Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Направление подготовки бакалавриата

01.03.02 Прикладная математика и информатика

Отчет о практике по научно-исследовательской работе

Анализ и моделирование сетей передачи данных

Выполнил:
студент 4 курса группы 22403

Игнатьев Е. А. _____
подпись

Место прохождения практики:
Кафедра информатики и математического обеспечения
Период прохождения практики:
03.02.20-03.05.20

Руководитель:
О. Ю. Богоявленская, к.т.н., доцент

_____
подпись

Итоговая оценка

_____
оценка

Петрозаводск — 2020

Содержание

Введение

Цель практики: Разработать приложение, определяющее распределение размера скользящего окна.

Задачи практики:

1. Изучить теорию сетей передачи данных;
2. Изучить теорию марковских цепей;
3. Рассмотреть работу алгоритма ЛРСУ (Линейный Рост Степенное Убывание);
4. Изучить алгоритм скользящего окна с обратной связью.

В настоящее время возросла потребность людей и предприятий в мгновенной передачи данных на огромные расстояния. Для того, чтобы передавать и обмениваться информацией, современные устройства нуждаются в сети передачи данных, гарантирующей быструю и надежную доставку информации получателю.

Сети передачи данных представляют собой группу устройств связи, которые объединены между собой при помощи каналов передачи данных [7]. Также сюда входят различные коммуникационные устройства, гарантирующие обмен сообщениями между устройствами, например: коммутаторы, шлюзы, хабы.

Телекоммуникация - дистанционная передача данных на базе компьютерных сетей и современных технических средств связи, в которой информация может поступать в самых различных видах: звуки, цифровые сигналы, изображения и печатные слова. Самой распространённой телекоммуникационной сетью является интернет [7].

В современных сетях передачи данных используется такое понятие как *скользящее окно* - оно определяет интенсивность потока посылаемых пакетов.

При отправке пакетов через сеть, транспортный протокол, например, TCP, должен обеспечивать надежность и целостность отправляемых данных. Возникает проблема анализа современных сетей передачи данных, а также проблема управления сетевым трафиком, в частности возникает потребность в приложении, определяющем распределение размера скользящего окна.

1 Обзор модели TCP/IP

TCP/IP — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет. Название TCP/IP происходит из двух важнейших протоколов семейства — Transmission Control Protocol (TCP) и Internet Protocol (IP), которые первыми были разработаны и описаны в данном стандарте [8].

1.1 Обзор протокола TCP

~~TCP протокол базируется на IP~~, но добавляет две важные вещи:

1. Установление соединения — ~~это~~ позволяет ему, в отличие от IP, гарантировать доставку;
2. Порты — для обмена данными между приложениями, а не просто узлами.

Протокол TCP предназначен для обмена данными — он является «надежным» протоколом, потому что:

1. Обеспечивает надежную доставку данных, так как предусматривает установление логического соединения;
2. Делит передаваемый поток байтов на части — сегменты — и передает их нижнему уровню, а на ~~приемной~~ стороне снова собирает их в непрерывный поток байтов;
3. Нумерует сегменты и подтверждает их прием квитанцией, а в случае потери организует повторную передачу.

1.1.1 Надежная доставка сегментов

Под надёжной доставкой понимается автоматическая повторная пересылка недошедших сегментов. Каждый сегмент маркируется при помощи специального поля — порядкового номера (sequence number). После отправки некоторого количества сегментов, TCP

на отправляющем узле ожидает подтверждения от получающего, где указывается порядковый номер следующего сегмента, который адресат желает получить. В случае, если такое подтверждение не получено, отправка автоматически повторяется. После нескольких неудачных попыток, протокол считает, что адресат недоступен, и сессия разрывается [9].

Таким образом, надёжная доставка означает, что разработчик, использующий TCP на транспортном уровне, знает, что если сессия не разорвалась, то всё, что он поручил отправить, будет доставлено получателю без потерь [9].

1.1.2 Упорядочивание сегментов

Каждый сегмент на нижних уровнях TCP/IP обрабатывается индивидуально. То есть, как минимум, он будет запакован в индивидуальный пакет. Пакеты идут по сети, и промежуточные маршрутизаторы в общем случае уже ничего не знают о том, что запаковано в эти пакеты. Часто пакеты с целью балансировки нагрузки могут идти по сети разными путями, через разные промежуточные устройства, с разной скоростью. Таким образом, получатель, декапсулировав их, может получить сегменты не в том порядке, в котором они отправлялись [9].

TCP автоматически пересоберёт их в нужном порядке, используя всё то же поле порядковых номеров, и передаст сегменты после склейки на уровень приложений.

1.1.3 Контроль за скоростью передачи данных

Контроль за скоростью передачи позволяет корректировать скорость отправки данных в зависимости от возможностей получателя. Благодаря механизму скользящего окна, TCP может работать с сетями разной надёжности. Данный механизм позволяет менять количество пересылаемых байтов, на которые надо получать подтверждение от адресата. Чем больше размер окна, тем больший объём информации будет передан до получения подтверждения. Если же TCP видит, что данные теряются, размер окна автоматически уменьшается [9].

1.1.4 Структура TCP-сегмента

Заголовок TCP-сегмента имеет следующую структуру:

Source Port (16 бит)			Destination Port (16 бит)
Sequence Number (32 бита)			
Acknowledgement Number (32 бита)			
Header Length (4 бита)	Reserved (6 бит)	Control (6 бит)	Window (6 бит)
Checksum (16 бит)			Urgent (16 бит)
Options (0 или 32 бита)			

- Source Port - номер порта получателя
- Destination Port - номер порта отправителя
- Sequence Number - порядковый номер сегмента
- Acknowledgement Number - номер подтверждения (номер сегмента, который должен придти следующим)
- Header Length - длина заголовка
- Reserved - зарезервированные дополнительные биты
- Control - флаги
- Window - размер окна
- Checksum - контрольная сумма
- Urgent - флаг важности (срочности) данного сегмента
- Options - дополнительные опции при их наличии

1.1.5 Механизм действия протокола

В отличие от традиционной альтернативы — UDP, который может сразу же начать передачу пакетов, TCP устанавливает соединения, которые должны быть созданы перед передачей данных. TCP-соединение можно разделить на 3 стадии:

- Установка соединения
- Передача данных

- Завершение соединения

Процесс начала сеанса TCP (также называемый «рукопожатие» (англ. handshake)) состоит из трёх шагов:

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN.
    - Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет для обслуживания нового клиента;
    - В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED;
    - В случае неудачи сервер посылает клиенту сегмент с флагом RST.
2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK.
    - Если клиент одновременно получает и флаг ACK, то он переходит в состояние ESTABLISHED;
    - Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться;
    - Если клиент не получает ответа в течение 10 секунд, то он повторяет процесс соединения заново.
3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED.
    - В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED.

