Ярославский государственный университет имени П. Г. Демидова

На правах рукописи

Алексеев Игорь Вадимович

Адаптивная схема управления потоком для транспортного протокола в сетях с коммутацией пакетов

05.13.17 - теоретические основы информатики

ДИССЕРТАЦИЯ

на соискание ученой степени

кандидата физико-математических наук

Научный руководитель: кандидат физико-математических наук, профессор Соколов В. А.

Научный консультант: доктор физико-математических наук, профессор Тимофеев Е. А.

Ярославль - 2000 г.

Реферат

В работе предложен новый алгоритм управления скоростью передачи данных для транспортного протокола сети с коммутацией пакетов. Разработанный нами новый протокол получил название ARTCP - Adaptive Rate TCP. Нами показано, что трафик протокола ARTCP обладает свойством самоподобия, разработана и реализована программная модель транспортного протокола, проведен ряд модельных экспериментов показывающих преимущества ARTCP по сравнению с TCP.

Разработанный алгоритм управления потоком транспортного протокола характеризуется рядом существенных отличий от традиционных методов управления потоком протокола TCP, а именно:

- Скорость отправки ARTCP сегментов в сеть управляется не размером окна передачи (как в TCP) а индивидуальной задержкой каждого сегмента. Изменение скорости отправки потока выражается в изменении его скважности (межсегментного временного интервала).

- Индикатором текущего состояния сети и соответственно, наступления перегрузки служит не потеря пакета, а изменение скважности потока сегментов, измеряемое получателем, а также изменение времени транзита сегментов, измеряемое отправителем.

- Функционирование ARTCP не зависит от потока подтверждений для синхронизации отправки новых сегментов в сеть.

Таким образом, в ARTCP устранена логическая зависимость алгоритмов коррекции ошибок передачи и управления потоком. Это дает существенные преимущества протоколу ARTCP, особенно в приложениях, где потеря пакета не является индикатором перегрузки, например, в беспроводных сетях. Кроме того, при работе в традиционных сетях алгоритм ARTCP оказывается более эффективным, так как он минимизирует среднюю длину очередей в маршрутизаторах и не доводит сеть до состояния перегрузки в процессе определения максимальной доступной соединению доли пропускной способности, что особенно важно для сосуществования потоков данных и мультимедиа. Также отсутствие необходимости в синхронизации по подтверждениям дает возможность эффективно применять ARTCP для систем с асимметричными каналами.

В настоящей работе показано, что трафик, генерируемый протоколом TCP, обладает свойством самоподобия, поэтому единственным способом его исследования является модельный эксперимент, поскольку развитого аналитического аппарата, применимого к самоподобному процессу, на данный момент не существует.

Для проведения модельного эксперимента нами разработана и реализована объектно-ориентированная программная модель сетевой архитектуры, которая моделирует основные свойства сети, определяющие функционирование транспортного протокола, а именно: задержку, мультиплексирование, потери и ошибки передачи. Наша программная модель позволяет конструировать любую топологию сетевых соединений.

Цель модельного эксперимента, осуществленного в рамках данной работы - определить значения важнейших характеристик транспортного протокола: вероятность потери сегментов, среднюю длину очереди, эффективность использования пропускной способности канала, показатель равноправия разделения ресурсов и, в сравнении с имеющимися данными по стандартному TCP, показать преимущества нового протокола ARTCP. По данным модельного эксперимента делаем вывод, что превосходство ARTCP по отношению к TCP, наиболее очевидно для беспроводных систем, однако и в обычных проводных сетях применение ARTCP имеет явные преимущества: меньшая по сравнению с TCP средняя длина очереди, полное отсутствие потерь сегментов.

Содержание

Введение

1.1. Предмет исследования

Одним из важнейших направлений научно-технического прогресса в настоящее время являются коммуникационные системы, представляющие собой сети передачи информации. Координацию процессов передачи информации в распределенной системе, которой является сеть, осуществляют коммуникационные протоколы.

Принято разделять коммуникационные протоколы по степени общности задач, решаемых ими, на несколько уровней, упорядоченный набор которых образует сетевую архитектуру. Самой распространенной и универсальной сетевой архитектурой является архитектура TCP/IP [43, 1]. В рамках TCP/IP все системы в сети делятся на конечные системы, между которыми происходит информационный обмен, и промежуточные системы, не являющиеся конечными или исходными точками обмена. Конечные системы называются узлами сети, а промежуточные - маршрутизаторами.

Двусторонний поток информации между парой смежных систем в сети обеспечивается каналом, связывающим две системы. Каналы характеризуются скоростью информационного потока (пропускной способностью), задержкой передачи и вероятностью битовых ошибок. В каждой точке подключения маршрутизатора к каналу имеется буфер, в котором организуется очередь данных ожидающих отправки по этому каналу. Буферное пространство и пропускная способность (ПС) представляют собой разделяемые ресурсы сети. Если скорость прибытия информации в маршрутизатор превышает максимально возможную скорость ее отправки, то происходит перегрузка сети, выражающаяся в переполнении буферов и потерях информации.

Протокол транспортного уровня занимает важнейшее положение в любой сетевой архитектуре, в том числе и в TCP/IP, поскольку он обеспечивает надежную и эффективную передачу информации непосредственно между конечными системами сети. Для этого транспортный протокол задает согласованный набор правил поведения для участников информационного обмена. Эти правила регулируют совместный доступ узлов к разделяемым ресурсам сети, поэтому эффективность транспортного протокола определяет эффективность работы всей сети в целом. Программа, реализующая алгоритм протокола, называется объектом протокола.

