# Введение

**Актуальность области исследования**. В наши дни производительность сетей передачи данных достигла показателей, позволяющих передавать аудио- и видеоданные реального времени. Количество таких сервисов неумолимо растет и все больше операторов связи предоставляют такие услуги своим клиентам. В связи с валовым ростом интереса к получению данных услуг, актуальной становится задача адекватной оценки предоставляемого провайдером сервиса. Эта задача особенно актуальна в случае предоставления мультимедийного сервиса реального времени, который в значительной степени чувствителен к задержкам передачи трафика.

Необходимо отметить, что к настоящему времени теория оценки параметров качества предоставляемого мультимедийного сервиса в пакетных сетях недостаточно развита. Некоторые критические параметры состояния сети, такие как, например, вариация задержки в канала, не имеют однозначного способа оценки, что говорит об актуальности области исследования.

**Объектом исследования** являются известные методы подсчета вариации задержки в компьютерных сетях при передаче голосового и видеотрафика.

**Предметом** исследования является изучение влияния выбора методики расчёта вариации задержки на конечные метрики оценки качества канала связи..

**Цель исследования** – провести сравнительный анализ методов расчёта вариации задержки в компьютерных сетях и, на основе проведенного анализа, доказать необходимость стандартизации этих методик.

Для достижения цели исследования решались следующие **задачи**:

1. Выполнить аналитический обзор литературы, посвящённой проблеме расчета реального значения вариации задержки в компьютерных сетях.
2. Собрать репрезентативные данные о задержках в компьютерных сетях для дальнейших расчетов вариации задержки по выявленным методикам.
3. Провести расчет вариации задержки на основе полученных данных о задержках в компьютерных сетях.
4. Сравнить полученные результаты расчётов.
5. Проанализировав результаты сделать выводы о применимости методов расчета вариации задержки и, возможно, о необходимости их стандартизации.

**Теоретическая и методологическая основа исследования. При** Перечисляются использованные методы и средства исследовании, а также применяемые в работе инструменты исследования: аппарат теории вероятностей, теории массового обслуживания (ТМО), теории случайных процессов, теории планирования машинного эксперимента, методы численного анализа и имитационного моделирования, а также даётся перечень использованных инструментов, например: название IDE, типы СУБД, языки программирования, библиотеки, математическое ПО (Matlab, Mathcad, Octave, R,…) и т.п. (1-2 абзаца). Если тяжело наскрести хотя бы два пункта, можно подобрать что-нибудь из следующего списка: метод обобщений, формально-логический метод, структурно-функциональный анализ, общенаучный метод абстрагирования, метод сравнений и аналогий, метод перехода от общего к частному, от абстрактного к конкретному, от идеального к материальному.

**Актуальность темы исследования.** Здесь нужно обосновать не актуальность области исследования (как было в начале ВВЕДЕНИЯ), а актуальность конкретной выбранной темы исследования. Например, для темы ВКР «Повышение плотности записи информации на перфокартах» очень просто обосновать актуальность области исследования, т.к. область связана с развитием систем хранения данных, что на данный момент очень актуально. Однако обосновать актуальность такой темы исследования будет очень сложно, ведь перфокарты уже давно не используются (1 абзац).

**Степень теоретической разработанности темы**. Описывается степень проработанности темы другими авторами и краткая история развития тематики. Перечислить наиболее авторитетных за последние 10 лет авторов книг или авторов диссертаций (кандидатских и докторских) на выбранную или схожую тему (1 абзац).

**Информационная база исследования**. Написать, на какие конкретно труды и материалы в первую очередь опирался автор ВКР, например: книги, статьи, отчёты, доклады, статистические материалы, исследовательские сайты, репозитории, нормативные акты (законы, кодексы). В отличие от предыдущего, в этом пункте нужно не просто назвать авторов, а указать названия конкретных работ, которые использовались в качестве основы в ВКР (1 абзац).

**Научная новизна исследования.** Показать, что именно сделал автор, чего раньше никто не делал ... (2-3 абзаца).

**Практическая значимость исследования**. Указать, кому, в каком виде и в каких именно областях в практическом смысле пригодятся результаты, полученные автором в ВКР (1 абзац).

**Апробация результатов исследования**. Указать, используются ли где-нибудь на практике полученные в ВКР результаты. Перечислить конференции, на которых делались доклады по теме ВКР. Указать количество опубликованных работ (при наличии) – список названий публикаций писать не надо, его надо поместить в общий список литературы, а здесь просто поставить ссылку в квадратных скобках (1 абзац).

**Объем и структура работы**. Указать краткое содержание каждой главы: по одному длинному предложению на каждую главу. Предложения должны начинаться: «В первой главе раскрывается/описывается/приводится…» (1 абз.).

**ГЛАВА 1. ОБЗОР ЛИТЕРАТУРЫ И ПОСТАНОВКА ЗАДАЧИ ИССЛЕДОВАНИЯ.**

**1.1 История развития области исследования.**

История сетей коммуникации насчитывает более полутора веков, начиная свой отсчет с изобретения телеграфа. С тех пор технологии значительно шагнули далеко вперед. Структура сети полностью менялась с приходом новых технологий.

Более ста лет назад, телефонные сети общего пользования начали формировать всемирную сеть с коммутацией каналов. Полоса пропускания такой сети была фиксированной, что идеально подходило для передачи трафика реального времени, такого как голос.

Спустя 50 лет, специалисты представили миру сеть с коммутацией пакетов. Такая сеть позволяла делить поток данных на отдельные порции и отправлять их в пункт назначения различными путями.

Появление IP Quality of Service не было неожиданностью. Еще отцы основатели Интернета предвидели его необходимость и включили поле Type of Service в заголовок IP-пакета для дальнейшего облегчения внедрения QoS. Спецификация протокла IP описывает поле ToS, как индикатор желаемого качества обслуживания. Этот параметр используется для выбора параметров передачи датаграммы через ту или иную сеть.

В конце 80х годов 20го века Интернет использовался лишь в академических и военных целях, что означало ограниченное количество передаваемых через него данных. Поэтому в поле ToS не было необходимости, и почти все реализации протокла IP игнорировали его. При отправке пакета это поле оставалось пустым, а маршрутизаторы не использовали его при выборе дальнейшего маршрута до пункта назначения.