При обмене данными приёмник использует номер последовательности, содержащийся в получаемых сегментах, для восстановления их исходного порядка. Приёмник уведомляет передающую сторону о номере последовательности, до которой он успешно получил данные, включая его в поле «номер подтверждения». Все получаемые данные, относящиеся к промежутку подтверждённых последовательностей, игнорируются. Если полученный сегмент содержит номер последовательности больший, чем ожидаемый, то данные из

сегмента буферизируются, но номер подтверждённой последовательности не изменяется. Если впоследствии будет принят сегмент, относящийся к ожидаемому номеру последовательности, то порядок данных будет автоматически восстановлен исходя из номеров последовательностей в сегментах [10].

Для того, чтобы передающая сторона не отправляла данные интенсивнее, чем их может обработать приёмник, TCP содержит средства управления потоком. Для этого используется поле «окно». В сегментах, направляемых от приёмника передающей стороне, в поле «окно» указывается текущий размер приёмного буфера. Передающая сторона сохраняет размер окна и отправляет данных не более, чем указал приёмник. Если приёмник указал нулевой размер окна, то передача данных в направлении этого узла не происходит, пока приёмник не сообщит о большем размере окна [10].

Завершение соединения можно рассмотреть в три этапа:

- Посылка серверу от клиента флага FIN на завершение соединения.
- Сервер посылает клиенту флаги ответа ACK и FIN - соединение закрыто.
- После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.

1.1.6 Congestion Control

TCP Congestion Control - алгоритмы, которые пытаются сделать ~~все возможное, чтобы~~ обеспечить наиболее быструю скорость передачи данных между двумя узлами, передающими данные через TCP. Они управляют размером TCP-окна и могут ориентироваться на RTT (Round Trip Time — время от отправки запроса до получения ответа), потерю пакетов, время ожидания отправки пакета из очереди и т.д [11].

Управление потоком осуществляется путём ограничения количества сегментов данных, передаваемых за один раз, а также запроса подтверждений получения до отправки следующих сегментов.

Для управления потоком TCP в первую очередь определяет количество сегментов данных, которое может принять устройство назначения.

Во время задержки в получении подтверждения отправитель не отправляет дополнительных сегментов. В те периоды, когда сеть чрезвычайно загружена или ресурсы получа-

ющего узла на пределе, длительность задержки может увеличиться. По мере увеличения длительности задержки эффективная скорость передачи данных для этого сеанса снижается. Замедление передачи данных при каждом сеансе помогает уменьшить конфликт ресурсов на сетевом устройстве и устройстве назначения в случае запуска нескольких сеансов связи.

Протокол TCP использует размер окна, чтобы попытаться управлять скоростью передачи данных в соответствии с максимальным значением потока, которое поддерживается сетью и устройством назначения, одновременно с этим уменьшая потери данных и количество их повторных пересылок [11].

1.1.7 Sliding Window Protocol

Протокол скользящего окна является функцией пакетной передачи данных протоколов. Данный протокол используется там, где требуется надежная и упорядоченная доставка пакетов, например, на канальном уровне (уровень 2 OSI). Он также используется для повышения эффективности, когда канал может иметь высокую задержку [12].

Пакетные системы основаны на идее отправки пакета данных вместе с дополнительными данными, которые позволяют получателю гарантировать, что пакет был принят правильно контрольной суммой. Когда получатель проверяет данные, он отправляет сигнал подтверждения ACK обратно отправителю, чтобы указать, что он может принять следующий пакет. В простом автоматическом протоколе повторного запроса (ARQ) отправитель останавливается после каждого пакета и ожидает получения ACK от получателя. Это гарантирует, что пакеты поступают в правильном порядке, поскольку за один раз может быть отправлен только один пакет [12].

Время, которое требуется для приема сигнала ACK, может представлять значительное количество времени по сравнению со временем, необходимым для отправки пакета. В этом случае общая пропускная способность может быть намного ниже теоретически возможной. Чтобы решить эту проблему, протоколы скользящего окна позволяют отправлять выбранное количество пакетов без ожидания ACK. Каждый пакет получает порядковый номер, и ACK отправляют обратно этот номер. Протокол отслеживает, какие пакеты были подтверждены и когда они получены, затем отправляет больше пакетов. Таким образом, окно скользит по потоку пакетов, составляющих передачу [12].

2 Теория марковских цепей

2.1 Основные понятия

Цепью Маркова называют такую последовательность случайных событий, в которой вероятность события зависит только от состояния, в котором процесс находится в текущий момент и не зависит от ~~ранних состояний~~. Конечная дискретная цепь определяется:

1. **Множеством состояний** S, в котором событием является переход из одного состояния в другое в результате испытания;

2. **Вектором начальных вероятностей** $p^0 = \{p^0(1), ..., p^0(n)\}$, определяющим вероятности $p^0(i)$ того, что в начальный момент времени $t = 0$ процесс находился в состоянии s_i;

3. **Матрицей переходных вероятностей** $P = \{p_{ij}\}$, которая характеризует вероятность перехода процесса с текущим состоянием s_i в следующее состояние s_j, при этом: $\sum_{j=1}^{n} P_{ij} = 1$.

Пример матрицы переходных вероятностей с множеством состояний $S = \{S_1, ...S_5\}$, вектором начальных вероятностей $p^0 = \{1, 0, 0, 0, 0\}$:

$$P = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{array} \begin{array}{c} \begin{array}{ccccc} s_1 & s_2 & s_3 & s_4 & s_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0,4 & 0,3 & 0,2 & 0,1 \\ 0 & 1 & 0 & 0 & 0 \\ 0,1 & 0,9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Рис. 1. Матрица P

С помощью вектора начальных вероятностей и матрицы переходов можно вычислить стохастический вектор p^n — вектор, составленный из вероятностей $p^n(i)$ того, что процесс окажется в состоянии i в момент времени n. Получить p^n можно с помощью формулы $p^n = p^0 \times P^n$.

Векторы p^n при росте n могут сходиться к некоторому вероятностному вектору ρ, который можно назвать **стационарным распределением** цепи. Стационарность проявля-

ется в том, что взяв $p^0 = \rho$, мы получим $p^n = \rho$ для любого n.

