Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

В.С. Заборовский, В.А. Мулюха, Ю.Е. Подгурский

СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ

МОДЕЛИРОВАНИЕ И АНАЛИЗ КОМПЬЮТЕРНЫХ СЕТЕЙ: ТЕЛЕМАТИЧЕСКИЙ ПОДХОД

Учебное пособие

Санкт-Петербург
Издательство СПбГПУ
2010

УДК 004.7:004.942(075.8)
ББК 32.973.202
З 12

Заборовский В.С., Мулюха В.А., Подгурский Ю.Е. Сети ЭВМ и Телекоммуникации. Моделирование и анализ компьютерных сетей: телематический подход. Учеб. пособие. СПб: Изд-во СПбГПУ, 2010. 93 с.

Рецензенты:
Зав. каф. РВКС, проф., д.т.н. Ю.Г. Карпов
Зам. директора по научной работе СПИИРАН, заслуженный деятель науки,
проф., д.т.н. А.В. Смирнов

Данное пособие содержит курс лабораторных работ по освоению инструментальных средств анализа и моделирования трафика. Включает описание возможностей программы MTraffic для анализа различных параметров сети, программы Network Simulator 2, для моделирования сетевых процессов и различных механизмов протокола управления передачей данных и содержит методические рекомендации и материалы к циклу работ студентов.

В пособие предлагается телематический подход к изучению материала, заключающийся в комплексном рассмотрении телекоммуникационной составляющей и изучении сетевых сервисов.

Лабораторные работы в учебном пособии предназначены для практического освоения материала студентами старших курсов специальностей «Информатика и вычислительная техника», «Сети ЭВМ и телекоммуникации».

Табл. 1. Ил. 25. Библиогр.: назв. 12

Печатается по решению редакционно-издательского совета Санкт-Петербургского государственного политехнического университета.

ОГЛАВЛЕНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

АКФ – АвтоКорреляционная Функция

БУЗ – Быстро Убывающая Зависимость

ДС – Динамическая Система

МУЗ – Медленно Убывающая Зависимость

ПО – Программное Обеспечение

ПС – Пропускная Способность

УЛК – Учебно-Лабораторный Комплекс

ЭВМ – Электронная Вычислительная Машина

ARP – Address Resolution Protocol

BGP – Border Gateway Protocol

DARPA – Defense Advanced Research Projects Agency

ICMP – Internet Control Message Protocol

ID – Identification

IP – Internet Protocol

LAN – Local Area Network

NS2 – Network Simulator – 2

OSPF – Open Shortest Path First

RTCP – Realtime TCP

RTO – Retransmission TimeOut

RTT – Round Trip Time

SAMAN – Simulation Augmented by Measurement and Analysis for Networks

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

WDM – Wavelength Division Multiplexing

WiMAX – Worldwide Interoperability for Microwave Access

ВВЕДЕНИЕ

Анализ возможностей эффективного применения современных компьютерных сетей в условиях постоянного усложнения их топологии, появления новых технологий передачи данных, протоколов и приложений является сложной научно-технической задачей. Так как современные компьютерные сети представляют собой объекты, характеризующиеся естественной распределенностью, реконфигурируемостью транспортной инфраструктуры, децентрализованностью и кооперативностью управления проведение «натурных» экспериментов для исследования свойств процессов, изучения протоколов и особенностей функционирования приложений вызывает известные организационно-технические сложности. В этих условиях совместное применение средств моделирования и экспериментального анализа результатов работы сети позволят получить объективные данные о свойствах сетевых процессов и возможностях их прикладного использования.

Постоянный контроль работы сети на основе анализа характеристик трафика, режимов передачи и работоспособности приложений помогает администратору выявить проблемные участки сетевой инфраструктуры, процессы в которых представляют угрозу информационной безопасности, приводят к снижению производительности и надежности функционирования сети в целом.

Анализ процессов в современной компьютерной сети можно разделить на несколько этапов. Первый этап заключается в идентификации типов сетевых процессов, которые можно отнести к трем основным категориям – трафик реального времени с высокими требованиями к величине средней задержки в передачи пакетов, трафик изохронного типа с минимально возможными вариациями задержек и трафик передачи данных с высокими требованиями к пропускной способности. Такое деление позволяет использовать термин производительность для характеризации различных аспектов качества сетевого сервиса для конкретных типов сетевых приложений. Вторым этапом изучения

процессов в компьютерных сетях является статистический анализ трафика. На данном этапе сетевой трафик рассматривается либо как последовательность пакетов, либо как совокупность временных интервалов между пакетами. С точки зрения описания процессов эти два типа статистических характеристик имеют различные модели представления в форме дискретных или непрерывных распределений. Третий этап анализа процессов заключается в исследовании фрактальных свойств сетевого трафика. Природа фрактальных свойств трафика связана с наличием обратной связи в механизме управления параметрами протокола TCP и возникающих в результате этого медленно убывающих корреляционных зависимостей между состоянием виртуального транспортного соединения. Для анализа фрактальных свойств сетевых процессов можно использовать методы нелинейной динамики и модели состояний в фазовом пространстве дробной размерности. Исследование свойств аттракторов этих процессов позволяет спрогнозировать возникновение бифуркаций и смены режимов работы сети при помощи системы нелинейных динамических уравнений, а также свойств масштабной инвариантности (самоподобия) статистических характеристик и показателя Херста.

Особенности функционирования современных компьютерных сетей нельзя изучать без совместного рассмотрения свойств сетевой инфраструктуры, используемых протоколов и прикладных сервисов. Рассмотрение процессов на различных уровнях межсетевого взаимодействия в последнее время принято связывать с развитием телематического подхода, объединяющего средства передачи данных и виртуализации ресурсов для обеспечения высокого качества сетевых сервисов. В основе телематики лежит совместный анализ возможностей передачи данных по телекоммуникационным каналам связи и осуществления транзакций в виртуальном пространстве для получения запрашиваемых информационных услуг.

Термин телематика появился благодаря переводу французского слова télématique, которое появилось из соединения слов télécommunication и

informatique, и было предложено С. Норой и А. Минк в 1978 году в отчете под названием «Информатизация общества».

Таким образом, сфера телематики охватывает широкий и постоянно развивающийся спектр сетевых услуг и механизмов доступа к информационным ресурсам, включая службы электронной почты, цифровых данных, факсимильных, аудио- и видеосообщений.

При использовании телематического подхода можно выделить несколько системных уровней рассмотрения:

- техническое обеспечение (оборудование, программное обеспечение, протоколы взаимодействия, стандарты и т.д.);
- коммуникационные возможности (типы передаваемой информации, режимы межсетевого взаимодействия, скорость передачи данных и т.п.);
- информационные услуги (электронная почта, видеоконференции, социальные сети, форумы и т. п.);
- прикладные задачи (распределенные вычисления, защита информации).

В современной глобальной информационной сети все эти уровня реализованы на базе стека протоколов TCP/IP. Именно поэтому Интернет можно считать типичным представителем телематических сетей или современной средой доступа к различным телематическим услугам. Особенностью этого стека протоколов является иерархическая структура адресного пространства на сетевом уровне и возможность выбора методов доставки данных на транспортном уровне в зависимости от требований качества телекоммуникационного сервиса, надежности, допустимых задержек и пропускной способности.

Стек TCP/IP поддерживает все современные стандарты физического и канального уровней для локальных сетей – Ethernet, WiFi, WiMAX, сетевых кластеров – InfiniBand, глобальных сетей – WDM, B-ISDN и протоколы для спутниковых каналов связи в различных диапазонах частот. В этом стеке на сетевом уровне обеспечивается маршрутизация IP пакетов с использованием различных алгоритмов выбора оптимальных путей их доставки, реализуемых в

протоколах OSPF и BGP. Протокол TCP управляет надежной доставкой пакетов с использованием механизма обратной связи и адаптивной настройки пропускной способности виртуальных транспортных соединений в зависимости от загруженности физических каналов связи и узлов коммутации пакетов.

В пособии изучаются различные версии протокола TCP, отличающиеся алгоритмами формирования окна перегрузки, которое является основным механизмом реакции на потери и задержки в распространении пакетов. Учитывая широкое распространение телематических сервисов изучение механизмов сетевого информационного взаимодействия и средств моделирования процессов в высокоскоростных компьютерных сетях является важной составляющей подготовки бакалавров и магистров по направлению «Информатика и вычислительная техника» и «Сети ЭВМ и телекоммуникации».

Освоение материала учебного пособия позволит студентам глубже изучить возможности сетевых технологий, в том числе связанные с использованием мобильных устройств передачи цифровых данных, мультисенсорных сетей и средств распределенных вычислений. Наряду с этим, изучение свойств протокола TCP позволит приступить к исследованиям проблем информационной безопасности в сети Интернет, включая обеспечение доступности, целостности и конфиденциальности информационных ресурсов и сетевой инфраструктуры.

Авторы выражают благодарность сотрудникам и студентам кафедры телематики, без помощи которых написание данного учебного пособия было бы невозможно. Особо хотелось бы поблагодарить Сережину Ю.А. и Ратникову В.А., за разработку материалов практических заданий, используемых в данном пособии.

1. МОДЕЛИРОВАНИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ

1.1. СЕТЕВОЙ СИМУЛЯТОР NS2 (NETWORK SIMULATOR - 2)

Общие сведения

Сетевой симулятор NS2 (далее, симулятор) представляет собой программное средство для моделирования и анализа функционирования цифровых сетей с коммутацией пакетов. Первая версия симулятора появилась в 1989 году, когда быстрое распространение технологии межсетевого взаимодействия (internetworking) вызвало необходимость исследования и совершенствования протоколов составных сетей. Широкие возможности симулятора для исследования корректности и эффективности протоколов различных уровней, а также для моделирования разнородных приложений способствовали его быстрому распространению. С 1995 года совершенствование симулятора поддерживается агентством перспективных исследований министерства обороны (DARPA) США в рамках проектов VINT (Virtual InterNetwork Testbed) и SAMAN (Simulation Augmented by Measurement and Analysis for Networks), а также национальным научным фондом США (проект CONSER – Collaborative Simulation for Education and Research).

Симулятор предназначен для использования, как в исследовательских целях – для оценки влияния различных факторов на эффективность разрабатываемых протоколов и приложений, так и в учебных целях – для пояснения работы конкретных протоколов и алгоритмов управления процессами в сетях.

В зависимости от целей исследования, процессы в сети могут моделироваться на различных уровнях взаимодействия, с учетом особенностей традиционных и перспективных приложений, протоколов и технологий (WWW, Multicast, Mobile Networking, Satellite Networking, LAN и др.).

Основными особенностями симулятора NS2 являются:

- Модульный принцип построения и открытая архитектура симулятора, соответствующие многоуровневой архитектуре сетей и позволяющие легко расширять его функциональные возможности путем добавления новых модулей и модификации имеющихся.
- Широкий диапазон методов и средств абстрагирования, предоставляющих возможности изменения уровня абстракции, как при анализе результатов, так и, непосредственно, при моделировании, что позволяет идентифицировать существенные эффекты на уровне макромодели, а затем применять детальное моделирование для их более тщательного исследования.
- Наличие средств анимации, обеспечивающих возможность наблюдения конкретных реализаций и позволяющих выделить наиболее важные и интересные моменты поведения сетевой модели. (Отметим, что при рассмотрении только агрегированных статистических оценок много существенных явлений остается незамеченными).
- Наличие библиотеки сетевых топологий и генераторов трафика, облегчающих создание моделей сетей со сложной топологией и смешанной нагрузкой.
- Состав смешанного Интернет-трафика постоянно меняется (в настоящее время преобладают WWW-приложения, в ближайшем будущем, возможно, основную долю составят аудио- и видеоприложения). Топология сетей также постоянно изменяется. Важно облегчить исследователям возможность опережающего исследования влияния этих тенденций на эффективность работы сетей. Проблема задания топологии и динамики ее изменения особенно значима для мобильных сетей.
- Широкая известность NS2, как эталонного средства моделирования, свидетельствующая о высокой степени достоверности результатов.
- Свободное распространение пакета.

Симулятор NS2 относится к свободно распространяемым программным средствам (freeware). На сайте разработчиков [3] доступно несколько версий симулятора, ориентированных на различные операционные системы и

аппаратные платформы. Наиболее полной является версия, ориентированная на семейство ОС UNIX, которая и рассматривается в данном пособии.

Как правило, совместно с симулятором NS2 используются дополнительные программные средства графического отображения результатов моделирования. В первую очередь, к ним относятся утилита сетевой анимации nam (network animator) и утилита построения графиков X-graph. Кроме того, вместе с симулятором распространяются средства отладки моделей, конвертации выходных данных, генераторы топологий сетей и сценариев моделирования, а также ряд других пакетов.

Состав симулятора постоянно обновляется, в данном пособии рассмотрены лишь основные его возможности. Более полная информация об установке, составе, настройке и использовании отдельных пакетов, а также список основных объектов и функций симулятора содержится в [4]. Далее мы более подробно остановимся на TCP-объектах в симуляторе, а также на новых механизмах, появившихся в симуляторе последних версий.

TCP объекты в NS2

Объекты типа TCP представляют собой класс объектов-агентов, моделирующих транспортный протокол TCP [5]. В настоящее время NS2 поддерживает несколько типов TCP-агентов односторонних передач:

- Agent/TCP – “Tahoe” TCP-отправитель;
- Agent/TCP/Reno – “Reno” TCP-отправитель;
- Agent/TCP/Newreno – “Reno” TCP-отправитель с модификацией;
- Agent/TCP/Sack1 – TCP с выборочным повтором (RFC2018);
- Agent/TCP/Vegas – TCP Vegas;
- Agent/TCP/Fack – TCP Reno с «последующим подтверждением»;
- Agent/TCP/Linux – TCP-передатчик с поддержкой SACK, который использует TCP с перезагрузкой контрольных модулей из ядра Linux.

Односторонние агенты приема:

- Agent/TCPSink;
- Agent/TCPSink/DelAck;
- Agent/TCPSink/Sack1;
- Agent/TCPSink/Sack1/DelAck.

Двунаправленный агент в текущей версии поддерживается только агентом формы TCP Reno:

- Agent/TCP/FullTcp.

Симулятор также поддерживает несколько версий абстрактных TCP-отправителей. Эти объекты не являются моделями реального TCP. Они не содержат описания динамического окна, не устанавливают и не завершают соединение и не передают никаких данных. По умолчанию TCP-агентом является отправитель версии Tahoe.

TCP-агент поддерживает дополнительные конфигурационные переменные. Каждая из переменных, приведенных в этом списке, может быть переменной класса или переменной экземпляра класса. Изменение переменной класса меняет значение по умолчанию для всех агентов, созданных в последствии. Изменение переменной экземпляра конкретного агента влияет только на этот агент.

Например:

```
Agent/TCP set window_ 100; # влияет на класс
$tcp set window_ 20;       # меняет window_ только для
                           объекта $tcp
```

Конфигурационные параметры:

- window_ – верхняя граница окна приемника (Advertisment Window) TCP-соединения.
- maxcwnd_ – верхняя граница окна переполнения TCP-соединения. Для отмены ограничения устанавливается в 0 (значение по умолчанию).
- windowInit_ – начальное значение окна переполнения для медленного старта.
- windowOption_ – тип алгоритма, использующегося для управления окном переполнения.

- windowThresh_ – постоянная сглаживающего фильтра, используемого для вычисления awnd (см. далее). Применяется для исследования различных алгоритмов управления окном.
- overhead_ – диапазон случайно распределенной переменной, используемой для задержки каждого выходного пакета. Применяется только в версии tcp Tahoe, в tcp Reno не используется.
- ecn_ – указатель использования (true/false) механизма явного оповещения (explicit congestion notification), в дополнение к отбрасыванию пакетов, при насыщении.
- packetSize_ – размер пакетов источника.
- tcpTick_ – интервал таймера TCP, используемый для оценки времени RTT (round-trip time). По умолчанию установлена величина 100ms.
- bugFix_ – указатель (true/false) запрета механизма быстрой повторной передачи при потере пакетов в одном окне данных. Установка в true запрещает быстрый повтор передачи нескольких пакетов, потерянных в одном окне данных.
- maxburst_ – максимальное число пакетов, которое может посылать источник в ответ на одно полученное подтверждение. При отсутствии ограничения устанавливается в 0.
- slow_start_restart_ – признак использования (1/0) механизма медленного старта. Включено по умолчанию.
- MWS (Maximum Window Size) – константа, определяющая максимальный размер окна (в пакетах), допустимый в симуляторе. По умолчанию MWS=1024 пакетам. Для TCP Tahoe параметр «window» представляет размер указанного окна получателя, которое должно быть меньше чем MWS-1. Для TCP Reno значение «window» должно быть меньше, чем (MWS-1)/2.

Переменные состояния:

- dupacks_ – число дублирующих подтверждений (ACK), полученных после прихода последнего недублирующего подтверждения.

- seqno_ – наивысший последовательный номер сегмента (sequence number) для данных источника TCP.
- t_seqno_ – текущий последовательный номер сегмента (пакета) пакета.
- ack_ – наивысшее значение из полученных подтверждений.
- cwnd_ – текущее значение окна переполнения.
- awnd_ – текущее значение окна переполнения при использовании усреднения. Используется для исследования различных алгоритмов увеличения окна.
- ssthresh_ – текущее значение порога медленного старта.
- rtt_ – оценка значения round-trip time.
- srtt_ – оценка сглаженного значения round-trip time.
- rttvar_ – оценка среднего отклонения значений round-trip time.
- backoff_ – экспоненциальная постоянная задержки для round-trip time.

Наиболее часто модифицируемые переменные – window_ и packetSize_. Эти значения сильно влияют на поведение TCP, их установка связывает модель с реальным TCP, функционирующим в мировой сети. TCP с большим значением размера пакета, большим окном и меньшим RTT (результат топологии и перегрузки) более агрессивны в запросах к пропускной способности сети.

Задание:

Для освоения симулятора NS2 и изучения его функционала рекомендуется проделать курс лабораторных работ, описанных в учебном пособии [4].

1.2. Различные версии протокола TCP.

TCP Tahoe

В отличие от более ранних реализаций протокола TCP, в версии TCP Tahoe добавлен ряд механизмов управления передачей. К ним относятся:

- медленный старт (Slow-Start);
- предотвращение перегрузок (Congestion Avoidance);

- быстрый повтор передачи (Fast Retransmit);
- новый метод оценки длительности цикла передачи (RTT – Round Trip Time), используемой для установки таймера повторной передачи (RTO – Retransmission TimeOut).

Если буфер промежуточного телематического устройства переполнен, какое-то число сегментов данных будет потеряно. При этом может быть запущено несколько сценариев. Основной вариант – медленный старт, запускается в рамках классического алгоритма TCP-Tahoe при потере сегмента и сопряженным с ним таймаутом (RTO) у отправителя, так как отправитель не получит сигнала подтверждения – ACK для потерянного сегмента. При потере пакета CWND делается равным 1, а порог медленного старта (ssthresh) – половине значения CWND, при котором произошло переполнение буфера телематического устройства, хотя следует отметить, что эффективность канала максимальна при наибольшем значении CWND. Оптимальное значение CWND может быть выбрано лишь при исчерпывающем знании прогноза состояния виртуального канала и всех промежуточных телематических устройств, постольку такая информация обычно недоступна, система переходит в режим освобождения буфера (CWND=1).

Смысл этого алгоритма заключается в удержании значения CWND в области максимально возможных значений, с учетом состояния загруженности сети. По существу эта оптимизация осуществляется с помощью потери пакетов.

Потерянный пакет и все, посланные после него, пакеты (вне зависимости от того, подтверждено их получение или нет) пересылаются повторно. При большой вероятности потери это существенно снижает пропускную способность и увеличивает и без того высокую загрузку канала. Если потеря была связана с началом сессии обмена с конкурирующим клиентом, потеря пакета вызовет таймаут и виртуальный канал будет блокирован, что вызовет резкое падение скорости передачи.

TCP Tahoe агент осуществляет контроль перегрузки и оценку RTT следующим образом: окно перегрузки увеличивается на 1 пакет с каждым ACK,

полученным в течение медленного старта (когда cwnd_ < ssthresh_) и увеличивается на 1/cwnd_ для каждого нового ACK, полученного в фазе предотвращения перегрузок (когда cwnd_ ≥ ssthresh_).