Необходимость в QoS для сети Интернет возрастала с ее выходом из академической среды в более широкое использование в коммерческих и частных целях. Для передачи данных в Интернет используется стек TCP/IP, что делает его гибким и надежным. Но несмотря на это, сеть подвержена перегрузкам. Чаще всего в роли так нзываемого «бутылочного горлышка» выступают маршрутизаторы, соединяющие две сети с различной пропускной способностью. Этой проблемой заинтересовался Джон Нейгл в середине 80х годов.

Документ Нейгла, Управление Перегрузкой IP/TCP Сетей (RFC 896) описывает то, что он назвал «проблемой небольших пакетов», которая заключается в том, что приложение неоднократно посылает данные небольшими порциями, часто размером в 1 байт. Так как TCP пакеты имеют 40 байт заголовка (20 байт TCP, 20 байт IPv4), это приводит к тому, что передается пакет размером 41 байт, несущий в себе 1 байт полезной информации, то есть к огромным накладным расходам. Эта ситуация часто встречается в сессии Telnet, где большинство нажатий клавиш генерируют один байт данных, который немедленно передается. Кроме того, по медленным каналам связи многие такие пакеты могут находиться в пути в одно и то же время, что может привести к перегруженности сети.

Алгоритм Нейгла работает путём объединения нескольких небольших исходящих сообщений, а затем отправки их всех сразу. В частности, пока существует отправленный пакет, для которого отправитель еще не получил никакого подтверждения о доставке, отправитель должен держать в буфере следующие данные для отправки, до тех пор, пока не наберется достаточно данных на полный пакет, который можно отправить единожды. Данный алгоритм считается первой вехой в истории QoS.

В 1986 году американский ученый Ван Якобсон разработал новый набор инструментов QoS — механизмы предотвращения перегрузок, которые по сей день используются в реализации протокола TCP. Этими механизмами являются медленный старт и алгоритм контроля перегрузок, которые помогли сети Интернет избежать коллапса при значительном росте трафика в 1988—1989 годы. Еще два механизма, быстрая повторная передача и быстрое восстановление, были добавлены для обеспечения более высокой производительности при потере пакетов.

Несмотря на то, что описанные выше механизмы в значительной мере увеличивали производительность сети, оставалась проблема перегрузок при маршрутизации. Именно поэтому в 1990 году внимание исследователей было сконцентрировано вокруг маршрутизаторов. Когда передача данных сталкивается с проблемой «бутылочного горлышка» для приёма и отправки пакетов на маршрутизаторах, то обычно используется метод FIFO: первый пришел - первый ушёл (First In — First Out). При интенсивном трафике это создаёт заторы, которые разрешаются крайне простым образом: все пакеты, не вошедшие в буфер очереди FIFO (на вход или на выход), игнорируются маршрутизатором и, соответственно, теряются безвозвратно. Алгоритмы взвешенной справедливой очереди и взвешенного произвольного раннего обнаружения были разработаны для решения этой проблемы.

Следующим этапом развития стало изобретение сквозного (на всём пути следования трафика) качества обслуживания. Архитектура Intserv, разработанная инженерным советом Интернета, была призвана предоставить средства для передачи в сеть своих требований к качеству обслуживания трафика. С целью сообщения маршрутизаторам сети потребностей конечных узлов по качеству обслуживания потоков был разработан дополнительный протокол — RSVP. Хотя в середине 90-х годов прошлого века идея IntServ и RSVP вызывала большие надежды, со временем интерес к этой архитектуре угас. Главной причиной стала проблема масштабируемости, вызванная необходимостью хранить и поддерживать информацию о состоянии передачи в каждом маршрутизаторе. Эта проблема, переносимая на такие глобальные сети, как Интернет, отдаляет RSVP от реальности.

Одной из последних вех в QoS является внедрение модели DiffServ. Дифференцированое обслуживание предполагает разделение трафика на классы на основе требований к качеству обслуживания. Каждый класс трафика дифференцируется и обрабатывается сетью в соответствии с заданными для этого класса механизмами QoS. Подобная схема обеспечения качества обслуживания (QoS) довольно часто называется схемой CoS.

Архитектура DiffServ подразумевает определение поля DiffServ (DS), которое заменяет поле типа обслуживания в протоколе IPv4, используемое при принятии решений о пересылке данных через каждый узел для классификации пакетов и функций согласования трафика, например таких, как измерение, маркирование, формирование и контроль.

Стандарты RFC не устанавливают метода реализации пересылок данных через каждый узел, перенося всю ответственность за это на производителя оборудования. Компания Cisco реализует технологию организации постановки в очередь, которая может основывать пересылку данных через каждый узел на базе IP-приоритетов или значений кода DSCP в заголовке IP-пакета. Основываясь на коде DSCP или IP-приоритете, трафик может быть вставлен в определенный класс обслуживания. Пакеты в рамках какого-либо класса обслуживания обрабатываются одинаково.

Дифференцированное обслуживание удобно применять в сетях с интенсивным

трафиком приложений. В этом случае важно обеспечить отделение административного трафика сети от всего остального трафика и назначить ему приоритет, позволяющий в любой момент времени быть уверенным в связности узлов сети.

**1.2. Основные характеристики канала связи**

Внедрение механизмов QoS предполагает обеспечение со стороны сети соединения с определенными ограничениями по производительности. Основными характеристиками производительности сетевого соединения являются полоса пропускания, задержка, вариация задержки и уровень потери пакетов.

**1.2.1 Полоса пропускания**

Термин полоса пропускания (bandwidth) используется для описания номинальной пропускной способности среды передачи информации, протокола или соединения. Этот термин достаточно эффективно определяет “ширину канала”, требующуюся приложению для взаимодействия по сети. Как правило, каждое соединение, нуждающееся в гарантированном качестве обслуживания, требует от сети резервирования минимальной полосы пропускания. К примеру, приложения, ориентированные на передачу оцифрованной речи, создают поток информации интенсивностью 64 Кбит/с. Эффективное использование таких приложений становится практически невозможным вследствие снижения полосы пропускания ниже 64 Кбит/с на каком-либо из участков соединения.