Транспортным протоколом в архитектуре TCP/IP является TCP (Transmission Control Protocol) [4, 5, 6], который обеспечивает надежную двустороннюю связь с контролем

скорости передачи. Источник TCP потока получает информацию от пользователя в виде последовательности битов, формирует из нее блоки конечной длины, называемые сегментами и отправляет их к TCP получателю. Получатель, принимая сегменты, формирует из них исходную последовательность и передает ее своему пользователю.

Для осуществления обмена TCP устанавливает логическое соединение между парой узлов сети, на каждом из которых исполняется алгоритм TCP. Поток сегментов по TCP соединению может проходить через упорядоченную последовательность маршрутизаторов и каналов. Пропускная способность соединения в целом ограничена минимальной из ПС каналов, через которые проходит соединение. Алгоритм управления потоком, являющийся частью TCP, стремится отправлять данные со скоростью, не превышающей меньшее из ПС соединения и скорости потребления информации получателем.

Набор соединений транспортного протокола, разделяющих общий канал, представляет собой сложную самоорганизующуюся систему в смысле Г. Хакена [101]. Поведение каждого из объектов протокола в этой системе определяется алгоритмом протокола, однако, поведение всей системы, как целого, вообще говоря, не описывается совокупностью действий ее компонентов. Каждый объект протокола стремится максимально эффективно адаптироваться к доступным ресурсам сети в условиях кооперации с другими объектами этого протокола.

На сегодняшний момент известен ряд существенных недостатков алгоритма управления потоком протокола TCP:

1. Для оценки доступной ПС алгоритм управления потоком TCP постоянно увеличивает скорость отправки сегментов, искусственно вызывая перегрузку сети. Это приводит к частым потерям пакетов и, при устойчивом переполнении буферов, к увеличению задержек сегментов в сети.

2. TCP интерпретирует потерю сегмента как признак перегрузки сети и реагирует на любую потерю данных снижением скорости передачи, что ведет к существенным ограничениям эффективности TCP в сетях, где вероятность потери сегментов из-за возникновения ошибок отлична от нуля. Это относится, в частности, ко всем беспроводным сетям.

3. Локальные неравномерности в отправке сегментов TCP приводят к повышению вероятности потери сегментов при максимальном заполнении буферов.

Устранение приведенных выше недостатков TCP является темой большого числа исследований. В работах на эту тему предлагаются разные варианты усовершенствования транспортного протокола. Большинство протоколов, предлагаемых для использования в сетях с ненулевой вероятностью битовых ошибок, не являются совместимыми с TCP и требуют введения дополнительных элементов в структуру сети, усложняя ее и нарушая основной принцип транспортного протокола, состоящий в том, что на транспортном уровне соединение устанавливается между непосредственным источником и получателем информации.

Таким образом, важнейшей задачей является разработка нового транспортного протокола в рамках архитектуры TCP/IP, который был бы более эффективен, чем TCP. Новый протокол должен быть универсальным в смысле возможности использования его как в проводных, так и беспроводных сетях, что особенно важно в свете дальнейшего развития сетевых технологий и расширения областей их применения.

В диссертации разработан новый транспортный протокол ARTCP. В среде языка C++ создан класс, полностью описывающий протокол ARTCP, который может стать основой реализации протокола. Разработана универсальная объектно-ориентированная имитационная модель для конструирования сетей с топологией любой сложности. Проведенные эксперименты работы протокола ARTCP для ряда сценариев показали, что он почти всегда превосходит стандартный протокол TCP.

1.2. Научная новизна работы

Основные научные результаты диссертации состоят в следующем:

Разработан протокол ARTCP, использующий темпоральные показатели потока в качестве входного параметра для алгоритма управления потоком и сочетающий оконный механизм контроля скорости с диспетчеризацией каждого сегмента.

Построена имитационная программная модель (ИПМ), позволяющая моделировать все компоненты сети, влияющие на функционирование транспортного протокола. ИПМ является универсальным инструментом для исследования взаимодействий в сетях и позволяет строить топологические схемы большой сложности.

По данным модельного эксперимента в ИПМ определены важнейшие характеристики ARTCP, а также показано наличие свойства самоподобия у трафика ARTCP. Результаты эксперимента позволяют утверждать, что ARTCP превосходит TCP по основным критериям.

1.3. Практическая ценность результатов

Практическая ценность предлагаемой схемы управления потоком очень высока. Во-первых, протокол ARTCP не доводит сеть до состояния перегрузки для выявления доступной максимальной пропускной способности, поэтому потери пакетов в стабильном состоянии работы сети вообще не происходят. Таким образом, повышается эффективность использования сетевой инфраструктуры, что дает прямой экономический эффект.

Во-вторых, протокол ARTCP не интерпретирует потерю пакета как признак перегрузки сети, что позволяет эффективно применять его в каналах с ненулевой вероятностью ошибок. В настоящее время развитие инфраструктурной части сетей сориентировано в немалой степени именно на беспроводные системы, поэтому ARTCP может найти в таких сетях широкое применение.

В-третьих, средняя длина очередей в маршрутизаторах сети при использовании ARTCP минимальна, поскольку ARTCP не только адаптирует скорость отправки пакетов в сеть к максимальной скорости обслуживания потока, но и обладает механизмом компенсации перегрузки. Таким образом, среднее время транзитной задержки пакетов в сети снижается.

В-четвертых, алгоритм ARTCP не противоречит созданию совместимой с TCP реализации. Поэтому внедрение ARTCP может происходить постепенно, в первую очередь на тех узлах, где применение нового протокола наиболее выгодно.

