеличено по любому требованию к системе. В позиции "count" осуществлялся подсчет количества сформированных пакетов трафика. На выходе фишки кадров Ethernet трафика изменяли свой цвет на новый цвет frm1. К структуре добавлялось еще одно поле – порядковый номер кадра.

Запись сгенерированного трафика в файл осуществлялась с помощью встроенных функций пакета CPN Tools – мониторов. Листинг монитора на языке CPN ML представлен ниже:

```
fun obs (bindelem) =
  let
    fun obsBindElem (GeneratorTraf'Write_in_fille(1, {delay1,
delay2, dst, i, qos,src,szfrm})) = "1`("^Int.toString(i)^",
"^Int.toString(src)^",                      "^Int.toString(dst)^",
"^Int.toString(qos)^",                      "^Int.toString(szfrm)^",
"^ModelTime.toString(time())^",0)++\n"
    | obsBindElem _ = ""
  in
    obsBindElem bindelem
  end.
```

В результате исследования полученного на выходе генератора сетевого трафика были получены следующие характеристики трафика в виде гистограмм на рисунках 19, 20, 21, которые подтвердили правильность его работы.

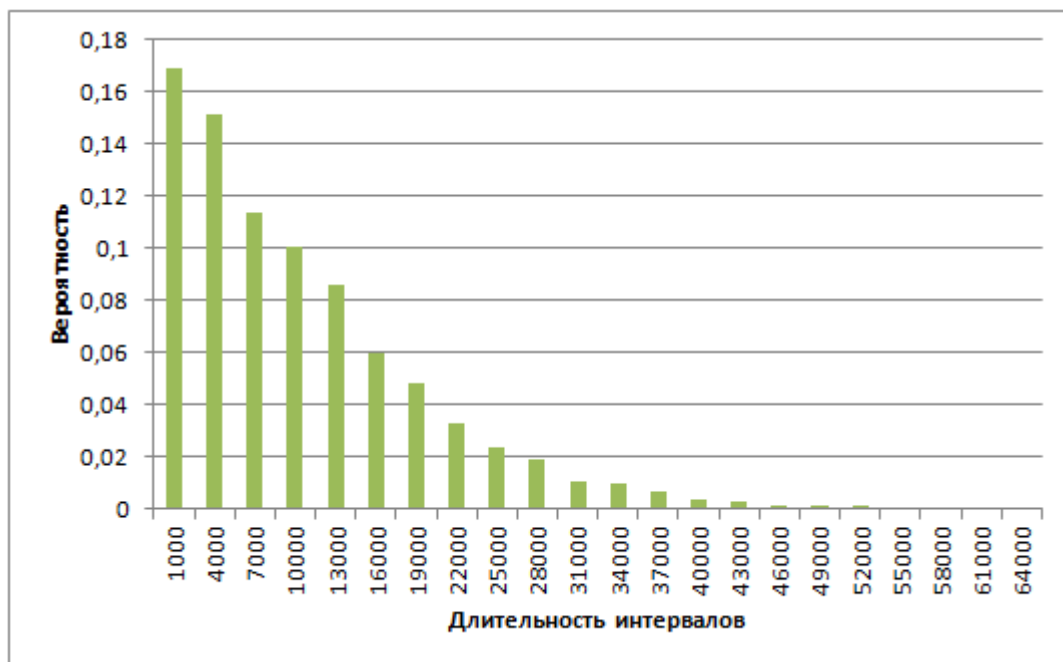


Рисунок 19 — Гистограмма распределения интервалов между моментами поступления кадров