**1.2.2 Потеря пакетов**

Уровень потери пакетов (packet loss) определяет количество пакетов, отбрасываемых сетью во время передачи. Основными причинами потери пакетов являются перегрузка сети и повреждение пакетов во время передачи по линии связи. Чаще всего отбрасывание пакетов происходит в местах перегрузки, где число поступающих пакетов намного превышает верхнюю границу размера выходной очереди. Кроме того, отбрасывание пакетов может быть вызвано недостаточным размером входного буфера.

Как правило, уровень потери пакетов выражается как доля отброшенных пакетов за определенный интервал времени.

**1.2.3 Задержка и джиттер при передаче пакетов**

Задержка при передаче пакета (packet delay), или латентность (latency), на каждом переходе состоит из задержки сериализации, задержки распространения и задержки коммутации. Ниже приведены определения каждого из названных выше типов задержки.

1. Задержка сериализации (serialization delay). Время, которое требуется устройству на передачу пакета при заданной ширине полосы пропускания. Задержка сериализации зависит как от ширины полосы пропускания канала передачи информации, так и от размера передаваемого пакета. Например, передача пакета размером 64 байт при заданной полосе пропускания 3 Мбит/с занимает всего лишь 171 нс. Обратите внимание, что задержка сериализации очень сильно зависит от полосы пропускания: передача того же самого пакета размером 64 байт при заданной полосе пропускания 19.2 Кбит/с занимает уже 26 мс. Довольно часто задержку сериализации называют еще задержкой передачи (transmission delay).
2. Задержка распространения (propagation delay). Время, которое требуется пере-анному биту информации для достижения принимающего устройства на другом конце канала. Эта величина довольно существенна, поскольку в наилучшем случае скорость передачи информации соизмерима со скоростью света. Обратите внимание, что задержка распространения зависит от расстояния и используемой среды передачи информации, а не от полосы пропускания. Для линий связи глобальных сетей задержка распространения измеряется в миллисекундах. Для трансконтинентальных сетей Соединенных Штатов характерна задержка распространения порядка 30 мс.
3. Задержка коммутации (switching delay). Время, которое требуется устройству, получившему пакет, для начала его передачи следующему устройству. Как правило, это значение меньше 10 нс. Обычно каждый из пакетов, принадлежащий одному и тому же потоку трафика, передается с различным значением задержки. Задержка при передаче пакетов меняется в зависимости от состояния промежуточных сетей.

В том случае, если сеть не испытывает перегрузки, пакеты не ставятся в очередь в маршрутизаторах, а общее время задержки при передаче пакета состоит из суммы задержки сериализации и задержки распространения на каждом промежуточном переходе. В этом случае можно говорить о минимально возможной задержке при передаче пакетов через заданную сеть. Следует отметить, что задержка сериализации становится незначительной по сравнению с задержкой распространения при передаче пакета по каналу с большой пропускной способностью. Если же сеть перегружена, задержки при организации очередей в маршрутизаторах начинают влиять на общую задержку при передаче пакетов и приводят к возникновению разницы в задержке при передаче различных пакетов одного и того же потока. Изменение задержки при передаче пакетов получило название джиттера (packet jitter). Джиттер имеет большую важность, поскольку именно он определяет максимальную задержку при приеме пакетов в конечном пункте назначения. Принимающая сторона, в зависимости от типа используемого приложения, может попытаться компенсировать вариацию задержки за счет организации приемного буфера для хранения принятых пакетов на время, меньшее или равное верхней границе джиттера.

К этой категории относятся приложения, ориентированные на передачу/прием непрерывных потоков данных, например Internet-телефония или приложения, обеспечивающие проведение видеоконференций.

Причиной джиттера могут быть разные факторы, обусловленные спецификой передачи информации в пакетных сетях: это и большая загрузка узлов коммутации, где пакеты выстраиваются в очередь и ожидают своей отправки в следующую точку агрегации и маршрутизации сетевого трафика, это и диверсификация маршрутов следования пакетов (пакеты, следовавшие друг за другом в источнике, могут быть переданы разными маршрутами к получателю, а значит, и вероятно обработаны разным количеством сетевых узлов, каждый из которых вносит свою, случайную задержу обработки пакета).

**1.3 Методы расчета вариации задержки**

При заключении договора на предоставление услуг связи между поставщиком и клиентом заключается договор SLA (Service Level Agreement), в котором явно указывается то качество услуг, которое поставщик обязан предоставить потребителю. Не последнюю роль в данном соглашении играют метрики QoS, в том числе и вариация задержки. В документе RFC5481 упоминается, что обычно требования к джиттеру формулируются в виде количества пакетов имеющих джиттер не больший, чем установлено соглашением на определённом промежутке времени. Однако, существует множество методов расчёта вариации задержки, которые будут рассмотрены в следующем пункте.

**1.3.1. Inter-Packet Delay Variation**

Самым простым известным методом измерения джиттера является метод измерения вариации задержки между двумя следующими друг за другом пакетами.

IPDV(i) = Delay(i) – Delay(i-1),

где Delay(i) – односторонняя задержка пакета, для которого производится измерение, а Delay(i-1) – задержка при передаче предыдущего пакета.

Значения IPDV могут быть как положительными, так и отрицательными, но несмотря на это принято брать абсолютное значение.

**1.3.2 Packet Delay Variation**

Термин Packet Delay Variation (PDV) впервые был определен в документе Y.1540 : Internet protocol data communication service - IP packet transfer and availability performance parameters. Эти термином обозначается разница задержек пакета, для которого производится расчёт, и минимальной задержки пакета на всем интервале измерений.

PDV рассчитывается для пары, где одним из пакетов является пакет, для которого происходит вычисление, а вторым является так называемый эталонный пакет.

PDV(i) = Delay(i) – Delay(ref),

В Y.1540 приводится несколько способов выбора эталонного значения задержки:

* минимальная задержка;
* средняя задержка на всем интервале измерения;
* задержка при передаче первого пакета.

**1.3.3.Рассчет джиттера в протоколе RTP**

RTP – протокол передачи данных реального времени. Данный протокол широко используется для передачи аудио- и видеотрафика в таких программных продуктах, как, например, Astesisk. На практике протокол RTP не отделим от протокола RTCP (RTP control protocol). Последний служит для мониторинга QoS и для передачи информации об участниках обмена в ходе сессии. [RFC1889].