1.4. Апробация работы

По результатам, полученным в ходе работы, были сделаны доклады на международном семинаре IEEE «Интернет: технологии и сервисы», а также на семинарах ЯрГУ: "Моделирование и анализ информационных систем", "Нейронные сети". Новый протокол вызвал большой интерес и одобрение со стороны экспертов в области телекоммуникаций.

1.5. Содержание работы

Для создания алгоритма нового протокола и разработки его имитационной модели необходимо рассмотреть все компоненты транспортного протокола. В **части 1.7 введения (коммуникационные транспортные протоколы)**, даны важнейшие определения, принципы и алгоритмы, которые используются далее по тексту. В **части 1.8 (управление потоками в коммуникационных системах)** рассмотрены работы в области управления скоростью передачи в распределенных сетях построенных на принципе коммутации пакетов. Самоподобие сетевого трафика и важнейшие работы в этой области описаны в **части 1.9**

(свойство самоподобия сетевого трафика) введения. На основе типичных задач и путей решения проблемы управления потоками можно осуществить постановку задачи - создание протокола, свободного от недостатков TCP, это сделано в **главе 1**. В **главе 2** дано описание алгоритма предлагаемого протокола ARTCP, и его функциональных компонентов. В **главе 3** приведено описание разработанной имитационной модели сети и аспекты ее объектной реализации. В **главе 4** приведены результаты модельных экспериментов, дано сравнение свойств ARTCP и TCP, а также показано свойство самоподобия ARTCP трафика. Работа завершается **выводами**.

1.6. Благодарности

Хочу выразить искреннюю благодарность своему научному руководителю проф. В. А. Соколову за ценные советы и помощь с работой. Особенно хочется отметить помощь и поддержку, оказанную научным консультантом проф. Тимофеевым Е. А. Отдельное спасибо моим коллегам по Центру Интернет ЯрГУ, в частности, проректору по информатизации Русакову А. И. за помощь и поддержку, а также руководителю проекта "Региональный кластер научных вычислений"[1] (грант РФФИ № 98-07-90171) Захаровой М. Н. за предоставление возможности осуществлять разработку программной модели и модельный эксперимент, а также всем, кто оказывал мне помощь и поддержку. Написание данной диссертации является частью работ выполняемых по проекту "Развитие высокоскоростного сегмента Ярославской региональной опорной сети на основе ATM технологий" (грант РФФИ № 98-07-90307).

1.7. Коммуникационные транспортные протоколы

1.7.1. Современные коммуникационные сети

Важность сетей передачи данных и мультимедиа информации невозможно переоценить сегодня, в начале информационной эпохи развития человечества. Системы сбора, обработки и распределения информации являются важнейшей точкой приложения научного знания. Развитие технологии устранило различия между обработкой и передачей информации, а наиболее актуальной задачей в коммуникации стала спецификация и верификация коммуникационных протоколов, а также повышение их эффективности.

Современное определение сети передачи данных такое: сеть это набор автономных вычислительных устройств физически и логически связанных между собой. Иными словами

---

[1] Региональный кластер научных вычислений "Cyclone" состоит из трех Alpha-станций под управлением ОС Digital UNIX. Подробная информация по адресу http://cluster.yars.free.net/cluster

набор равноправных узлов имеющих возможность обмениваться информацией. Такое определение относится к так называемой сети передачи данных, однако, не ограничивается только компьютерными данными и может быть расширено для любой сети, построенной по принципу коммутации пакетов, в том числе и универсальной сети с интеграцией сервисов.

Сети передачи данных и мультимедиа информации в современном обществе применяются повсеместно – в производстве и в сфере управления, в исследовательской среде и в быту. Экономическая выгода от повышения эффективности коммуникационных протоколов может оказаться очень существенной, особенно с учетом дальнейшего роста и развития коммуникационных систем.

Любая коммуникационная система состоит из двух основных компонентов: аппаратной и логической частей.

1.7.2. Аппаратная инфраструктура сетей

Аппаратная часть это физическая инфраструктура, посредством которой осуществляется распространение физических сигналов кодирующих информационные сообщения. Физическая инфраструктура может либо осуществлять информационный обмен между всеми участниками сети одновременно (широковещательная сеть), либо только между двумя узлами (сеть типа точка-точка). В зависимости от масштаба и топологии сетевой инфраструктуры производят ее дальнейшую классификацию: локальные вычислительные сети (ЛВС), территориальные сети (WAN), сотовые сети и т.д.

Многие свойства семейства протоколов для коммуникационной системы находятся в прямой зависимости от ее физической среды и топологии, поэтому, параметры физической инфраструктуры сети тщательно учитываются при разработке ее протоколов.

1.7.3. Системы протоколов

Важнейшим компонентом сети являются коммуникационные протоколы, составляющие ее логический компонент. В качестве неформального определения протокола можно привести следующее: протокол представляет собой соглашение по процедуре обмена информацией в распределенной системе.

Задачей любого протокола является управление совместным использованием ресурсов. Протоколы сходны с языками. Определение протокола дается следующим образом - через определение его функциональных компонентов [3]:

- Протокол определяет точный формат допустимых сообщений (синтаксис);
- Протокол определяет процедурные правила для обмена информацией (грамматика);

- Протокол определяет словарь правильных сообщений и их значения (семантика);

Разработка, описание и верификация протоколов составляют наиболее сложную задачу в решении проблемы создания коммуникационных систем.

1.7.3.1. Методы формального описания

Существуют несколько методов для формального описания протоколов. Это языки SDL (Specification and description language), Lotos и Estelle. Наиболее часто применяется язык SDL, который является стандартом ITU[2]. Существуют две формы SDL – графическая и программная [55]. Естественно, что из всех компонентов протокола именно процедурные правила могут быть даны с помощью того или иного языка формального описания.

