МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ

КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ

Факультет прикладной информатики

Кафедра компьютерных технологий и систем

К у р с о в а я р а б о т а

по дисциплине

"ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ, СЕТИ И ТЕЛЕКОММУНИКАЦИИ"

**Сравнительный анализ производительности методов типа "Алоха"**

Исполнитель: ХОРУЖАЯ А. М.,

факультет "Прикладная информатика",

учебная группа ПИ1901

Руководитель: д. э. н. Ткаченко В. В.

**Краснодар 2020**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc40272402)

[1. ЦЕЛИ И ЗАДАЧИ РАБОТЫ 4](#_Toc40272403)

[2. ТЕОРЕТИЧЕСКИЙ БЛОК 5](#_Toc40272404)

[2.1. ПРОТОКОЛ HDLC 5](#_Toc40272407)

[2.2. МЕТОДЫ СЛУЧАЙНОГО ДОСТУПА К СЕТИ 6](#_Toc40272408)

[2.3. ЧИСТАЯ АЛОХА 7](#_Toc40272409)

[2.4. СИНХРОННАЯ АЛОХА 10](#_Toc40272410)

[3. ОПИСАНИЕ АЛГОРИТМА 12](#_Toc40272411)

[4. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 13](#_Toc40272412)

[ЗАКЛЮЧЕНИЕ 15](#_Toc40272413)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 16](#_Toc40272414)

[ПРИЛОЖЕНИЕ: 17](#_Toc40272415)

[ПРИЛОЖЕНИЕ А. 17](#_Toc40272416)

[ПРИЛОЖЕНИЕ Б. 19](#_Toc40272417)

[ПРИЛОЖЕНИЕ В. 29](#_Toc40272418)

## ВВЕДЕНИЕ

В постиндустриальном мире компьютерное оборудование заполняет практически все сферы жизни и деятельности людей. Поднимается вопрос о том, как объединить компьютеры в единый механизм в пределах одного предприятия.

Необходимо организовать общение между компьютерами – процесс, который позволит обмениваться пакетами данных.

Такой принцип позволяет повысить эффективность работы и выполнить ранее непосильные задачи, за счет объединения рабочих ресурсов нескольких машин. Существует несколько методов применяемых на практике, но эта работа будет основываться на изучении методов чистой и синхронной Алохи.

В процессе работы будут алгоритмизированы данные методы, на основе результатов будет проведен анализ эффективности.

ALOHAnet — первая компьютерная сеть передачи данных с пакетной коммутацией, использовавшая в качестве среды доступа к ней беспроводную технологию. Была разработана и введена в эксплуатацию в 1968—1970-х годах группой учёных Гавайского университета под руководством Нормана Абрамсона в рамках исследовательского проекта THE ALOHA SYSTEM, основной целью которого было изучение возможностей использования радиопередачи как альтернативы проводным коммуникациям. Концептуальные наработки и решения, реализованные в ходе этого проекта, во многом легли в основу таких технологий и протоколов как Ethernet, Wi-Fi и сотовые сети. 6]

## ЦЕЛИ И ЗАДАЧИ РАБОТЫ

Главная задача – написание программ, реализующих работу компьютерной сети по передаче набора кадров: типы «Алоха». Необходимо обработать наборы из 100, 1000 и 10000 кадров:

Исходные данные, согласно варианту:

- величины нормированной пропущенной нагрузки: 4) 0,01; 0,5; 2,0;

- величины временных интервалов: 4) 1; 4; 8.

Необходимо найти:

а) время, необходимое для передачи всего набора кадров;

б) среднее время, необходимое для передачи кадра;

в) нормированную производительность протокола передачи.

Подсчитать количество коллизий, и результаты оформить в виде таблицы. Построить и сравнить графики зависимости производительности от пропущенной нагрузки.

Сравнить результаты моделирования с теоретическими расчетами и проанализировать эффективность алгоритмов.

Работа имеет особенность – фактически, в ней не будет использоваться локально-вычислительная сеть. Реализация будет осуществлена на основе математической модели. Факторы, не имеющие особого влияния, будут считаться близкими к 0.

Окончательным результатом будет являться выявление самого эффективного метода.