Протокол RTCP так же содержит в себе механизм расчета вариации задержки. Джиттер рассчитывается на основании предыдущего значения и настоящего значения задержки пакета.

Джиттер *Ji* для *i*-го пакета определяется по формуле:

где:

– отклонение от ожидаемого времени прибытия i-го пакета.

Отклонение от ожидаемого времени прибытия i-го пакета определяется по формуле:

, где:

R – время прибытия пакета в метках времени RTP,

*S* – временная метка RTP, взятая из пакета.

**1.4. Выводы по главе**

1. История области исследования берет начало в 80х годах прошлого века и до сих пор актуальна.
2. Среди всех метрик QoS вариация задержки является особо важной при передаче трафика реального времени.
3. На данный момент существует 5 методов расчёта джиттера и не имеется единого стандарта, на который можно опираться при описании качества предоставляемых поставщиком услуг передачи данных.

**Глава 2. Описение метода сбора данных сетевых задержек и рассчета вариации задержки различными методами**

Для достижения целей выпускной квалификационной работы был разработан стенд по расчету вариации задержки различными методами, который берез в качестве входных данных реальные значения задержек, измеренные при помощи инструмента Scamper.

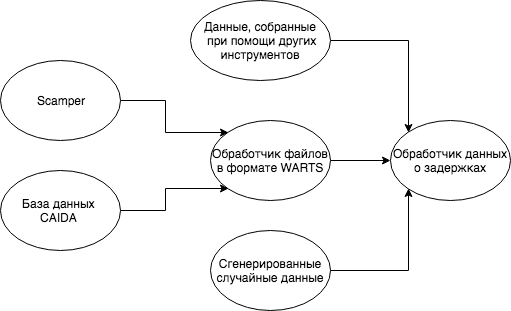


Рисунок \* - Схема экспериментального стенда

Так же данные о задержках можно либо собирать при помощи других инструментов и конвертировать их в формат, используемый обработчиком, либо сгенерировать случайные данные. Однако, при проведении экспериментов были использованы дампы данных в формате warts с открытого сайта научной организации CAIDA. Далее warts файлы поступают на обработку скрипта на языке Python, который генерирует файл, содержащий только значения задержек, по одному значению в каждой строке. Затем запускается скрипт на языке R, который производит расчеты вариации задержки различными методами и их сравнение.

* 1. **Структура компонентов эксперементального стенда**

2.1.1. Орагнизация по исследованию Интернета CAIDA

Center for Applied Internet Data Analysis (CAIDA) – научная организация, основанная в 1997 году, которая проводит исследование состояния компьютерных сетей и строит исследовательскую инфраструктуру для сбора данных и распространения их в научной среде. Исследовательская группа, расположенная в федеральном центре суперкомпьютеров Сан-Диего, проектирует, разворачивает и поддерживает растущее количество сервисов для вычислений, анализа и визуализации данных. Так же исследовательская группа распространяет измерительную аппаратуру и внедряет ее в сети по всему мир. Исследователи CAIDA разрабатывают новые технологии сбора, анализа и визуализации получаемых данных.

CAIDA исследует теоретические и практические аспекты работы Интернета:

* формирование понимая глобальных функций инфраструктуры глобальной сети, ее поведения, использования и эволюции;
* способствование созданию совместной инфраструктуры для распространения и анализа научных данных;
* создание целостного представления о науке об Интернете;
* информирование исследователей, производителей и законодателей в сфере информационных технологий о проблемах глобальной сети.

Исследование, проводимое CAIDA, охватывает топологию, маршрутизацию, безопасность и экономику Интернета. Деятельность CAIDA по разработке программного и аппаратного обеспечения и распространению информации обеспечивает данными не только исследовательский центр университета Сан-Диего, но и ученых по всему миру.

Так же, CAIDA занимается картографией Интернета, составляя карту глобальной сети, используя при этом свою инфраструктуру. Основной задачей является улучшение точности и поддержание карты в актуальном состоянии. Так же исследовательской группой совместно с учеными из Массачусетского Технологического Института ведется работа по нанесению на карту всех соединений между различными провайдерами услуг связи, для исследования перегрузок, вызванных пирингом и работой сервисов CDN.

CAIDA проводит мониторинг безопасности и стабильности глобальной сети и имеет свою открытую базу данных, которые могут быть свободно использованы для исследователями процессов, происходящих в глобальной сети, учеными со всего мира. Так же CAIDA разрабатывает множество инструментов для сбора или анализа таких данных.

* + 1. **Утилита Scamper**

Scamper – это инструмент сбора данных о состоянии сети для дальнейшего анализа ее топологии и производительности. Данное программное обеспечение было разработано программистом из Новой Зеландии Мэттью Лаки для CAIDA при участии ученых из Открытого Центра Инноваций Университета Ваикато и активно используется начиная с сентября 2008 года. Уже к августу 2010 года было активно 48 точек по всему миру под управлением трех команд, из которых производились замеры. Данное программное обеспечение распространяется по стандартной общественной лицензии GNU GPL v2+. В отличие от его предшественника skitter, он позволяет зондирование, как IPv4 адресов, так и IPv6. Данный инструмент позволяет производить измерения множеством методов:

* Traceroute
* Ping
* MDA traceroute
* Alias resolution
* Sting

Изначально scamper был разработан для сбора данных о производительности сетей в рамках проекта исследовательской группы CAIDA для изучения глобальной топологии сети Интернет. Трассировка пакетов в scamper имеет множество функций: она поддерживает обе версии сетевого протокола IP (четвертую и шестую); измерения могут производиться при помощи трех протоколов канального уровня: UDP, TCP и ICMP, такое разнообразие протоколов позволяет получать данные практически из любой конечной точки в большинстве автономных систем.

Метод Ping позволяет измерять сквозные задержки и потерю пакетов при передаче по сети, определять доступность IP адресов и классифицировать поведение хостов, оценивая их ответы на запросы. В дополнение к традиционному методу эхо-запросов по протоколу ICMP, scamper реализует так же измерение посредством UDP, TCP и TTL-limited запросов, которые могут быть использованы в случае, если обычные ICMP запросы остаются без ответа.