Наиболее сложной проблемой разработки или повышения эффективности протокола является, таким образом, создание большого набора процедурных правил, который обладает следующими свойствами:

- Он должен быть минимальным, т.е. не содержать недостижимых элементов или участков кода
- Он должен быть логически связным
- Он должен удовлетворять условию полноты
- Этот набор должен быть легко реализуем (аппаратным или программным образом)

1.7.3.3. Пять элементов протокола

В дополнение к определению, данному выше, разработка протокола предполагает детальную спецификацию сервиса предоставляемого протоколом пользователю, которым может быть прикладная программа или протокол более высокого уровня, а также знание характеристик среды, в которой протокол будет выполняться.

Таким образом, выделяют пять компонентов протокола, каждый из которых должен быть задан тем или иным методом формального описания:

1. Сервис, предоставляемый пользователю
2. Предполагаемая характеристика среды исполнения протокола
3. Словарь допустимых сообщений
4. Кодировка сообщений – формат каждого сообщения из словаря
5. Процедурные правила, контролирующие обмен сообщениями

Формализованное описание сервиса и характеристик среды исполнения протокола находят отражение в процедурных правилах. Для реализации своей задачи – предоставления

---

[2] International Telecommunications Union – международная организация разрабатывающая стандарты в области телекоммуникаций

сервиса верхнего уровня протокол должен выполнять ряд задач низкого уровня – синхронизации, распознавания и коррекции ошибок. То, как протокол выполняет эти задачи, зависит от свойств среды его исполнения.

Словарь допустимых сообщений и их кодировка также связаны с предположениями о среде исполнения протокола – среде передачи данных. Вероятность возникновения битовых ошибок и средняя длина поля последовательности битовых ошибок влияют на набор, формат и кодировку сообщений протокола, а также выбор алгоритма распознавания или коррекции ошибок.

1.7.3.4. Определение протокола через "связанные конечные автоматы"

На низком уровне абстракции протокол можно представить в виде конечных автоматов. В своей монографии [3] Хольцман предлагает так называемую модель связанных конечных автоматов для формализованного описания протокола. В этом случае задачи разработки, формальной верификации и проверки на совместимость сводятся к нахождению желаемых/нежелаемых состояний и переходов. Каждый из связанных конечных автоматов может получать символы на вход, осуществлять переход в новое состояние и генерировать символьную информацию на выходе. Отдельные конечные автоматы связываются между собой посредством ограниченных FIFO очередей, через которые выходной сигнал одного автомата поступает на вход другого. Так моделируется асинхронная связь, характерная для большинства коммуникационных систем.

Определение очереди: очередь сообщений (символов) представляется тройкой (S, N, C), где S - ограниченное множество определяющее словарь очереди, N - целое число позиций в очереди, C - упорядоченное множество элементов из S. S и C: множества сообщений. Если модель системы требует наличия нескольких очередей, то их словари должны образовывать непересекающиеся множества. Если M множество всех системных очередей, индекс $1 \le m \le |M|$ нумерует очереди, а $1 \le n \le N$ позиции внутри очереди, C_n^m - n-ный символ в m-той очереди. Системный словарь тогда выражается через словари каждой из очередей и нулевой символ ε: $V = \mathrm{Y}_{m=1}^{|M|} S^m \mathrm{Y}\, \varepsilon$. В [3] дается определение связанных конечных автоматов как набора (Q, q_0, M, T), где Q - конечное непустое множество состояний, q_0 - элемент Q - начальное состояние, M - множество символьных очередей, определенных выше, T - отношение перехода. T имеет два аргумента T(q, a), где q текущее состояние и a - действие (одно из: ввод, вывод, пустое действие). Реализация перехода с действиями типа ввод/вывод зависит от состояния символьных очередей. При их реализации изменяется состояние одной из очередей. Отношение перехода T задает множество (которое может быть пустым) всех

возможных результирующих состояний. Достаточным условием реализации T(q, ε) является то, что текущее состояние есть q.

Расширение модели связанных конечных автоматов множеством внутренних переменных, а набора действий булевыми операциями и операциями присвоения значения дает эквивалентную модель, но существенно уменьшает число состояний. Для такой модели определены алгоритмы минимизации и исполнения. Однако представление реальных сложных протоколов, например TCP, в виде конечных автоматов крайне сложно и неэффективно, поэтому для их разработки и спецификации применяют менее формальные методы.

1.7.3.5. Иерархии протоколов

Принципы разработки и исследования любой сложной системы предполагают расчленение ее на части меньшего объема и сложности, которые можно исследовать по отдельности. Имея в виду принципы взаимодействия частей, можно обобщить знание на всю систему. Системы протоколов в коммуникации имеют сложную иерархическую структуру. Каждый из протоколов в такой системе представляет собой определенный уровень абстракции, при этом нижележащие уровни скрывают низкоуровневые задачи от верхних уровней. Иерархический принцип позволяет выразить логическую структуру протоколов, разделяя задачи нижних и верхних уровней.

Рис. 1. Пример инкапсуляция данных одного протокола в формате сообщений другого. Протокол 2 инкапсулирует сообщения протокола 1.

Таким образом, для снижения сложности разработки коммуникационные системы организуются в виде нескольких уровней (слоев). Количество, содержание и функции слоев различны для различных типов сетей. Целью существования каждого из уровней является предоставление определенных типов сервисов уровням расположенным выше их и защита верхних уровней от деталей реализации этих сервисов.

Уровень N на одном устройстве логически находится в состоянии обмена с уровнем N на другом сетевом устройстве. Набор законов и соглашений, принятых при таком обмене называется протоколом уровня N.