## ТЕОРЕТИЧЕСКИЙ БЛОК



### ПРОТОКОЛ HDLC

High-Level Data Link Control (HDLC) — бит-ориентированный протокол канального уровня сетевой модели OSI, разработанный ISO. Текущим стандартом для HDLC является ISO 13239. HDLC может быть использован в соединениях с множественным доступом, но в настоящее время в основном используется в соединениях точка-точка с использованием асинхронного сбалансированного режима (ABM). [3]

Имеет 3 типа станций:

1. **Primary terminal (Первичная(ведущая) станция)** отвечает за работоспособность и восстановление, поддерживает отельные связи в соединениях точка-многоточка.
2. **Secondary terminal (Вторичная (ведомая) станция)** полностью подконтрольна ведущей, поддерживает 1 сеанс связи.
3. **Combined terminal (Комбинированная станция)** гибрид предыдущих. В соединении точка-точка производит команды и ответы.

Поддерживает 3 режима логического соединения:

1. Режим нормального ответа (NRM), требует инициации передачи в виде явного разрешения на передачу от ведущей станции.
2. Режим синхронного ответа (ARM), даёт возможность ведомой станции самой инициировать передачу.
3. Асинхронный сбалансированный режим (ABM), используется комбинированными станциями, т.к. передача может быть инициализирована с любой стороны.

Немного познакомившись с этим протоколом, перейдем к интересующей нас, передаче кадров. Следуя концепции многоуровневой архитектуры, стандартизируется применение на каждом уровне архитектуры четырех основных примитивов услуг, чтобы предусмотреть взаимодействие между пользователями услуг на одном уровне и поставщиками услуг на нижестоящем уровне. Эти примитивы являются основными элементами определения обмена между пользователями услуг.

Основные элементы:

– запрос;

– признак;

– ответ;

– подтверждение.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Флаг FD | Адрес | Управляющее поле | Информационное поле | FCS | Флаг FD |
| 8 бит | Кратно 8 битам | 8 или 16 бит | 0 или более бит, кратно 8 | 16 бит | 8 бит |

Т.1. Структура кадров, включая флаги.

Где:

* Флаги – коды 01111110, обрамляющие HDLC-кадр. Позволяют приемнику определить начало и конец кадра.
* Адрес – идентификация одного из нескольких возможных устройств.
* Управляющее поле – определяет тип кадра по первым битам. (0 – информационный, 10 – управляющий, 11 – ненумерованный)
* Информационное поле – передача сети пакетов протоколов вышележащих уровней. (IP, IPX…)
* Поле FCS – контрольная последовательность, необходимая для обнаружения ошибок. [3]

### МЕТОДЫ СЛУЧАЙНОГО ДОСТУПА К СЕТИ

Двумя основными способами доступа к общей среде передачи являются управляемый доступ с применением опроса и случайный доступ. В свою очередь существуют различные типы стратегий случайного доступа.

Методы случайного доступа полностью децентрализованы. Пользователь может передавать когда угодно, лишь с незначительными ограничениями, зависящими от метода доступа.

Из-за случайности моментов времени, в которые пользователи могут решить начать передачу, независимо от метода не исключена возможность того, что два или несколько пользователей могут выйти на связь в пересекающиеся промежутки времени. Это приводит к столкновениям (коллизиям), которые сначала должны быть распознаны, а затем разрешены. При увеличении нагрузки увеличивается и вероятность коллизий, что приводит к возможной неустойчивости работы рассматриваемых механизмов.

В результате производительность ограничивается некоторым максимальным значением, меньшим пропускной способности канала, и это значение в каждом случае зависит от первоначального механизма доступа и алгоритма разрешения коллизий.

Сначала методы случайного доступа были предложены для случаев, когда многие пользователи пытаются довольно редко передать пачки сообщений или когда друг с другом связываются небольшое число ЭВМ. Но применительно к производственным процессам, которые требуют строгого управления задержкой доступа, более предпочтителен управляемый доступ. Рассмотрим два простейших типа стратегии случайного доступа: *чистую Алоху и синхронную Алоху.* [1]

### ЧИСТАЯ АЛОХА

Эта схема сначала была применена для доступа к общему каналу сотрудниками Гавайского университета в начале 1970-х годов. По этой схеме пользователь, желающий передать сообщение, делает это когда угодно. В результате могут наложится во времени два или несколько сообщений, вызвав столкновение (коллизию).