Помимо упомянутых выше способов scamper реализует многопутевой алгоритм детектирования, позволяющий определить все интерфейсы на всех возможных путях следования пакета от источника до пункта назначения. Это достигается путем умышленного варьирования идентификатора потока, который маршрутизатор может рассчитывать при балансировке нагрузки. Запросы с разными идентификаторами могут быть отправлены разными маршрутами, тем самыми выявить все узлы, через которые можно достичь пункта назначения. В дополнение к стандартным методам варьирования контрольных сумм ICMP и портов назначения в UDP, в scamper так же реализован метод изменения исходящего порта, что позволяет обходить некоторые сетевые экраны, которые могут блокировать запросы, принимая их за сканирование портов злоумышленниками.

Alias resolution – это метод трассировки, позволяющий определить связи между интерфейсами маршрутизаторов. Данный метод позволяет определить адреса интерфейсов, принадлежащих одному маршрутизатору, данная информация является ключевой при построении карты топологии Интернета.

Scamper позволяет получать выходные данные двух видов: обычный текстовый формат в кодировке ASCII и бинарный. Данные в текстовом виде имеют формат, сходный с выводом стандартных утилит ping и traceroute, и удобны для использования scamper в интерактивном режиме. Бинарный вид удобен для использования исследователями сети, благодаря возможности получения большего количества деталей и удобству хранения в файлах формата warts [warts]. Scamper включает в себя библиотеку для обработки бинарных данных именно в этом формате, а так же, участниками CAIDA была разработана библиотека для языка Ruby, позволяющая писать на нем программы для анализа выходных данных. Так же scamper предоставляет API для реализации новых методов сбора данных и сохранения их в бинарном формате. Библиотека не поддерживает функцию чтения данных из текстового файла, это было сделано намерено, чтобы заставить разработчиков отказаться от использования текстового формата и поощрить использование бинарного.

Пакет scamper включает множество утилит:

* scamper: основная утилита для сбора данных,
* sc\_analysis\_dump: утилита для конвертации данных в формат, удобный для обработки на языке Perl,
* sc\_filterpolicy: драйвер для исследования систем на предмет соответствия политикам фильтрации,
* sc\_remoted: утилита для подключения и сбора данных с удаленных сборщиков данных,
* sc\_tbitblind: драйвер для исследования систем на предмет уязвимости к атаке «слепой подмены»,
* sc\_tbitpmtud: драйвер для исследования систем на предмет чувствительности к слишком большим ICMP пакетам,
* sc\_tracediff: утилита для отслеживания изменений маршрутизации в сети,
* sc\_warts2csv: утилита для конвертации warts фалов в формат csv,
* sc\_warts2json: утилита для конвертации warts файлов в формат JSON,
* sc\_warts2pcap: утилита для конвертации warts файлов в формат pcap,
* sc\_warts2text: утилита для конвертации warts файлов в обычный текстовый формат.
* sc\_wartscat: утилита для конкатенации нескольких файлов в формате warts
* sc\_wartsfix: утилита для восстановления поврежденных файлов в формате warts.
  + 1. **Конвертация warts файлов**

Для расчета вариации задержки необходимо получить данные о задержках при передаче пакетов. Так как основной скрипт-обработчик, который производит расчет вариации задержки различными методами, принимает на вход текстовые файлы, в которых каждое значение задержки располагается на отдельной строке, необходимо привести все данные к этому формату. Для эти целей был разработан скрипт «warts\_converter.py» на языке Python версии 2.7. В качестве аргумента скрипт принимает путь до папки с warts-файлами. Помимо базовых модулей в скрипте используются python-модули os, sys и scamper-pywarts[pywarts].

Модуль os в Python предназначен для взаимодействия программы с операционной системой, в частности, для управления файлами. Иногда нужно автоматизировать какие-то действия с файлами, например, создать автоматический установщик, или программу, делающую бэкап документов на второй раздел диска. В данном скрипте, модуль os используется для операций с путями файлов, получения имен файлов, расположенных в папке, и создания новых папок на файловой системе.

Модуль sys содержит информацию о среде выполнения программы, об интерпретаторе Python. В скрипте-конвертере использование данного модуля обусловлено необходимостью получения параметров запуска скрипта, а именно папки, в которой находятся файлы в формате warts, которые будут подвергаться конвертации.

Модуль scamper-pywarts – это библиотека с открытым исходным кодом на языке Python, предназначенная для чтения и обработке файлов в формате warts. Данный модуль просто в использовании и имеет множество достоинств:

* реализация на чистом языке Python, отсутствие зависимостей;
* поддерживает обе версии языка Python;
* легко расширяемый, благодаря открытости исходного кода;
* обладает высокой производительностью (для обработки warts-файла, содержащего данные трассировок, размером 80 мегабайт необходимо меньше минуты);
* реализует потоковую обработку данных, так как единовременно в памяти находится только одна запись. Это позволяет обрабатывать большие warts-файлы при ограниченном ресурсе виртуальной памяти и получать данные напрямую на выходе утилиты scamper.

В качестве аргумента данный скрипт принимает путь до директории с warts файлами и приводит все файлы в ней к виду, пригодным для обработки скриптом на языке R. Ниже приведен исходный код описанного скрипта:

**import** warts  
**from** warts.traceroute **import** Traceroute  
**from** os **import** listdir, makedirs, path  
**from** sys **import** argv  
  
  
*# method for warts file conversion to format used by R processing script***def** make\_pings\_file(warts\_dir, filename):  
 directory = **'rtt\_files\_last\_hop'** rtt\_filename = filename.replace(**'.warts'**, **'.txt'**)  
 **if not** path.exists(directory):  
 makedirs(directory)  
 **if** path.exists(path.join(directory, rtt\_filename)):  
 **print**(**"{} exists"**.format(rtt\_filename))  
 **return** 1  
 **with** open(path.join(warts\_dir, filename), **'r'**) **as** fd,\  
 open(path.join(directory, rtt\_filename), **'w'**) **as** outfile:  
 **while** True:  
 **try**:  
 record = warts.parse\_record(fd)  
 **if** record **is** None:  
 **break  
 if** isinstance(record, Traceroute) **and** record.hops:  
 **try**:  
 used\_hop  
 **except**:  
 used\_hop = record.hops[-1].address  
 outfile.write(str(record.hops[-1].rtt) + **'\n'**)  
 **else**:  
 **if** record.hops[-1].address == used\_hop:  
 **print**(**'the hop address is {} and rtt is {}'**.  
 format(record.hops[-1].address, record.hops[-1].rtt))  
 outfile.write(str(record.hops[-1].rtt) + **'\n'**)  
 **else**:  
 **print**(**"Route has been changed from {norm} to {changed}"**.  
 format(norm=used\_hop, changed=record.hops[-1].address))  
 **except** Exception:  
 **continue  
 return** 0  
  