Реакция на перегрузку: TCP Tahoe считает пакет потерянным (от перегрузки), когда он получает 3 (значение по умолчанию переменной NUMDUPACKS в tcp.h) дублирующих повторных подтверждения, или когда истекает время повторной передачи (TO – таймаут). В каждом случае TCP Tahoe реагирует, устанавливая ssthresh_ в половину текущего размера окна (минимум из cwnd_ и window_) или в 2, в зависимости от того, что больше. Затем он снова присваивает cwnd_ значение windowInit_. Это обычно приводит к переходу TCP в режим медленного старта.

Оценка Round-Trip Time и RTO

Для оценки round-trip time используются 4 переменных, и они определяют время повторной передачи:

rtt_ – значение RTT,

srtt_ – сглаженное значение RTT,

rttvar_ – среднее отклонение значений RTT,

tcpTick_ – интервал таймера TCP,

backoff_ – экспоненциальная постоянная задержки для RTT.

TCP инициализирует rttvar_ = 3/tcpTick_ и backoff_ = 1. Когда таймер будущих повторных передач установлен, его timeout устанавливается в текущее время + max(bt(a + 4v + 1), 64) секунд, где:

b – текущее значение backoff_,

t – значение tcpTick_,

a – значение srtt_,

v – значение rttvar_.

Отсчет RTT приходит с новым ACK. Отсчет RTT рассчитывается как разница между текущим временем и полем «времени эха» в пакете ACK. Когда первый отсчет взят, его значение используется как инициализируемое для srtt_.

Половина первого отсчета используется для инициализации rttvar_. Для последующих отсчетов, значения изменяются следующим образом:

$$srtt = \frac{7}{8} * srtt + \frac{1}{8} * sample;$$
$$rttvar = \frac{3}{4} * rttvar + \frac{1}{4} * |sample - srtt|.$$

Конфигурация.

Запуск TCP симуляции предполагает создание и конфигурирование агента, прикрепление источника данных (генератора трафика) и их запуск.

Источник данных TCP.

TCP-агент не генерирует никаких прикладных данных, вместо этого пользователь может соединить любой генерирующий трафик модуль с TCP-агентом для генерации данных. Обычно используется FTP или Telnet. FTP представляет данные заданного объема, а telnet выбирает размер передачи случайно из tcplib (см. файл tcplib-telnet.cc.)

TCP Reno

Объекты типа TCP/Reno являются подклассом объектов TCP, моделирующим протокол TCP Reno. TCP Reno очень похож на TCP Tahoe агент, за исключением того, что он содержит механизм быстрого восстановления (fast recovery). Новый алгоритм не требует освобождение канала и его медленного (slow-start) заполнения после потери одного пакета. Отправитель переходит в режим быстрого восстановления, после получения некоторого предельного числа дублирующих подтверждений (как правило, этот предел – tcprexmtthresh, устанавливается равным трем). После получения указанного числа дублирующих подтверждений, отправитель повторяет передачу одного пакета и уменьшает окно насыщения (cwnd) в два раза. Но, в отличие от версии TCP Tahoe, не переходит к алгоритму медленного старта. Он уменьшает размер окна до половины текущего размера и устанавливает ssthresh_ в соответствии с этим значением.

В TCP Reno при нормальной ситуации размер окна меняется циклически. Он увеличивается до тех пор, пока не произойдет потеря сегмента, а при потере происходит его сокращение. TCP Reno имеет две фазы изменения размера окна: фаза медленного старта и фаза избегания перегрузки. При получении отправителем подтверждения доставки в момент времени t + tA (сек.), текущее значение размера окна перегрузки cwnd_(t) преобразуется в cwnd_(t + tA) согласно:

$$cwnd(t+t_A)=\begin{cases}\text{фаза медленного старта (cwnd(t) < ssth(t)):}\\ cwnd(t)+1,\\ \text{фаза избегания перегрузки (cwnd(t) } \geq \text{ ssth(t)):}\\ cwnd(t)+\dfrac{1}{cwnd(t)},\end{cases}$$

где ssth(t) (ssthresh(t)) – значение порога медленного старта в пакетах, при котором TCP переходит из фазы медленного старта в фазу избегания перегрузки, когда в результате таймаута определяется потеря пакета, значения cwnd(t) и ssth(t) обновляются следующим образом:

$$cwnd(t)=1;$$
$$ssth(t)=cwnd(t)/2.$$

С другой стороны, когда TCP детектирует потерю пакета согласно алгоритму быстрой повторной передачи, cwnd(t) и ssth(t) обновляются иначе:

$$ssth(t)=cwnd(t)/2;$$
$$cwnd(t)=ssth(t).$$

TCP-Reno после этого переходит в фазу быстрого восстановления. В этой фазе размер окна увеличивается на один пакет, всякий раз, когда получается ACK. Таймаут обрабатывается в соответствии с механизмом, описанным для алгоритма TCP-Tahoe:

$$cwnd(t) = 1;$$
$$ssth(t) = cwnd(t)/2.$$

Для этих объектов не определены никакие дополнительные методы, конфигурационные параметры и переменные.

TCP NewReno

Объекты TCP/NewReno являются подклассом объектов TCP, моделирующим модифицированную версию протокола BSD TCP Reno. Они основаны на TCP Reno агенте, но с модификацией действий, предпринимаемых при получении пакетов подтверждений ACK. Полученный ACK с наивысшим посланным номером в последовательности пакетов автоматически подтверждает и все ранее прибывшие пакеты.

При получении трех дублированных подтверждений (dupACK) отправитель считает пакет потерянным и посылает его повторно. После этого отправитель может получить дополнительные дублированные подтверждения, так как получатель продолжает осуществлять подтверждение пакетов, которые находятся в пути, когда отправитель перешел в режим быстрой повторной передачи (fast retransmit). В случае потери нескольких пакетов из одного окна отправитель получит данные о следующем потерянном пакете, когда приходит подтверждение для повторно посланного пакета. Если потерян один пакет и не было смены порядка пакетов, тогда в качестве подтверждения пакета придет подтверждение на последний пакет серии, что автоматически будет означать успешную доставку всех предыдущих пакетов до перехода в режим быстрой повторной передачи (fast retransmit). Однако если потеряно несколько пакетов, тогда подтверждение повторно посланного пакета подтверждает доставку некоторых, но не всех пакетов, такие подтверждения называются частичными.

Разработчики назвали алгоритм быстрого восстановления NewReno, так как он значительно отличается от базового алгоритма Reno. Предложенный алгоритм дает определенные преимущества по сравнению с каноническим Reno

при самых разных сценариях. Но есть один случай, когда классический Reno превосходит NewReno – изменение порядка следования пакетов.

Конфигурационные параметры:

newreno_changes_ – указатель версии протокола. Установка в 0 соответствует основной версии NewReno, установка в 1 приводит к использованию дополнительных алгоритмов NewReno.

TCP Sack

Объекты типа TCP/Sack1 – подкласс объектов TCP, моделирующий протокол BSD TCP Reno с селективным подтверждением. Объекты Sack1 наследуют все функциональные особенности TCP объектов. Этот агент включает выборочные повторы, основанные на выборе ACKs, осуществленном получателем. Алгоритм TCP Sack использует поле «Опции» заголовка сегмента TCP для дополнительной информации о полученных пакетах стороной-получателем. Если произошла потеря, то каждый сегмент дублирующего ACK (dupACK), отправляемый получателем, содержит информацию о пакете, вызвавшем посылку данного сегмента. Таким образом, отправитель, получив данный пакет, имеет информацию не только о том, какой сегмент был потерян, но также и о том, какие сегменты успешно достигли получателя. Благодаря этому избегается ненужная повторная посылка сегментов, успешно буферированных на стороне получателя. Избыточные данные сохраняются в TCP-заголовке, 40 байт на сегмент. В переменную могут быть записаны два числа – 0 (выключено) и 1 (включено). Значение по умолчанию – 1 (включено).

Как и Reno, TCP Sack входит в режим быстрого восстановления при получении трех дублирующихся подтверждений (dupACK). Во время быстрого восстановления отправитель поддерживает переменную pipe, отображающую число пакетов, находящихся в сети. Данная переменная увеличивается каждый раз, когда новый сегмент был отправлен и уменьшается, если было получено очередное подтверждение. Передача нового пакета в сеть разрешена, если значение pipe меньше окна перегрузки.

Отправитель также поддерживает структуру данных (scoreboard), которая запоминает подтверждения из опции SACK прибывающих подтверждений. Если отправителю разрешена передача, он передает следующий пакет из списка пакетов, считаемых потерянными. Если таких пакетов нет, то посылается новый пакет.

Для этих объектов не определено никаких дополнительных методов, конфигурационных параметров и переменных.

TCP Fack

Объекты типа TCP/Fack – подкласс объектов TCP, моделирующий протокол BSD TCP Reno с механизмом контроля перегрузки при помощи упреждающих подтверждений (Forward Acknowledgement Congestion Control – FACK).

Переменная tcp_fack ответственна за систему упреждающих подтверждений (Forward Acknowledgement) в Linux. Forward Acknowledgement - – это специальный алгоритм, который работает поверх SACK и предназначен для контроля «заторов».

Главная идея алгоритма FACK состоит в отслеживании наибольшего номера выборочно подтвержденной последовательности как признака того, что все предыдущие не подтвержденные сегменты были потеряны. Это позволяет оптимизировать восстановление потерь. Однако этот алгоритм становится неработоспособным в случае неупорядоченного потока данных и тогда он автоматически отключается для данного конкретного соединения.

Изначально алгоритм FACK был разработан Мэттью Матисом (Matthew Mathis) с соавторами.

Переменная может принимать два значения – 0 (выключено) и 1 (включено). Значение по умолчанию – 1 (включено). Эта опция используется только тогда, когда включена переменная tcp_sack.

Конфигурационные параметры:

- ss-div4 – делит ssthresh_ на 4 (вместо 2), если переполнение выявляется в пределах 1/2 RTT от медленного старта (1=Enable, 0=Disable).
- rampdown – медленно уменьшает окно переполнения вместо того, чтобы время от времени делить его пополам (1=Enable, 0=Disable).

TCP Vegas

Объекты типа TCP/Vegas – подкласс объектов TCP, моделирующий протокол Vegas TCP.

TCP/Vegas контролирует размер окна путем контроля отправителем RTT для пакетов, посланных ранее. Увеличение нагрузки сети влечет за собой заполнение буферов промежуточных маршрутизаторов и, следовательно, увеличение времени пребывания пакета в канале связи (RTT). Если обнаруживается увеличение RTT, система узнает, что сеть приближается к перегрузке и сокращает размер окна данных. Если RTT уменьшается, отправитель определит, что сеть преодолела перегрузку, и увеличит размер окна. Следовательно, размер окна в идеальной ситуации будет стремиться к требуемому значению. В частности, на фазе избегания перегрузки, размер окна будет равен:

$$cwnd(t+t_A)=\begin{cases} cwnd(t)+1, \text{если } diff < \dfrac{\alpha}{base\_rtt} \\ cwnd(t), \text{если } \dfrac{\alpha}{base\_rtt} \le diff \le \dfrac{\beta}{base\_rtt}; \quad diff = \dfrac{cwnd(t)}{base\_rtt} - \dfrac{cwnd(t)}{rtt}, \\ cwnd(t)-1, \text{если } \dfrac{\beta}{base\_rtt} < diff \end{cases}$$

где rtt – зарегистрированное в текущий момент RTT,

base_rtt – наименьшее встретившееся в данном цикле RTT,

α и β – заданные константы.

Эта модификация TCP требует высокого разрешения таймера отправителя. Конфигурационные параметры TCP/Vegas:

- v_alpha_,

- v_beta_,
- v_gamma_,
- v_rtt_.

TCP FullTcp

Объекты типа TCP/FullTcp являются подклассом объектов TCP, более детально моделирующим существующие реализации протокола TCP. Отличия от других агентов TCP заключаются в следующем:

- Моделируется обмен пакетами установления и завершения соединения (SYN/FIN);
- Поддерживается двунаправленная передача данных;
- Последовательная нумерация (sequence number) относится к байтам, а не к пакетам.

Генерация SYN пакетов (и их ACKs) может иметь критическое значение при попытке моделирования реального поведения TCP в случае использования множества коротких передач данных. Эта версия TCP в настоящее время по умолчанию посылает данные на 3-ем сегменте инициированного троекратного рукопожатия, и ее поведение несколько отличается от реального TCP. Типичное TCP соединение происходит с отправкой SYN-а активным элементом, пассивный элемент отвечает с SYN+ACK, активный отвечает с ACK и спустя некоторое время посылает первый сегмент данных (в соответствии с первым запросом). Таким образом, эта версия TCP посылает данные раньше, чем реальный TCP протокол. В настоящее время FullTCP настроен только под алгоритм контроля перегрузки TCP Reno, но в скором времени должны стать доступными и другие версии (Tahoe, SACK, Vegas и т.д.)

Конфигурационные параметры:

- segsperack_ – число сегментов, на которое генерируется одно подтверждение;
- segsize_ – размер сегмента;

- tcprexmtthresh_ – число дублирующих подтверждений, после получения которых осуществляется переход в режим быстрого повтора передачи;
- iss_ – начальный последовательный номер первого байта передаваемых данных;
- nodelay_ – отключение (false) алгоритма Найджела (Nagle);
- data_on_syn_ – разрешение/запрет (True/False) передачи данных совместно с пакетом SYN;
- dupseg_fix_ – исключение быстрого восстановления при получении дублирующих подтверждений;
- dupack_reset_ – сброс счетчика дублирующих подтверждений при получении сегментов нулевой длины, содержащих дублирующие подтверждения.

TCP Sink

Объекты типа TCPSink представляют собой подкласс объектов агентов, моделирующий протокол TCP на стороне приемника. Отметим, что объекты TCP (кроме FullTcp) моделируют только однонаправленные TCP-соединения, в которых TCP-источник посылает пакеты данных, а приемник – только подтверждения (ACK-пакеты). Для этих объектов не определено никаких методов и переменных.

Конфигурационные параметры:

- packetSize_ – размер используемых пакетов подтверждений в байтах;
- maxSackBlocks_ – максимальное число блоков данных, которое может быть подтверждено в опции SACK. Этот параметр используется только подклассом объектов TCPSink/Sack1. Эта величина не может быть увеличена для TCPSink объекта после того, как объект создан. (После создания TCPSink объекта величина может быть уменьшена, но не увеличена).

Объекты типа TCPSink/DelAck являются подклассом объектов TCPSink, моделирующим TCP-приемник с механизмом задержанных подтверждений. Они наследуют функциональность TCPSink объектов.

Конфигурационный параметр:

interval_ – временная задержка перед генерацией подтверждения на отдельный пакет. Если следующий пакет прибывает до истечения данного времени, то подтверждение генерируется немедленно.

Объекты типа TCPSink/Sack1 – подкласс объектов TCPSink, моделирующий TCP-приемник с селективным подтверждением пакетов. Для этих объектов не определено никаких дополнительных методов, конфигурационных параметров и переменных.

Объекты типа TCPSink/Sack1/DelAck – подкласс объектов TCPSink/Sack1, моделирующий TCP-приемник с задержанным селективным подтверждением пакетов. Конфигурационный параметр у данного объекта такой же, как у TCPSink/DelAck.

1.3. Объекты типа DelayBox

DelayBox – это узел NS2, который должен быть помещен между источником и приемником. С использованием DelayBox, пакеты TCP-потока могут быть задержаны, потеряны или принудительно отправлены через топологию, так называемого «бутылочного горлышка» прежде, чем будут переданы следующему узлу. Для задания задержки, потери и скорости передачи в «бутылочном горлышке» могут быть использованы различные типы распределений. Каждый поток пакетов между источником-приемником берет свои характеристики из заданного распределения. Задержки в DelayBox определяются для всего виртуального соединения (под виртуальным соединением в данном пособии понимается совокупность логически связанных пакетов, принадлежащих одному соединению транспортного уровня), а не для отдельных пакетов. Так как DelayBox должен распознавать потоки данных – переменная fid_ (идентификатор потока flow identifier) должна быть установлена для каждого потока в симуляции. DelayBox может быть использован и с Tcp-, и с FullTcp-агентами.

Рассмотрим топологию «бутылочное горлышко» более подробно. Эта топология – одна из наиболее часто используемых в симуляциях, изображена на Рис. 1.1. Узел PC1 посылает данные узлу PC2 через маршрутизирующий узел R1, который в свою очередь соединен с маршрутизатором R2. Объем данных, производимый узлами-источниками обычно больше, чем пропускная способность линии между маршрутизаторами R1 и R2, что создает своего рода «бутылочное горлышко». Эта связь также может обладать определенной заданной вероятностью потери пакетов и односторонней задержкой.

Обычные линии связи в NS2 между узлами не обладают потерей пакетов, а только односторонней задержкой.

Рис. 1.1. Схематичное изображение механизма бутылочного горлышка

Реализация DelayBox включает в себя две таблицы: таблицу правил и таблицу потока. Вводные данные для таблицы правил добавляются пользователем на языке OTcl в симуляционном скрипте и дают структурированное представление о том, как должны обрабатываться потоки от источника к приемнику. В таблицу должны быть внесены данные об источнике, приемнике, задержке в DelayBox (в мс), потерях (доле отброшенных пакетов) и пропускной способности «бутылочного горлышка» (в МБ/с). Поле пропускной способности – опционально.

Вводные данные в таблице потока описывают, как каждый поток должен быть обработан. Конкретные значения получаются путем выборки из распределения, заданного в таблице правил. Поля таблицы потока: источник, приемник, ID потока, задержка, доля потерь и пропускная способность в «бутылочном горлышке» (если необходимо). Потоки Full-TCP определяются по началу приема первого SYN-а потока с новым ID и заканчиваются после отсылки первого FIN-а. Пакеты после первого FIN-а не задерживаются и не отбрасываются в DelayBox. Для остальных Tcp-агентов, потоки определяются по началу приема первых 40 байтов пакета потока с новым ID. Tcp-агенты в NS2 не включают схему «троекратного рукопожатия» и FIN-пакетов, потому потоки Tcp-агента никогда не считаются законченными и не удаляются из таблицы потока.

DelayBox также содержит несколько очередей для обработки пакетов с задержкой, по одной очереди на каждое соединение в таблице потока. Эти очереди – очереди с возможной дельтой, в которых время доставки пакета содержится только в первом передаваемом пакете. Все остальные пакеты

хранятся в очереди с разницей (дельтой) между временем, когда они должны быть переданы и временем, когда предыдущий пакет должен быть передан. Время, когда последний пакет должен быть передан, хранится в переменной deltasum_, название которой отражает ее суть. Это время есть сумма всех значений дельта в очереди (включая время передачи первого пакета). Если скорость для потока в «бутылочном горлышке» была задана, задержка обработки вычисляется для каждого пакета делением размера пакета на скорость потока в «бутылочном горлышке». Когда пакет получен, вычисляется его время передачи (текущее время + задержка). Это время передачи – момент, когда первый бит пакета начнет передаваться. Пакеты, пришедшие в очередь после этого, будут задержаны на время, необходимое для передачи всех битов пакета через «бутылочное горлышко»

Существует 2 сценария определения дельты пакета: если пакет нужно переслать прежде последнего бита последнего пакета в очереди, то он должен стоять в очереди, но будет готов к посылке, как только предыдущий пакет закончит свою передачу. А если пакет нужно переслать позже последнего бита последнего пакета в очереди, то его дельта – будет разница между временем его передачи и временем передачи предыдущего пакета.

Если текущий пакет – единственный в очереди, DelayBox назначает таймер для посылки пакета. Когда этот таймер истечет, DelayBox передаст пакет стандартному передатчику пакетов для отправки и начнет искать следующий пакет в очереди для передачи, устанавливать таймер для его передачи. Таким образом, все пакеты получают необходимую задержку.

Пакеты, которые должны быть отброшены, не задерживаются и не передаются. Все пакеты в очереди получены от одного соединения, задерживаются на одинаковое время (за исключением задержки, зависящей от размера пакета) и отбрасываются с одинаковой вероятностью.

Потери пакетов в DelayBox не записываются в файл trace-queue.

1.4. Исследование характеристик производительности, адаптивности и надежности различных версий TCP (Transmission Control Protocol)

Исследование характеристик различных версий протокола TCP

Протокол TCP – протокол с обратной связью, которая обеспечивает гарантированную доставку сообщений. Пакеты-подтверждения (ACK) позволяют источнику точно знать, доставлен ли пакет.

Режим передачи с гарантией доставки имеет ряд существенных недостатков:

- сеть дополнительно загружается пакетами-подтверждениями;
- TCP интерпретирует потерю пакета, как признак перегрузки сети и реагирует снижением скорости передачи;
- при максимальном заполнении буферов (приемника, окна и передатчика) велика вероятность потери пакетов.