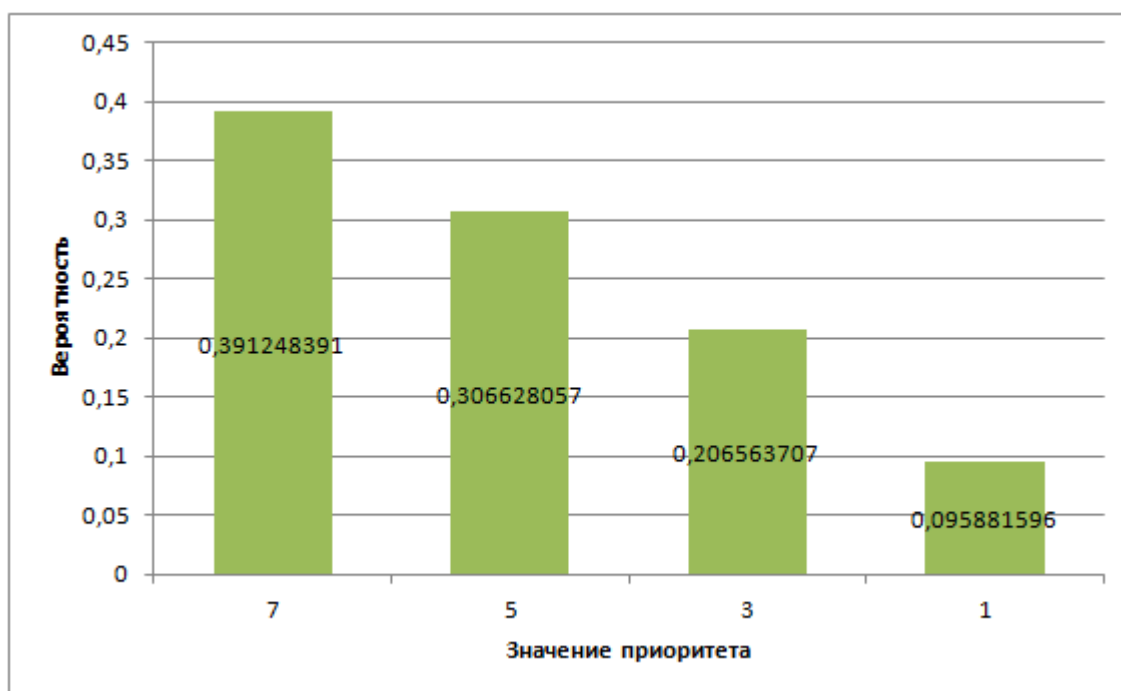


Рисунок 20 — Гистограмма распределения значений приоритетов кадров трафика

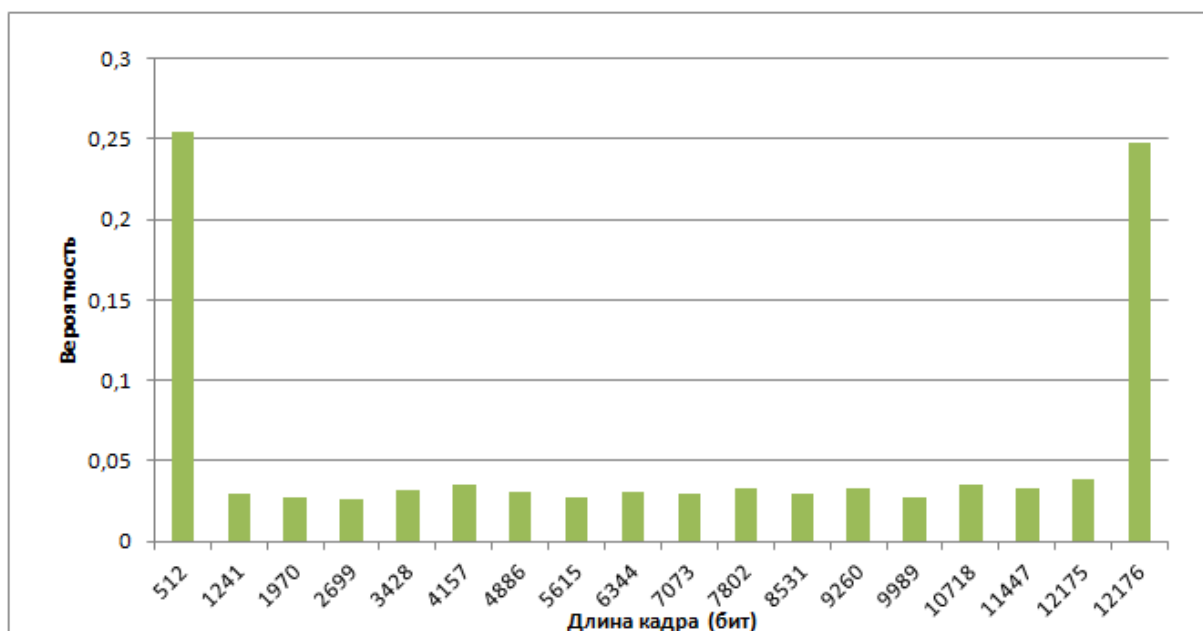


Рисунок 21 — Гистограмма распределения длин кадров сетевого трафика

### 3.3 Генератор сетевого трафика с "тяжелыми хвостами"

Для исследования моделей алгоритмов управления очередями, описанных во второй главе, наиболее интересным сетевым трафиком является так называемый трафик с "тяжелыми хвостами", где распределение интервалов между моментами поступления кадров формируется по распределению Парето.

Распределение Парето в теории вероятностей – двухпараметрическое семейство абсолютно непрерывных распределений, являющихся степенными. Называется это распределение по имени итальянского инженера Вилфредо Парето [20].

Для генератора сетевого трафика также использовался пакет CPN Tools. Данный пакет моделирования содержит в себе готовые функции различных распределений, но не всех, в том числе и распределения Парето. Было принято решение реализовать формирование интервалов между моментами поступления кадров отдельно, например, в среде Mathcad и затем перенести полученные значения в модель, реализованную в пакете CPN Tools.