2.2 Алгоритм ЛРСУ

Для разработки приложения, определяющего распределение размера скользящего окна, необходимо ~~использовать~~ алгоритм "Линейный Рост Степенное Убывание"(англ. AIMD) на основе марковских процессов. Суть алгоритма заключается в том, что если от получателя пришло подтверждение об успешной доставке пакетов, то данный алгоритм линейно увеличивает интенсивность потока. При потере пакета, ЛРСУ снижает интенсивность потока отправляемых пакетов в $n>0$ раз.

Модель ЛРСУ основывается на следующих преположениях:

1. Потери пакетов происходят согласно некоторому известному распределению $f_i i = 1, 2, ...,$ где f_i - вероятность того, что i пакетов были последовательно доставлены получателю
2. Пропускная способность сетевого маршрута имеет явный, заранее известный предел
3. Линейный рост скользящего окна ограничен некоторой величиной w_{max}
4. Отправитель всегда имеет данные для отправки получателю

Данные предположения соответствуют принятым принципам реализации алгоритма ЛРСУ [5].

2.3 Описание модели на основе ЛРСУ

На основе этих данных можно описать задачу теории массового обслуживания:

1. Система имеет управляемый входной поток, детерминированное обслуживание, обратную связь и неограниченную очередь
2. Линия связи является обслуживающим устройством
3. Популяция заявок составляет множество пакетов отправителя
4. Время обслуживания - детерминированное t_0

В случайные моменты времени поступают сигналы обратной связи - как положительные, так и отрицательные. Заявки регулируются алгоритмом ЛРСУ в зависимости от типа сигнала. Все заявки обслуживаются в порядке поступления. Чтобы построить модель производительности протокола, необходимо охарактеризовать выходящий из данной системы поток данных [5].

Данная задача теории массового обслуживания будет решаться с использованием алгоритма ЛРСУ на основе марковских процессов и уравнений Колмогорова.

3 Постановка задачи

3.1 Математическая постановка

Имеется размер скользящего окна w в момент времени $t > 0$. Моменты времени, когда изменяется w, образуют собой последовательнос[illegible] Минимальный размер окна равен 2. Последовательность w_i образует марковскую цепь [5]. Ступенчатый случайный процесс $\{w(t)\}$ является полумарковским случайным процессом. p_{uv}^k - вероятность перехода цепи $\{w_i\}$ из состояния u в состояние v за k шагов.

Распределение π_w, которое является стационарным для марковской цепи $\{w_i\}$, удовлетворяет уравнениям Колмогорова, приведенным в [5].

3.2 Определение распределения

В данном пункте приведен алгоритм на псевдоязыке программы, определяющей распределение размера скользящего окна. Код программы приведен в Приложении 1:

- Определение вероятностей потери пакетов
- Вычисление рекуррентных соотношений
- Вычисление распределения

3.3 Примеры

В этом разделе приводятся графические примеры распределения размера скользящего окна с использованием разработанного приложения. На рис. 2 представлено распределение размера окна π_w для вероятности потери пакета $p = 0.0006$ и $w_{max} = 120$.

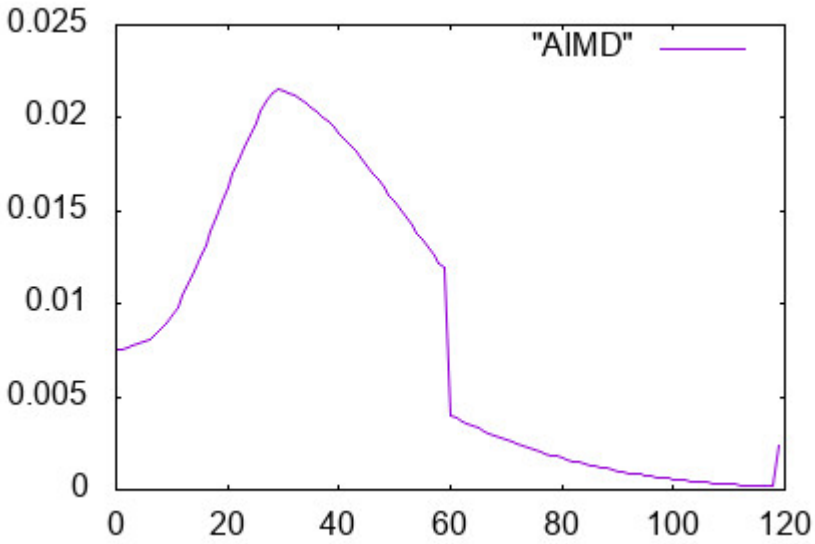

Рис. 2. Распределение π_w

На рис. 3 представлено распределение для вероятности $p = 0.001$ и $w_{max} = 120$.

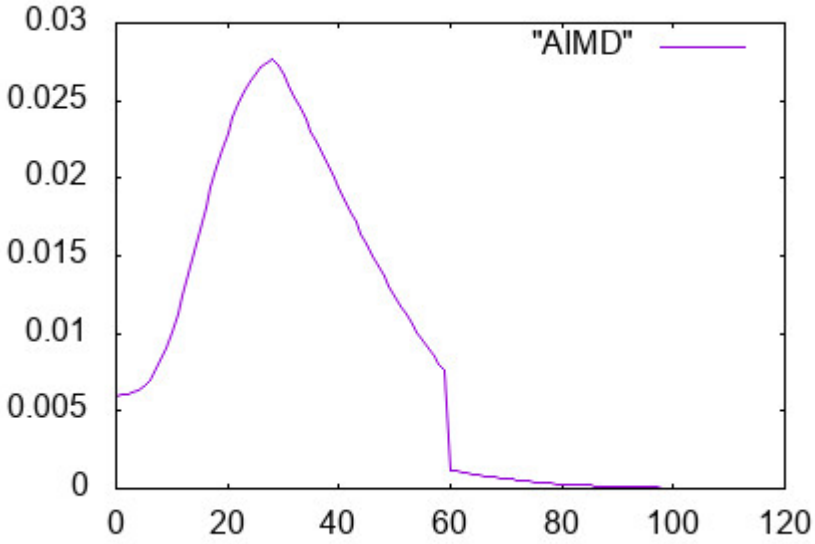

Рис. 3. Распределение π_w

На рис. 4 представлено распределение для вероятности $p = 0.0016$ и $w_{max} = 120$.