Распознавание коллизий и сообщение о них пострадавшим пользователям в первоначальной системе Алоха направлялись по радио на центральный пункт. Также это могло осуществляться применением положительных подтверждений в сочетании с перерывом. При обнаружении столкновения пострадавшие станции предпринимают попытки повторной передачи потерянного сообщения, но они должны распределять время попыток случайным образом, следуя некоторому алгоритму, уменьшающему возможность возникновения нового конфликта.

Стратегия доступа типа Чистая Алоха позволяет добиться производительности самое большее 1/2e ≈ 0,18 пропускной способности канала. Введем сначала некоторые определения. За доступ к каналу состязаются N станций. Станция передает, в среднем, λ пакетов в секунду (интенсивность обращений к сети). Величина 1/m представляет собой пропускную способность канала (μ) в передаваемых пакетах в секунду. Рассмотрим частный случай, при котором все передаваемые сообщения (пакеты) имеют среднюю длину, соответствующую *m* единицам времени передачи. Будем считать, что интенсивность нагрузки S (эквивалентно ρ - нормированной по μ нагрузке) характеризует использование канала вновь поступающими пакетами

***S ≡* ρ *= N*λ*m***

Величина 1/*m*, которая обозначается μ, представляет собой пропускную способность канала в передаваемых пакетах в секунду. Таким образом, *N*λ*/*μ *= N*λ*m* - относительное использование канала, или производительность, нормированная относительно μ. Общая интенсивность пакетов, передаваемых в канал, включая вновь генерируемые и передаваемые повторно, имеет некоторое значение λ’ > λ (из-за коллизий от каждого компьютера будет передаваться больше сообщений из-за необходимости возобновлять поток). Тогда фактическая интенсивность нагрузки, или использование канала, является параметром *G*, который равен

***G* = *N*λ*’m***

Рассмотрим типичное сообщение длительностью *m*, показанное на рисунке 1.

m

m

время

Рис. 1. Столкновение двух сообщений.

Оно подвергается столкновению с другим сообщением, если эти два сообщения будут наложены одно на другое в любой точке. Легко заметить, передвигая пунктирное сообщение во времени, что столкновение может произойти в промежутке времени продолжительностью 2*m* с. Вероятность того, что в промежутке 2*m* с не произойдет столкновения, равна e-2Nλ’m=e-2G.

График 1. Характеристика производительности Чистая Алоха.

Отношение S/G представляет долю сообщений из числа передаваемых в канал, которые проходят успешно. Это число должно быть равно вероятности отсутствия столкновений. Таким образом уравнение производительности для чистой Алохи:

**S = Ge-2G**

Здесь S - нормированная производительность (средняя скорость поступления пакетов, деленная на максимальную производительность 1/m), а G - нормированная пропущенная нагрузка. Таким образом, S – независимая переменная, а G - ее функция. График зависимости G от S имеет вид двузначной кривой (график 1).

График 1. Характеристика производительности Чистая Алоха

Отметим, что S имеет максимум S = 0,5e-1≈ 0,18 при G = 0,5. Судя по формуле (4.11) или кривой при малой поступающей нагрузке S столкновения происходят редко и G ≈ S. Когда S начинает расти, приближаясь к максимальному значению 0.18, число столкновений быстро увеличивается, что ведет в свою очередь к росту вероятности столкновения. Система теряет устойчивость, S падает, а G увеличивается до больших значений**.** [1]

### СИНХРОННАЯ АЛОХА

Максимально возможная производительность схемы чистой Алохи может быть удвоена с помощью простого приема разметки шкалы времени и разрешения пользователям начинать попытки передачи сообщений только в начале каждого временного интервала m (равного длительности сообщения). Эта схема требует, чтобы работа всех пользователей системы была синхронизирована во времени. Пример работы такой системы показан на рисунке 2, на котором одно сообщение передано успешно, а с другим произошло столкновение.

Успешная передача

Столкновение

…

время

4m

3m

2m

m

0

-m

Канальные интервалы

Рис. 2. Передача при синхронной Алохе.

Поскольку сообщения могут быть переданы только в размеченные промежутки времени, столкновения происходят лишь когда одна или несколько попыток передачи совершаются в том же промежутке.

Вероятность успешной передачи задается в виде e-G, а производительность для синхронной Алохи имеет вид:

**S = Ge-G**

Нормированная производительность S достигает максимального значения 1/e ≈ 0,368 при G = 1. Зависимость пропущенной нагрузки от производительности для синхронной Алохи показана на рисунке 4, где она сравнивается с соответствующей зависимостью для чистой Алохи.