Ниже представлен фрагмент реализации генератора случайных чисел по распределению Парето в Mathcad.

$$a = \frac{20000}{3} \quad b = 3$$

$$N = 5000 \quad n = 0 \dots N-1$$

$$x_n = \frac{a}{\sqrt[b]{1 - rnd(1)}}$$

Параметр  $N$  – количество сформированных значений,  $a$  и  $b$  – параметры распределения. Выходная последовательность значений была записана файл.

Остальные параметры кадра формировались, как описано в подразделе 3.2. В целом модель генератора сетевого трафика осталась такая же, за исключением добавления нового блока, который объединяет последовательность кадров трафика с соответствующими значениями интервала между моментами поступления кадров, полученными в Mathcad. На рисунке 22 представлен генератор сетевого трафика с "тяжелыми хвостами".



При срабатывании перехода read из файла считываются значения интервала между моментами поступления кадров и помещаются в позицию Interval. Далее срабатывает переход pareto, в котором считанные значения интервалов в соответствии с порядковым номером подставляются в соответствующее поле фишки f кадра Ethernet.

На рисунке 23 показана полученная гистограмма распределения интервалов между моментами поступления кадров по распределению Парето, которая подтвердила правильность работы модели генератора сетевого трафика.



Рисунок 23 — Гистограмма распределения интервалов между моментами поступления кадров по распределению Парето

Идея разработки универсального генератора сетевого трафика заключалась в том, чтобы объединить все достоинства различных пакетов и сред моделирования, например, как было описано выше пакета CPN Tools и Mathcad. В итоге получилась модель универсального генератора сетевого трафика со всеми настраиваемыми параметрами кадра,

формируемыми по любому известному распределению или закону, что делает такой сетевой трафик более адекватным свойствам реального трафика Ethernet.

#### **4 Исследование моделей алгоритмов управления очередями**

В рамках выпускной квалификационной работы были проведены исследования построенных моделей алгоритмов управления очередями в коммутаторах, описанных в подразделах 2.2.1, 2.2.2 и 2.2.3.

Работа моделей алгоритмов TailDrop и RED была исследована на тестовом наборе регулярного трафика.

С помощью генератора сетевого трафика был сформирован поток, состоящий из пятисот кадров регулярного трафика. С периодом в 25000 тактов формировалось по четыре кадра с небольшой задержкой между друг другом. Параметры кадра были заданы жестко, длина кадра равна 6400 бит.

Длина максимальной очереди в алгоритме TailDrop равнялась 70000 бит.

Параметры алгоритма RED были следующие:

- минимальный порог 30000 бит;
- максимальный порог 70000 бит;
- максимальное значение вероятности сброса 0.9.

На рисунках 24 и 25 представлены графики зависимости изменения очереди выходного буфера коммутатора от времени, соответственно полученные в результате обработки трафика алгоритмами TailDrop и RED, а также моменты времени отбрасывания пакетов с их порядковыми номерами.