В реальности никакого обмена данными не происходит между слоями на одном уровне. Вместо этого каждый уровень передает данные и контрольную информацию на уровень находящийся прямо под ним, пока не будет достигнут низший уровень, который поместит информацию непосредственно на физический носитель для передачи ее по сети.

Между каждой парой смежных слоев находится интерфейс, определяющий какими операциями и сервисами нижнего уровня может воспользоваться вышележащий уровень. В процессе разработки сети очень важным моментом является четкое и чистое определение интерфейсов между слоями. Для этого каждый уровень должен реализовывать некоторый набор хорошо продуманных функций. Тщательная проработка интерфейсов не только делает возможной замену одного типа внутренней реализации уровня на другой при условии сохранения семантики интерфейса, но и способствует сведению к минимуму количества контрольной информации, пересылаемой между уровнями. В этом случае замена может быть осуществлена прозрачно для смежных уровней.

Набор слоев и соответствующих им протоколов называется архитектурой сети. Упорядоченный набор протоколов определенных в рамках конкретной сетевой архитектуры называется стеком протоколов. Каждый из N уровней одной системы осуществляет обмен с уровнем N конечной системы, используя для этого сервисы уровня N-1, при этом каждый из промежуточных уровней инкапсулирует[3] данные верхнего уровня внутри своего формата сообщений (рис. 1.).

Таким образом, на каждом уровне N процедура, отправляющая или получающая данные от уровню N другой системы, на самом деле отправляет информацию для обработки на уровень N-1 своей системы, снабдив данные своей контрольной информацией.

1.7.3.6. Общие свойства уровней иерархической системы

В функции каждого из уровней входят несколько типов операций, которые, реализуясь по-разному на каждом уровне абстракции, имеют сходные признаки. Так каждому уровню необходим механизм для идентификации отправителей и получателей – определенная система адресации и мультиплексирования потоков. Уровни должны работать с несколькими режимами обмена данными: симплекс, полудуплекс, дуплекс, а также поддерживать набор приоритетов и логических соединений. Очень важной функцией

[3] помещает в поле полезной нагрузки своего формата сообщения

присутствующей на каждом из уровней является обнаружение и коррекция ошибок. Поскольку максимальная длина сообщений поддерживаемых на каждом из уровней различна, то реализация уровней должна предусматривать сборку и разборку длинных сообщений. Каждый из уровней реализует контроль скорости передачи так, чтобы не допустить, во-первых, переполнения сети и потерю данных в ней и, во-вторых, переполнения медленного получателя и потерю данных у получателя. Естественно, что и отправитель и получатель сообщений должен осуществлять промежуточную буферизацию данных. В данной работе предлагается новая схема организации контроля скорости передачи потока данных протокола транспортного уровня.

1.7.3.7. Сервисы и интерфейсы

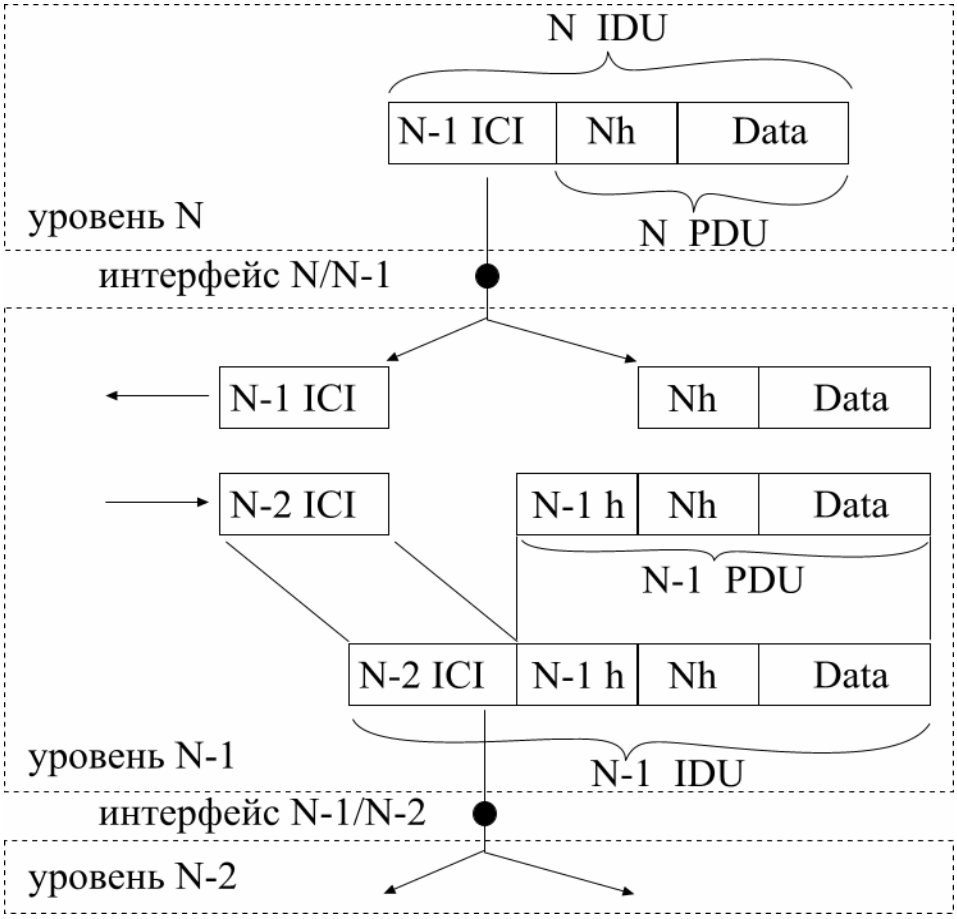

Рис. 2. Преобразование потока информации на интерфейсе NSAP между смежными уровнями сетевой архитектуры.