Транспортные протоколы создают большую долю трафика в сети Интернет, поэтому эффективность механизма управления потоком определяет то, насколько эффективно будут использоваться ресурсы сети.

Попытки устранения приведенных выше недостатков TCP привели к созданию разных вариантов усовершенствования транспортного протокола.

Таким образом, эта работа предусматривает сравнительный анализ разработанных версий транспортного протокола в рамках архитектуры TCP/IP, а также рассмотрение вопроса их эффективности по различным характеристикам.

Рассмотрим три основных свойства TCP – производительность, надежность и адаптивность [2], а также механизмы, благодаря которым обеспечиваются эти свойства.

Надежность

Надежность предполагает наличие механизма контроля и коррекции ошибок.

Контроль ошибок

TCP – достоверный протокол транспортного уровня. Это означает, что прикладная программа доставляет поток данных к TCP, к прикладной программе на другом конце в порядке, без ошибок и без потери любой части или дублирования.

TCP обеспечивает достоверность, используя контроль ошибок. Контроль ошибок включает в себя механизмы обнаружения:

- искаженных сегментов;
- потери сегментов, нарушения порядка следования сегментов;
- дублирования сегментов.

Контроль ошибок также включает механизм для коррекции ошибок после того, как они обнаружены.

Обнаружение и коррекция ошибок

Обнаружение ошибок в TCP достигается с помощью использования трех простых инструментов: контрольной суммы, наличия пакетов подтверждения и контроля по времени (time-out). Каждый сегмент включает в себя поле контрольной суммы, которое используется для проверки искаженности сегмента. Если сегмент искажен, он удаляется пунктом назначения TCP. TCP использует пакеты подтверждения для получения сведений о том, что сегмент достиг пункта назначения неискаженным. Отрицательное подтверждение не используется в TCP. Если сегмент не подтвержден прежде чем окончится контрольное время, это считается признаком искажения или потери и сегмент будет передан повторно.

Возможные ошибки:

- искаженный сегмент;
- потеря сегмента;
- дублированный сегмент;

- сегмент с нарушением порядка;
- потеря подтверждения.

Исследовать надежность каждой версии протокола TCP можно при помощи наблюдения за работой механизма обратной связи, то есть за отправкой, получением пакетов и подтверждений при различных типах ошибок передачи.

Адаптируемость

Адаптируемость протокола TCP выражается в наличии механизма скользящего окна перегрузки. Различные версии TCP обладают различными алгоритмами вычисления изменений окна. В зависимости от условий передачи, отправитель рассчитывает, какое количество данных в текущий момент может быть успешно передано через конкретное виртуальное соединение, и тем самым вычисляет размер окна перегрузки.

Если представить виртуальное соединение как последовательность номеров пакетов (см. Рис. 1.2), то с течением времени размер окна сдвигается вправо, по мере того, как приемник будет подтверждать данные. Взаимное перемещение двух границ окна увеличивает или уменьшает его размер.

Рис. 1.2. Схема использования скользящего окна

Для описания перемещения границ окна вправо и влево используются три термина.

1. Окно закрывается, когда его левая граница, двигаясь вправо, совпадает с правой. Это происходит, когда данные отправлены и подтверждены.

2. Окно открывается, когда его правая граница сдвигается вправо, при этом данные могут быть отправлены. Это происходит, когда принимающий процесс читает подтвержденные данные, освобождая тем самым место в приемном буфере TCP, и пропускная способность виртуального соединения позволяет передавать данные без потерь.

3. Окно сжимается, когда его правая граница передвигается влево. На практике так делать не рекомендуют, так как TCP-соединение может быть установлено с хостом, который не поддерживает подобную опцию.

Если приняты пакеты подтверждения (ACK), которые требуют перемещения левой границы окна влево, это дублированные ACK (dupACK), то они отбрасываются.

Если левая граница окна совпала с правой, то окно закрывается – это называется нулевым окном. При этом отправитель прекращает передачу данных.

Отправитель не обязан передавать сразу полное окно данных, размер лишь ограничивает максимальное количество неподтвержденных данных, которое может передать источник. Сегмент данных от получателя, содержащий подтверждение данных, увеличивает окно перегрузки вправо. Это происходит из-за того, что размер окна связан с номером последовательности, которая была подтверждена. Получатель не должен ждать, пока окно заполнится перед отправкой пакета подтверждения (на практике, большинство реализаций посылают пакет подтверждения для каждых двух сегментов данных, которые были получены).

Размер окна получателя – это еще одно ограничение на максимальное количество неподтвержденных данных, которое предлагает получатель. На практике оно чаще всего зависит от загруженности сетевого интерфейса и компьютера получателя в целом. Этот параметр тоже может оказывать существенное влияние на производительность TCP.

Одним из главных достоинств TCP является то, что он подстраивается под состояние среды передачи данных. Загруженность среды передачи данных

может меняться в процессе работы TCP-соединения, но протокол адаптируется к этим изменяющимся условиям. Таким образом, исследуя размер окна (с помощью утилиты Xgraph выводя переменную cwnd) для различных версий протокола TCP, можно проследить реакцию различных версий протокола на наличие перегрузки в канале связи и сравнить адаптируемость этих версий.

Производительность

Важнейшим параметром любого сетевого соединения является его пропускная способность, потому для исследования и сравнения производительности различных версий TCP можно наблюдать пропускную способность для разработанной модели сети.

В наибольшей степени на производительность сети в целом влияют:

- используемые коммуникационные протоколы и их параметры;
- доля и характер широковещательного трафика, создаваемого различными протоколами;
- топология сети и используемое коммуникационное оборудование;
- интенсивность возникновения и характер ошибочных ситуаций;
- конфигурация программного и аппаратного обеспечения конечных узлов.

Размер передаваемого пакета может существенным образом повлиять на эффективную пропускную способность протокола, а значит и на производительность сети. Выясним на примере, как изменится эффективная пропускная способность протокола Ethernet, если вместо кадров минимальной длины при обмене данными будут использоваться кадры максимальной длины с полем данных в 1500 байт, как это определено в стандарте.

Размер пакета конкретного протокола обычно ограничен максимальным значением поля данных (MaximumTransferUnit, MTU), определенным в стандарте на протокол.

Необходимо отметить, что повышение размера кадра увеличивает пропускную способность сети только в том случае, когда данные в сети редко

искажаются или теряются, то есть при устойчивой, надежной работе сети. В противном случае увеличение размера пакета может привести не к увеличению, а к снижению пропускной способности, так как приложению будет необходимо повторно передавать большие порции информации в сеть. Для каждого уровня искажений данных можно подобрать рациональный размер пакета, для которого пропускная способность сети будет максимальной.

Работа с пакетами больших размеров повышает производительность сети не только за счет уменьшения накладных расходов на передачу служебной информации заголовка. При использовании больших пакетов повышается производительность коммуникационного оборудования, работающего с кадрами и пакетами, то есть мостов, коммутаторов и маршрутизаторов. Это происходит из-за того, что при передаче одного и того же объема информации число используемых больших пакетов существенно меньше, чем число маленьких, а так как коммуникационное оборудование тратит определенное время на обработку каждого пакета, то и временные потери продвижения пакетов мостами, коммутаторами и маршрутизаторами при использовании больших пакетов будут меньше.

Для сравнительного анализа различных версий протокола TCP необходимы одинаковые параметры сети, поэтому размер пакета в предлагаемых заданиях – постоянен (за исключением работ, предполагающих исследование производительности).

При сравнении характеристик различных версий протокола TCP цель моделирования определяется выбранным параметром сравнения, а также отличиями в алгоритмах различных версий, которые могут влиять на этот параметр. Как сказано выше, для протокола TCP отличительными характеристиками являются: адаптивность, надежность и производительность.

Для исследования адаптивности будем рассматривать изменения окна приемника, как реакцию на потери или задержки пакетов при передаче. Это предполагает отслеживание переменной cwnd. Выполнение скрипта позволит

получить график изменения cwnd – a, следовательно, определить, как протокол подстраивается под изменения в сети.

Для исследования производительности будем рассматривать пропускную способность.

Исследование надежности предполагает рассмотрение механизма подтверждений – ACK. Определяем реакцию каждой конкретной версии TCP на потерю пакета (или потерю ACK).

Задание

Цель работы:

- Приобретение навыков исследования протоколов транспортного уровня с помощью симулятора NS2;
- Ознакомление с особенностями различных версий протокола транспортного уровня – TCP: TCP Reno, TCP New Reno, TCP Sack, TCP Fack, TCP FullTCP, TCP Vegas;
- Изучение и сравнение основных характеристик различных версий протокола TCP.

Тестовая конфигурация:

Для исследования характеристик различных версий протокола TCP используем два варианта базовой топологии, приведенные ниже.

Модель сети с использованием классических механизмов NS2 представлена на Рис. 1.3:

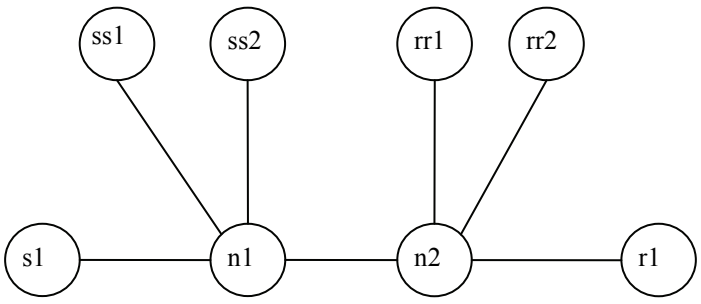

Рис. 1.3. Базовая топология сети для сравнения различных версий протокола TCP

Сеть состоит из 8 узлов: источников (s1, ss1, ss2), маршрутизирующих узлов (n1, n2) и приемников (rr1, rr2 и r1).

На каждом из узлов ss1, ss2, находятся источники трафика с экспоненциальным распределением on/off интервалов, прикрепленные к UDP-агентам. Эти источники используются для создания помех и побочного трафика – имитации работы реальной сети.

Параметры источников ss1 и ss2:

- Размер пакетов 300 и 1000 байт соответственно;
- Среднее значение длительности интервала генерации “On” burst-time 0,2 и 0,1 сек соответственно;
- Среднее значение длительности интервала ожидания “Off” idle-time 0,1 и 0,1 сек соответственно;
- Скорость передачи rate 1500 и 1000 Кбит/сек соответственно.

Источники TCP трафика имеют следующие параметры:

- Размер пакетов – 1000 байт;
- Максимальный размер окна – 20 пакетов;
- Число генерируемых пакетов источника – 100000;
- Источник начинает работу через 0.2 сек и завершает через 30 сек.

Линии связи имеют полосу пропускания 100 Мб/с и задержку распространения 10 мс, за исключением линии связи между маршрутизирующими узлами. Эта линии связи имеет следующие параметры: полосу пропускания 2 Мб/с и задержку распространения 10 миллисекунд. Размер очереди между узлами n1 и n2 – 40 пакетов. Источники трафика начинают работу в разное время (через 0.1 сек – ss1 и ss2, через 0.2 сек – s1) после начала моделирования и продолжают работу до 30 сек. Моделирование прекращается через 31 сек.

С помощью утилиты nam иллюстрируется передача пакетов от источников к приемникам, а также пакеты подтверждения (ACK).

После выполнения сценария, результаты работы симулятора записаны в trace файл out-{версия_TCP}.nam.

Источники ss1 и ss2 используются для моделирования различной загруженности канала и создаваемый ими трафик в выходном файле можно не отслеживать. Будем наблюдать за состоянием очереди в линии n1-n2.

Адаптивность

Для наблюдения параметра cwnd, при формировании выходных данных для nam и xgraph можно воспользоваться следующим вариантом процедур finish и plotWindow:

```
proc finish {} {
 global ns nf tcp(версия TCP)
 $ns flush-trace
 close $nf
 close $tcp(версия TCP)
 exec nam out-(версия TCP).nam
 exec xgraph tcp(версия TCP) -geometry 800x400 -t "cwnd" -x
"Time, secs" -y "Window, pkts"
 exit 0
 }
```

Процедура установления значений параметров для вывода окна Xgraph:

```
proc plotWindow {tcpSource file _cwnd_} {
 global ns
 set time 0.01
 set now [$ns now]
 set cwnd [$tcpSource set cwnd_]
# _cwnd_ - переменная, хранящая площадь под графиком cwnd
 set temp $_cwnd_
 set _cwnd_ [expr ($temp + $cwnd*$time)]
 set temp $_cwnd_
 set cwnd_rez $_cwnd_
 puts $file "$now $cwnd $cwnd_rez"
# выносим _cwnd_ в вывод файла
```

```
 $ns at [expr $now+$time] "plotWindow $tcpSource $file
$_cwnd_"
}
```

В данном скрипте, создание и прикрепление источников к узлу оформлено в виде процедуры, параметрами которой являются ссылки на соответствующие узлы:

```
proc attach-expoo-traffic { node sink size burst idle rate }
{
   set ns [Simulator instance]
   set source [new Agent/CBR/UDP]
   $ns attach-agent $node $source
   set traffic [new Traffic/Expoo]
   $traffic set packet-size $size
   $traffic set burst-time $burst
   $traffic set idle-time $idle
   $traffic set rate $rate
   $source attach-traffic $traffic
   $ns connect $source $sink
   return $source
}
```

Для запуска симуляции с требуемыми параметрами необходимы следующие команды (например, для версии TCP Newreno):

```
$ns at 8.0 "finish"
$ns at 0.0 "plotWindow $tcp1 $tcpNewreno $cwnd_rez"
$ns run
```

Для установки параметров протокола и запуска источника версии TCP FullTCP необходимо написать команды:

```
set src1 [new Agent/TCP/FullTcp]
set sink1 [new Agent/TCP/FullTcp]
$src1 set maxcwnd_ 10
$src1 set packetSize_ 1000
```

```
$src1 set fid_ 1
$sink1 set fid_ 1
$ns attach-agent $s1 $src1
$ns attach-agent $r1 $sink1
$ns connect $src1 $sink1
$sink1 listen
$ns at 0.2 "$src1 advance 200"
```

Надежность

В заданиях [4] входные данные для xgraph формировались непосредственно командами симулятора. Часто удобно для дальнейшего использования, в том числе для визуального отображения, выходные данные симулятора предварительно обработать (отфильтровать, записать в определенном формате и т.д.). Это можно сделать при помощи следующего скрипта:

Процедура обработки выходных данных

```
proc finish {file mod} {
```

Создаем и подготавливаем выходной файл данных

```
 exec rm -f temp_(версия TCP).rands
 set f [open temp_(версия TCP).rands w]
 puts $f "TitleText: $file"
 puts $f "Device: Postscript"
 exec rm -f temp_(версия TCP).p
 exec touch temp_(версия TCP).p
```

Обрабатываем файл данных симулятора out_версия TCP.tr и заносим из него данные о полученных/отправленных пакетах очереди во временный файл temp_(версия TCP).p – нас интересуют поставленные в очередь и отправленные пакеты TCP.

```
  exec awk {
     {
     if (($1 == "+" || $1 == "-") && \
```

```
          ($5 == "tcp"))\
           print $2, ($8-1)*(mod+10) + ($11 % mod)
   }
} mod=$mod out_(версия TCP).tr >temp_(версия TCP).p
```

Заносим данные об отброшенных пакетах TCP во временный файл temp_(версия TCP).d – далее этот файл будет использоваться для подсчета количества отброшенных пакетов.

```
 exec rm -f temp_(версия TCP).d
 exec touch temp_(версия TCP).d
  exec awk {
    {
     if (($1 == "d" ) && \
         ($5 == "tcp"))\
         print $2, ($8-1)*(mod+10) + ($11 % mod)
   }
} mod=$mod out_(версия TCP).tr >temp_(версия TCP).d
```

Обрабатываем 2-ой поток от приемника к источнику:

```
exec rm -f temp_(версия TCP).p2
exec touch temp_(версия TCP).p2

 exec awk {
    {
    if (($1 == "-") && \
        ($5 == "ack"))\
    print $2, ($8-1)*(mod+10) + ($11 % mod)
    }
 }  mod=$mod out2_(версия TCP).tr >temp_(версия TCP).p2
```

Заносим данные из временных файлов temp.p и temp.d в выходной файл для xgraph *temp_(версия TCP)*.rands:

```
 puts $f \"packets
```

```
 flush $f
 exec cat temp_(версия TCP).p >@ $f
 flush $f
 puts $f \n\"acks
 exec cat temp_(версия TCP).p2 >@ $f
puts $f \n\"drops
 flush $f

 exec head -1 temp_(версия TCP).d >@ $f
 exec cat temp_(версия TCP).d >@ $f
 close $f
 set tx "time (sec)"
 set ty "packet number (mod $mod)"
```

Запускаем xgraph с входным файлом temp.rands:

```
 exec xgraph -bb -tk -nl -m -zg 0 -x $tx -y $ty temp_(версия
TCP).rands &
 exit 0
}
```

Для реализации подобной процедуры, необходимо также указать следующее:

```
set label "TCP"
set mod 80
…
```

Рассматриваем 2 потока – от источника к приемнику и от приемника к источнику:

```
exec rm -f out_(версия TCP).tr
set fout [open out_(версия TCP).tr w]
$ns trace-queue $n1 $n2 $fout

exec rm -f out2_(версия TCP).tr
set fout2 [open out2_(версия TCP).tr w]
$ns trace-queue $n2 $n1 $fout2
```

Для увеличения наглядности результатов можно сократить скорости передачи источников трафика нагрузки, пропускную способность канала n1-n2, а также общее число передаваемых пакетов источником s1 и время симуляции.

Производительность

Графический вывод пропускной способности предполагает свою отдельную процедуру.

Необходимо создать 2 потока – 1 для вывода собственно графика, другой – для вывода среднего значения пропускной способности:

```
set f0 [open out0_(версия TCP).tr w]
set f1 [open out1_(версия TCP).tr w]

proc finish {} {
 global f0 f1
 close $f0
 close $f1
 exec xgraph out0_(версия TCP).tr out1_(версия TCP).tr -
geometry 800x600 -x "Time, secs" -y "bytes" \
 -0 bytes_per_sec -1 average &
 exit 0
}
```

Также необходимо 2 монитора – один из которых будет выводить информацию с заданным периодом и необходим для построения графика, другой – будет накапливать информацию обо всех полученных битах и необходим для вывода среднего

```
set qm0 [$ns monitor-queue $n1 $n2 [$ns get-ns-traceall]]
set qm1 [$ns monitor-queue $n1 $n2 [$ns get-ns-traceall]]
```

Эта процедура осуществляет мониторинг очереди с помощью объектов qm0, qm1. Состояние очереди контролируется через заданный интервал

времени, определяемый переменной time и осуществляется запись результатов в выходной файл, определяемый переменными f0,f1:

```
proc trqueue {} {
 global qm0 qm1 f0 f1
 set ns [Simulator instance]
 set time 0.1
 set end_time 8.0
 set q1 [$qm0 set barrivals_]
 set q2 [$qm1 set barrivals_]
 set now [$ns now]
 puts $f0 "$now $q1"
 puts $f1 "$end_time [expr $q2/(80+$now)]"
 $qm0 reset
 $ns at [expr $now+$time] "trqueue"
}
```

Необходимо добавить команду $ns at 0.0 «trqueue» для запуска.