  
**def** main():  
 warts\_dir = argv[1]  
 warts\_files = listdir(warts\_dir)  
 **for** warts\_file **in** warts\_files:  
 make\_pings\_file(warts\_dir, warts\_file)  
 **return** 0  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
  
 main()

* + 1. **Программа расчета вариации задержки различными методами**

Программа обработки данных и расчета вариации задержки была реализована при помощи математического пакета R для анализа

R - язык программирования для статистической обработки данных и работы с графикой, но в тоже время это свободная программная среда с открытым исходным кодом, развиваемая в рамках проекта GNU. Среда программрования R поставляется в большинстве дистрибутивов GNU/Linux и находит свое применение везде при работу с данными. Это не только статистика в узком смысле слова, но и «первичный» анализ (графики, таблицы сопряжённости), и продвинутое математическое моделирование. Среда R так же может послужить бесплатной заменой популярным коммерческим пакетам, таким как Octave и MatLab. С другой стороны, вполне естественно, что основная вычислительная мощь R лучше всего его проявляется при статистическом анализе: от вычисления средних величин до вейвлет-преобразований временных рядов.

Благодаря тому, что реализация методов расчета джиттера вынесена в отдельные модули, хранящиеся в директории «methods», программа является масштабируемой, позволяя легко добавлять реализации новых методов, без необходимости детального изучения работы скрипта и внесения значительных изменений в код. На данный момент реализованы 4 способа расчета, описанные в первой главе работы, и представлены файлами: ipdv.R, jitter\_first.R, jitter\_min.R и jitter\_mean.R.

Основным скриптом, осуществляющим расчеты является main.R. Первым аргументом при запуске скрипта необходимо указать путь до директории с текстовыми файлами, в которых каждое значение задержки при передаче пакета располагается на отдельной строке. Все следующие аргументы являются кодовыми обозначениями способа расчета джиттера, сопоставить обозначения с методами расчета можно при помощи Таблицы \*.

Таблица \*

|  |  |
| --- | --- |
| Обозначение | Способ расчета |
| 1 | jitter\_min.R |
| 2 | jitter\_mean.R |
| 3 | jitter\_first.R |
| 4 | ipdv.R |

Результатом работы скрипта является набор графиков для каждого файла, на которых изображен распределенный во времени джиттер, рассчитанный всеми указанными при запуске скрипта методами. Так как имеют место резкие скачки значений задержек, для удобства отображения ось ординат является логарифмической. Все графики сохраняются в папке image, которая будет создана в случае ее отсутствия, в формате PNG.

Так же скрипт сохраняет в текстовый файл «quantiles.txt» значения квантилей (1 – 10-3), где в каждой новой строке указаны сначала имена исходных файлов с данными, а затем квантили вариации задержки, рассчитанной всеми выбранными способами, в порядке указания их при запуске скрипта. Исходный код описанного скрипта:

*# script to calculate jitter and draw plots*image\_dir = **"images"**arguments <- commandArgs(trailingOnly=TRUE)  
stopifnot(length(arguments) > 1)  
quantiles <- numeric()  
method\_list <- c(**"jitter\_min"**, **"jitter\_mean"**, **"jitter\_first"**, **"ipdv"**)  
line\_colors <- c(**"green"**, **"blue"**, **"red"**, **"yellow"**)  
filename <- arguments[1]  
  
stopifnot(exists(**"filename"**))  
stopifnot(file.exists(filename))  
  
delays <- as.integer(readLines(filename))  
endY = max(delays) + 0.05 \* (max(delays) - min(delays)) / 0.9  
  
  
dir.create(image\_dir, showWarnings = FALSE)  
image\_name = file.path(image\_dir, paste(basename(filename), **".png"**, sep=**""**))  
png(image\_name)  
plot(x=0, y=1, xlim=c(0, length(delays)), ylim=c(1 , endY), type=**"n"**, xlab=**"package number"**, ylab=**"jitter"**, log=**"y"**)  
  
**for** (method\_number **in** arguments[2:length(arguments)]){  
 **if** (strtoi(method\_number) %in% seq(1:4)){  
 method\_script <- paste(**"methods/"**, method\_list[strtoi(method\_number)], **".R"**, sep=**""**)  
 source(method\_script)  
 values <- getJitter(delays)  
 quantiles <- c(quantiles, round(quantile(values[2:length(values)], probs=0.99, names = FALSE)/1000 ,digits = 2))  
 points(1:length(values), values, type=**"l"**, col=line\_colors[strtoi(method\_number)])  
 } **else** {  
 print(paste(**"There are no method with number "**, method\_number, **". Please use numbers from 1 to 4"**, sep=**""**))  
 }  
}  
write(c(basename(filename), quantiles), file=**"quantiles.txt"**, append=TRUE)  
garbage <- dev.off()

* 1. **Верификация и валидация экспериментального стенда**

Так как одной из задач исследования является сбор репрезентативных данных о задержках при передаче пакетов, было решено использовать реальные данные, собранные в сети PlanetLab.

PlanetLab – сеть, используемая учеными для тестирования новых сетевых сервисов или модификации уже существующих. На текущий момент, использование PlanetLab в качестве платформы для тестирования вошло в список стандартной методологии исследования распределенных систем. За время своего существования PlanetLab использовалась для тестирования большого числа различных сервисов: p2p сетей, включая DHT, anycast, распределения файлов большого размера, анализа сети, диагностики аномалий и ошибок, и многих других.