График 2. Характеристика производительности; Синхронная Алоха.

Из приведенной характеристики очевидно, что ввиду двух возможных значений G при заданной производительности S для этой системы доступа также характерна неустойчивость. [1]

## ОПИСАНИЕ АЛГОРИТМА

В работе используются методы Алоха. Блок-схема алгоритма приведена в [Приложении А](#_ПРИЛОЖЕНИЕ_А.). По формулам, приведенным в теории, высчитываются принятые и отклоненные пакеты. На основе полученных и входных данных считаются все необходимые показатели(коллизии, нормированная производительность и т.д.). Полный и, при необходимости, прокомментированный код каждого из методов приведен в [Приложении Б](#_ПРИЛОЖЕНИЕ_Б.). Работа выполнена в среде Microsoft Visual Studio 2019 на языке C++. Сгруппированные результаты выводятся на консоль.

По введенным и распределенным данным, в первую очередь, рассчитывается вероятность прохождения кадра, по формуле (P = exp(-2 \* NTL)), где NTL – нормированная пропускная нагрузка, данная из условия задачи. Далее в зависимости от опытной вероятности прохождения кадра изменяются счетчики принятых пакетов и коллизий.

По полученным результатам определяется опытное значение нормированной пропускной нагрузки, нормированная производительность, время обработки 1 кадра и передачи всего пакета.

## АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Исходными данными, согласно варианту, являются:

1. NTL – нормированная пропускная нагрузка (0.01; 0.5; 2.0)
2. T – временной интервал (1; 4; 8)
3. NOP – количество пакетов(наборов) (100; 1 000; 10 000)
4. Cores – количество процессоров в системе (4)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Чистая Алоха (100 пакетов) | | | | | | | | |
| G | t | | | T | Tcp | S | Sтеор | Количество коллизий |
| 0.01 | 1 |  |  | 100 | 1 | 0.001 | 0.01 | 0 |
|  | 4 |  | 424 | 4.24 | 0.133 | 0.01 | 6 |
|  |  | 8 | 816 | 8.16 | 0.184 | 0.01 | 2 |
| 0.5 | 1 |  |  | 242 | 2.42 | 0 | 0.18 | 142 |
|  | 4 |  | 1048 | 10.48 | 0.166 | 0.18 | 162 |
|  |  | 8 | 2024 | 20.24 | 0.088 | 0.18 | 153 |
| 2 | 1 |  |  | 4816 | 48.16 | 0 | 0.04 | 4716 |
|  | 4 |  | 17812 | 178.12 | 0 | 0.04 | 4353 |
|  |  | 8 | 35048 | 350.48 | 0 | 0.04 | 4281 |

По результатам составлены таблицы. В которых помимо входных данных, занесены итоговые результаты. Т – время, для полной передачи пакета. Тср – среднее время, для передачи 1 кадра. S – нормированная производительность. Sтеор – теоретическая нормированная производительность. Стоит заметить, что результаты близкие к нулю, приняты за него (=0).

Ниже приведены результаты для 100 пакетов Синхронной и Чистой Алохи. Показатели для 1 000 и 10 000 пакетов будут расположены в [Приложении В.](#_ПРИЛОЖЕНИЕ_В.)

Т. 2. Результат анализа 100 пакетов метода Чистая Алоха.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Синхронная Алоха 100 кадров | | | | | | | | | |
| G | t | | | T | Tcp | S | Sтеор | Количество коллизий |
| 0.01 | 1 |  |  | 102 | 1.02 | 0.068 | 0.01 | 2 |
|  | 4 |  | 400 | 4 | 0.368 | 0.01 | 0 |
|  |  | 8 | 816 | 8.16 | 0.305 | 0.01 | 2 |
| 0.5 | 1 |  |  | 155 | 1.55 | 0.001 | 0.3 | 55 |
|  | 4 |  | 632 | 6.32 | 0.305 | 0.3 | 58 |
|  |  | 8 | 1304 | 13.04 | 0.354 | 0.3 | 63 |
| 2 | 1 |  |  | 776 | 7.76 | 0 | 0.27 | 676 |
|  | 4 |  | 3464 | 34.64 | 0.007 | 0.27 | 766 |
|  |  | 8 | 5296 | 52.96 | 0.137 | 0.27 | 562 |

Т. 3. Результат анализа 100 пакетов метода Синхронная Алоха.