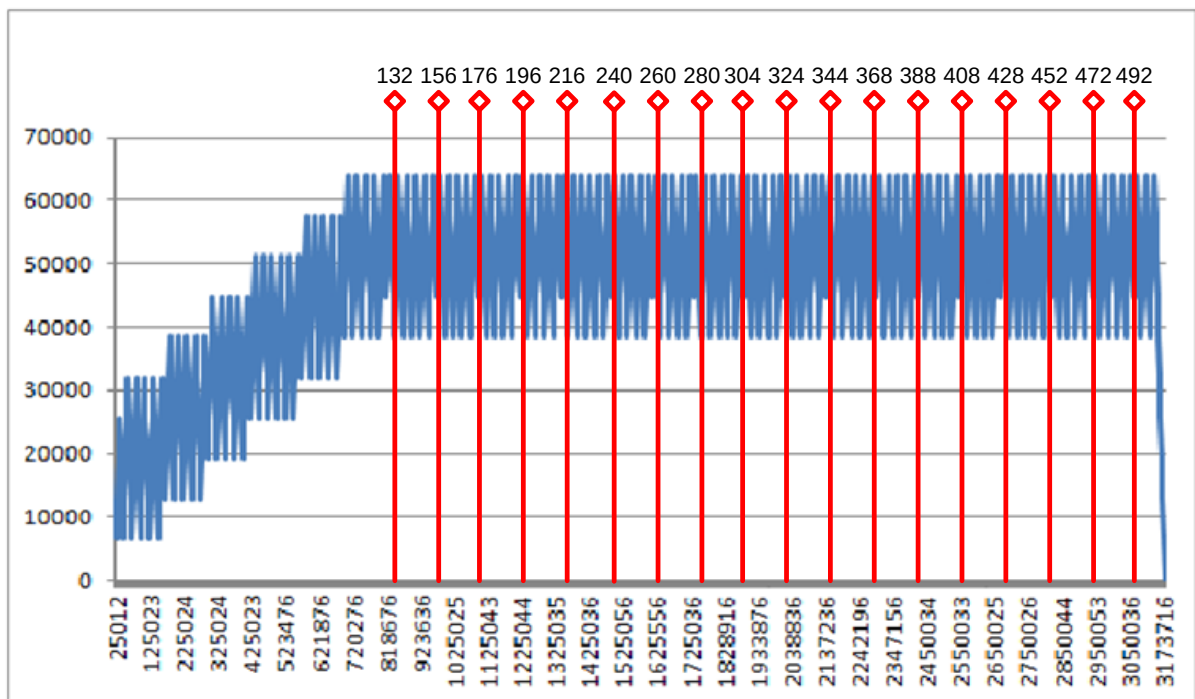


Рисунок 24 — График изменения очереди выходного буфера коммутатора и при алгоритме TailDrop

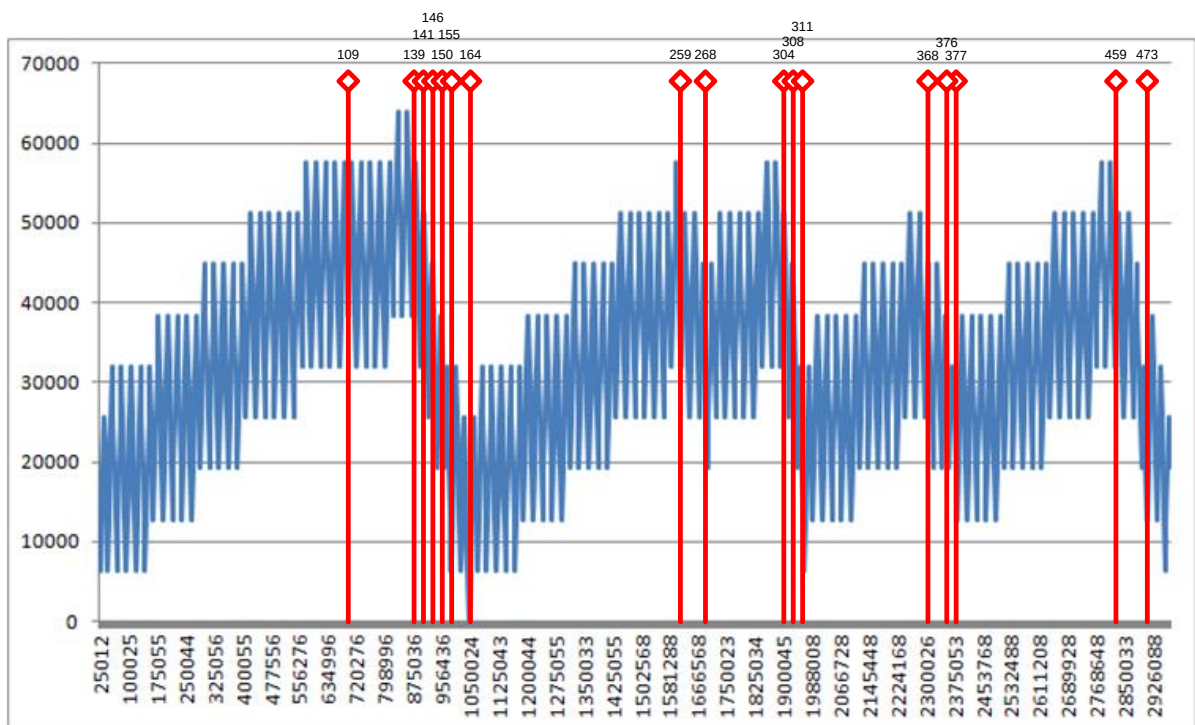


Рисунок 25 — График изменения очереди выходного буфера коммутатора при алгоритме RED

На графиках координатная ось X - это время, а координатная ось Y - это размер очереди в битах.

На рисунке 24 видно, что при использовании алгоритма TailDrop средняя длина очереди стремится к значению максимального порога и с определенной периодичностью, когда коммутатор не справляется с загрузкой канала, отбрасываются пакеты.

А при использовании алгоритма RED средняя длина очереди колеблется от 30000 бит до 50000 бит, тем самым не нагружая максимально очередь буфера. Кадры отбрасываются заблаговременно до возникновения всплеска длины очереди буфера в хаотичном порядке, сглаживая при этом нагрузку входного трафика. Это также позволяет избежать эффекта "зацикливания" трафика, тогда как при использовании алгоритма TailDrop и, например, протокола TCP, с такой же периодичностью, как при отбрасывании кадров, эти же кадры будут повторно отправлены от источников в сеть, и опять будет наблюдаться переполнение буфера.

Также при использовании алгоритма RED время обработки нашего тестового трафика меньше на 150000 тактов, чем при использовании алгоритма TailDrop, хотя количество отброшенных пакетов примерно одинаково.