Модель сети с использованием блока DelayBox

Прежде всего, для использования этого блока, в начало скрипта добавляются следующие команды:

```
remove-all-packet-headers; # удаляет все заголовки пакетов
add-packet-header IP TCP; # добавляет заголовки TCP/IP
set ns [new Simulator]
global defaultRNG
$defaultRNG seed 999
```

Изменится топология и количество узлов:

```
set s1 [$ns node]
set r1 [$ns node]
set db(0) [$ns DelayBox]
set db(1) [$ns DelayBox]
```

```
$ns duplex-link $db(0) $db(1) 2Mb 10ms DropTail
$ns duplex-link-op $db(0) $db(1) orient right
$ns duplex-link $s1 $db(0) 100Mb 10ms DropTail
$ns duplex-link-op $s1 $db(0) orient right
$ns duplex-link $r1 $db(1) 100Mb 10ms DropTail
$ns duplex-link-op $r1 $db(1) orient right
```

Поскольку создание помех и ошибок теперь осуществляется блоком DelayBox, необходимо задать лишь один, главный, источник трафика.

```
set snk1 [new Agent/TCPSink]
$ns attach-agent $r1 $snk1
```

Задаем переменные, отвечающие за параметры «бутылочного горлышка»:

```
set recvr_delay [new RandomVariable/Uniform]; # задаем
задержку приемника
$recvr_delay set min_ 1
$recvr_delay set max_ 20
set sender_delay [new RandomVariable/Uniform]; # задаем
задержку отправителя
$sender_delay set min_ 20
$sender_delay set max_ 100
set recvr_bw [new RandomVariable/Constant]; # скорость
приемника 100 Mbps
$recvr_bw set val_ 100
set sender_bw [new RandomVariable/Uniform]; # скорость
отправителя 1-20 Mbps
$sender_bw set min_ 1
$sender_bw set max_ 20
set loss_rate [new RandomVariable/Uniform]; # задаем долю
потерь
$loss_rate set min_ 0
$loss_rate set max_ 0.05
```

Задаем правила для DelayBox:

```
$db(0) add-rule [$s1 id] [$r1 id] $recvr_delay $loss_rate
$recvr_bw
$db(1) add-rule [$s1 id] [$r1 id] $sender_delay $loss_rate
$sender_bw
```

Выводим задержки в файлы:

```
$db(0) set-delay-file "db0.out"
$db(1) set-delay-file "db1.out"

# Завершающие команды:
$ns at 0.2 "$ftp1 start"
$ns at 10.0 "$ftp1 stop"
$ns at 0.2 "$ftp1 produce 100000"
$ns at 1000.0 "$db(0) close-delay-file; $db(1) close-delay-
file; exit 0"
$ns at 8.0 "finish"
$ns at 0.0 "plotWindow $tcp1 $tcp(версия TCP)_db $cwnd_rez "
$ns run
```

Задание к работе:

Исследуйте характеристики (адаптивность, производительность, надежность) различных версий протокола TCP с использованием вышеупомянутых моделей сети и объясните полученные результаты.

Расшифровка результатов моделирования.

Надежность

При выполнении скриптов, исследующих надежность TCP, будут созданы trace-файлы стандартного формата симулятора с данными мониторинга очереди соединения между узлами n1 и n2. Процедура finish создает вспомогательные файлы temp_(версия TCP).p и temp_(версия TCP).d с информацией, соответственно, об успешно переданных и отброшенных пакетах в очереди

рассматриваемого соединения. Далее эти данные объединяются и заносятся в соответствующем формате во входной файл для xgraph – temp_(версия TCP).rands. Таким образом, во входном файле имеется 2 набора данных (полученные/отправленные очередью пакеты данных и отброшенные пакеты).

Для удобства вывода, номера пакетов выводятся с периодом 80 (80 mod). В окне xgraph отложено время моделирования, по оси Y – номера пакетов.

Графики показывают поставленные в очередь пакеты и уже посланные (красные точки), отброшенные обозначаются синими кругами, пакеты АСК – выделены салатовым.

Первые пакеты практически не задерживаются в очереди и метки, отображающие их движение, на графике сливаются. С увеличением загруженности очереди время между поступлением в нее пакета и его отправлением начинает увеличиваться, что видно по увеличению расстояния на графике между метками.

Надежность TCP обеспечивается наличием обратной связи, то есть пакетами подтверждения (АСК).

Наиболее надежным является протокол с меньшими потерями, поэтому необходимо подсчитать количество потерянных пакетов и составить сводную таблицу. При этом следует подсчитать количество отброшенных пакетов на различных интервалах времени, например, 4 секунды, 15 секунд, 30 секунд и объяснить полученные результаты.

При недостаточно крупном изображении потерь, график в окне xgraph можно увеличить, выделив требуемую часть мышкой.

Адаптивность

После выполнения скриптов work9_(версия TCP).tcl результаты моделирования будут выведены в окне xgraph. На полученных графиках можно проследить ряд применяемых в протоколе TCP алгоритмов:

- Slow Start (медленный старт);
- Fast Retransmit (быстрый повтор передачи);

- Congestion Avoidance (предупреждение перегруженности).

На графиках видно, что часть пакетов передаются без потерь, при этом, окно TCP экспоненциально возрастает в соответствии с алгоритмом медленного старта. После получения подтверждения от приемника окно увеличивается на 1 пакет. В дальнейшем, в фазе медленного старта (размер окна увеличивается на 1 максимальный сегмент (MSS) для каждого принятого ACK; таким образом, размер окна отправителя TCP увеличивается по экспоненте) число переданных пакетов, определяемое текущим окном перегрузки, каждый раз удваивается, пока доставка всех пакетов подтверждается приемником.

При благоприятном исходе размер окна перегрузки достигает максимально возможного, но не более заявленного со стороны приемника окна получателя. Алгоритм формирования cwnd по сигналу обратной связи в виде пакетов подтверждения называется алгоритмом скользящего окна. При этом возможны два сценария развития процесса передачи данных: режим тайм-аута и режим быстрой повторной передачи. При возникновении тайм-аута один или часть пакетов теряются, или чрезмерно задерживаются (время задержки превышает тайм-аут). В этом случае пакеты подтверждения не отсылаются и на стороне источника снова формируются окно перегрузки cwnd = 1 пакету, а также устанавливается порог медленного старта sstresh = cwnd/2, cwnd – окна перегрузки, при котором произошел тайм-аут (см. Рис. 1.4). После него рост окна перегрузки в режиме предотвращения перегрузки становится линейным (окно отправителя увеличивается на один пакет за каждый цикл RTT (round trip time)). В результате потерянные пакеты, а также посланные после них, пересылаются повторно.

В режиме повторной передачи получатель до истечения времени тайм-аута сигнализирует о приходе пакетов с нарушением порядка следования, что может быть следствием флуктуации задержки или потери пакетов. В этом случае, получатель отсылает три или более задублированных пакета подтверждения и, не дожидаясь тайм-аута, значение окна перегрузки уменьшается в соответствии

с предусмотренным для данной версии TCP алгоритмом с дальнейшим линейным увеличением окна перегрузки.

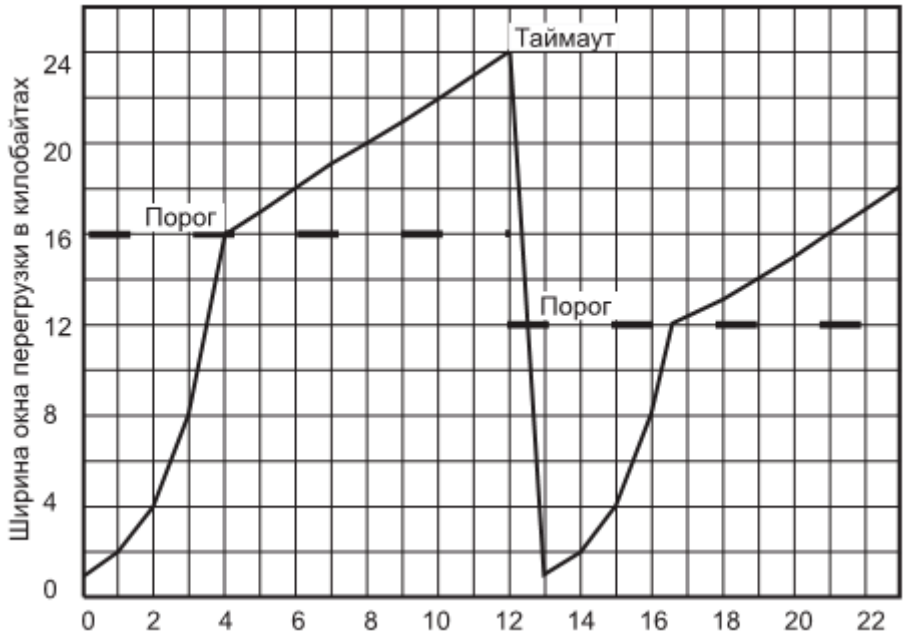

Рис. 1.4. Эволюция размера окна перегрузки при медленном старте

В случае если АСК теряется:

- Когда подтверждение теряется, отправитель вынужден ожидать следующего подтверждения для восстановления потери. Частые потери подтверждений могут привести к значительному времени ожидания, замедляя передачу и снижая пропускную способность системы. Более того, поскольку алгоритмы TCP – медленный старт и устранение перегруженности увеличивают окно передачи, основываясь на количестве полученных подтверждений, потеря подтверждений замедляет рост окна, уменьшая скорость передачи.
- В случае потери всех подтверждений пакетов окна данных, отправитель обнаруживает потерю, только после окончания интервала таймаута.

Создайте скрипты work9_2_Reno.tcl, work9_2_NewReno.tcl, work9_2_Vegas.tcl, work9_2_Fack.tcl, work9_2_Sack1.tcl, work9_2_FullTCP.tcl для моделирования данной ситуации, и отследите изменение размера окна (cwnd). Также необходимо провести моделирование с использованием блока

DelayBox для каждой версии протокола TCP. Наиболее эффективным будет считаться протокол с большей площадью под графиком cwnd (больше переданных пакетов) и лучше отрабатывающий алгоритмы TCP. На графиках вывода отметьте режимы TCP и приведите объяснения перехода от режима к режиму. Сравните площади под графиками функциональных зависимостей размера окна от времени – с помощью просмотра значения третьего столбца последней строки файлов вывода симулятора «tcp(версия TCP)», «tcp(версия TCP)_db».

Производительность

Существует несколько причин резкого сокращения пропускной способности канала, возникающего при потере пакетов данных.

Во-первых, потеря каждого пакета данных (в отличие от потерь подтверждений) приводит к необходимости одной или более повторной передачи. При потере пакета, не смотря на использование механизмов быстрого восстановления после быстрого повтора передачи, протокол TCP уменьшает окно передачи вдвое и работает с окном меньшим максимально допустимого в течение нескольких циклов RTT. Последовательные периодические потери, возникающие до того, как TCP достигнет своего максимального окна передачи, могут привести к значительному уменьшению скорости передачи.

Во-вторых, поскольку механизм быстрой повторной передачи повторяет передачу пакета только после получения трех дублирующих подтверждений, быстрый повтор передачи становится не эффективным, при сокращении TCP окна меньше 4. В этом случае, после потери пакета, протокол TCP может восстановить передачу, только после окончания интервала таймаута. Таймаут также возникает в случае потери нескольких пакетов из одного окна данных. При этом, не смотря на то, что источник данных может отследить потерю первого пакета и осуществляет восстановление с помощью быстрого повтора передачи, часто, после восстановления, размер окна TCP не достаточно велик, для передачи трех двойных подтверждений, необходимой для восстановления

после потери второго пакета. Иногда даже 1% потери пакетов данных могут привести к 50% снижению эффективной пропускной способности [1].

При выводе графиков пропускной способности в окне xgraph также будет выводиться среднее значение пропускной способности виртуального канала на интервале моделирования. Увеличением окна измерьте это значение (выделено салатовым в конце временной шкалы) и составьте сводную таблицу результатов и объясните полученные результаты.

Программа практической работы

1. В соответствии с приведенными выше моделями и заданием создайте файлы

Адаптивность (также с использованием DelayBox):

work7_1_Reno.tcl, work7_1_Reno_db.tcl,

work7_1_NewReno.tcl, work7_1_NewReno_db.tcl,

work7_1_Vegas.tcl, work7_1_Vegas_db.tcl,

work7_1_Fack.tcl, work7_1_Fack_db.tcl,

work7_1_Sack1.tcl, work7_1_Sack1_db.tcl,

work7_1_FullTCP.tcl, work7_1_FullTCP_db.tcl

Проведите сравнение эффективности работы различных версий протокола по функциональной зависимости изменения размера окна от времени.

2. Модифицируйте скрипты предыдущего задания в соответствии с требованиями для лабораторных посвященных исследованию механизма надежности. Сохраните скрипты в виде файлов.

Надежность (также с использованием DelayBox):

work7_2_Reno.tcl, work7_2_Reno_db.tcl,

work7_2_NewReno.tcl, work7_2_NewReno_db.tcl,

work7_2_Vegas.tcl, work7_2_Vegas_db.tcl,

work7_2_Fack.tcl, work7_2_Fack_db.tcl,

work7_2_Sack1.tcl, work7_2_Sack1_db.tcl,

work7_2_FullTCP.tcl, work7_2_FullTCP_db.tcl

В выводах приведите описание механизма обеспечения надежности с помощью АСК-ов и сводную таблицу результатов, а также поясните полученные результаты.

3. Создайте скрипты для исследования производительности TCP-соединений. Создайте файлы

Производительность (также с использованием DelayBox):

work7_3_Reno.tcl, work7_3_Reno_db.tcl,

work7_3_NewReno.tcl, work7_3_NewReno_db.tcl,

work7_3_Vegas.tcl, work7_3_Vegas_db.tcl,

work7_3_Fack.tcl, work7_3_Fack_db.tcl,

work7_3_Sack1.tcl, work7_3_Sack1_db.tcl,

work7_3_FullTCP.tcl, work7_3_FullTCP_db.tcl

Площадь под графиком пропускной способности отражает эффективность протокола. Составьте сводную таблицу средних эффективностей различных версий протокола, сравните работу различных версий протокола между собой и объясните полученные результаты.

По результатам выполнения всех работ по сравнению различных версий TCP – приведите сводную таблицу и кратко опишите плюсы и минусы каждой версии протокола. Определите, какие механизмы, и при каких условиях, оказывают наибольшее влияние на работу TCP.

2. СРЕДСТВА АНАЛИЗА ПРОЦЕССОВ В КОМПЬЮТЕРНЫХ СЕТЯХ

2.1. Многофункциональная программа MTraffic

MTraffic – Многофункциональная программа для анализа трафика в компьютерных сетях, разработанная в СПбГПУ на кафедре телематики для учебно-лабораторного комплекса по исследованию процессов в компьютерных сетях. Программа имеет пользовательский графический интерфейс и зарегистрирована в государственном реестре программных продуктов.

Функциональные возможности:

- исследование характеристик, используемых администраторами для отслеживания работоспособности сети (интенсивность трафика, значения размеров пакетов, количество пакетов, пришедших за заданный интервал времени, интервал времени между приходом пакетов, вычисление максимальных, минимальных показателей по каждому из параметров и т.д., вывод графиков по запросу пользователя);
- исследование трафика с использованием методов статистического анализа (подсчет дисперсии, среднего, показателя Хэрста, коэффициентов пачечности и пиковости);
- исследование трафика с использованием методов нелинейной динамики (анализ числовых рядов, поиск внутренней зависимости в них (корреляции), построение уравнений динамической системы). Для ускорения вычислений предусмотрено распараллеливание наиболее ресурсоёмкого фрагмента кода и автоматический расчёт его на кластере.

Структура MTraffic приведена на Рис. 2.1.

Разработанное инструментальное средство имеет модульную структуру:

- Common_parametrs.m (общий анализ),
- Classical_analyze.m (методы статистического анализа),
- rtcp_test.m. (анализ протокола RTCP),

- polyrebuild_diff.m (восстановление уравнения дифференциальным методом),
- polyrebuild_razn.m (восстановление уравнения разностным методом),
- Size_analyzer.m (обработка временного ряда),
- MTraffic.m (главное меню).

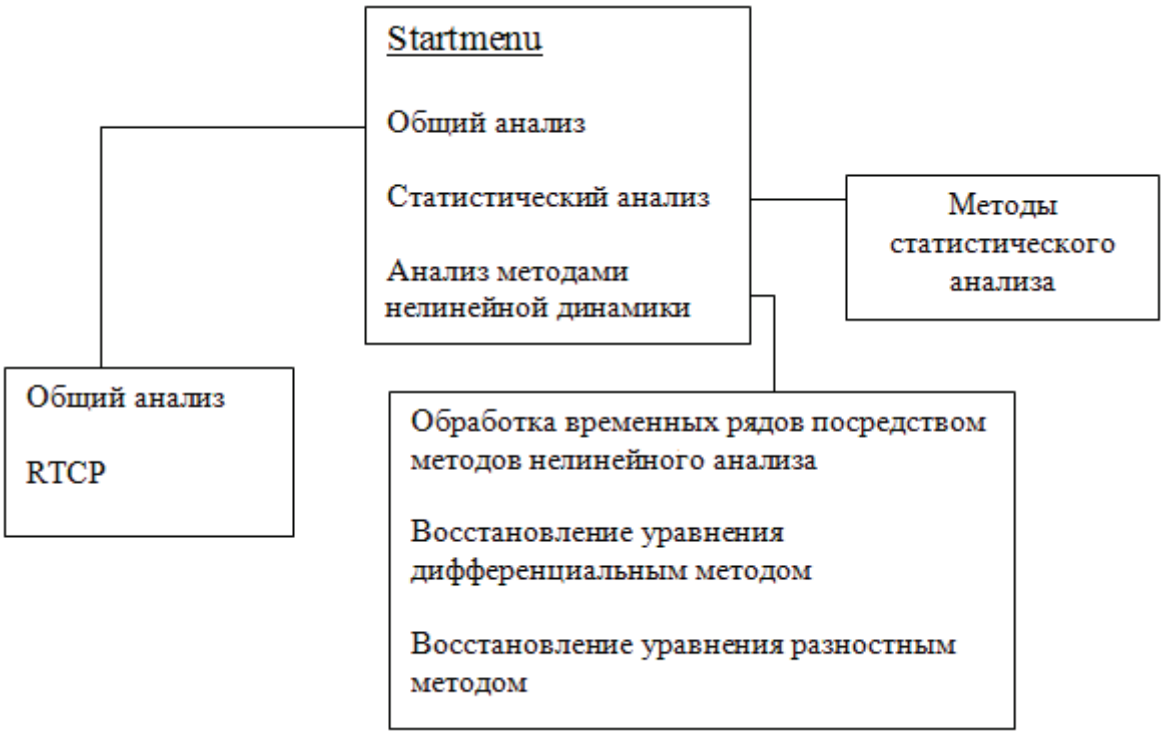

Рис. 2.1. Структурная схема программы MTraffic

2.2. Общий анализ трафика различных типов

Основные задачи данной работы:

- ознакомление со структурой программного комплекса MTraffic;
- получение навыков работы с модулем общего анализа;
- получение навыков оценки состояния сети;
- сравнение характеристик для трафика различных типов (nonreal-time, realtime).

Методика выполнения работы:

1. Откройте дамп в программе Wireshark (работа с программой Wireshark подробнее описана в [5]).

2. Проанализируйте дамп, обратите внимание на следующие характеристики:

- частота встречаемости протоколов;
- доля IP-трафика ко всему остальному;
- между какими IP-адресами происходит передача пакетов наиболее интенсивно;
- определите, какой тип трафика встречается в дампе (real-time, передача большого массива данных).

В зависимости от типа предоставляемого сервиса выделяются две основные категории трафика:

1. Трафик реального времени (real-time), предоставляющий мультимедийные услуги для передачи информации между пользователями в реальном масштабе времени.

2. Трафик обычных данных (nonreal-time), который образуется традиционными распределенными услугами современной телекоммуникационной сети, такими как электронная почта, передача файлов, виртуальный терминал, удаленный доступ к базам данных и др.

В качестве примеров услуг, генерирующий трафик реального времени, можно привести следующие: IP-телефония, высококачественный звук, видеотелефония, видеоконференцсвязь, дистанционное (удаленное) медицинское обслуживание (диагностика, мониторинг, консультация), видеомониторинг, широковещательное видео, цифровое телевидение, вещание радио- и телевизионных программ.

В таблицу сведены типичные значения для всех вышеперечисленных параметров мультимедийных услуг.

Таблица. Параметры трафика мультимедийных услуг (типичные значения)

Тип мультимедийного сервиса	Параметры мультимедийных трафиков		
	$V_{\max}$, Мбит/с	$V_{ср}$, Мбит/с	$T_{ср}^{(p)}$, с
IP-телефония	0,064	0,064	100
Высококачественный звук	1	1	53
Видеотелефония	10	2	1
Видеоконференция	10	2	1
Дистанционное медицинское обслуживание	10	2	1
Видеомониторинг	10	2	–
Вещание радио- и телевизионных программ	34	34	–
Цифровое телевидение	34	34	–

Нередко для передачи трафика реального времени используется протокол RTP (например, H.323, SIP). Приложения обычно используют RTP поверх протокола UDP для того, чтобы использовать его возможности мультиплексирования и проверки целостности при помощи контрольной суммы. Но RTP может использоваться и поверх любого другого транспортного протокола.

3. Запустите программу Matlab. Зайдите в каталог MTraffic. Запустите файл MTraffic.m.

Структурная схема программного комплекса приведена на рис. 2.1. Главное меню программы MTraffic.m показано на рис. 2.2.

Рис. 2.2. Главное меню программы MTraffic

4. В диалоговом окне (рис. 2.2) нажмите кнопку «Общий анализ»

Внешний вид модуля, отвечающего за общий анализ, приведен на рис. 2.3.