Предоставление сервисов верхним уровням является главной задачей каждого из уровней. Активные элементы внутри каждого из уровней называются объектами протокола. Объект может представлять собой программный процесс или часть функциональности аппаратуры, например интеллектуальный контроллер ввода/вывода. Объекты уровня N

реализуют сервисы, используемые уровнем N+1. В этом случае уровень N является поставщиком услуг, а уровень N+1 пользователем. Уровень N для выполнения своей задачи по предоставлению набора сервисов уровню N+1 может сам выступать пользователем услуг уровня N-1. Возможно предоставление нескольких типов сервиса, например быстрой и ненадежной связи наряду с медленной и надежной. Сервисами уровня можно воспользоваться через интерфейс, называемый точкой доступа к сервису (ТДС). Каждый из возможных ТДС уровня N это интерфейс, на котором объект уровня N+1 может иметь доступ к сервисам уровня N. Каждая ТДС идентифицируется уникальным адресом. Чтобы два смежных уровня могли обмениваться информацией необходимо наличие набора правил регламентирующих распределение доступа и интерфейс между ними (рис. 2.).

IDU – формат блока данных на соответствующем интерфейсе

ICI – контрольная информация для нижележащего уровня

PDU – сообщение, определенное протоколом соответствующего уровня

IDU – сообщение в формате интерфейса между уровнями

h – заголовок, несущий контрольную информацию для протокола соответствующего уровня

Data – передаваемые данные

Механизм обмена следующий: сущность уровня N снабжает данные заголовком несущим информацию для протокола уровня N. На приемной стороне соединения заголовок Nh будет использован уровнем N для восстановления данных в исходном виде, после чего сами данные будут переданы пользователю уровня N. Данные совместно с заголовком образуют сообщение протокола уровня N (PDU). Кроме того, уровень N передающей стороны снабжает N PDU дополнительной контрольной информацией для уровня N-1 (N-1 ICI). Структура, содержащая ICI и PDU, образует так называемое сообщение в формате интерфейса между уровнями (IDU) которое передается нижележащему уровню.

На уровне N-1 от сообщения отделяется контрольная информация N-1 ICI, которая определяет способ обработки данного сообщения. После этого уровень N-1 генерирует собственную контрольную информацию для уровня N-2 (N-2 ICI) и добавляет ее к сообщению вместе с заголовком протокола уровня N-1 (N-1 h). После этого получившаяся структура данных передается на уровень N-2 для дальнейшей обработки.

Рассмотрим как определяется сервис, предоставляемый уровнем своему пользователю.

1.7.3.8. Типы соединений

Уровни иерархической архитектуры могут предоставлять два кардинально различающихся типа сервисов уровням находящимся над ними: сервис с установлением логического соединения и сервис, при котором логическое соединение не устанавливается.

В случае сервиса с установлением логического соединения обмен данными может начаться лишь после того, как от отправителя до получателя установлен логически выделенный канал. Протокол транспортного уровня TCP функционирует именно по такой схеме.

Сервис без установки логического соединения основан на модели почтовой системы. Каждое сообщение имеет полный адрес получателя и в потоке сообщений не соблюдается очередность доставки. Таким характеристикам соответствует протокол IP применяемый в сетях с коммутацией пакетов, который предоставляет свой сервис протоколу TCP.

Причина, по которой изучению аспектов взаимодействия протоколов смежных уровней придается много внимания в том, что это взаимодействие является одним из необходимых компонентов протокола, а именно характеристикой среды его исполнения.

1.7.3.9. Примитивы сервисов

Сервис формально определяется набором примитивов, определяющих операции, доступные пользователю. Примитивы являются командами объекту сервиса совершить определенное действие или выдать отчет о выполнении действия. Одним из способов классификации операций сервиса является организация их примитивов в четыре основных класса:

Запрос	объект должен выполнить определенную задачу
Индикация	Запрашивающий должен быть проинформирован о событии
Ответ	Запрашивающий желает прореагировать на событие
Подтверждение	Прибытие ответа на предыдущий запрос

Пример – установление и прекращение связи для простейшего протокола могут быть активированы с помощью следующего набора: (примитивы могут иметь параметры)

CONNECT.request – запрос на установление соединения

CONNECT.indication – сигнал вызываемой стороне

CONNECT.response – применяется вызываемой стороной для подтверждения/отмены установления соединения

CONNECT.confirm – сообщение вызывающей стороне о приеме запроса на соединение

DATA.request – запрос на передачу данных

DATA.indication – сигнал о приеме данных

DISCONNECT.request – запрос на разъединение

DISCONNECT.indication – сигнал другой стороне о разъединении

1.7.3.10.Связь сервисов и протоколов

Сервис это набор примитивов (элементарных действий) которые уровень N предоставляет пользователю – т.е. уровню N+1. Сервис относится к интерфейсу между двумя уровнями. Протокол, с другой стороны, есть набор правил, определяющих формат и значение сообщений, обмениваемых между объектами коммуникационных узлов, находящимися на одинаковом уровне и связанных через сеть. Объекты пользуются протоколами для реализации своих сервисов. По аналогии с объектными языками программирования сервис является классом, в котором определены операции, которые могут быть осуществлены, но не оговариваются детали их внутренней реализации, каковая и является тем или иным протоколом.

1.7.4. Эталонные модели

1.7.4.1. Модель ISO OSI RM

Международной организацией по стандартизации (ISO) в целях унификации методов разработки протоколов была предложена так называемая эталонная модель взаимодействия открытых систем (OSI RM) (рис. 3.).