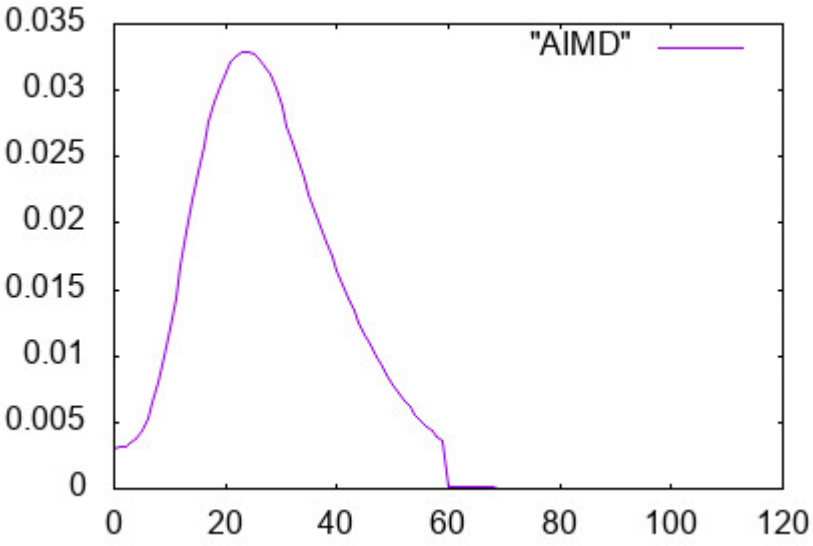

Рис. 4. Распределение π_w

На рис. 5 представлено распределение для вероятности $p = 0.002$ и $w_{max} = 120$.

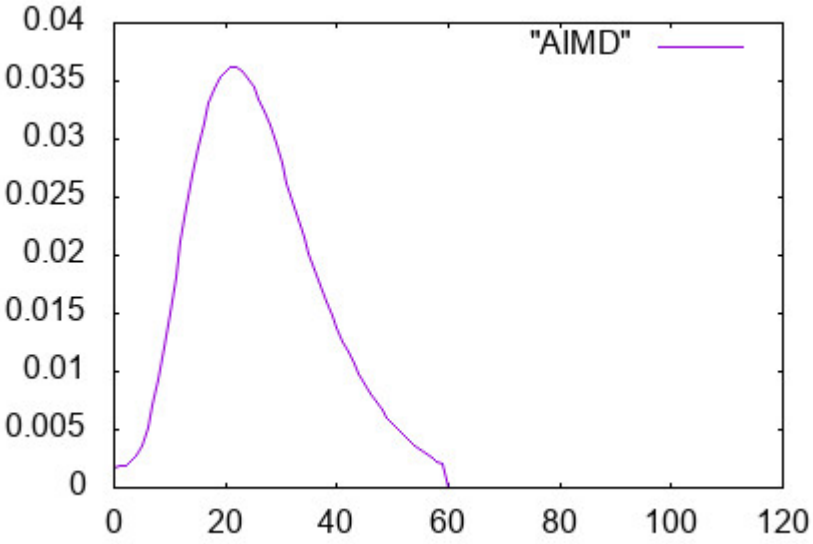

Рис. 5. Распределение π_w

Можно заметить, что при более высоких значениях вероятности потери пакета поведение окна становится неустойчивым и распределение теряет экстремум в точке $w_{max}/2$.

4 Модель пуассоновского потока

Для дальнейшей работы модель распределения размеров скользящего окна, указанная в пункте [4.2] настоящей работы, была изменена в соответствии с моделью пуассоновского потока. Иначе говоря, в новой модели распределение размера подчиняется не линейному алгоритму, а распределению Пуассона.

Для сравнения были выбраны нормальное и непрерывное равномерное распределения. Для обоих случаев были выбраны плотности вероятностей: $e^{-\frac{(x-\varphi)^2}{2\sigma^2}} \frac{1}{\sigma\sqrt{2\pi}}$ - в первом случае, $(b-a)^{-1}$ - во втором случае.

Соответственно, функция поиска распределения претерпела некоторые изменения, которые приведены в Приложении 2:

- Вычисление интеграла по формуле Симпсона
- Вычисление рекуррентных соотношений
- Нахождение распределения

Программа, которая находит распределение размера скользящего окна для непрерывного распределения, указана в Приложении 3.

4.1 Примеры и сравнение

В данном разделе приводятся графические примеры распределения размера скользящего окна с использованием нового функционала приложения, а также сравнительный анализ нового приложения с прошлым.

Так как потери пакетов происходят при значении w, то можно вывести зависимость между вероятностью потери пакета p и множителем λ.

Для вычисления зависимости используется следующая формула: $RTT\frac{w}{2} = \frac{1}{\lambda}$, где RTT - длина двойного пути, используется среднее значение (30 мс).

Так как вероятность потери пакета и размер окна связаны формулой $\frac{3w^2}{4} = \frac{1}{p}$, то, проведя вычисления, получаем зависимость λ от p: $\lambda = \frac{\sqrt{3p}}{RTT}$.

В таблице 1 приведены результаты вычислений мат. ожидания и дисперсии при различных вводных данных, где:

- w_{max} - максимальный размер окна
- p - вероятность потери пакета
- λ - константа-множитель
- E - мат. ожидание
- σ - среднеквадратическое отклонение

Параметры	ЛРСУ		Равномерное		Нормальное	
	E	σ	E	σ	E	σ
$w_{max} = 30$, $p = 0.0025$, $\lambda = 0.0086$	19.41	8.63	6.79	4.94	22.53	8.14
$w_{max} = 30$, $p = 0.005$, $\lambda = 0.012$	13.16	7.43	5.35	3.6	21.26	8.50
$w_{max} = 30$, $p = 0.025$, $\lambda = 0.027$	6.98	3.24	3.33	1.69	17.82	8.25
$w_{max} = 30$, $p = 0.05$, $\lambda = 0.039$	5.15	2.59	2.85	1.22	16.98	7.11
$w_{max} = 60$, $p = 0.0025$, $\lambda = 0.0086$	18.30	9.64	6.77	5.00	31.04	17.42
$w_{max} = 60$, $p = 0.005$, $\lambda = 0.012$	15.13	6.80	5.35	3.6	27.64	16.70
$w_{max} = 60$, $p = 0.025$, $\lambda = 0.027$	7.62	3.57	3.33	1.69	20.89	12.69
$w_{max} = 60$, $p = 0.05$, $\lambda = 0.039$	5.17	2.61	2.85	1.22	19.81	9.83
$w_{max} = 120$, $p = 0.0025$, $\lambda = 0.0086$	24.87	10.51	6.77	5.00	33.55	23.90
$w_{max} = 120$, $p = 0.005$, $\lambda = 0.012$	17.85	7.73	5.35	3.60	28.96	20.97
$w_{max} = 120$, $p = 0.025$, $\lambda = 0.027$	7.62	3.56	3.33	1.69	21.19	13.62
$w_{max} = 120$, $p = 0.05$, $\lambda = 0.039$	5.21	2.56	2.85	1.22	20.01	10.19

4.1.1 Примеры при $w = 30$

В данном разделе приводятся сравнительные графики при $w = 30$.