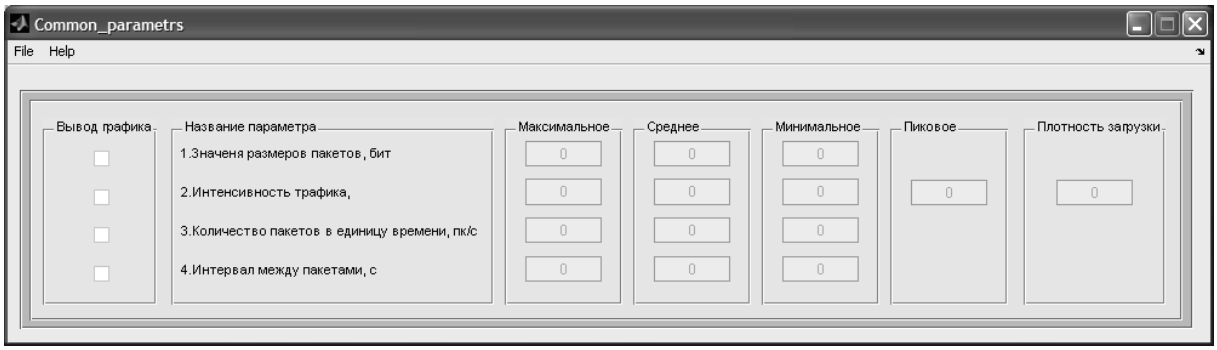

Рис. 2.3. Графический интерфейс модуля общего анализа

Панель меню состоит из двух разделов:

Files:

- Open – загрузка данных;
- Save – сохранение рассчитанных коэффициентов;
- Exit – завершение работы модуля.

Help:

- Help MTraffic – справка по программе, открывается на локальной странице, посвященной модулю;
- About MTraffic – о программе.

Поля:

- Вывод графиков – набор переключателей. Переключатель в установленном режиме выводит график требуемой характеристики.
- Название параметра – наименование характеристики, для которой производятся расчеты и построение графиков.
- Максимальное/Среднее/Минимальное/Пиковое/Плотность загрузки.

5. Загрузите данные в программу: File⇒Open.

Данные для модуля должны состоять из двух колонок (без наименований колонок): относительное время прихода пакета (relative time) и его размер в байтах (length byte).

Как только модуль закончит вычисления, станет доступен набор переключателей «Вывод графика».

6. Проанализируйте полученные данные.

7. Повторите пункты 1-6 для следующего дампа.

8. Сравните полученные результаты обоих дампов.

9. Есть ли в дампе протокол RTCP? Проведите исследование пакетов с этим протоколом в программах Wireshark и RTCP (MTraffic)

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.

RTCP выполняет четыре функции:

1. Главной задачей данного протокола является обеспечение обратной связи для контроля качества при рассылке данных.

2. RTCP имеет постоянный идентификатор транспортного уровня для RTP источника, который называется каноническим именем или cname. Этот дополнительный идентификатор нужен, так как SSRC-идентификатор может быть изменен в случае, если будет зафиксировано столкновение или источник будет вынужден перезапуститься.

3. Первые две функции требуют, чтобы все участники посылали RTCP-пакеты, следовательно, скорость передачи должна контролироваться для того, чтобы RTP мог работать с большим числом участников. При посылке каждым участником своих управляющих пакетов всем остальным любой партнер может независимо определить полное число участников сессии. Это число используется при вычислении частоты посылки пакетов.

4. Четвертая опционная функция служит для передачи минимальной управляющей информации, например идентификаторов участников, для графического интерфейса пользователя. Это полезно для «слабо управляемых» сессий, когда участники входят и выходят без должного контроля и без согласования параметров.

Пакет отчета отправителя состоит из трех секций (см. рис. 2.4), за которыми может следовать четвертая, которая определяется, если необходимо, профайлом. Наибольший интерес, в случаи невозможности детального анализа всего пакета целиком, представляет секция блока отчета 1 – это третья секция, состоящая из нуля или более блоков отчета о приеме в зависимости от числа источников, пакеты от которых приняты с момента последнего отчета. Каждый блок отчета о приеме несет в себе статистику получения RTP-пакетов, поступающих от одного из источников синхронизации. Получатель не сохраняет статистику, когда источник изменяет свой ssrc-идентификатор.

Каждый блок состоит из:

- SSRC_n (идентификатор источника): 32 бита.

SSRC-идентификатор источника, к которому относится информация, содержащаяся в блоке отчета о получении.

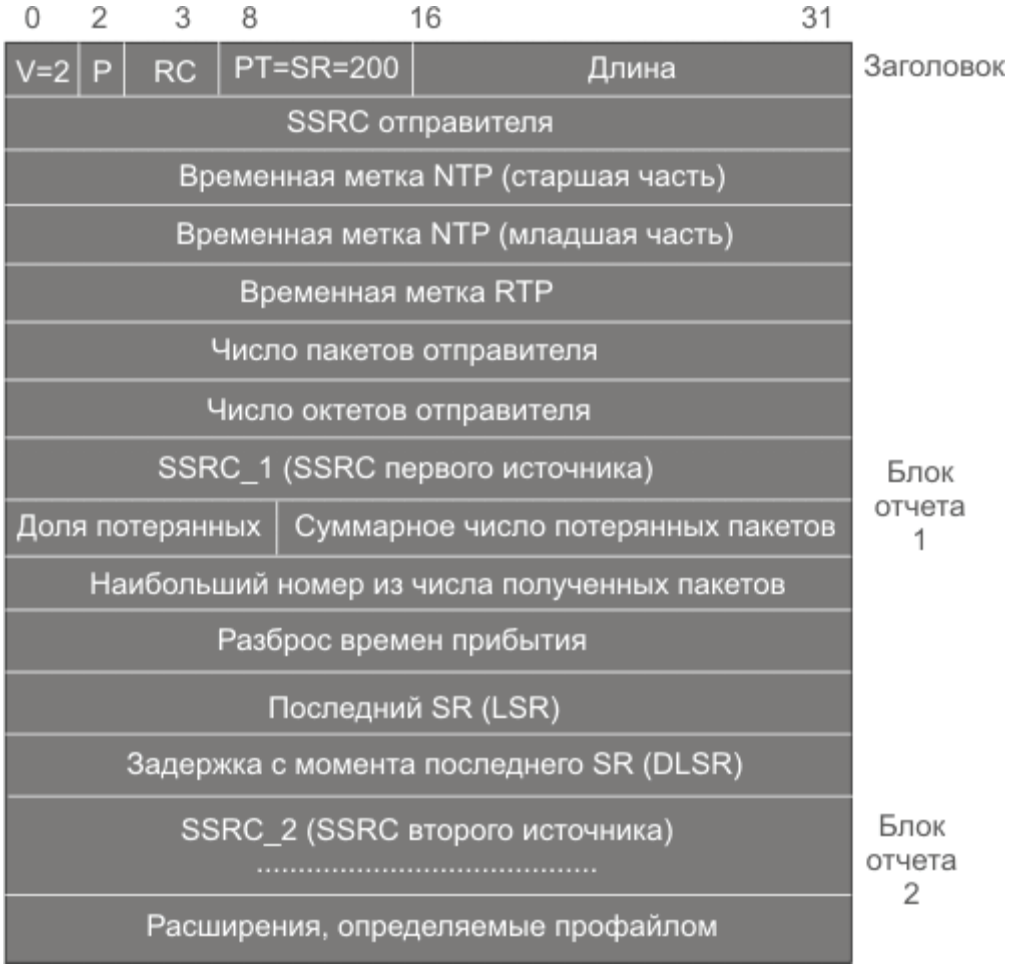

Рис. 2.4. Формат RTCP пакета сообщения отправителя

- Часть потерянных (пакетов): 8 бит.

Доля информационных RTP-пакетов от источника ssrc_n, потерянных с момента посылки предыдущего SR или RR-пакетов, представленная в виде числа с фиксированной запятой, помещенной слева. Эта доля получается в результате деления числа потерянных пакетов на ожидаемое число пакетов. Если потери из-за дубликатов оказались отрицательны, доля потерянных пакетов объявляется равной нулю. Заметим, что от источника, все пакеты которого были потеряны при транспортировке, отчета о приеме послано не будет.

- Суммарное число потерянных пакетов: 24 бита.

Полное число информационных RTP-пакетов от источника SSRC_n, которые были потеряны с момента начала передачи. Это число определяется как разность между ожидаемым и полученным числами пакетов, где число полученных включает в себя и дубликаты. Таким образом, пакеты, пришедшие с опозданием, не считаются потерянными, а число потерянных пакетов может оказаться отрицательным, если получены дубликаты пакетов. Число ожидаемых пакетов определяется как разность между номером последнего полученного пакета и номером первого пакета.

- Наибольший номер из числа полученных пакетов: 32 бита.

Младшие 16 бит содержат наибольший порядковый номер полученного от источника SSRC_n информационного RTP-пакета. Старшие 16 бит несут в себе число циклов нумерации (переполнения счетчика номеров пакетов). Заметим, что различные получатели в рамках одной и той же сессии генерируют разные коды циклов нумерации (расширений), если они начали свою работу в разное время.

- Разброс времени доставки: 32 бита.

Оценка статистической вариации периода прихода RTP-пакетов, измеряемого с помощью временных меток и характеризуемого целым числом. Разброс периода прихода пакетов j определяется как усредненное отклонение разности D расстояния между пакетами со стороны получателя по отношению к той же величине для стороны отправителя. Эта величина характеризует относительный разброс времени транспортировки пакетов.

Если S_i равно временной метке *i*-го пакета RTP, а R_i – время прибытия в единицах временной метки пакета *i*, тогда для двух пакетов *i* и *j* *D* может быть выражено, как показано в формуле:

$$D_{i,j} = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i).$$

Разброс времени доставки вычисляется непрерывно для каждого пребывающего от SSRC_n пакета *i*, используя разность *D* для данного пакета и предыдущего пакета *i-1* согласно формуле:

$$j = j + \frac{\left|D_{i-1,i}\right| - j}{16}.$$

Вычисление разброса времени доставки позволяет мониторам, независимым от профайла, осуществлять интерпретацию докладов, приходящих от различных приложений.

10. Откройте дамп в Wireshark, создайте фильтр отображения, чтобы был показан только протокол RTCP.

11. Выполните Export ⇒ File. Установите переключатель в положение Displayed, оставьте галочки как показано на рис. 2.5. Сохраните файл в формате *.txt.

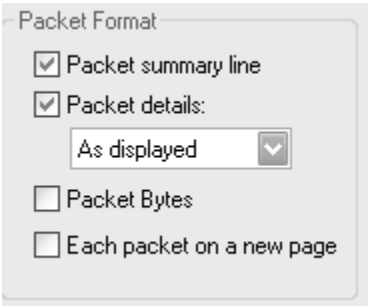

Рис. 2.5. Меню программы Wireshark Export File

12. Нажмите кнопку RTCP в MTraffic (рис. 2.6) и загрузите полученный дамп.

Поле «вывод графиков» – набор переключателей. Переключатель в установленном режиме выводит график требуемой характеристики.

Численные значения для характеристик «доля потерянных пакетов», «разброс времен прибытия пакетов», «задержка с момента последнего SR-отчета» выводятся усредненными. В поле для численного значения

характеристики «суммарного числа потерянных пакетов» выводится значение, полученное из последнего SR-отчета.

13. Проанализируйте полученные данные.

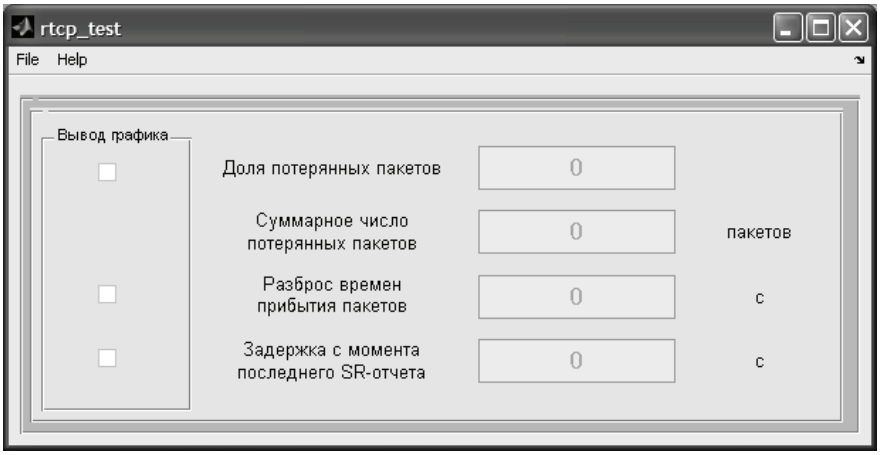

Рис. 2.6. Графический интерфейс модуля для анализа заголовка протокола RTCP

Варианты заданий

- Для дампов voice.pcap, voice_skype_data.pcap voice_data.pcap проделать пункты 1-7. Сравнить данные. Как сказывается передача обычных данных на данные реального времени и есть ли отличия в характеристиках, если передача данных ведется другим приложением,
- Для дампов voice.pcap, voice_video.pcap, voice_data.pcap проделать пункты 1-7. Сравнить данные.
- Для дампов voice_video.pcap, voice_video_skype, data.pcap, voice_video_data.pcap проделать пункты 1-7. Сравнить данные.
- Для дампов voice.pcap, IPTV.pcap, ftp.pcap, http.pcap проделать пункты 1-7. Сравнить данные.

В отчете привести:

1. Анализ характеристик из пункта 2.
2. Результат выполнения пункта 6,7,8,13.
3. Выводы.

2.3. Анализ трафика статистическими методами

Для обретения навыков анализ реального трафика статистическими методами, студенты должны решить ряд задач:

- знакомство с методами статистического анализа;
- знакомство с подпрограммой статистического анализа;
- получение навыков статистического анализа.

Методика анализа трафика статистическими методами

1. Запустите программу Matlab. Зайдите в каталог MTraffic. Запустите файл MTraffic.m. В диалоговом окне (рис. 2.2) нажмите кнопку «Статистический анализ».

В открывшемся диалоговом окне (рис. 2.7) загрузите дамп, формат дампа такой же, как и для общего анализа.

Поля графиков:

- график сигналов,
- график автокорреляционной функции,
- график спектра сигнала.

Процессы, происходящие в компьютерных сетях, представляются в моделях как непрерывные или дискретные случайные процессы. В общем случае полностью описать случайный процесс практически невозможно и реально приходится ограничиваться использованием только некоторых характеристик (моменты невысоких порядков – математическое ожидание, дисперсия, автокорреляционная функция и т.д.) Трафик – дискретный случайный процесс.

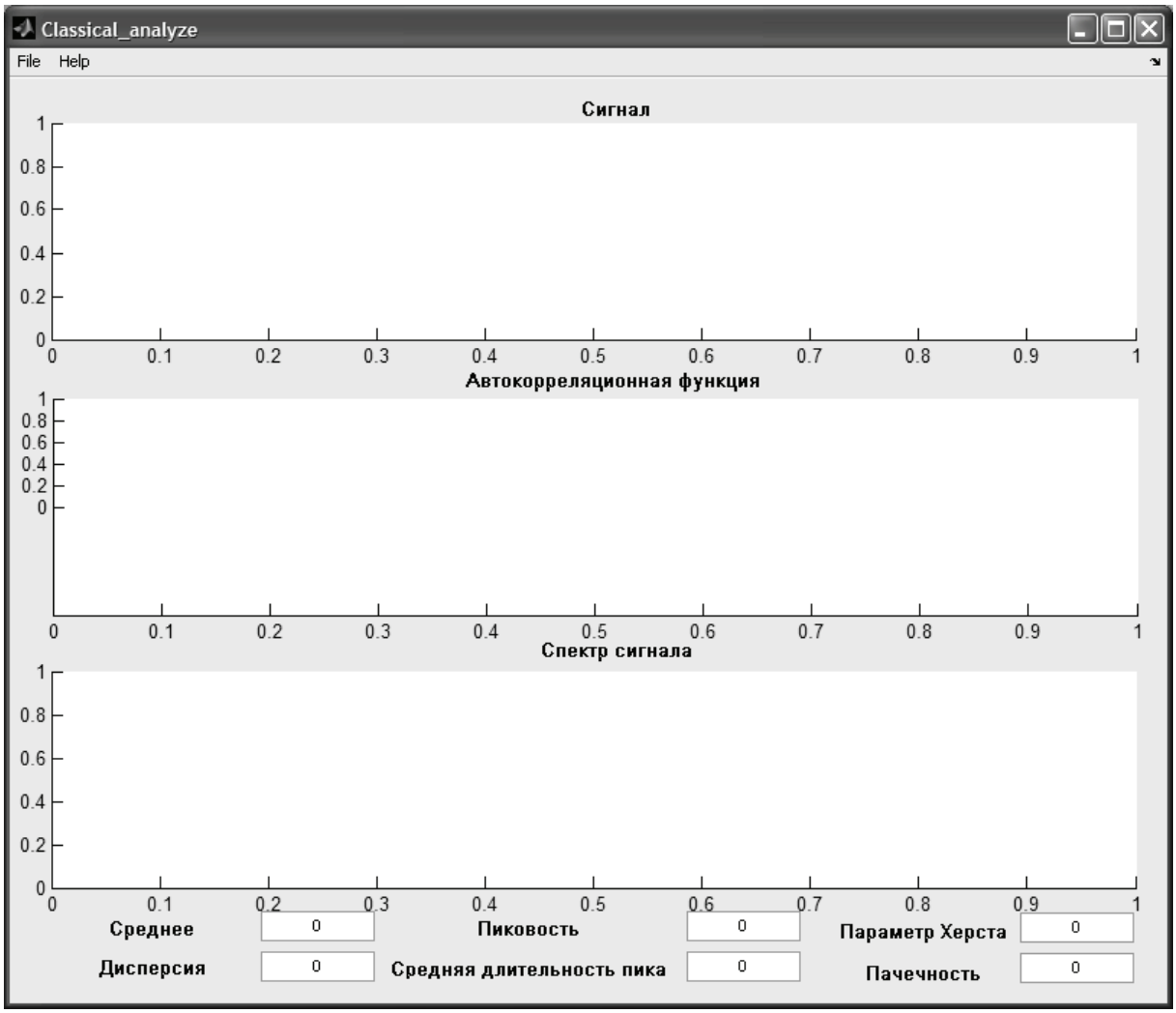

Рис. 2.7. Графический интерфейс модуля статистического анализа

Среднеквадратическое отклонение определяется по формуле :

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

где $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ и n – число элементов в выборке.

Дисперсия вычисляется по формуле:

$$D[x] = M[(x - M[x])^2],$$

где $M[x] = \bar{x}$ – среднее.

Автокорреляция – корреляционная связь между значениями одного и того же случайного процесса в разнесенные моменты времени. Автокорреляционная функция (АКФ) характеризует эту связь. В общем случае АКФ характеризует внутреннюю зависимость между временным рядом и тем же рядом, но сдвинутым на некоторый промежуток (сдвиг) времени, который называется лагом. Вычисления АКФ проводятся по классической формуле:

$$r(k) = \frac{\sum_i (x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_i (x_i - \bar{x})^2},$$

где $\bar{x}$ – выборочное среднее, k = 0, 1, 2, … – смещение АКФ во времени.

Понятие медленно убывающей зависимости (автоковариации) имеет ключевое значение в теории самоподобных процессов и фактически описывает интересное в отношении прогнозирования свойство – продолжительную память. На интуитивном уровне данное свойство можно объяснить следующим образом: будущее процесса определятся его прошлым, причем с убывающей степенью влияния по мере того, как прошлое удалено от настоящего. Таким образом, процесс с продолжительной памятью хорошо «помнит» свое недавнее прошлое, но как бы «постепенно забывает» свои давно минувшие состояния по мере продвижения времени в будущее.

Процессы с медленно убывающей зависимостью (МУЗ) характеризуются автокорреляционной функцией, которая убывает гиперболически (по степенному закону) при увеличении сдвига АКФ во времени (лага). В отличие от процессов с МУЗ процессы с быстро убывающей зависимостью (БУЗ) (Short-Range Dependence) обладают экспоненциально спадающей АКФ (рис. 2.8). Самоподобный в широком смысле процесс характеризуется инвариантностью АКФ при изменении уровня агрегирования при условии МУЗ.

Известно, что в частотной области МУЗ отражается на характерном степенном законе поведения спектральной плотности рассматриваемого процесса.