## **Заключение**

В ходе выполнения выпускной квалификационной работы магистра были разработаны модели алгоритмов управления очередями в коммутаторах TailDrop и RED с полной буферизацией и с так называемой обработкой кадров "на лету" в среде моделирования CPN Tools, была усовершенствована модель генератора сетевого трафика Ethernet.

В рамках выпускной работы выполнено исследование поведения разработанных моделей на различных наборах сетевого трафика.

При разработке моделей были учтены все требования, изложенные в техническом задании.

Построенные модели алгоритмов управления очередями в коммутаторах позволяют оценить вероятностно-временные характеристики подобных систем, а также оценить влияние проектируемых систем на параметры качества обслуживания в сетях с поддержкой QoS.

## **Список использованных источников**

1. QoS. URL: <https://ru.wikipedia.org/wiki/QoS> (дата обращения: 03.05.18).
2. Кучерявый, Е.А. Управление трафиком и качество обслуживания в сети Интернет / Е.А. Кучерявый. – СПб.: Наука и Техника, 2004. – 336 с.
3. Яновский, Г.Г. Качество обслуживания в сетях IP. / Г.Г. Яновский. // Вестник связи. №1, 2008. – С. 65-74.
4. Олифер, В.Г., Олифер, Н.А. Компьютерные сети. Принципы, технологии, протоколы / В.Г. Олифер, Н.А. Олифер. – 4-е изд. – СПб.: Питер, 2010. – 943 с.
5. Floyd S., Jacobson V. Random Early Detection gateways for Congestion Avoidance // IEEE/ACM Transactions on Networking.— 1993.— Vol. 1, no. 4. - P. 397-413.
6. Коваленко Т.Н. Модель активного управления очередями в распределенных инфокоммуникационных системах, представленная сетью Петри [Электронный ресурс] / Т.Н. Коваленко // Проблемы телекоммуникаций. – 2012. – № 2 (7). – С. 58 – 67. – Режим доступа к журн.: [http://pt.journal.kh.ua/2012/2/1/122\\_kovalenko\\_queue.pdf](http://pt.journal.kh.ua/2012/2/1/122_kovalenko_queue.pdf).
7. Configuring Weighted Random Early Detection. URL: [https://www.cisco.com/c/en/us/td/docs/ios/12\\_2/qos/configuration/guide/fqos\\_c/qcfwred.pdf](https://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfwred.pdf) (дата обращения: 01.04.18)
8. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management : Rep. / ICSI ; Executor: S. Floyd, R. Gummadi, S. Shenker : 2001.
9. MatLab. URL: <https://ru.wikipedia.org/wiki/MATLAB> (дата обращения: 25.05.18)
10. Сети Петри. URL: [https://ru.wikipedia.org/wiki/Сети\\_Петри](https://ru.wikipedia.org/wiki/Сети_Петри) (дата обращения: 25.05.18)

11. Зайцев, Д.А. Сети Петри и моделирование систем. Метод. указания к практическим занятиям и лаб. работам / Д.А. Зайцев - Одесса: ОНАС им. А.С. Попова, 2007. - 42 с.
12. Зайцев Д.А., Шмелева Т.Р. Моделирование телекоммуникационных систем в CPN Tools / Д.А. Зайцев, Т.Р. Шмелева - Одесса: ОНАС им. А.С. Попова, 2009. - 72 с.
13. Кизилев Е.А. Алгоритмы и модели диспетчеризации очередей в компьютерных сетях с поддержкой качества обслуживания: диссертация на соискание учёной степени кандидата технических наук. ПГУ, Пенза, 2017.
14. Черных, И.В. Моделирование электротехнических устройств в MatLab, SimPowerSystems и Simulink. / И.В. Черных. – М.: ДМК Пресс, 2014. – 288 с.
15. Дьяконов, В. Simulink 4. Специальный справочник / В. Дьяконов. – СПб.: Питер, 2002. – 528 с.
16. Кизилев, Е.А. Моделирование механизмов qos в коммутаторах Ethernet цветными сетями Петри / Домнин А.Л., Кизилев Е.А., Пушкарев В.А. // XVII Всероссийская научно-методическая конференция «Телематика '2010» Труды XVII Всероссийской научно-методической конференции (Санкт-Петербург, 21-24 июня 2010 г.). - СПб.: СПбГУ ИТМО, 2010. – С. 29 – 35.
17. Кизилев, Е.А. Моделирование механизма взвешенного справедливого обслуживания очередей в компьютерных сетях / Е.А. Кизилев, Н.Н. Коннов, В.Б. Механов // Межвузовский сборник научных трудов «Вычислительные системы и технологии обработки информации» Вып. 10. – Пенза: Изд-во ПГУ, 2011. – С. 26 – 35.
18. Никишин, К. И. Моделирование трафика сети Ethernet цветными сетями Петри/ К. И. Никишин, Н. Н. Коннов, А. Л. Домнин// современные проблемы компьютерных наук (СПКН-2013) : сб. материалов I Междунар. науч.-практ. конф., посвящ. 70-летию образования

Пензенского государственного университета (г. Пенза, 29–30 октября 2013 г.) / под ред. Л. Р. Фионовой – Пенза : Изд-во ПГУ, 2013 – С. 120–124.