На сайте imdc.datcat.org [datcat] были найдены дампы трассировок, сделанных в сети PlanetLab в апреле 2017 года. Данные представляют собой warts файлы, содержащие результаты трассировок DNS-сервера компании Google по адресу 8.8.8.8, замеренные с 107 узлов сети PlanetLab. Так как целью исследования являлось изучение случайной природы значений RTT на ближних узлах сети, то максимальное значение TTL пакета было установлено равным 4. Измерение проводилось в течение недели, по одному замеру в секунду. Эти файлы довольно велики, они содержат более 100 тысяч трассировок каждый.

Для расчетов было решено выбрать задержки до второго узла следования маршрута. Так как маршрут следования пакет может меняться, скрип, обрабатывающий warts файлы, разделяет задежрки по адресам назначения, а затем выбирает наиболее часто используемый адрес, отбрасывая незначительное количество запросов, сделанных по другому адресу назначения, чем предотвращает возможную ошибку использования задержек до разных узлов при расчете джиттера Так же обрезать полученные после конвертации файлы, взяв первые 10 тысяч строк. Такой размер будет более чем достаточен, так как требования рабочих характеристик для двухточечного изменения задержки пакета IP (определенные в Рек. МСЭ-Т Y.1540) основаны на верхнем ограничении 1 − 10–3 квантиля от распределения базового значения для потока. Использование 1 − 10–3 квантиля позволяет работать с короткими интервалами оценки (например, для оценки этого ограничения минимально допустимым является образец с 1000 пакетов). Сделать это можно при помощи команды sed.

find . -type f -exec sed -i "" '10001,$d' {} \;

Для валидации работы стенда было предложено сгенерировать файл с одинаковыми задержками и рассчитать для него вариацию задержки различными способами, очевидно, что джиттер для абсолютно равномерного трафика будет равен нулю. Для проверки был сгенерирован файл same.txt содержащий 10 тысяч одинаковых значений, равных 100 при помощи консольной команды:

for i in $(seq 1 10000); do echo -e "100" >> same.txt; done

По результатам обработки файла скриптом на языке R, были получены данные, представленные в Таблице 2

Таблица 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя файла/метод рассчета | 1 | 2 | 3 | 4 |
| same.txt | 0 мс | 0 мс | 0 мс | 0 мс |

Так же был построен график на котором видно, что линии, изображающие джиттер, сливаются в одну и совпадают с осью абсцисc.

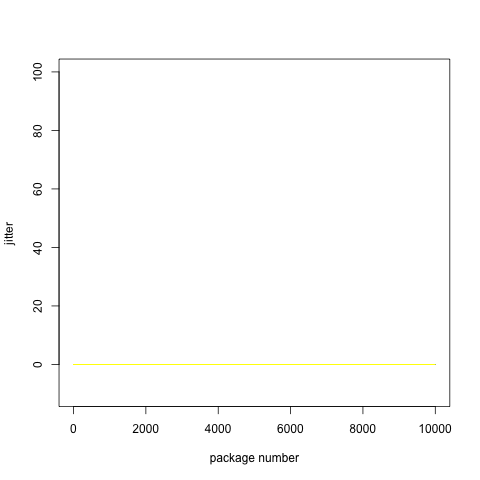


Рисунок \* - Визуальное представление значений джиттера, рассчитанного различными способами при постоянных значениях задержки

На основании этих результатов можно сделать вывод, что экспериментальный стенд работает корректно.

**2.3 Используемый математический аппарат**

В созданном экспериментальном стенде функции расчета джиттера были вынесены в отдельные модули, расположенные в папке «methods». Все методы реализованы функцией «getJitter», принимающей на вход вектор значений задержек и возвращающий вектор, где первый элемент является кодовым обозначение метода расчета, а все последующие – рассчитанными значениями вариации задержки.

**2.3.1. Описание реализации различных методов расчета вариации задержки**

Метод расчета вариации задержки под номером 1 состоит в вычислении разницы между значением задержки передачи пакета и минимальным значением задержки на интервале от начала измерения до текущего пакета. Ниже приведен код реализации на языке R.

# function to calculate jitter via 1st method

getJitter <- function(delays){

jitter <- numeric()

jitter[1] <- 1

for (i in 2:length(delays)){

minDealay <- min(delays[1:i-1])

jitter[i] <- delays[i] - minDealay

}

return(jitter)

}

Метод расчета вариации задержки под номером 2 состоит в вычислении разницы между значением задержки передачи пакета и средним значением задержки на интервале от начала измерения до текущего пакета. Так как среднее значение может быть больше, текущего значения, в качестве результата берется абсолютное значение. Ниже приведен код реализации на языке R.

# function to calculate jitter via 2nd method

getJitter <- function(delays){

jitter <- vector()

jitter[1] <- 2

for (i in 2:length(delays)){

meanDealay <- mean(delays[1:i-1])

jitter[i] <- abs(delays[i] - meanDealay)

}

return(jitter)

}

Метод расчета вариации задержки под номером 3 состоит в вычислении разницы между значением задержки передачи пакета и значением задержки при передаче первого пакета. Так как первое значение может быть больше, текущего значения, в качестве результата берется абсолютное значение. Ниже приведен код реализации на языке R.

# function to calculate jitter via 3rd method

getJitter <- function(delays){

jitter <- vector()

jitter[1] <- 3

for (i in 2:length(delays)){

jitter[i] <- abs(delays[i] - delays[1])

}

return(jitter)

}

Метод расчета вариации задержки под номером 4 состоит в вычислении разницы между значениям задержки передачи текущего и предыдущего пакетов. Как указано в первой главе, значение IPDV может принимать отрицательные значения, но часто для расчетов берется абсолютное значение. Ниже приведен код реализации на языке R.

# function to calculate jitter via 4th method

getJitter <- function(delays){

jitter <- vector()

jitter[1] <- 4

for (i in 2:length(delays)){

jitter[i] <- abs(delays[i-1] - delays[i])

}

return(jitter)

}

Метод расчета из пункта 1.3.3 завязан на значениях внутренних полей заголовков RTP и не был рассмотрен в данной работе, так как существует сложность с получением актуальных дампов трафика протоколов RTP и RTCP.

**2.3.2 Статистическая обработка результатов экспериментов**

В документе «Y.1541 - Требования к сетевым показателям качества для служб, основанных на протоколе IP» в Таблице 1 указано, что верхнее значение 99.99-процентиля вариации задержки для нулевого и первого классов QoS равно 50 мс. Отсюда очевидно, что для анализа полученных результатов необходимо посчитать данный квантиль для всех методов рассчета джиттера.