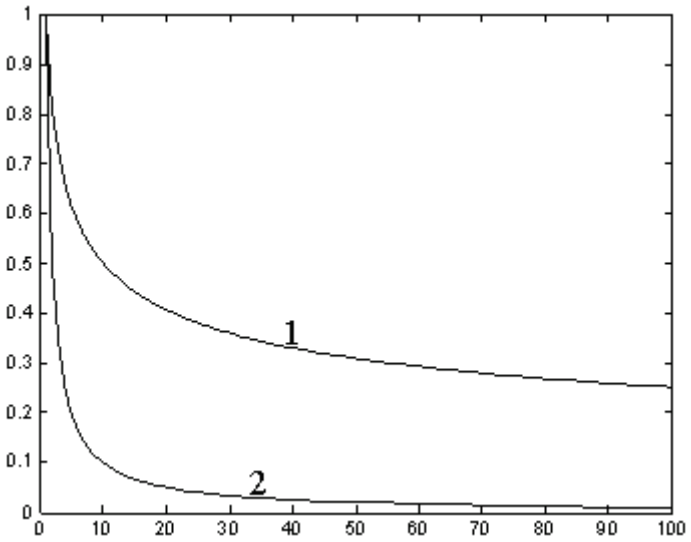

Рис. 2.8. АКФ БУЗ (кривая №2) и АКФ МУЗ (кривая №1)

Мощным инструментом обработки данных, определенных дискретной зависимостью $y(x_i)$ или непрерывной функцией *f(x)* (полученной, например, посредством интерполяции), является спектральный анализ, имеющий в своей основе различные интегральные преобразования. Спектральный анализ используется как в целях подавления шума, так и для решения других проблем обработки данных. Спектром совокупности данных *y(x*) называют некоторую функцию другой координаты (или координат) *F(w)*, полученную в соответствии с определенным алгоритмом. Примерами спектров являются преобразование Фурье и вейвлет-преобразование. Каждое из интегральных преобразований эффективно для решения своего круга задач анализа данных.

Задачами, непосредственно связанными со спектральным анализом, являются проблемы сглаживания и фильтрации данных. Они заключаются в построении для исходной экспериментальной зависимости $y(x_i)$ некоторой (непрерывной или дискретной) зависимости *f(x)*, которая должна приближать ее, учитывая к тому же, что данные $(x_i y_i)$ получены с некоторой погрешностью, выражающей шумовую компоненту измерений. При этом функция *f(x*) с помощью того или иного алгоритма уменьшает погрешность, присутствующую в данных $(x_i y_i)$. Такого типа задачи называют задачами фильтрации.

Случайную реализацию *x(t)* можно разложить по детерминированным ортогональным функциям:

$$x(t) = \sum c_n \varphi_n(t) .$$

Коэффициенты такого разложения c_n будут случайными величинами. Для гармонического разложения коэффициенты определяются из формулы:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{X}(\omega) e^{i\omega t} d\omega , \; \dot{X}(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt .$$

Ввиду случайности спектральной плотности $\dot{X}(\omega)$ и равенства нулю ее среднего значения при усреднении по всем реализациям при $\bar{x} = 0$ (ввиду случайности и независимости фаз спектральных составляющих в различных реализациях) она не используется для характеристики случайного процесса. Поэтому для случайного процесса *x(t)* вводится понятие спектральной плотности мощности, связанной с автокорреляционной функцией преобразованием Фурье (соотношение Винера – Хинчина):

$$K_k(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} W_k(\omega) e^{-i\omega\tau} d\omega , \; W_k(\omega) = \int_{-\infty}^{\infty} K_k(\tau) e^{-i\omega\tau} d\tau .$$

Спектральная плотность мощности определяется из последнего соотношения по функции корреляции определяемой для эргодического процесса в пределах одной реализации:

$$K_k(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t) x(t+\tau) dt .$$

При нулевом среднем значении $\overline{x(t)} = 0$ получим формулу:

$$\overline{x^2}(t) = \sigma_k^2 + \overline{x(t)}^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} W_k(\omega)d\omega .$$

Чем шире энергетический спектр случайного процесса, тем быстрее меняется *x(t)* и меньше время корреляции, и наоборот.

Параметр Херста является мерой самоподобия или статистической инерции процесса. Оценки Херст-параметра (H) основываются на идее измерения наклона линейного приближения на графике. Примером такой оценки является, так называемая вариограмма или R/S оценка. График зависимости R/S от дискретного времени N в логарифмическом масштабе по обеим шкалам использует тот факт, что для самоподобной последовательности данных диапазон изменения масштаба или R/S-статистика растет согласно степенному закону с экспонентой H как функция числа включенных точек (N). Таким образом, график R/S в зависимости от N на графике в логарифмическом масштабе имеет наклон, который является оценкой H.

Далее, пусть имеется дискретная реализация наблюдений $x_1, x_2, \ldots, x_N$ в соответствующие моменты времени $t_1, t_2, \ldots, t_N$, где N – объем выборки отсчетов.

Обозначим через $g_j(t)$ накопленное отклонение процесса $x_i(t)$ от среднего $\bar{x}$ к моменту времени t_j :

$$g_j = \sum_{i=1}^{j}(x_i - \bar{x}) .$$

Разность между максимальным и минимальным накопленным отклонением $g_j(t)$ определяется как размах накопленного отклонения R по формуле

$$R = \max_{1\le j\le N}\{g_j\} - \min_{1\le j\le N}\{g_j\} .$$

Параметр Херста (Hurst) H определяется из соотношения:

$$\frac{R}{S} = (aN)^H,$$

где R – размах отклонения, S – стандартное отклонение, N – число членов временного ряда, a – константа.

Для самоподобных процессов параметр Херста оценивается из формулы логарифмированием правой и левой части отношения $R/S \sim (N/2)^H$.

Используя значение показателя Херста, выделяют три типа случайных процессов:

$0 \le H < 0{,}5$ – случайным процесс является антиперсистентным или эргодическим рядом, который не обладает самоподобием;

$H = 0{,}5$ – полностью случайный ряд, аналогичный случайным смещениям частицы при классическом броуновском движении;

$H > 0{,}5$ – персистентный (самоподдерживающийся) процесс, который обладает длительной памятью и является самоподобным.

Исследователи отметили, что параметр Херста для сетевого трафика находится в интервале (0.5, 1). На качественном уровне такой самоподобный трафик имеет постоянный «взрывной» характер (burstiness), то есть обладает высокой пачечностью на всех масштабах временной оси.

Коэффициент пачечности (пачечность) для заданного процесса соответствует отношению пиковой интенсивности процесса поступления заявок на обслуживание к его среднему значению. Самоподобие можно расценивать, как фундаментальное статистическое свойство сетевого трафика, которое необходимо учитывать на практике.

2. Загрузите данные.

Данные для модуля должны состоять из двух колонок (без наименований колонок): относительное время прихода пакета (relative time) и его размер в байтах (length byte).

После обработки входных данных и проведения подсчетов характеристик, вывода графиков сигнала и спектра сигнала, пользователю будет предложено установить значения среднего выборочного и сдвига во времени для расчета

автокорреляционной функции (рис. 2.9). По умолчанию будет установлены значения:

- выборочное среднее – подсчитанное среднее значение для сигнала,
- временной сдвиг – 1.

Пользователь может воспользоваться дополнительно модулем для расчета сдвига во времени (lag), нажав кнопку «Расчёт лага».

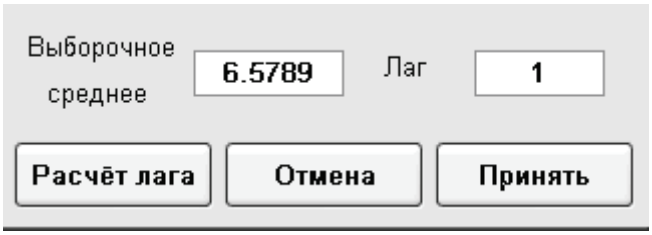

Рис. 2.9. Установка значения выборочного среднего и сдвига АКФ во времени

Далее производится расчет АКФ и выводится ее график.

3. Является ли процесс персистентным?

Варианты заданий

- Определите, как меняются характеристики в зависимости от длины выборки.
- Произведите разбивку дампа на файлы, где выборки будут одинаковой длины (к примеру, для дампа за 1 час взять 6 выборок по 10 минут), и для каждой произведите оценку характеристик. Как согласуются вновь полученные характеристики с характеристиками для исходного дампа?
- Проследите динамику изменения АКФ в зависимости от выборочного значения, сдвига во времени и влияния на оценку МУЗ (LRD).

В отчете привести:

1. Название дампа, тип трафика, к которому относятся данные (real-time, nonreal-time).
2. Используемые значения выборочного среднего и сдвига во времени.
3. Анализ полученных характеристик.
4. Выводы.

2.4. Анализ трафика методами нелинейной динамики

Целью данной работы будет являться получение студентами навыков прогнозирования и оценки длины прогноза для реального сетевого трафика методами нелинейной динамики.

Основные задачи, решаемые в работе:

- ознакомление с анализом данных методами нелинейной динамики;
- ознакомление с подпрограммой анализа методами нелинейной динамики;
- получение навыков прогнозирования и оценки длины прогноза.

Методика выполнения работы

Один из методов исследования системы – построение и изучение её фазовой траектории. Выводы о характере системы можно сделать из наличия или отсутствия у неё аттрактора в фазовом пространстве, а также из вида этого аттрактора, если он присутствует. Аттрактор – множество точек фазового пространства динамической системы, к которому она стремится с течением времени. Существует несколько видов аттракторов (рис. 2.10).

Если со временем система переходит в состояние равновесия, то её аттрактором будет являться точка (рис. 2.10, кривая а). Точка – один из основных видов аттракторов.

Также существуют следующие аттракторы:

- цикл – замкнутая кривая в фазовом пространстве, образ движения, повторяющегося с периодом T (рис. 2.10, кривая б);
- тор – «бесконечно тонкая нитка, наматывающаяся на бублик», образ квазипериодических движений (с двумя характерными периодами T_1 и T_2, находящимися в иррациональном соотношении) (рис. 2.10, кривая в). Тор может иметь три и более измерений – представлять сложное движение с тремя, четырьмя и т.д. некратными частотами синусоидальных компонент;

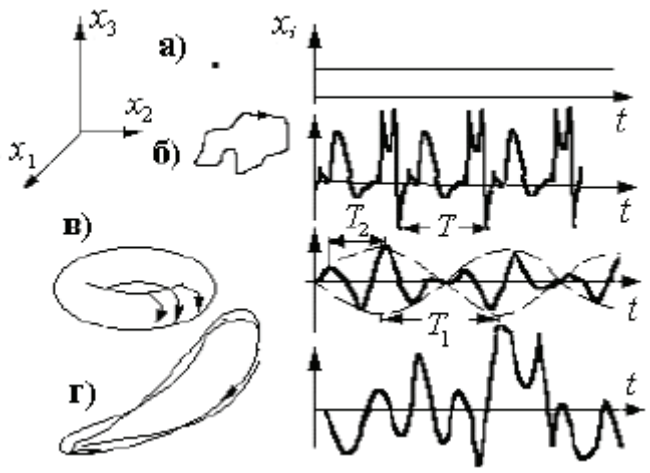

Рис. 2.10. Примеры характерных множеств в фазовом пространстве систем с непрерывным временем и временных реализаций движений, которым они соответствуют

- фрактально устроенное множество, сосредоточенное в ограниченной области фазового пространства, образ хаотических колебаний – странный аттрактор (рис. 2.10, кривая г).

Реализуемые в динамических системах варианты установившихся движений и соответствующие им аттракторы ограничены размерностью динамической системы (ДС). Так в фазовом пространстве систем с непрерывным временем (с операторами, представленными дифференциальными уравнениями) при D = 1 могут существовать лишь состояния равновесия, при D = 2 – точки равновесия и циклы, при D ≥ 3 – все перечисленные ранее предельные множества. Эти сведения могут помочь определиться с выбором размерности модели на практике. Обнаружение, например, хаотических колебаний свидетельствует о том, что для моделирования объекта с помощью нелинейных дифференциальных уравнений первого порядка их потребуется не менее трёх.

Вид аттрактора для систем дискретным временем можно представить, разрезав левые картинки на рис. 2.10 плоскостью (сечение Пуанкаре). Точка и однооборотный цикл дадут в разрезе точку. Более сложные циклы – несколько точек. Траектория на торе точками прокола секущей плоскости «нарисует»

замкнутую кривую, которая, в дискретной системе, представляет в фазовом пространстве квазипериодическое движение. Хаотический аттрактор будет представлять собой сложно структурированный (часто, самоподобный) набор точек.

Если множество в восстановленном фазовом пространстве неупорядоченно, то, скорее всего исследуемый сигнал случаен.

Кроме визуально фиксируемых различий портреты в фазовом пространстве обладают набором количественных мер. Наиболее популярными среди них являются размерности.

Целочисленную топологическую размерность D_T можно определить по индуктивному принципу [7]: для точки устанавливается размерность $D_T=0$; множество, которое может быть разделено на непересекающиеся части подмножеством размерности D_T, имеет размерность D_T+1. В соответствии с этими правилами гладкая линия имеет топологическую размерность $D_T=1$, поверхность – $D_T=2$, объём – $D_T=3$. Соответственно, точка равновесия, цикл и тор имеют топологические размерности 0, 1 и 2.

Структура странных аттракторов качественно отличается от перечисленных множеств. Они фрактальны (самоподобны), что требует введения более сложных мер – фрактальных размерностей. Наиболее простая из них – ёмкость, оценивает только геометрию аттрактора. Вводятся и обобщённые размерности, учитывающие посещаемость его подмножеств изображающей точкой.

Для определения ёмкости предельное множество в D-мерном фазовом пространстве покрывается D-мерными кубами (т.е. отрезками, квадратами, трёхмерными кубами и т.д.), со стороной ε. Вместо кубов можно использовать D-мерные шары или множества другой формы.

Большинство известных фрактальных множеств имеет дробную размерность и их можно «уложить» в пространство, размерность которого равна фрактальной размерности, дополненной до целой величины. Так, Канторово множество уже не конечный набор точек, но еще и не линия.

Более тонкая характеристика – размерность Хаусдорфа – обобщает ёмкость на случай покрытия элементами произвольной формы и размера. Обе величины часто совпадают, но не всегда [8]. Точная численная оценка хаусдорфовой размерности, как правило, невозможна. Обобщённые размерности Реньи D_q «учитывают» частоту посещения изображающей точкой той или иной области аттрактора [9, 10].

Пусть аттрактор разбит на N непустых кубов (ячеек) размера ε. Обозначим долю времени, проводимого изображающей точкой в ячейке номер i, через p_i. Это нормированная плотность точек в ячейке – оценка вероятности её посещения. Тогда

$$D_q = \frac{1}{q-1} \lim_{\varepsilon \to 0} \frac{\ln \sum_{i=1}^{N(\varepsilon)} p_i^q}{\ln \varepsilon}, \qquad (2.1)$$

где D_q – обобщенная размерность, ε – размер ячеек, N – количество ячеек, p_i – вероятность посещения точкой ячейки.

Выделяют специальные виды обобщенной размерности: при q = 0 – это описанная выше ёмкость, при q = 1 – информационная размерность (в смысле предела при $q \to 1$), при q = 2 – корреляционная размерность. Последняя характеризует асимптотическое поведение пар точек на аттракторе. Действительно, величины p_i^2 можно интерпретировать, как вероятность найти две изображающие точки внутри i-го куба размера ε. Как раз эту величину легче всего оценить. Прямое использование определения (2.1) приводит к вычислительным трудностям при D > 3. Поэтому развиты различные численные методы оценки размерностей по фрагменту дискретизованной во времени фазовой траектории $x(t_1), x(t_2), \ldots, x(t_N)$.

Одним из самых известных является алгоритм Грассбергера – Прокаччиа для оценки корреляционной размерности

$$C(\varepsilon) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Theta(\varepsilon - \|x(t_i) - x(t_j)\|),$$

где N – количество точек, Θ – функция Хевисайда (Θ(s) = 0, s ≤ 0, Θ(s) = 1, s > 0), ||.|| – норма вектора, $x(t_i)$ – элемент вектора состояния.

Физический смысл этой величины – оценка вероятности того, что две точки, произвольно выбранные на аттракторе в соответствии с его вероятностной мерой, окажутся на расстоянии, меньшем ε. При ε → 0 имеет место $C(\varepsilon) \approx A\varepsilon^{D_2}$. Тогда корреляционную размерность D_2 можно оценить как наклон графика lnC(lnε) при малом ε. На практике число точек траектории N ограничено, поэтому размер ячейки ε нельзя сделать сколь угодно малым. Причём для надёжной оценки размерности требуется тем большее число точек, чем больше размерность.

Именно этот метод используется в программе для определения корреляционной размерности аттрактора. Строится несколько корреляционных интегралов для различных размерностей и определяется поведение угла наклона линейного участка графика корреляционного интеграла с увеличением размерности. Когда рост угла наклона прекращается, то его значение принимается за величину корреляционной размерности D. Основываясь на оценке размерности D и теореме о вложении, можно заключить, что размерность фазового пространства этой динамической системы не превышает 2D+1.

Для целочисленной оценки размерности наблюдаемого движения сверху используются и другие идеи. Один из наиболее популярных – метод ложных близких соседей – основан на проверке того свойства, что фазовая траектория, восстановленная в пространстве достаточной размерности не должна иметь самопересечений. Он применяется для восстановления фазовой траектории по временной реализации единственной переменной.

Другой известный метод – метод главных компонент – заключается в выделении в фазовом пространстве направлений, вдоль которых, в основном, развивается движение изображающей точки, на основе анализа корреляций между компонентами вектора состояния.

Кроме размерности из графиков корреляционного интеграла можно также судить о детерминированности/случайности сигнала и о наличии в сигнале шумов.

Если сигнал детерминированный, то все графики корреляционного интеграла практически сливаются в одну линию. Если же сигнал случаен, то при увеличении размерности вектора состояния график корреляционного интеграла должен проходить всё ниже и ниже.

Если сигнал зашумлён, то угол наклона линейной части графика, соответствующей малым ε, будет расти, а в области больших ε он достигнет насыщения. В незашумлённом сигнале угол наклона всего линейного участка графика с ростом ε меняется одинаково.

Рассмотрим более конкретно одну из схем реконструкции отображения по наблюдаемому временному ряду $x_1, x_2, \ldots, x_N$ [11]. Пусть размерность фазового пространства модели то уже выбрана. Используя для реконструкции векторов состояния метод запаздываний со сдвигом p = 1, построим набор векторов (2.2), отвечающих последовательным точкам траектории.

$$y(n) = (x_n, x_{n-1}, x_{n-p}, \ldots, x_{n-m+1}) \equiv (y_1(n), y_2(n), \ldots, y_m(n)). \qquad (2.2)$$

Соотношения, связывающие компоненты вектора состояния в два последовательных момента времени, можно представить формулой (2.3).

$$\begin{cases} y_1(n+1) = y_2(n), \\ y_2(n+1) = y_3(n), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_{m-1}(n+1) = y_m(n), \\ y_m(n+1) = f(y_1(n), y_2(n), \ldots, y_m(n)). \end{cases} \qquad (2.3)$$

Функцию *f*, которая должна быть определена посредством обработки реализации, представим в виде ряда до кубических членов

$$f(y_1, y_2, ..., y_m) = \sum_{i=1}^{m} A_i y_i + \sum_{i=1}^{m}\sum_{j=1}^{m} B_{ij} y_i y_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} C_{ijk} y_i y_j y_k \,. \qquad (2.4)$$

Фигурирующие здесь неизвестные коэффициенты A_i, B_{ij}, C_{ijk}, должны быть подобраны так, чтобы модель (2.3) обеспечивала наилучшее соответствие с наблюдаемым временным рядом, критерием чего является минимизация суммы

$$S = \sum_{n=m}^{N-1} (x_n - f(x_{n-m}, x_{n-m+1}, ..., x_{n-1}))^2 .$$

Для достижения минимума все частные производные от S по коэффициентам A_i, B_{ij}, C_{ijk} должны обращаться в нуль [12], что дает систему линейных алгебраических уравнений для определения этих неизвестных коэффициентов.

Аналогичная схема для непрерывного времени получается, если для построения вектора состояния использовать производные от наблюдаемой переменной $x(t)$ по времени и наложить условие:

$$y(t) = (x, \dot{x}, \ddot{x}, \dddot{x}, ..., x^{(m-1)}) \equiv (y_1(t), y_2(t), ..., y_m(t)) .$$

Тогда система уравнений для компонент вектора y записывается в следующем виде:

$$\begin{cases} \dot{y}_1 = y_2, \\ \dot{y}_2 = y_3, \\ ..... \\ \dot{y}_{m-1} = y_m, \\ \dot{y}_m = f(y_1, y_2, ..., y_m). \end{cases}$$

Представим функцию f в виде (2.4) и для наилучшего соответствия модели и реализации потребуем, чтобы интеграл в формуле (2.5) был минимален.

$$S = \int (x^{(m)} - f(x, \dot{x}, ..., x^{(m-1)}))^2 \, dt \,. \qquad (2.5)$$

Для этого частные производные от S по коэффициентам A_i, B_{ij}, C_{ijk} должны обращаться в нуль, что приводит к системе линейных алгебраических уравнений, из которой получаются указанные коэффициенты, определяющие конкретный вид уравнений динамической системы.