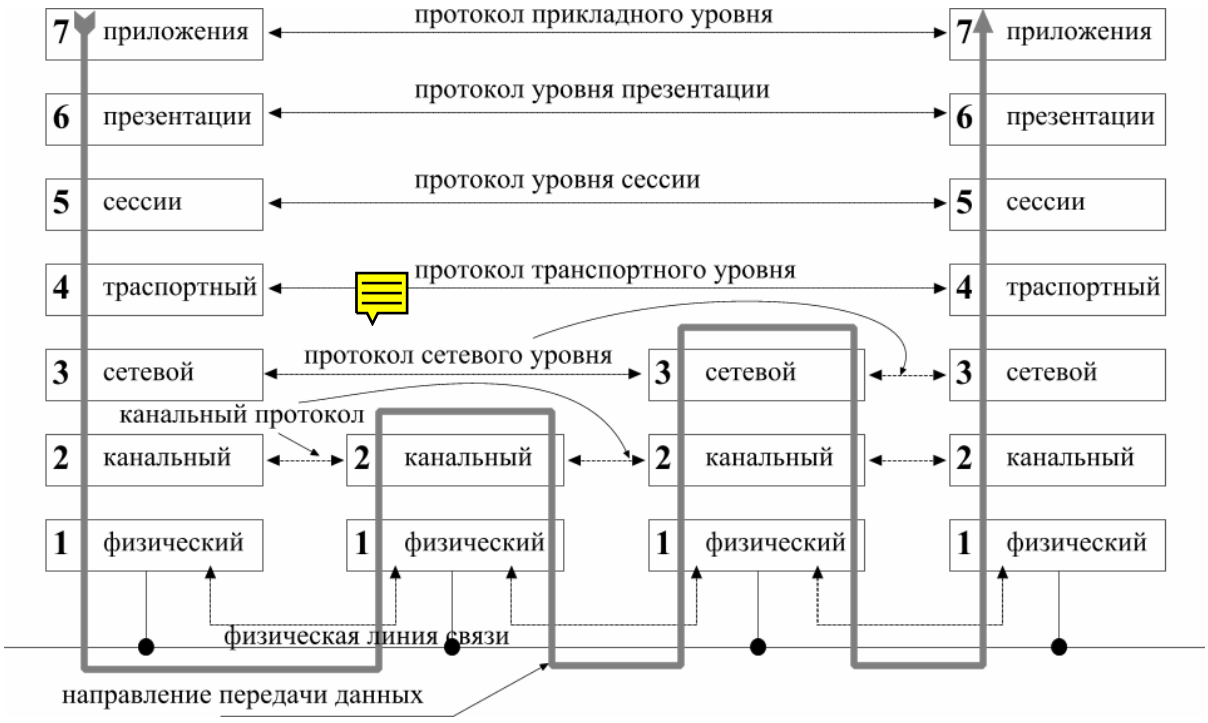

Рис. 3. Эталонная модель взаимодействия открытых систем (ЭМВОС). Уровни и направление передачи данных. Изображены также промежуточные системы канального и сетевого уровней.

Эталонная модель OSI RM состоит из семи уровней:

1.7.4.1.1. Физический уровень

В задачу этого уровня входит корректная передача битового потока по физической линии связи. Когда отправитель передает «1», бит принимающая сторона должна получить его как «1» бит, а не «0». Спецификации этого уровня задают как логические, так и физические и схемотехнические параметры. Физический уровень рассматривает информацию как непрерывный битовый поток.

1.7.4.1.2. Канальный уровень

Этот уровень создает логический канал, не имеющий ошибок и сбоев, для нужд сетевого уровня. Отправляемая информация разбивается на блоки, называемые кадрами, которые пересылаются поочередно. В некоторых случаях уровень создает и обрабатывает кадры с подтверждением корректного приема данных. Поскольку физический уровень имеет дело с неструктурированным потоком битов, то канальный уровень должен заботиться о создании и отслеживании границ кадров. Также канальный уровень может применять различные алгоритмы для определения и по возможности коррекции искажения информации.

Канальный уровень берет на себя все заботы об аккуратном предоставлении информации сетевому уровню. Другая проблема, которую приходится решать на многих уровнях, включая канальный, это, как не допустить переполнения медленного получателя данными от быстрого отправителя. Необходим некоторый механизм, позволяющий отправителю информации иметь данные о наличии свободного буферного пространства у получателя и действовать в соответствии с этим знанием. Сети с широковещательной передачей вносят дополнительные сложности в устройство канального уровня – управление множественным доступом к каналу.

1.7.4.1.3. Сетевой уровень

Задачей сетевого уровня является управление работой базовой подсети. Важнейшая проблема – рассчитать путь от точки отправления то получателя. Такой маршрут может быть основан на статических таблицах хранящихся в памяти устройств подсети или же маршрут может определяться в начале каждой сессии, альтернативным вариантом является высоко-динамическая маршрутизация, когда путь заново определяется для каждого отдельного пакета. Если в базовую подсеть попадает больше пакетов, чем сеть может обрабатывать, то возникает перегрузка, борьба или профилактика которой также является задачей сетевого уровня. Единица информационного обмена на сетевом уровне называется пакетом.

1.7.4.1.4. Транспортный уровень

Транспортный уровень предоставляет услуги (и скрывает от вышележащих уровней) по надежной транспортировке данных по сети. Данный уровень обеспечивает открытие, поддержку и нормальное отключение виртуальных каналов, передает сообщения об ошибках и сбоях, осуществляет управление скоростью потоков информации, чтобы избежать переполнения буферов сетевых устройств и приемников и, соответственно, потерь данных. Отличительной особенностью транспортного уровня является то, что объекты протокола транспортного уровня осуществляют обмен напрямую, вне зависимости от базовой подсети. Единица информационного обмена на транспортном уровне называется сегментом или датаграммой.