На рис. 6 представлено сравнение результатов при значении $p = 0.0025$ и $w_{max} = 30$.

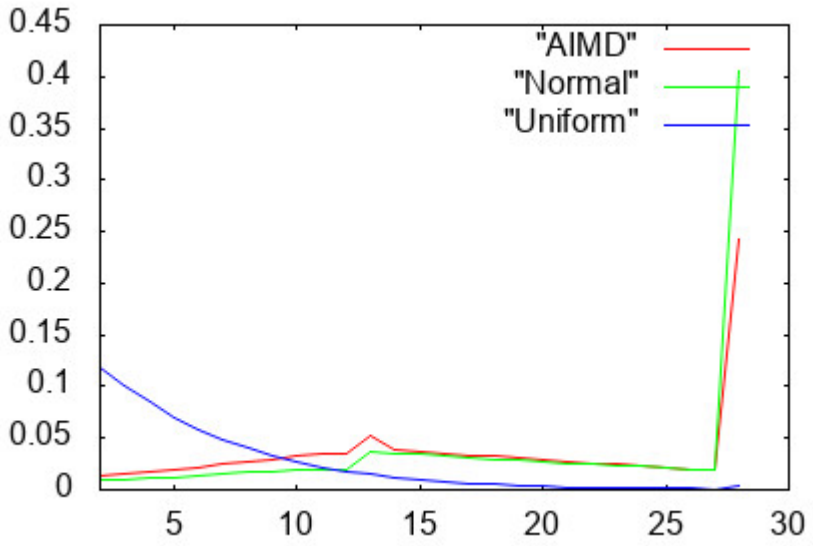

Рис. 6. Распределение π_w

На рис. 7 представлено сравнение результатов при значении $p = 0.005$ и $w_{max} = 30$.

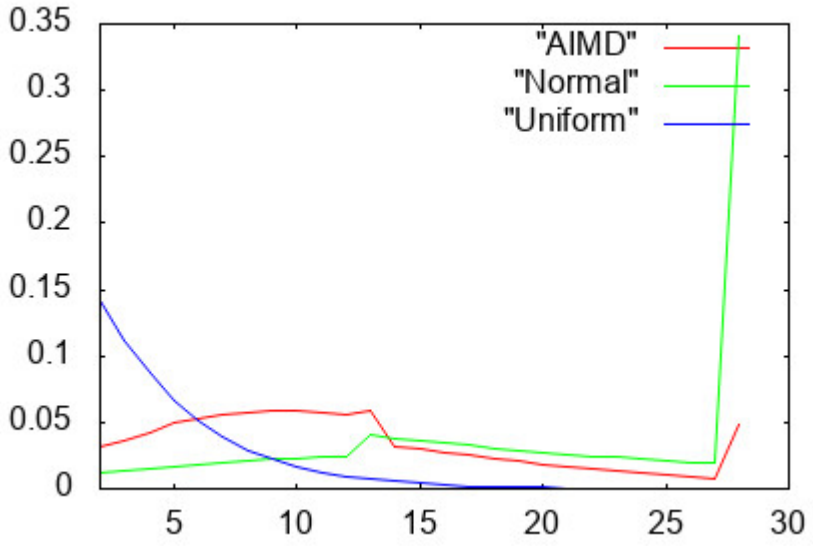

Рис. 7. Распределение π_w

На рис. 8 представлено сравнение результатов при значении $p = 0.025$ и $w_{max} = 30$.

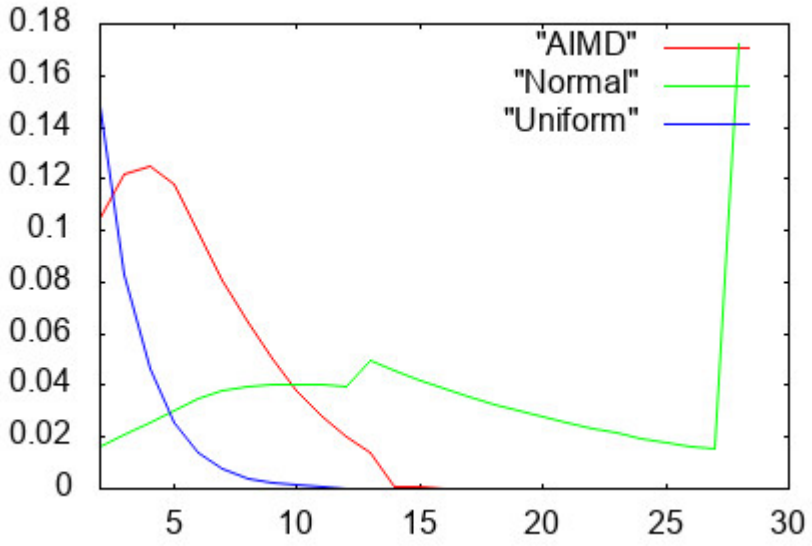

Рис. 8. Распределение π_w

На рис. 9 представлено сравнение результатов при значении $p = 0.05$ и $w_{max} = 30$.

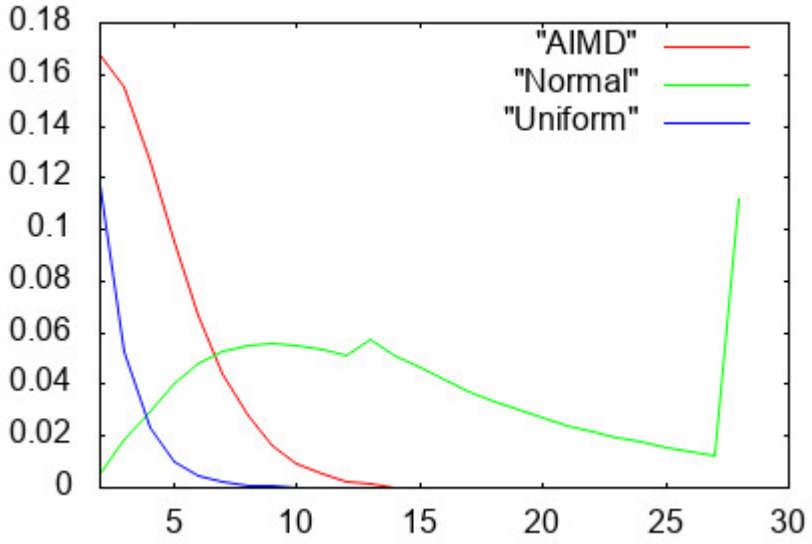

Рис. 9. Распределение π_w

4.1.2 Примеры при $w = 60$

В данном разделе приводятся сравнительные графики при $w = 60$.