Третий этап анализа, отвечающий за реализацию методов нелинейной динами имеет следующий графический интерфейс: модуль обработки временных рядов (рис. 2.11) и модули восстановления уравнения полинома (рис. 2.12) по двум методам, разностному и дифференциальному. Обе последние имеют единый графический интерфейс.

На основе временного ряда (зафиксированный в последовательные моменты времени ряд значений) подпрограмма обработки строит траекторию системы в фазовом пространстве, строит графики корреляционных интегралов для заданных размерностей, по которым вычисляется корреляционная размерность системы.

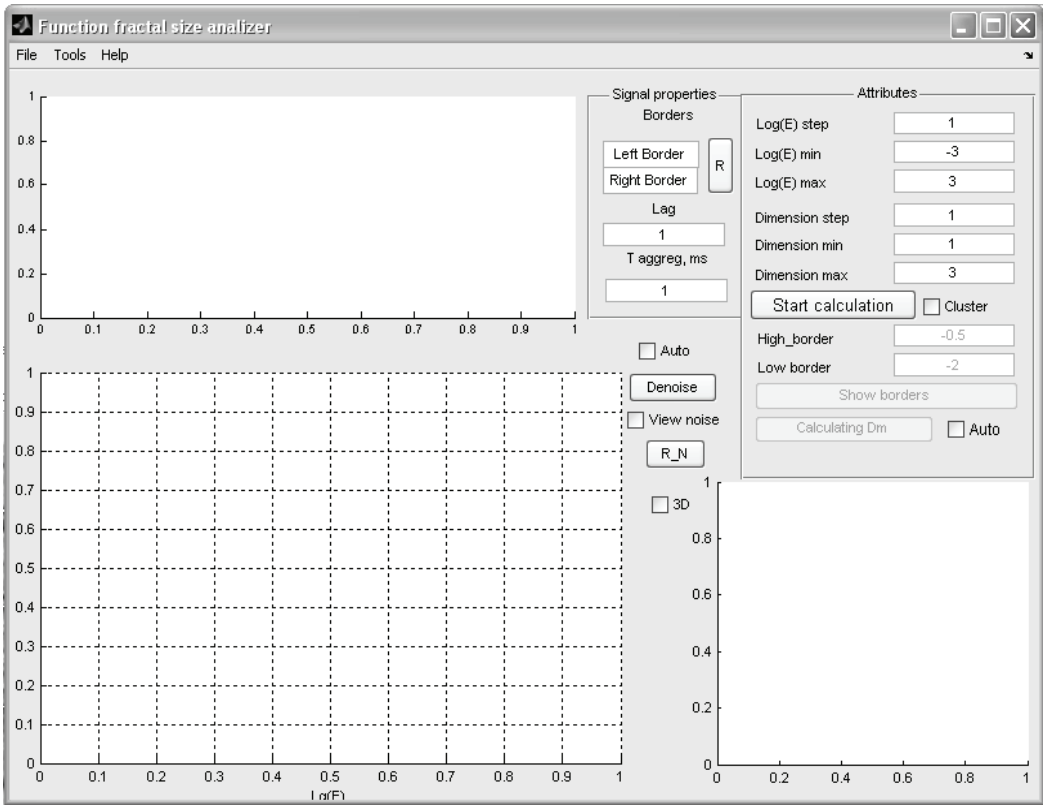

Рис. 2.11. Модуль обработки временных рядов

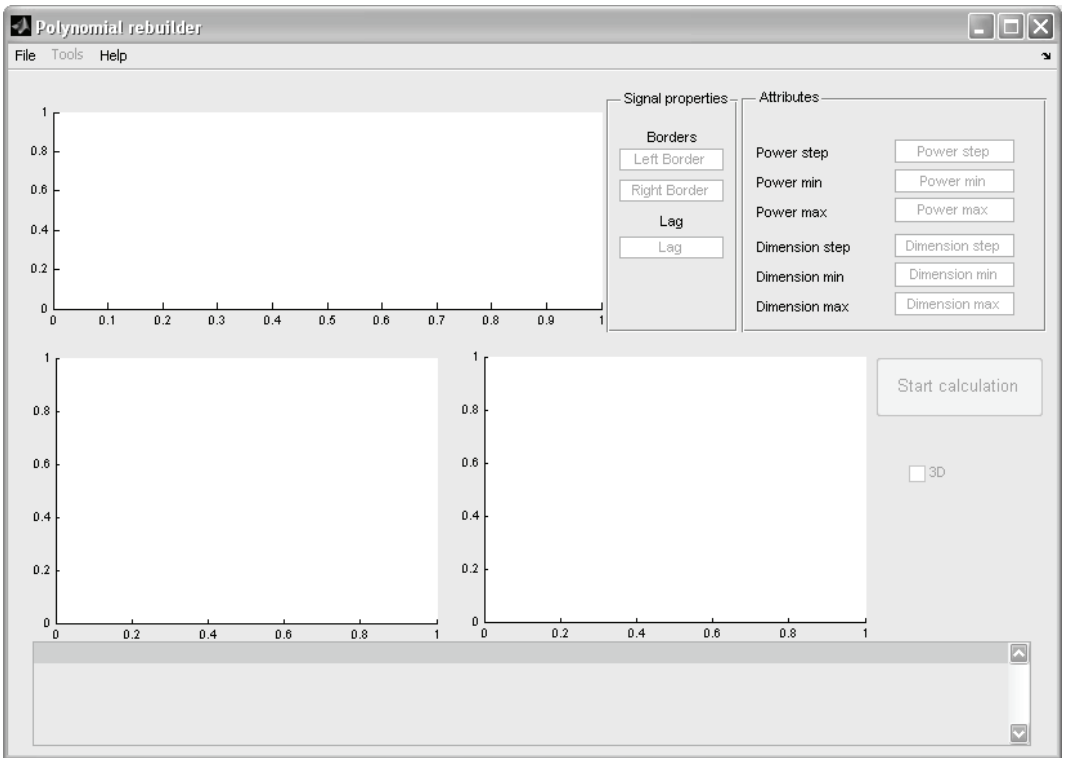

Рис. 2.12. Модуль восстановления уравнения полинома

Панель меню состоит из следующих разделов: File, Tools, Help.

File:

- New – позволяет загрузить примеры сигналов После выбора вида сигнала потребуется также указать количество точек.
- Load – в качестве входных данных могут быть дампы трафика, построенные программами Tcpdump или Wireshark; результат работы команды ping, а также любые числовые ряды, записанные в текстовом формате или в виде структуры. Необходимо указать вид вводимых данных, что и делается в этом меню.
- TCPDump – для дампов трафика возможно 4 вида представления: пакетная/байтовая агрегация/интерполяция.

Агрегация – это подсчёт количества пакетов или байт, пришедших за определённый интервал времени. Весь интервал наблюдения делится на равные интервалы агрегации.

При интерполяции рассматриваются интервалы времени между моментами приходов пакетов. Затем на каждый интервал делится единица в случае

пакетной интерполяции, или размер пакета в случае байтовой интерполяции. То есть получается средняя на этом интервале скорость передачи. Полученное значение откладывается по оси ординат и соответствует середине текущего интервала. Таким образом, получается ряд точек, ордината которых соответствует отношению размера пакета к интервалу, а абсцисса – середина интервала. Это исходный ряд для интерполяции.

- Save – сохранение сигналов трёх видов: исходный сигнал, очищенный от шума и сам удалённый шум.
- Exit – завершение работы подпрограммы.

Tools:

- Noise supression – шумоподавление.
- Norma – норма, используется при расчётах корреляционных интегралов.
- Lag – сдвиг во времени, задержка в соответствующем методе восстановления недостающих переменных, используется при расчётах корреляционных интегралов.

Help:

- Help – справка.
- About program – о программе.

Имеется три поля для вывода графиков:

1. Исследуемый сигнал, для которого производятся все расчеты.

2. Графики корреляционных интегралов, здесь можно выбирать границы линейных участков рассчитанных интегралов, и построит график размерности, где можно увидеть уровень насыщения и тем самым определить корреляционную размерность.

3. Фазовая траектория.

Поля:

- Signal properties:

Borders – границы обработки сигнала.

R – сброс.

Lag – сдвиг во времени.

T aggreg – время агрегации/интерполяции в зависимости от настроек, упомянутых ранее.

- Attributes

Log(E) step/min/max – шаг и пределы изменения эпсилон для корреляционного интеграла в логарифмическом масштабе.

Dimention step/min/max – шаг и пределы изменения размерности при построении корреляционных интегралов.

Start calculation – запуск вычисления корреляционных интегралов.

Cluster – позволяет распараллелить задачу вычисления.

High/Low border – границы графика на поле 2, в пределе которых участок интеграла считается линейным.

Show borders – показывает выбранные границы и отображает аппроксимирующие отрезки.

Calculation Dm – вычисление коррелирующей размерности от размерности пространства вложения.

Auto – вычисление корреляционной размерности с помощью автоматического поиска линейных участков корреляционных интегралов.

Прочие флаги и кнопки:

- Denoise – шумоподавление.
- View noise – отображение шума.
- R_N – сброс шумоподавления.
- 3D – трехмерное отображение фазового пространства.

Методика выполнения задания:

1. Загрузить исходных данных.

В случае использования tcp-дампа, выбрать его представление: пакетная/байтовая агрегация/интерполяция, задавая требуемый временной интервал. После вывода графика происходит активация всех управляющих флагов, а в каждое поле записывается значение по умолчанию (рис. 2.13).

При необходимости произвести шумоподавление (встроенная функция Matlab, по умолчанию уровень шума равен трем) сигнала сначала устанавливают его параметры.

2. Установите границы сигнала.

3. Установите сдвиг АКФ во времени, его подбирают по виду фазовой траектории или рассчитывают.

4. Подберите значения эпсилон.

Для этого берут достаточно большой интервал и большой шаг эпсилон, а интервал размерностей берут небольшой. И запускают вычисления. Затем корректируют значения эпсилон так, чтобы было удобно исследовать интегралы.

5. Подобрав подходящий интервал эпсилон, устанавливают его шаг, а также пределы и шаг размерности. И запустите вычисления.

6. Затем укажите горизонтальные границы (рис. 2.14), между которыми графики интегралов можно считать линейными (кнопка Show borders) и вычисляют размерность (кнопка Calculating Dm). Или выполняют автоматические расчёты размерности (флаг Auto и кнопка Calculating Dm).

После вычисления размерности вместо корреляционных интегралов отображается график размерности, и внешний вид подпрограммы будет как на рис. 2.15.

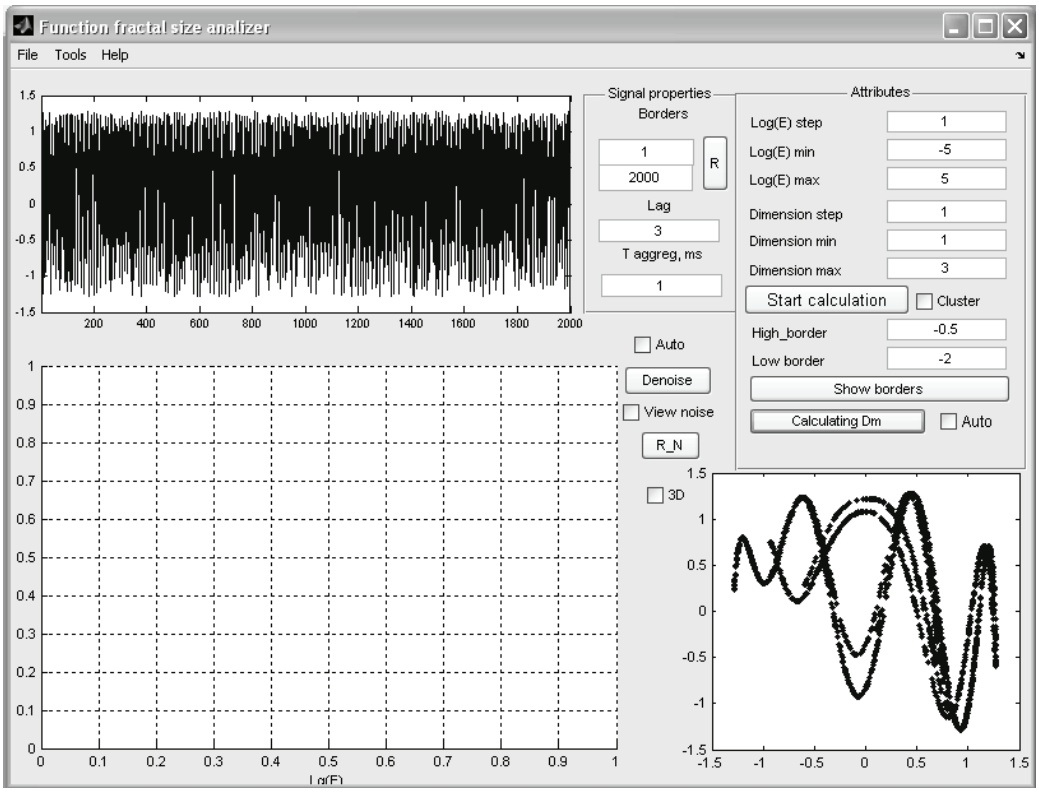

Рис. 2.13. Вид подпрограммы после загрузки данных

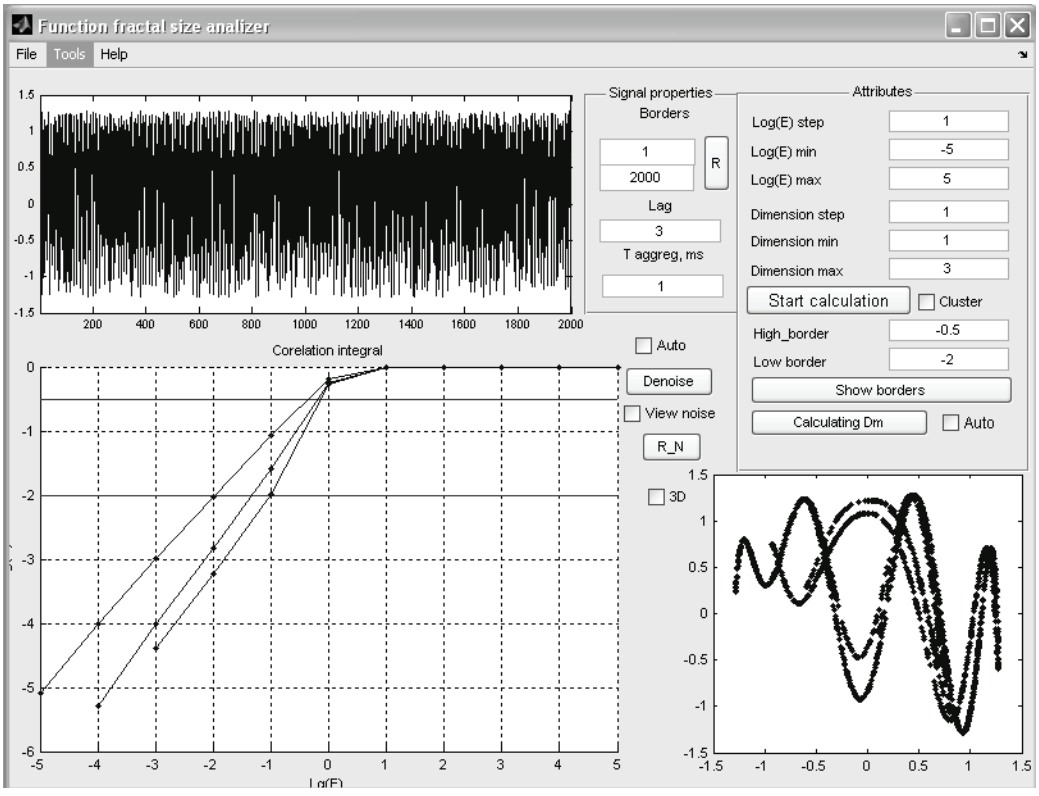

Рис. 2.14. Указание границ на графике интегралов

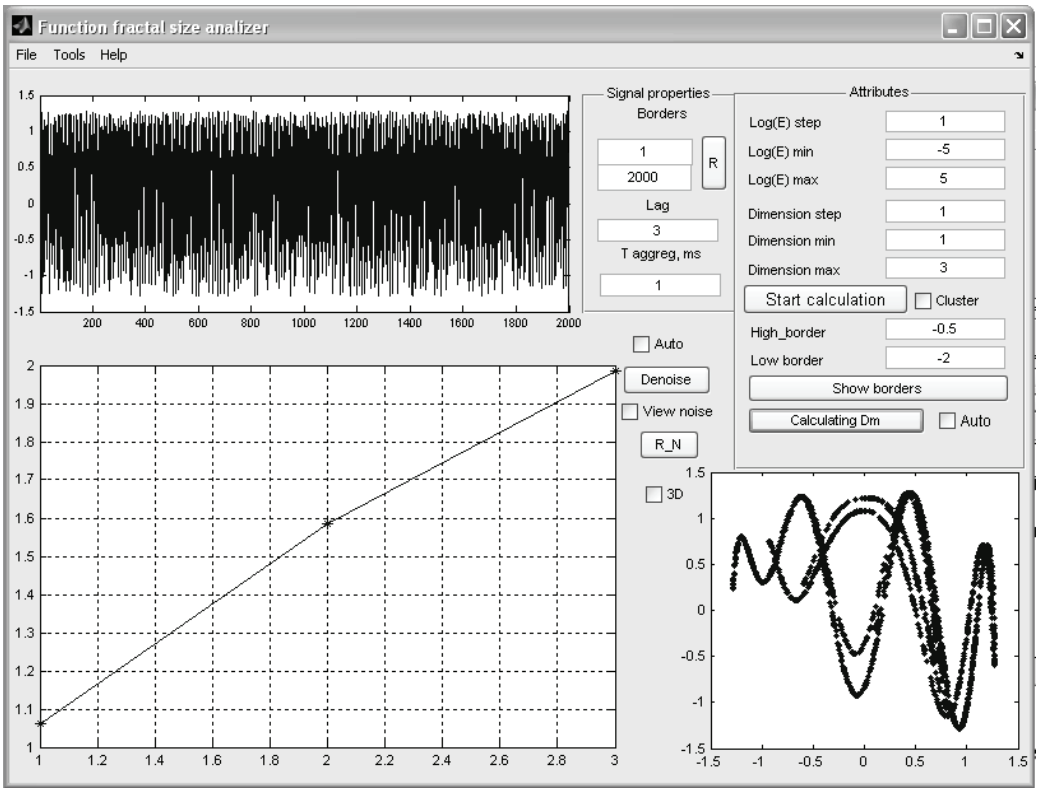

Рис. 2.15. Вид подпрограммы после вычисления размерности

Все выходные данные можно сохранить и использовать далее.

Подпрограмма восстановления уравнения полинома содержит в себе два метода: разностный и дифференциальный.

Панель меню состоит из аналогичных разделов: File, Tools, Help. Разделы File и Help имеют такую же структуру, что и в предыдущей подпрограмме.

Tools:

- Connect attractor dots – соединение точек.
- PowerDiminsion choose menu – выбор коэффициентов.

Поля для вывода графиков:

1. Исследуемый сигнал, для которого производятся все расчеты

2. Аттрактор исходного временного ряда

3. Аттрактор, построенный по найденному уравнению.

Есть еще одно поле – поле с уравнением восстановленного полинома и всех его найденных коэффициентов.

Поля:

- Signal properties

Borders – границы обработки сигнала.

Lag – сдвиг корреляционной функции во времени.

- Attributes

Power step/min/max – шаг и пределы изменения эпсилон для корреляционного интеграла в логарифмическом масштабе.

Dimention step/min/max – шаг и пределы изменения размерности при построении корреляционных интегралов.