Как видно из таблицы для метода чистой Алохи опытное значение производительности меньше, чем для метода синхронной Алохи. Можно заметить, что количество коллизий зависит от производительности и временного интервала.

Анализируя данные с более крупными пакетами ( таблицы 4 и 5 [Приложения В](#_ПРИЛОЖЕНИЕ_В.) ) напрашивается вывод, что производительность Чистой Алохи, все-таки уступает Синхронной, но это недостаточно соответствует теоретическим заключениям.

Если увеличить временной интервал, то увеличится количество коллизий и производительность. При G=2, производительность обоих методов практически равна 0, а количество коллизий достаточно большое. Что говорит об отсутствии эффективности.

Наборы из 10 000 кадров практически полностью соответствуют теоретическим, кроме строк с данными при G=2. Производительности практически 0.

Вывод: при обработке крупных пакетов ( > 10 000), практические показатели близки к теоретическим.

## ЗАКЛЮЧЕНИЕ

Главной задачей этой роботы являлось создание программы с работающими методами «Чистая Алоха» и «Синхронная Алоха». Входными данными являлись временные интервалы, количество кадров и нормированная нагрузка. Данная модель спроектирована и представлена в виде блок-схемы и кода, реализованного в среде MS VS 2019.

Исходящие данные распределены в таблицы для анализирования.

Сравнительный анализ производительности обоих методов показал, что производительность метода «синхронной Алохи» практически в 2 выше, и подтвердил тем самым теоретические данные, приводимые в основном используемом ресурсе ( 1 пункт из списка используемой литературы).

Наиболее точные результаты получаются при передаче более крупных пакетов (в нашем случае 10 000 кадров). Погрешности округлений сведены к минимуму.

В любом случае, неточности связаны с упрощенностью модели случайного доступа. Да, они повлияли на полученные результаты, но никак не смогли изменить конечные выводы.

Подводя итоги работы, можно сказать, что поставленная в самом начале задача – выполнена. Самый эффективный метод выявлен. А также, данная программа может быть использована в качестве основы для разработки приложений в этой предметной области.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Лойко В.И., Лаптев В.Н. - Компьютерные системы. Методические рекомендации для контактной и самостоятельной работы (электронное издание). – Краснодар: КубГАУ, 2020. – 125 с., ил., с34-38. [1].
2. Т. П. Барановская, В. И. Лойко, М. И. Семенов, А. И. Трубилин. Под ред. В. И. Лойко – Архитектура компьютерных систем и сетей. Учебное пособие – М: Финансы и статистика, 2007. – 291 с.: ил.
3. HDLC - Электронный ресурс – ВикипедиЯ [https://ru.wikipedia.org/wiki/HDLC], [3].
4. Семенов М.И, Лойко В.И., Барановская Т.П. Компьютерные системы и сети: Учебное пособие для студентов специальности 0719 – "Информационные системы в экономике" и др. экономических специальностей вузов. – Краснодар: КубГАУ, 2000. – 215с.
5. Функции канального уровня – электронный ресурс [https://it-black.ru/funktsii-kanalnogo-urovnya-modeli-osi/]
6. ALOHAnet – электронный ресурс – ВикипедиЯ [https://ru.wikipedia.org/wiki/ALOHAnet] [6]

## ПРИЛОЖЕНИЕ:

### ПРИЛОЖЕНИЕ А.

Изображена полная блок-схема программы. Насколько могли заметить, методы отличаются только в двух точках:

1. в расчете вероятности прохождения кадра ( P = exp(-2\*NTL) – для чистой и

P=exp(-NTL) – для синхронной)

1. в расчете нормированной производительности протокола передачи

( S= RG \* exp(-2 \* RG) – для чистой и S = RG \* exp(-RG) – для синхронной)

Поэтому смысла в двух одинаковых блок-схемах нет.