На рис. 10 представлено сравнение результатов при значении $p = 0.0025$ и $w_{max} = 60$.

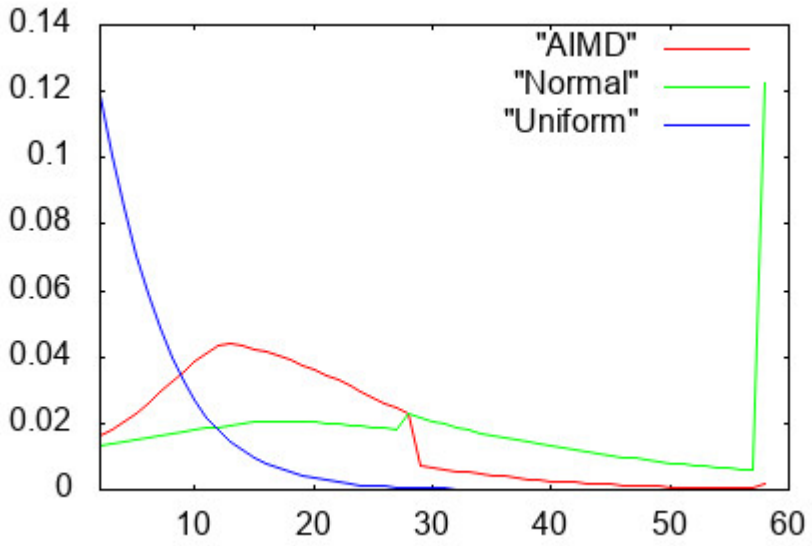

Рис. 10. Распределение π_w

На рис. 11 представлено сравнение результатов при значении $p = 0.005$ и $w_{max} = 60$.

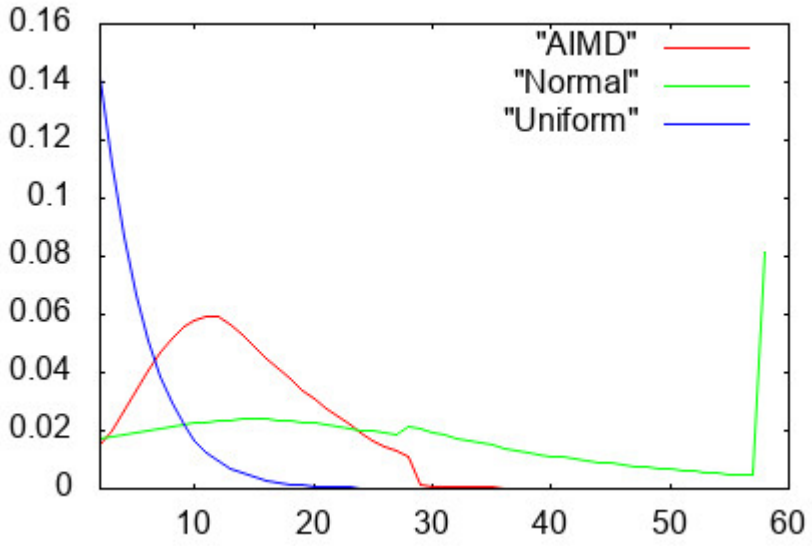

Рис. 11. Распределение π_w

На рис. 12 представлено сравнение результатов при значении $p = 0.025$ и $w_{max} = 60$.

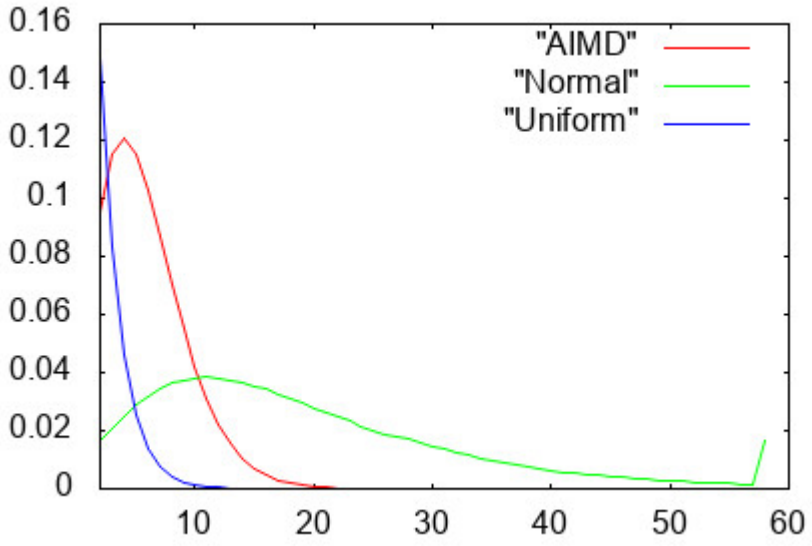

Рис. 12. Распределение π_w

На рис. 13 представлено сравнение результатов при значении $p = 0.05$ и $w_{max} = 60$.

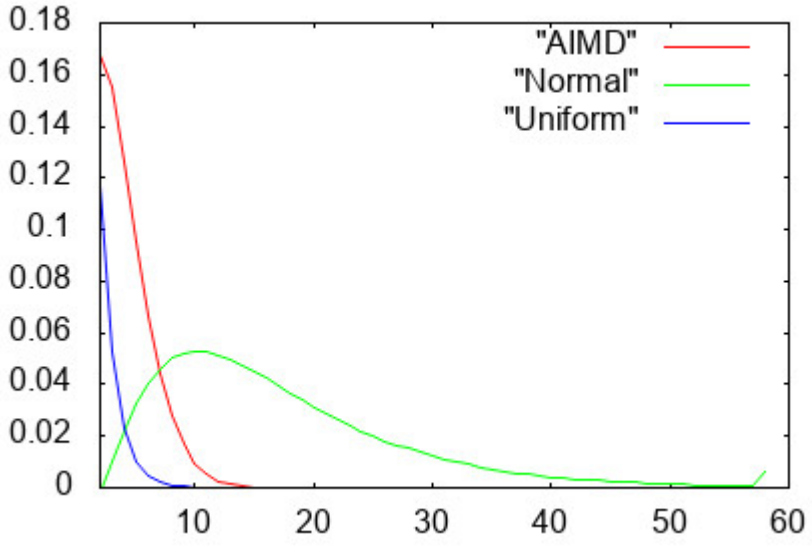

Рис. 13. Распределение π_w

4.1.3 Примеры при $w = 120$

В данном разделе приводятся сравнительные графики при $w = 120$.