Квантиль в математической статистике — значение, которое заданная случайная величина не превышает с фиксированной вероятностью. Если вероятность задана в процентах, то квантиль называется процентилем или перцентилем.

Математический пакет R содержит встроенную функцию расчета квантиля. Квантили рассчитываются в R при помощи функции quantile(), например

quantile(mtcars$mpg)

0% 25% 50% 75% 100%

10.400 15.425 19.200 22.800 33.900

При настройках, заданных по умолчанию, выполнение указанной команды приведет к расчету минимального (10.4) и максимального (33.9) значений, а также трех квартилей, т.е. значений, которые делят совокупность на четыре равные части - 15.4, 19.2 и 22.8. Разница между первым и третим квартилями носит название интерквартильный размах (ИКР; англ. interquartile range). ИКР является робастным аналогом дисперсии и может быть рассчитан в R при помощи функции IQR():

Так как нас интересует значение квантиля вероятности 99.99%, то функция выглядит следующим образом

quantile(values[2:length(values)], probs=0.99, names = FALSE)

* 1. **Выводы по главе**
  2. Был разработан экспериментальный стенд для сбора данных о задержках, расчета джиттера различными методами и анализа полученных результатов.
  3. Дампы трафик адекватны реальности, так как были собраны при помощи утилиты scamper и описывают реальные процессы происходящие в компьютерных сетях.
  4. Размер дампов в 10 тысяч записей достаточен, так как для расчета квантиля вероятности 99.99 процента достаточно всего 1000 значений вариации задержки.
  5. Экспериментальный стенд показывает ожидаемый результат при простейших входных данных, на основании чего можно считать, что методы рассчета вариации задержки были реализованы без ошибок.

**ГЛАВА 3. АНАЛИЗ РЕЗУЛЬТАТОВ РАСЧЕТОВ ВАРИАЦИИ ЗАДЕРЖКИ РАЗЛИЧНЫМИ МЕТОДАМИ НА ОСНОВЕ СОБРАННЫХ ДАННЫХ**

Для достижения ВКР необходимо произвести расчеты вариации задержки всеми описанными методами на собранных данных и проанализировать результаты.

* 1. **Порядок проведения экспериментов**

Ниже описан полная последовательность действий для воспроизведения результатов произведенных экспериментов.

**3.1.1 Сбор данных о задержках**

В качестве данных решено было взять трассировки с сайта datcat.com, описанные в Главе 2. Набор данных представляет из себя архив «ScamperPlantelabRTT\_20160407.tar.gz» размером 987 мегабайт. Разархивировать его в папку «warts\_files» можно при помощи команды

tar -xvzf ScamperPlantelabRTT\_20160407.tar.gz

После распаковки архива мы получим 107 файлов в формате warts общим размером 6.3 Гигабайта.

**3.1.2 Конвертация warts файлов**

Теперь необходимо конвертитровать данные из бинарного формата в текстовый, читаемый cкриптом на языке R. Для этих целей используется скрипт на языке Python версии 2.7, описанный в пункте 2.1.3. Скрипт запускается из консоли командой:

python2.7 warts\_converter.py warts\_files

Перед запуском скрипта необходимо удостовериться в наличие модуля scamper-pywarts. Сделать это можно при помощи менеджера пакетов pip, он сам проверит наличие необходимого модуля и скачает его из репозитория в случае отсутствия. Пример командны:

pip install scamper-pywarts

Работа скрипта может занять несколько часов, в зависимости от мощности компьютера. На выходе мы получим 102 текстовых файла в папке «rtt\_files», 5 бинарных файлов оказались испорчены, и скрипт отбросил их. Каждый файл содержит от 22 до 365 тысяч строк, а суммарный размер их уменьшился до 105 Мегабайт.

Так как было решено использовать выборки размером 10 тысяч значений задержки, необходимо обрезать полученные текстовые фалы до приемлемого размера, сделать это можно при помощи консольной команды:

cp rtt\_files rtt\_files\_10k && find rtt\_files\_10k -type f -exec sed -i "" '10001,$d' {} \;

Теперь мы имеем все необходимое для запуска скрипта, который производит необходимые расчеты вариации задержки различными способами.

**3.1.3 Расчет вариации задержки разными способами при помощи скрипта**

Для расчета вариации задержки различными способами используется скрипт на языке R, описанный в пункте 2.1.4. Так как на вход он принимает имя файла, содержащего значения задержек, необходимо передать ему имена всех файлов, содержащихся в папке «rtt\_files\_10k» при помощи команды find

for file in $(find rtt\_files\_first\_hop\_10k -name "\*.txt") ; do Rscript main.R $file 1 2 3 4 ; done

Команда выше будет запускать скрипт каждый раз подставляя в качестве первого аргумента новое имя файла из папки «rtt\_files\_10k». Работа скрипта может занять продолжительное время, зависящее от количества обрабатываемых файлов. По итогам обработки данных мы получаем папку «images» с графиками и файл «quantiles.txt», содержащий значения квантилей для всех методов расчета.

Программа расчет джиттера записывает значения в файл quantiles.txt каждое с новой строки. Для удобства дальнейшего анализа был написан простой shell-скрипт, который собирает значения в одну строку в соответствии с именем файла, данные из которого были обработаны. Код данного скрипта приведен ниже.

**#!/bin/bash  
  
while** true; **do** out=()  
 **for ((** i=0; i<5; i++ **))**; **do** read **&&** out+=( **"$REPLY"** )  
 **done  
 if (( $**{#out[@]} > 0 **))**; **then** printf **'%s ' "${out[@]}"** echo  
 **fi  
 if (( $**{#out[@]} < 5 **))**; **then** break; **fi  
done** *<quantiles.txt >quantiles2.txt*

**Бибилиографический список**

[warts] - <https://www.caida.org/tools/measurement/scamper/man/warts.5.pdf>

[pywarts] - <https://pypi.org/project/scamper-pywarts/>

[datcat] - http://imdc.datcat.org/collection/1-0721-1=Traceroutes-for-statistical-analysis-of-latency