Синхронная Алоха

Вывод результатов: RG, totalTime, timeOnePacket

RG = (float)((counterPacketSuccess + counterPacketDenied) \* cores \* ((float)1/T)) / NOP

S=RG \* exp(-2\*RG)

totalTime = (counterPacketSuccess + counterPacketDenied) \* T

timeOnePacket = totalTime \* 1.0 / NOP

i--

RP=rand() % 101

RP > prob

counterPacketDanied ++

counterPacketSuccess ++

RP <= prob

RP=rand() % 101

i=0; i<NOP; i++

i=0; P=exp(-NTL); prob=int(P\*100);

counterPacketSuccess=0; counterPacketDanied=0

Показатели:

NTL, T, NOP, cores

### ПРИЛОЖЕНИЕ Б.

Листинг метода «Чистая Алоха»:

#include "stdafx.h" //код переделан и готов к сдаче

#include <iostream>

#include <string>

#include <cmath>

#include <stdlib.h>

#include <time.h>

using namespace std;

//входные данные:

// NTL - нормированная пропускная нагрузка

// T - временной интервал

// NOP - количество пакетов

// n - постоянное количество процессоров в системе = 4

void pure\_Alloha(double NTL, int T, int NOP, int cores);

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(0));

cout << "Тест метода Чистая Аллоха" << endl;

cout << endl;

//Ввод показателей для расчета 4 варианта

double NTL\_1 = 0.01; //нормированная пропускная нагрузка

double NTL\_2 = 0.5;

double NTL\_3 = 2.0;

int T\_1 = 1; //временной интервал

int T\_2 = 4;

int T\_3 = 8;

int NOP\_1 = 100; // количество пакетов

int NOP\_2 = 1000;

int NOP\_3 = 10000;

int cores = 4; // количество процессоров в системе

// вывод показателей для 100 пакетов 1ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << " Для 100 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|' ;

pure\_Alloha(NTL\_1, T\_1, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_1, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_1, NOP\_1, cores);

cout << endl;

// вывод показателей для 100 пакетов 2ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << " Для 100 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_2, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_2, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_2, NOP\_1, cores);

cout << endl;

// вывод показателей для 100 пакетов 3ий интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << " Для 100 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_3, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_3, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_3, NOP\_1, cores);

cout << endl;

// вывод показателей для 1 000 пакетов 1ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 1 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_1, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_1, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_1, NOP\_2, cores);

cout << endl;

// вывод показателей для 1 000 пакетов 2ой интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 1 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_2, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_2, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_2, NOP\_2, cores);

cout << endl;

// вывод показателей для 1 000 пакетов 3ий интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 1 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_3, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_3, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_3, NOP\_2, cores);

cout << endl;

// вывод показателей для 10 000 пакетов 1ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 10 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_1, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_1, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_1, NOP\_3, cores);

cout << endl;

// вывод показателей для 10 000 пакетов 2ой интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 10 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_2, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_2, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_2, NOP\_3, cores);

cout << endl;

// вывод показателей для 10 000 пакетов 3ий интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 10 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_1, T\_3, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_2, T\_3, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

pure\_Alloha(NTL\_3, T\_3, NOP\_3, cores);

cout << endl;

system("pause");

return(0);

}

void pure\_Alloha(double NTL, int T, int NOP, int cores) {

{

int i = 0;

double P = exp(-2 \* NTL); // вероятность прохождения кадра

double prob = int(P \* 100);

int RP; // опытная вероятность прохождения кадра

int counterPacketSuccess = 0; //счестчик прошедших (успешных) пакетов

int counterPacketDenied = 0; // счетчик не прошедших пакетов

for (int i = 0; i < NOP; i++)

{

RP = rand() % 101;

if (RP <= prob)

{

counterPacketSuccess++; //счетчик принятых пакетов

}

else

{

counterPacketDenied++; //счетчик отклоненных пакетов

while (RP > prob)

RP = rand() % 101;

i--;

};

};

// опытное значение нормированной пропускной нагрузки

double RG = (float)((counterPacketSuccess + counterPacketDenied) \* cores \* ((float)1 / T)) / NOP;

double S = RG \* exp(-2 \* RG); //нормированная производительность

int totalTime = (counterPacketSuccess + counterPacketDenied) \* T;

double timeOnePacket = totalTime \* 1.0 / NOP;

cout << endl;

cout << "Результаты тестирования:" << endl;

cout << "Общее время передачи кадров: " << totalTime << endl;

cout << "Среднее время передачи одного кадра: " << timeOnePacket << endl;

cout << "Нормированная производительность протокола передачи=" << S << endl;

cout << "Количество не прошедших пакетов: " << counterPacketDenied << endl;

};

}

Листинг метода «Синхронная Алоха»:

#include "stdafx.h"

#include <iostream>

#include <string>

#include <cmath>

#include <stdlib.h>

#include <time.h>

using namespace std;