19. Никишин, К. И. Генератор трафика ethernet на основе цветных сетей петри / К.И.Никишин, Н.Н.Коннов // Модели, системы, сети в экономике, технике, природе и обществе. – 2016. - № 1(17). – С. 299–307.

20. Распределение Парето. URL: [https://ru.wikipedia.org/wiki/Распределение\\_Парето](https://ru.wikipedia.org/wiki/Распределение_Парето) (дата обращения: 02.04.18).

## **ПРИЛОЖЕНИЕ А**

### **ПРЕЗЕНТАЦИЯ**

# Моделирование и исследование алгоритмов управления очередями в коммутаторах

Магистрант Аладьев Ю.Ю.

Руководитель: к.т.н., профессор Коннов Н.Н.

## Цель работы

Цель: разработка моделей алгоритмов управления очередями в коммутаторах и исследование вероятностно-временных характеристик системы в зависимости от интенсивности поступающей нагрузки и выбранного алгоритма управления очередями.

Объект исследования: коммутатор Ethernet с поддержкой качества обслуживания.

Предмет исследования: алгоритмы управления очередями обработки сетевого трафика в коммутаторах

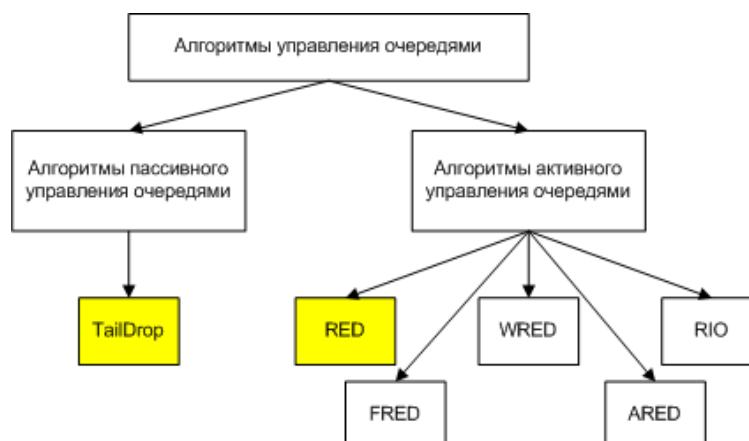
## Что такое качество обслуживания QoS

Quality of Service (QoS) — технология предоставления различным классам трафика различных приоритетов в обслуживании – качества обслуживания.

Механизмы реализации QoS:

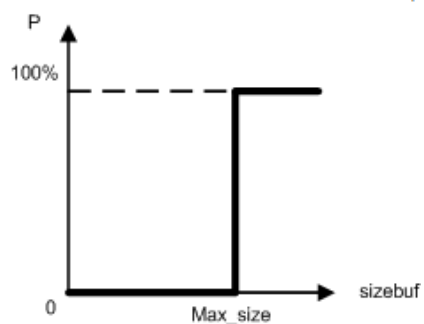
- механизмы маркирования;
- механизмы обслуживания очередей;
- механизмы управления очередями;
- механизмы управления полосой пропускания.

## Классификация алгоритмов управления очередями



## Алгоритм TailDrop

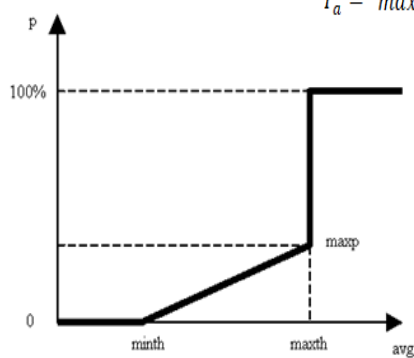
$$P_a = \begin{cases} 0, & \text{если } sizebuf < max\_size \\ 1, & \text{если } sizebuf > max\_size \end{cases}$$