На рис. 14 представлено сравнение результатов при значении $p = 0.0025$ и $w_{max} = 120$.

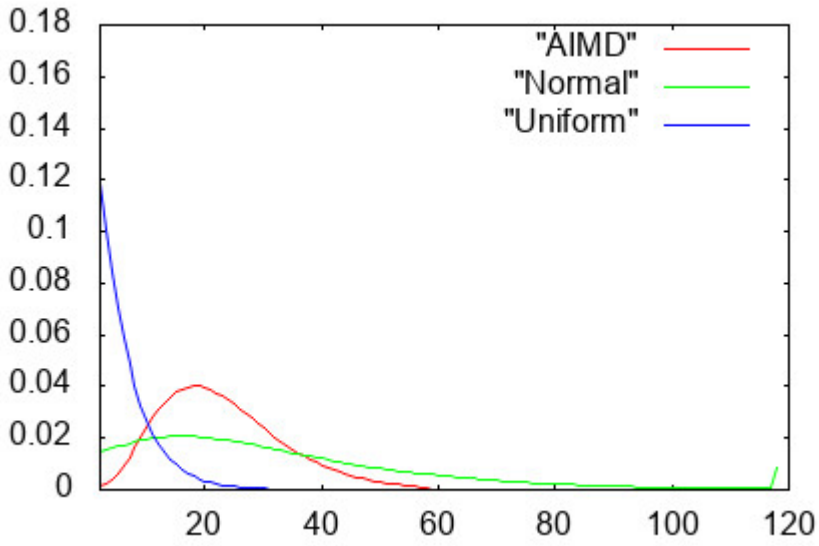

Рис. 14. Распределение π_w

На рис. 15 представлено сравнение результатов при значении $p = 0.005$ и $w_{max} = 120$.

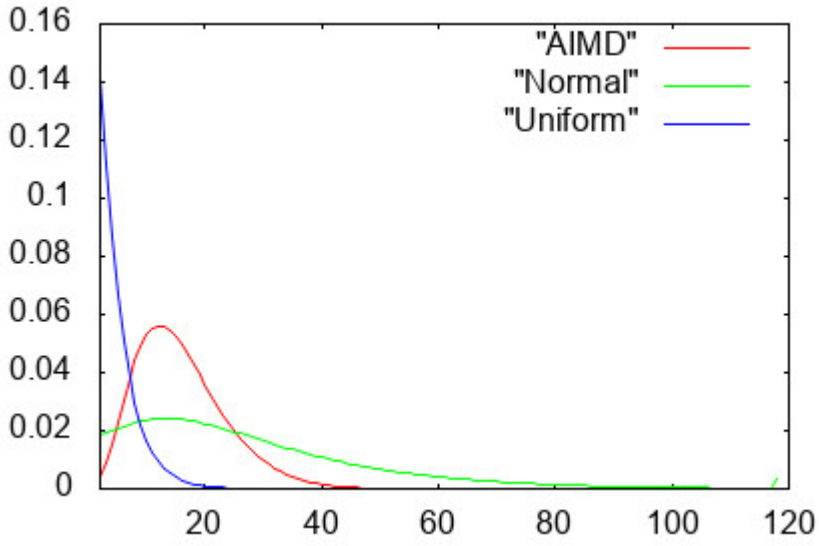

Рис. 15. Распределение π_w

На рис. 16 представлено сравнение результатов при значении $p = 0.025$ и $w_{max} = 120$.

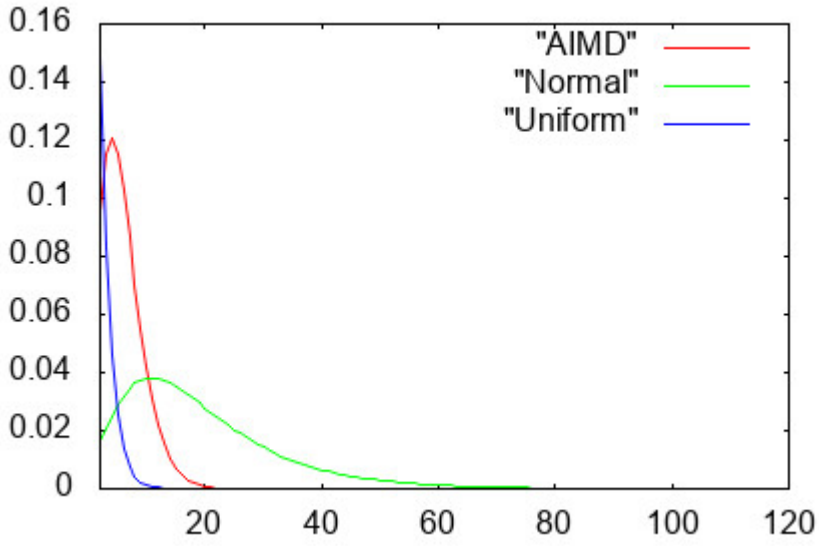

Рис. 16. Распределение π_w

На рис. 17 представлено сравнение результатов при значении $p = 0.05$ и $w_{max} = 120$.

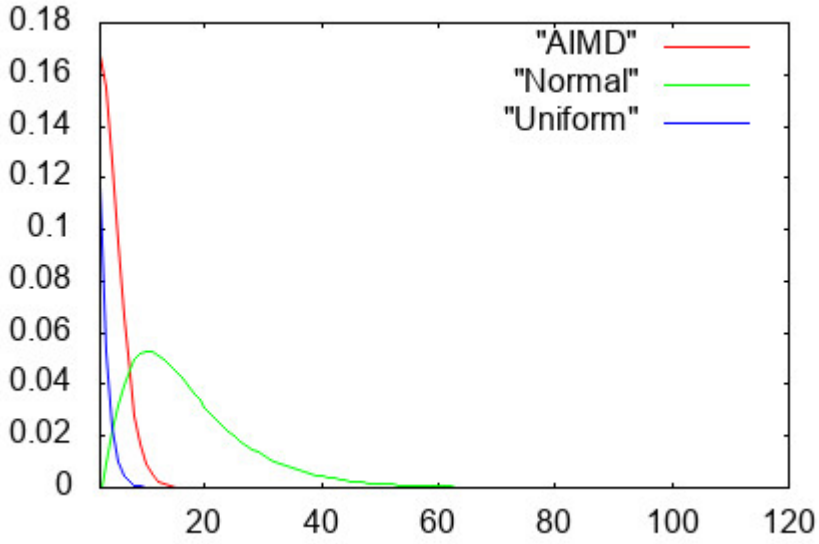

Рис. 17. Распределение π_w

5 Заключение

В ходе исследования были изучены основные принципы работы современных сетей передачи данных. Также был изучен принцип работы алгоритма ЛРСУ. На основе полученных знаний было разработано приложение, позволяющее находить распределение размера скользящего окна.