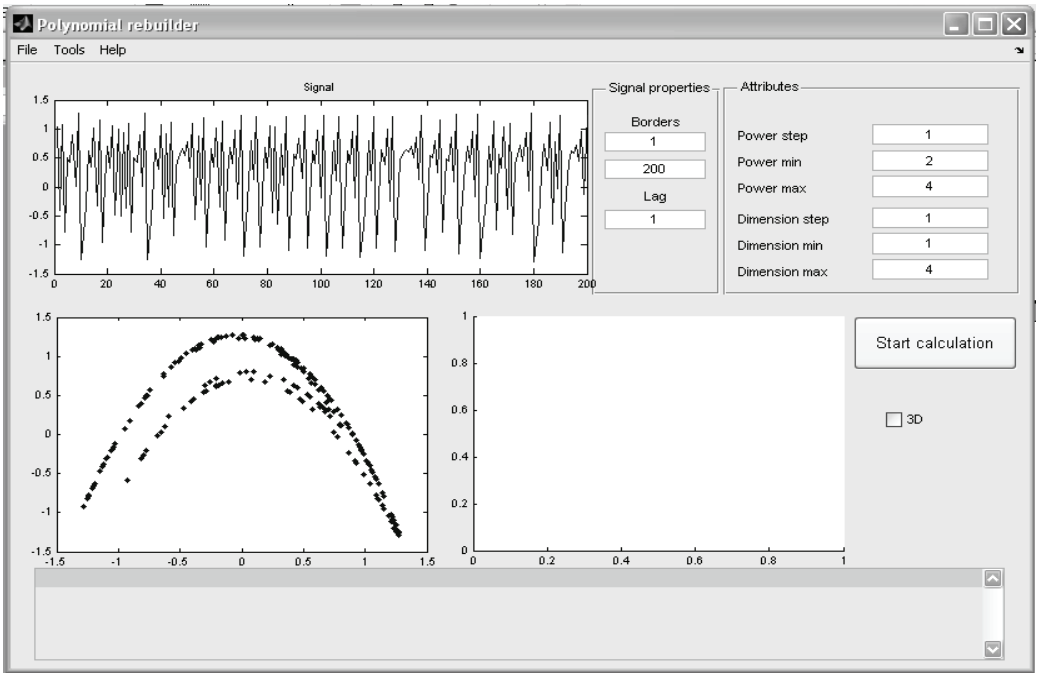

Рис. 2.16. Вид подпрограммы после загрузки данных

Прочие флаги и кнопки:

- Start calculation – запуск вычисления корреляционных интегралов.
- 3D – трехмерный отображение.

7. Загрузите данных.

После загрузки данных, они отображаются в верхнем левом углу рабочей области. Также становятся доступными основные элементы управления программой. Каждое поле имеет свое значение по умолчанию (рис. 2.16).

Если используется разностный метод, то после нажатия кнопки запуска пользователю будет предложено выбрать класс функций, которыми будет происходить восстановление уравнения – восстановление полиномами или дробно-рациональными функциями (рис. 2.17).

8. Выберите класс функции для восстановления уравнения (для разностного метода)

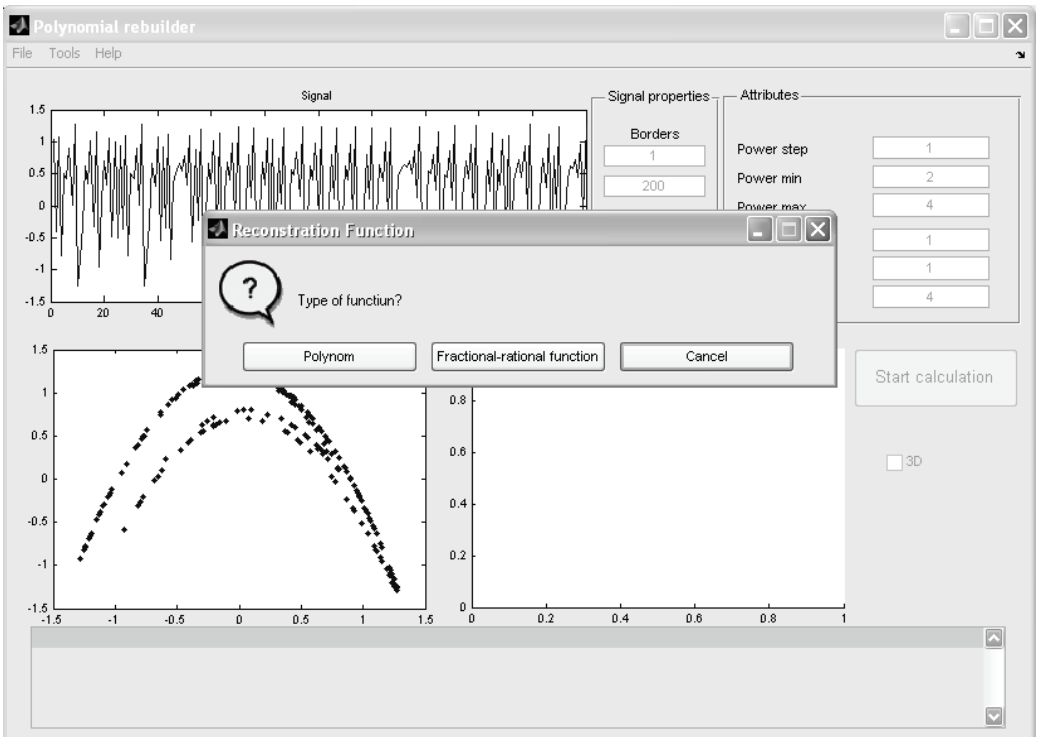

Рис. 2.17. Выбор класса функции для восстановления уравнения

После выбора начинается непосредственный процесс восстановления уравнения. Весь процесс вычислений разбит на два этапа: первый – это непосредственный поиск значений коэффициентов системы дифференциальных уравнений, а второй – реконструкция сигнала при помощи найденных коэффициентов и сравнение загруженного и восстановленного сигналов. По

завершению первого этапа появляется окно с результатами расчетов, позволяющее ему выбрать наиболее подходящий вариант (рис. 2.18).

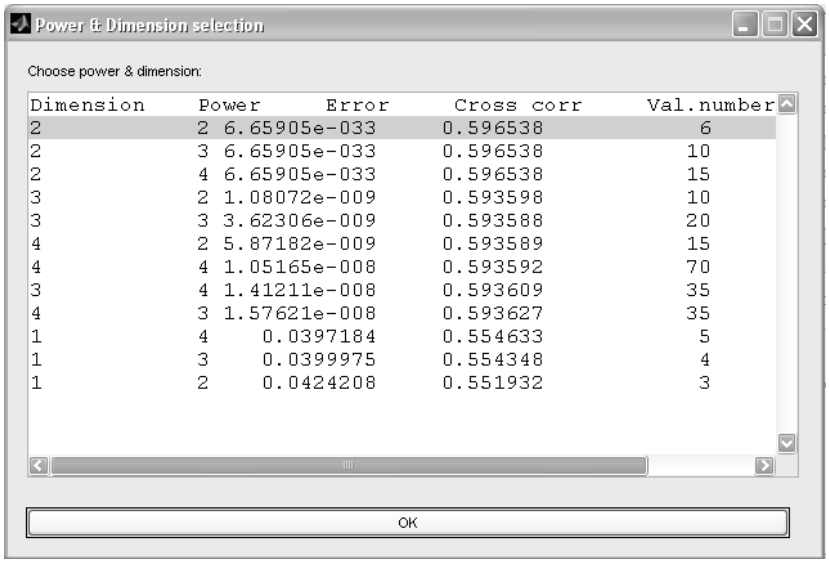

Рис. 2.18. Внешний вид окна выбора размерности и степени полинома

После выбора одного из предложенных вариантов восстановления уравнения, программа на основании выбранных данных и начальных условий пытается реконструировать входящий сигнал, а также произвести его сравнение двух сигналов. В результате чего, на экране появится два окна: окно сравнения характеристик исходного и полученного сигналов (рис. 2.19) и окно вывода результата и (рис. 2.20).

В окно сравнения характеристик три поля для вывода графиков: 1) графики исходного сигнала (кривая № 1.1) и полученного (кривая № 1.2), 2) графики функции автокорреляции исходного сигнала (кривая № 2.1) и взаимной корреляции сигналов (кривая № 2.2), 3) спектр исходного (кривая № 3.1) и полученного (кривая № 3.2) сигналов.

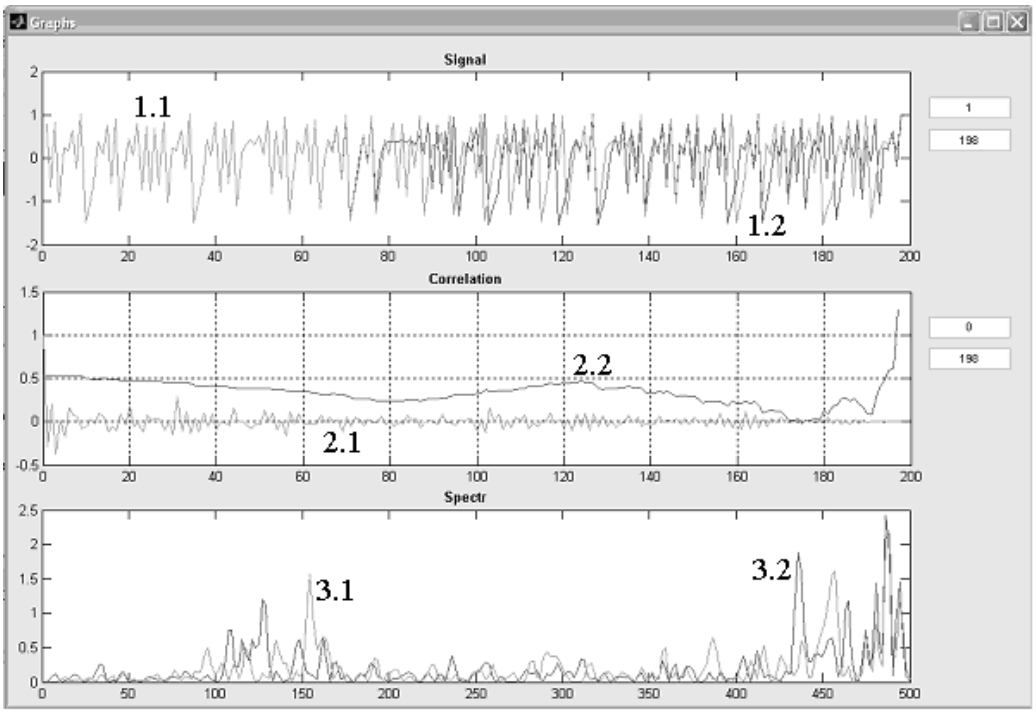

Рис. 2.19.Внешний вид окна сравнения сходного и восстановленного сигналов

В окне вывода результата в левом поле выводиться график аттрактора, построенного на основе входных данных, а в правом – на основе конструированных данных. Флаг 3D позволяет увидеть отображение аттракторов в трехмерном пространстве.

После окончания вычислений данные можно сохранить в текстовом виде в файле. А также можно загрузить новые входные данные или продолжить работу с существующими.

При использовании дифференциального метода порядок работы тот же. Отличается тем, что выводит только восстановленное уравнение (рис.2.31).

9. Оцените длину прогноза.

Варианты заданий

Оценить длину прогноза

- для разных дампов,
- для частей разной длины одного дампа,

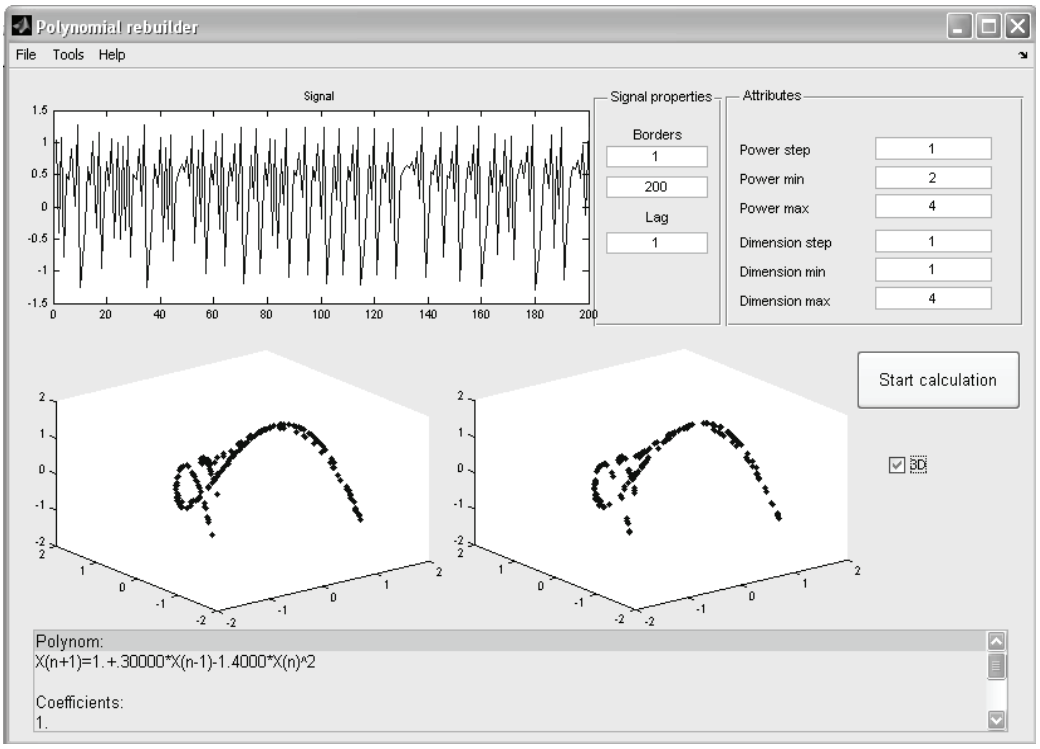

Рис. 2.20.Окно вывода результатов

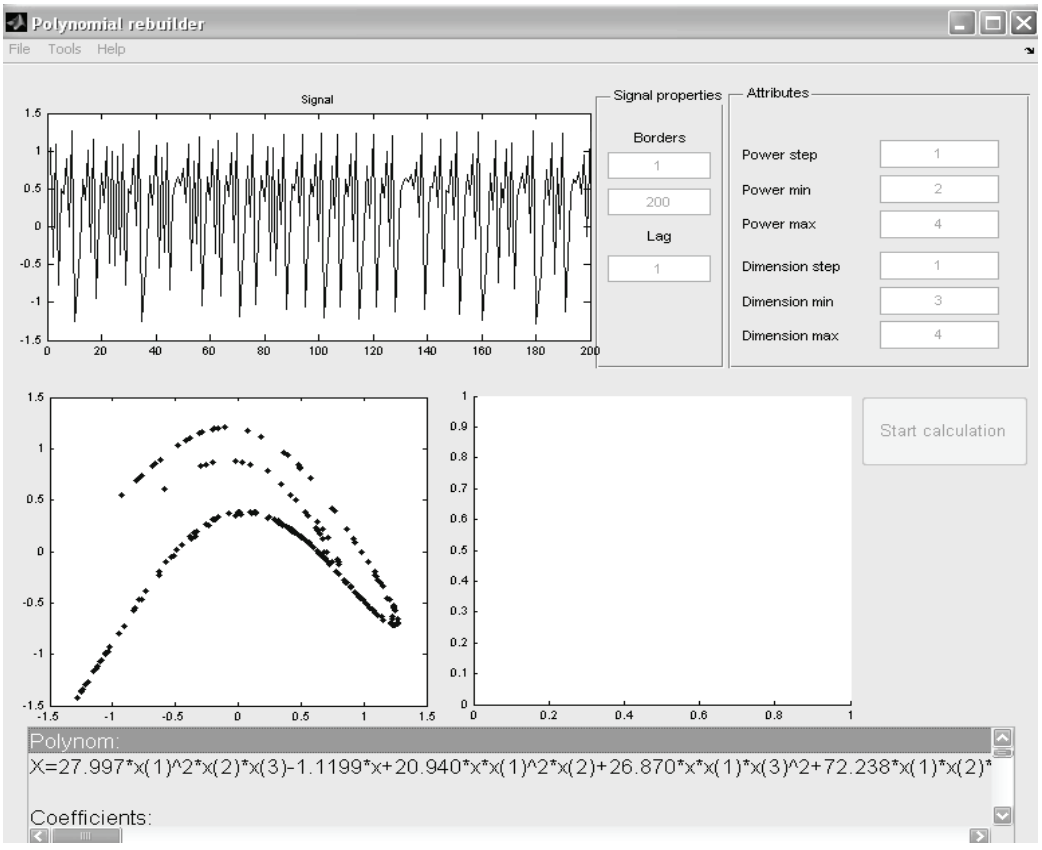

Рис. 2.21. Внешний вид подпрограммы

- для одинаковых кусков одного дампа и соответствие с характеристиками, полученными в первоначальном варианте,
- для одного дампа в зависимости от разных значений сдвига во времени,
- для одного дампа в зависимости от разных значений коэффициентов,
- для одного дампа в зависимости от разных значений от применения шумоподавления.

В отчете привести:

1. Название дампа, тип трафика (real-time, nonreal-time).
2. Вид фазовой траектории, сдвиг АКФ во времени, эпсилон (шаг, интервал), пределы и шаг размерности, горизонтальные границы (пункт 6).
3. Вид окна подпрограммы после вычисления размерности.
4. Метод восстановление уравнения полинома и класс функции для восстановления уравнения (для разностного метода).
5. Вид окна сравнения исходного и восстановленных сигналов.
6. Полином и коэффициенты.
7. Оценить длину прогноза.
8. Выводы.

2.5. Определение типа основного протокола транспортного уровня в типовой операционной системе.

Используя Wireshark и знания о различных механизмах TCP, покажите действие механизмов на примере захваченного трафика реальной операционной системы. На основании найденных результатов работы различных механизмов сделайте вывод о том, какая версия TCP используется в данной операционной системе. Обоснуйте и докажите свое предположение, демонстрируя наличие всех методов управления окном перегрузки, предписанных стандартом для данной версии и отсутствие других. Работу можно выполнять на любом доступном персональном компьютере.

СПИСОК ЛИТЕРАТУРЫ

1. Олифер Н.А., Олифер В.Г., Компьютерные сети. Принципы, технологии, протоколы. СПб: Питер, 2006 – 672 с.
2. Таненбаум Э. Компьютерные сети. – Изд. 4-е, СПб: Питер, 2010 – 992 с.
3. Официальный сайт сетевого симулятора Network Simulator [Электронный ресурс]. – Режим доступа: http://www.isi.edu/nsnam/ns, свободный.
4. Заборовский В.С., Сети ЭВМ и телекоммуникации. Моделирование и анализ сетей связи с коммутацией пакетов. Network Simulator (Сетевой симулятор ns2): Учебное пособие. СПб: Изд-во СПбГТУ, 2001 – 108 с.
5. Протокол управления передачей (Transmission Control Protocol). Перевод RFC 793 [Электронный ресурс]. – Режим доступа: http://www.protocols.ru/files/RFC/rfc-793.pdf, свободный.
6. Заборовский В.С., Мулюха В.А., Новопашенный А.Г. Сети ЭВМ и телекоммуникации. Анализ трафика в сетях коммутации пакетов. Учебное пособие. СПб: Изд-во СПбГПУ, 2010 – 90 с.
7. Кузнецов С.П. Динамический хаос. М.: Физматлит, 2001. 296 с.
8. Макаренко Н.Г. Эмбедология и нейропрогноз // Труды V Всеросс. научн.-тех. конф. «Нейроинформатика-2003». М., 2003. Ч. 1. С. 86-148.
9. Малинецкий Г.Г., Потапов А.Б. Современные проблемы нелинейной динамики. М.: Эдиториал УРСС, 2000. – 336 с.
10. Безручко Б.П., Смирнов Д.А. Математическое моделирование и хаотические временные ряды. – Саратов: Издательство ГосУНЦ «Колледж», 2005. – 320 с.
11. Крылов В.В., Самохвалова С.С. Теория телетрафика и ее приложения. – СПб.: БХВ-Петербург, 2005. – 288 с
12. Анищенко В. С., Знакомство с нелинейной динамикой. Лекции Соросовского профессора. – Изд-во ГосУНЦ «Колледж», Саратов, 2007. – 170 с.

В.С. Заборовский, В.А. Мулюха, Ю.Е. Подгурский

СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ МОДЕЛИРОВАНИЕ И АНАЛИЗ КОМПЬЮТЕРНЫХ СЕТЕЙ: ТЕЛЕМАТИЧЕСКИЙ ПОДХОД

Учебное пособие

Отпечатано с готового оригинал-макета, предоставленного авторами, типографии Издательства СПбГПУ.

195251, Санкт-Петербург, Политехническая ул., 29.