Зависимость вероятности сброса  
от длины очереди

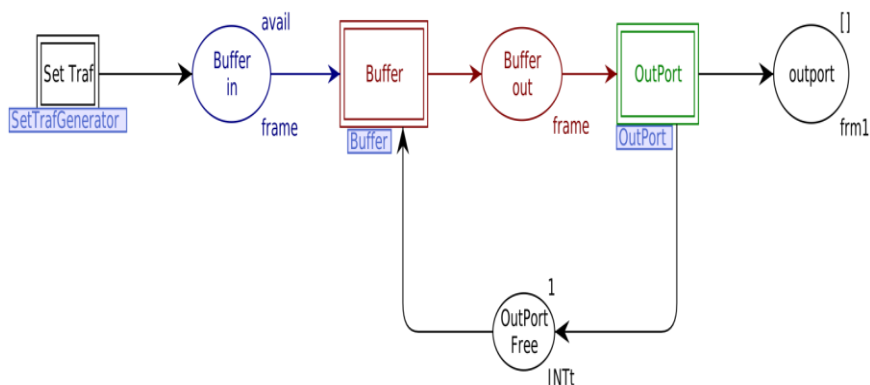
## Алгоритм RED

$$P_a = \begin{cases} 0, & \text{если } avg < min_{th} \\ max_p * \frac{avg - min_{th}}{max_{th} - min_{th}}, & \text{если } min_{th} \leq avg < max_{th} \\ 1, & \text{если } avg \geq max_{th} \end{cases}$$

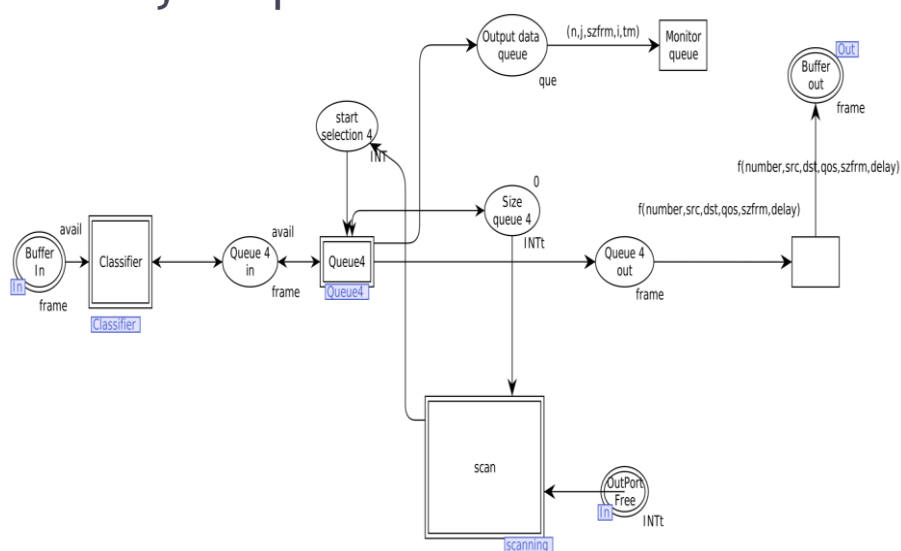


Зависимость вероятности сброса  
от усредненной длины очереди

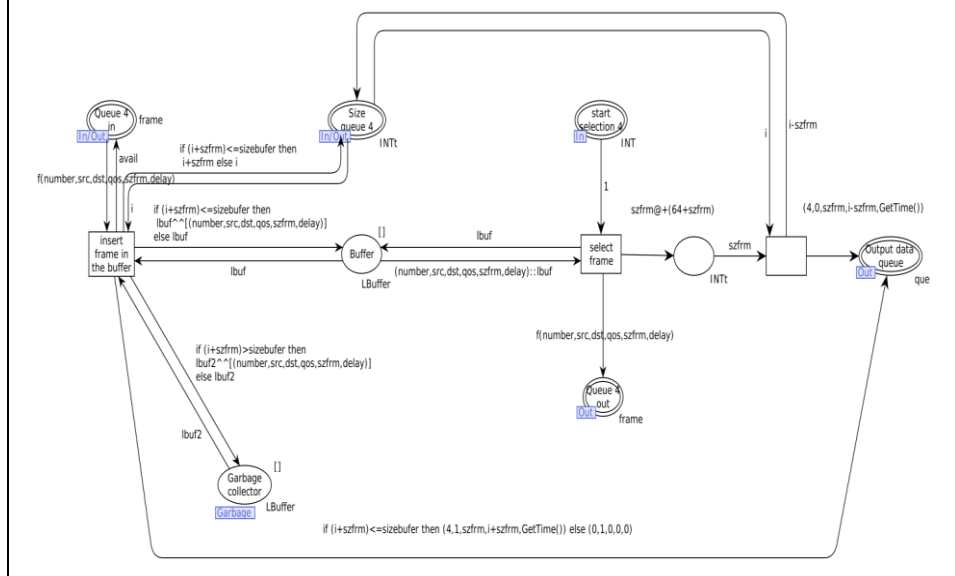
## Обобщенная структура модели коммутатора