Было разработано приложение, которое также находит распределение размера скользящего окна, но уже для других вводных данных и другой модели поведения передачи пакетов. В модели пуассоновского потока были получены результаты в нормальном и равномерном распределениях. Полученные данные наглядно показывают разницу математических моделей, показывая как преимущества, так и недостатки каждой модели.

В ходе исследований были получены результаты для различных функций распределения, также был проведен сравнительный анализ этих результатов.

Полученные навыки позволяют взаимодействовать с сетью передачи данных, например, управлять сетевым трафиком.

Приложение 1

На основе мат. модели [5], было разработано приложение, определяющее распределение размера скользящего окна. Функция была запрограммирована на языке C следующим образом:

```
j = wmax / 2;
s[j] = 1.0;

//Определение вероятностей потери пакетов
for (i = 1; i <= wmax; i++)
    k = 1;
    q = 1 - p;
    if (i == 1) {
        f[i] = q;
    } else {
        f[i] = q;
        while (k < i) {
            f[i] *= q;
            k++;
        }
    }
}

//Вычисление доп. переменных для рекуррентного представления распределения
for (i = 1; i <= wmax; i++) {
    k = i;
    if (i == 1) {
        F[i] = f[1];
    } else {
        F[i] = f[i];
        while (k > 1) {
```

```
            F[i] *= f[k - 1];
            k--;
        }
    }
}

//Получение нужных соотношений для получения распределения
K[wmax] = (F[wmax - 1]) / (1 - f[wmax]);
s[wmax] = K[wmax];
s[j] = K[j] = 1.0;

i = wmax - 1;
while (j < i && i < wmax) {
    K[i] = F[i - 1];
    s[i] = K[i];
    i--;
}

i = j;
while (i > 1) {
    K[i - 1] = (1 / (f[i - 1])) * (K[i] - (K[2 * i] * (1 - f[2 * i]) +
        + K[2 * i + 1] * (1 - f[2 * i + 1])));
    s[i - 1] = K[i - 1];
    i--;
}

for (i = 1; i <= wmax; i++) {
    sum += s[i];
}

//Вычисление искомого распределения
```

```
pi[j] = 1 / sum;


for (i = 1; i <= wmax; i++) {
    if (i == j) {
        continue;
    } else {
        pi[i] = pi[j] * K[i];
    }
}
```

7 Приложение 2

Для вычисления интеграла использовалась формула Симпсона, а само равномерное распределение было закодировано следующим образом:

```
#define N 10000
double a = 0, b = 50, h = 0.0, S = 0.0;

//Подинтегральная функция
double FU (double x, double lm) {
    double f;
    f = exp((-x) * lm) * (1 / b);
    return f;
}

//Функция, вычисляющая интеграл по формуле Симпсона
double FUNC (double x, double lm)
{
    S = 0;
    a = 0;
    b = 50;
    h = (b - a)/N;
    x = a + h;
    while (x < b)
    {
        S = S + 4 * FU(x, lm);
        x = x + h;
        if (x >= b) break;
        S = S + 2 * FU(x, lm);
        x = x + h;
    }
    S = (h/3) * (S + FU(a, lm) + FU(b, lm));
```

```
    return S;
}

...

j = wmax/2;
s[j] = 1.0;

integral = FUNC (x, lm);
int n = j - 1;

F[j - 1] = 1;

//Вычисление рекуррентных соотношений
for (i = j; i <= wmax - 1; i++) {
    F[i] = F[i - 1] * FUNC(0, lm);
}

for (i = j + 1; i <= wmax - 1; i++) {
    K[i] = F[i - 1];
}

K[wmax] = (F[wmax - 1]) / (1 - FUNC(wmax, lm));
K[j] = 1;
K[j - 1] = (1 - F[2 * j - 1]) / (FUNC(j - 1, lm));

for (i = j - 1; i > 2; i--) {
    K[i - 1] = (1 / (FUNC(i - 1, lm)) * (K[i] - (K[2 * i] *
        (1 - FUNC(2 * i, q)) + K[2 * i + 1] * (1 - FUNC(2 * i + 1, q)))));
}
```

...

8 Приложение 3

Для нормального распределения была изменена подинтегральная функция:

```
double FU (double x, double lm) {
    double f;
    f =  exp((-x) * lm) * ((1 / (sqrt(pi))) *
        exp((-0.5) * (x - 2) * (x - 2)));
    return f;
}
...
```

Остальные участки кода остались без изменений.

Список литературы

1. Олифер В.Г., Олифер Н.А. *Компьютерные сети. Принципы, технологии, протоколы*: Учебник для вузов. 3-е изд. — СПБ.: Питер, 2008. — 958 с.

2. Феллер В. *Введение в теорию вероятностей и ее приложения. В 2-х томах. Т. 1*: Пер. с англ. — М.: Мир, 1984. — 528 с., ил.

3. Гнеденко Б.В. *Курс теории вероятностей*: Учебник — Изд. 6-е, перераб. и доп. — М.: Наука. Гл. ред. физ.-мат. лит., 1988. — 448 с.

4. Вентцель Е.С. *Теория вероятностей*: Учеб. для вузов. — 7-е изд. стер. — М.: Высш. шк., 2001. — 575 с.

5. Богоявленская О.Ю., Анализ случайного потока, генерируемого транспортным протоколом с обратной связью, в сети передачи данных, *Автомат. и телемех.*, 2003, № 12, 60-68

6. [Электронный ресурс]: http://rain.ifmo.ru/cat/view.php/theory/processes-automata/markov-2008/

7. [Электронный ресурс]: https://www.sviaz-expo.ru/ru/articles/2016/seti-peredachi-dannyh/

8. [Электронный ресурс]: https://ru.wikipedia.org/w/index.php?title=TCP/IP

9. [Электронный ресурс]: http://ciscotips.ru/tcp

10. [Электронный ресурс]: https://ru.wikipedia.org/wiki/Transmission_Control_Protocol

11. [Электронный ресурс]: http://ccna.mpei.ac.ru/IntroductionToNetworkTech/

12. [Электронный ресурс]: https://en.wikipedia.org/wiki/Sliding_window_protocol