//входные данные:

// NTL - нормированная пропускная нагрузка

// T - временной интервал

// NOP - количество пакетов

// n - постоянное количество процессоров в системе = 4

void synchronous\_Alloha(double NTL, int T, int NOP, int cores);

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(0));

cout << "Тест метода Синхронная Аллоха" << endl;

cout << endl;

//Ввод показателей для расчета 4 варианта

double NTL\_1 = 0.01; //нормированная пропускная нагрузка

double NTL\_2 = 0.5;

double NTL\_3 = 2.0;

int T\_1 = 1; //временной интервал

int T\_2 = 4;

int T\_3 = 8;

int NOP\_1 = 100; // количество пакетов

int NOP\_2 = 1000;

int NOP\_3 = 10000;

int cores = 4; // количество процессоров в системе

// вывод показателей для 100 пакетов 1ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << " Для 100 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_1, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_1, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_1, NOP\_1, cores);

cout << endl;

// вывод показателей для 100 пакетов 2ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << " Для 100 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_2, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_2, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_2, NOP\_1, cores);

cout << endl;

// вывод показателей для 100 пакетов 3ий интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << " Для 100 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_3, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_3, NOP\_1, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_3, NOP\_1, cores);

cout << endl;

// вывод показателей для 1 000 пакетов 1ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 1 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_1, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_1, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_1, NOP\_2, cores);

cout << endl;

// вывод показателей для 1 000 пакетов 2ой интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 1 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_2, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_2, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_2, NOP\_2, cores);

cout << endl;

// вывод показателей для 1 000 пакетов 3ий интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 1 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_3, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_3, NOP\_2, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_3, NOP\_2, cores);

cout << endl;

// вывод показателей для 10 000 пакетов 1ый интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 10 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_1, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_1, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_1 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_1, NOP\_3, cores);

cout << endl;

// вывод показателей для 10 000 пакетов 2ой интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 10 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_2, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_2, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_2 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_2, NOP\_3, cores);

cout << endl;

// вывод показателей для 10 000 пакетов 3ий интервал

cout << "--------------------------------------------------------------------------------------------------------" << endl;

cout << "Для 10 000 пакетов:" << endl;

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_1 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_1, T\_3, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_2 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_2, T\_3, NOP\_3, cores);

cout << endl;

cout << "Для нормированной пропускной нагрузки = " << NTL\_3 << " и " << "временного интервала = " << T\_3 << '\n';

cout << '|';

synchronous\_Alloha(NTL\_3, T\_3, NOP\_3, cores);

cout << endl;

system("pause");

return(0);

}

void synchronous\_Alloha(double NTL, int T, int NOP, int cores) {

{

double P = exp(-NTL); // вероятность прохождения кадра

double prob = int(P \* 100);

int RP; // опытная вероятность прохождения кадра

int counterPacketSuccess = 0; //счестчик прошедших (успешных) пакетов

int counterPacketDenied = 0; // счетчик не прошедших пакетов

for (int i = 0; i < NOP; i++) //NOP - кол-во пакетов

{

RP = rand() % 101;

if (RP <= prob)

{

counterPacketSuccess++; //счетчик принятых пакетов

}

else

{

counterPacketDenied++; //счетчик отклоненных пакетов

while (RP > prob)

RP = rand() % 101;

i--;

};

};

// опытное значение нормированной пропускной нагрузки

double RG = (float)((counterPacketSuccess + counterPacketDenied) \* cores \* ((float)1 / T)) / NOP;

double S = RG \* exp(-RG); //нормированная производительность

int totalTime = (counterPacketSuccess + counterPacketDenied) \* T;

double timeOnePacket = totalTime \* 1.0 / NOP;

cout << endl;

cout << "Результаты тестирования:" << endl;

cout << "Общее время передачи кадров: " << totalTime << endl;

cout << "Среднее время передачи одного кадра: " << timeOnePacket << endl;

cout << "Нормированная производительность протокола передачи=" << S << endl;

cout << "Количество не прошедших пакетов: " << counterPacketDenied << endl;

};