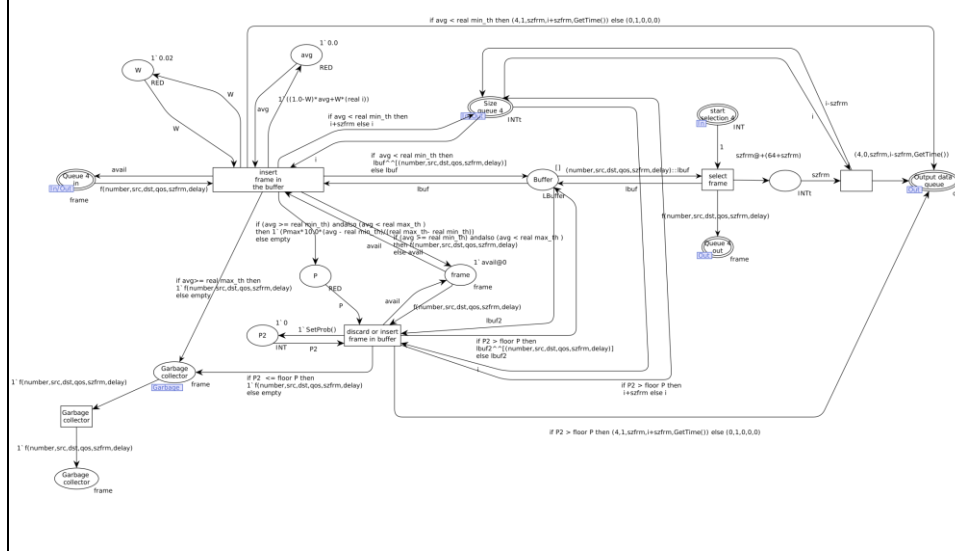
## Модель буфера выходного порта коммутатора



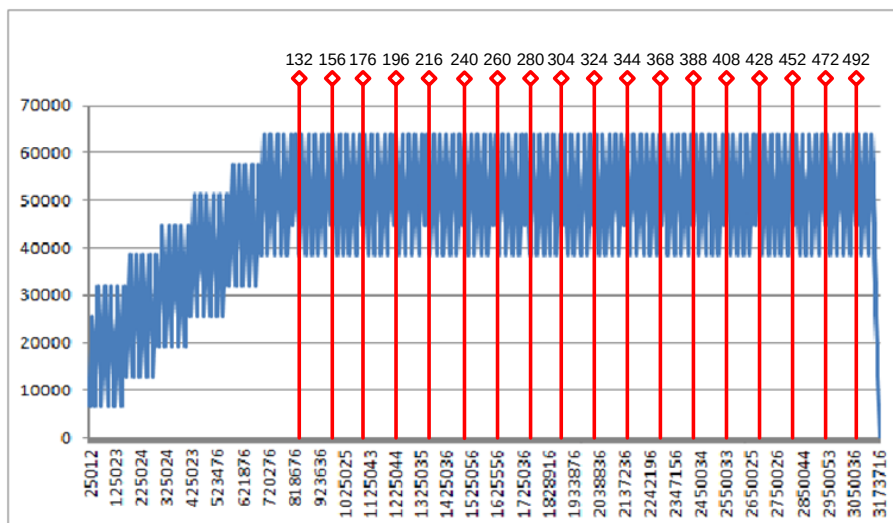
## Модель алгоритма TailDrop



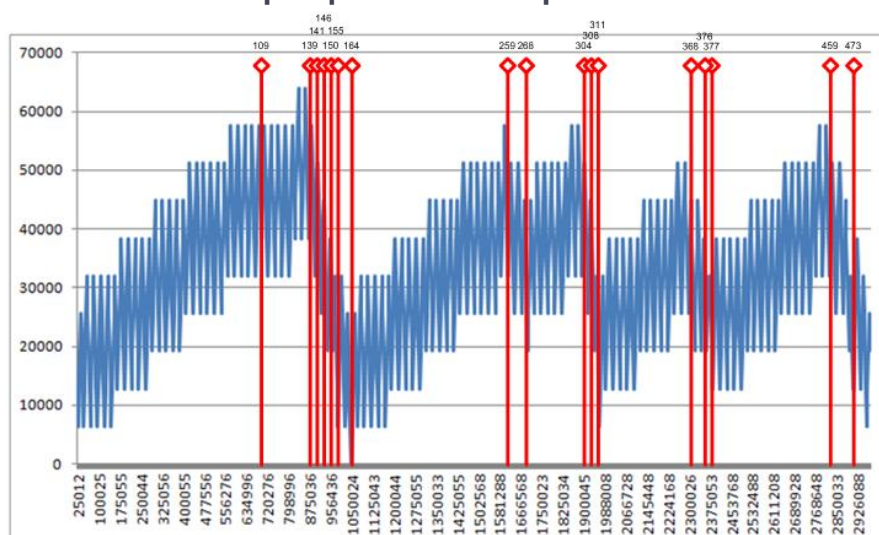
## Модель алгоритма RED



## Исследование на тестовом наборе сетевого трафика алгоритма TailDrop



## Исследование на тестовом наборе сетевого трафика алгоритма RED



**Спасибо за внимание!**