}

### ПРИЛОЖЕНИЕ В.

Для удобства анализирования, таблицы сгруппированы и расположены на разных страницах.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Чистая Алоха 1 000 кадров | | | | | | | | |
| G | t | | | T | Tcp | S | Sтеор | Количество коллизий |
| 0.01 | 1 |  |  | 1017 | 1.017 | 0.001 | 0.01 | 17 |
|  | 4 |  | 4128 | 4.128 | 0.133 | 0.01 | 32 |
|  |  | 8 | 8208 | 8.208 | 0.184 | 0.01 | 26 |
| 0.5 | 1 |  |  | 2712 | 2.712 | 0 | 0.18 | 1712 |
|  | 4 |  | 11244 | 11.244 | 0.145 | 0.18 | 1811 |
|  |  | 8 | 22256 | 22.256 | 0.090 | 0.18 | 1782 |
| 2 | 1 |  |  | 50447 | 50.447 | 0 | 0.04 | 49447 |
|  | 4 |  | 211716 | 211.716 | 0 | 0.04 | 51929 |
|  |  | 8 | 414992 | 414.992 | 0 | 0.04 | 50874 |

Т. 4. Результат анализа 1 000 пакетов метода Чистая Алоха.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Синхронная Алоха 1 000 кадров | | | | | | | | |
| G | t | | | T | Tcp | S | Sтеор | Количество коллизий |
| 0.01 | 1 |  |  | 1010 | 1.01 | 0.072 | 0.01 | 10 |
|  | 4 |  | 4048 | 4.048 | 0.368 | 0.01 | 12 |
|  |  | 8 | 8096 | 8.096 | 0.306 | 0.01 | 12 |
| 0.5 | 1 |  |  | 1679 | 1.679 | 0.009 | 0.3 | 679 |
|  | 4 |  | 6452 | 6.452 | 0.319 | 0.3 | 613 |
|  |  | 8 | 13408 | 13.408 | 0.361 | 0.3 | 676 |
| 2 | 1 |  |  | 6985 | 6.985 | 0 | 0.27 | 5985 |
|  | 4 |  | 30428 | 30.428 | 0.006 | 0.27 | 6607 |
|  |  | 8 | 60352 | 60.352 | 0.083 | 0.27 | 6544 |

Т. 5. Результат анализа 1 000 пакетов метода Синхронная Алоха.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Чистая Алоха 10 000 кадров | | | | | | | | |
| G | t | | | T | Tcp | S | Sтеор | Количество коллизий |
| 0.01 | 1 |  |  | 10212 | 1.0212 | 0.001 | 0.01 | 212 |
|  | 4 |  | 40808 | 4.0808 | 0.133 | 0.01 | 202 |
|  |  | 8 | 81376 | 8.1376 | 0.184 | 0.01 | 172 |
| 0.5 | 1 |  |  | 27721 | 2.7721 | 0 | 0.18 | 17721 |
|  | 4 |  | 110352 | 11.0352 | 0.011 | 0.18 | 17588 |
|  |  | 8 | 217192 | 21.7192 | 0.088 | 0.18 | 17149 |
| 2 | 1 |  |  | 506277 | 50.6277 | 0 | 0.04 | 496277 |
|  | 4 |  | 2002784 | 200.278 | 0 | 0.04 | 490696 |
|  |  | 8 | 4069880 | 406.988 | 0 | 0.04 | 498735 |

Т. 6. Результат анализа 10 000 пакетов метода Чистая Алоха.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Синхронная Алоха 10 000 кадров | | | | | | | | |
| G | t | | | T | Tcp | S | Sтеор | Количество коллизий |
| 0.01 | 1 |  |  | 10108 | 1.0108 | 0.071 | 0.01 | 108 |
|  | 4 |  | 40372 | 4.0372 | 0.368 | 0.01 | 93 |
|  |  | 8 | 80784 | 8.0784 | 0.305 | 0.01 | 98 |
| 0.5 | 1 |  |  | 16581 | 1.6581 | 0.008 | 0.3 | 6581 |
|  | 4 |  | 66620 | 6.662 | 0.315 | 0.3 | 6655 |
|  |  | 8 | 133208 | 13.3208 | 0.362 | 0.3 | 6651 |
| 2 | 1 |  |  | 72098 | 7.0298 | 0 | 0.27 | 62098 |
|  | 4 |  | 289316 | 28.9316 | 0.005 | 0.27 | 62329 |
|  |  | 8 | 578048 | 57.8048 | 0.010 | 0.27 | 62256 |

Т. 7. Результат анализа 10 000 пакетов метода Синхронная Алоха.