1.7.4.1.5. Сеансовый уровень

Этот уровень синхронизирует, открывает, закрывает и манипулирует сеансами связи, активными объектами уровня презентации (каковых может быть несколько). Уровень СЕАНСА присваивает степень срочности сообщениям, руководит ускоренной передачей сообщений об возникших ошибках вышележащим уровням.

1.7.4.1.6. Уровень презентации

Уровень презентации осуществляет кодирование данных, обеспечивая совместимость для различных форматов представления информации.

1.7.4.1.7. Уровень приложения

Этот уровень отличается от остальных тем, что его объектами являются непосредственно приложения, исполняемые на вычислительном узле. Функция этого уровня заключается в определении клиентов, которым необходима коммуникация, синхронизации коммуникации, определении наличие ресурсов для проведения сеанса коммуникации и детектирования ошибок.

1.7.4.2. Модель TCP/IP

В середине 1970-х американской военной исследовательской организацией DARPA было принято решение о создании сети с коммутацией пакетов для обеспечения соединения исследовательских учреждений на территории Соединенных Штатов. В то время исследователи и производители впервые столкнулись с проблемой организации в сеть разнообразных и гетерогенных компьютерных систем. С целью выработки протоколов связи для разнородных систем DARPA начала финансирование исследований проводимых в Станфордском университете по созданию семейства сетевых протоколов. Второй целью было создание сети имеющей возможность продолжать функционировать даже при выходе

из строя существенной доли ее аппаратной части. В результате этой работы появились протоколы семейства Internet protocols. Способность новой разработки соединять сети, основанные на разных технологиях, совершенно прозрачным образом была основной задачей разработчиков с самого начала. Наиболее практичными и широко используемыми членами этого семейства являются протоколы Transmission Control Protocol (TCP) - протокол контроля передачи и Internet protocol (IP) протокол интернета, а сама архитектура стала известной под названием эталонной модели TCP/IP.

Семейство протоколов TCP/IP применяется для коммуникации через любое количество промежуточных ЛВС. Протоколы TCP/IP идеально подходят как для коммуникации в ЛВС, так и для глобальных вычислительных сетей. Набор протоколов TCP/IP содержит спецификации не только протоколов низкого уровня, таких как TCP или IP, но также и таких широко распространенных приложений как электронная почта, эмуляция удаленного терминала, протокол передачи файлов и многое другое.

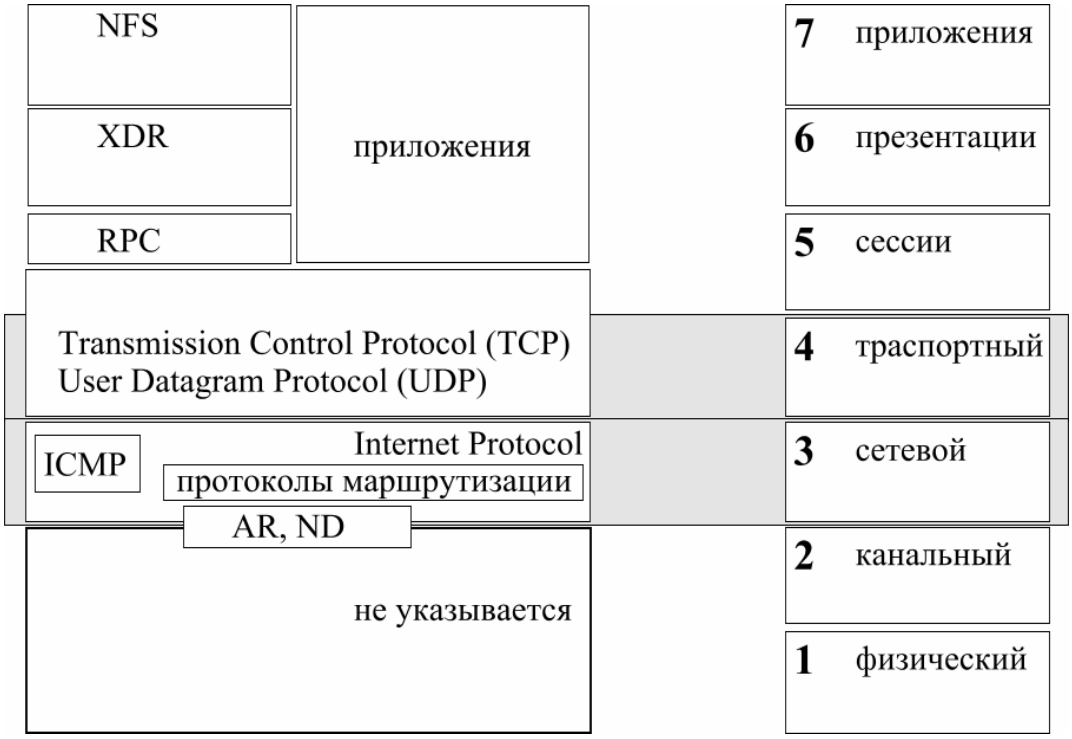

Рис. 4. Сопоставление эталонных моделей сетевых архитектур ЭМВОС и TCP/IP.

Как видно из рис. 4. уровни модели TCP/IP не полностью совпадают с уровнями модели OSI. Основные составляющие модели TCP/IP соответствуют сетевому и транспортному уровням ЭМВОС. Эти протоколы – IP и TCP являются ключевыми для концепции современной коммуникационной архитектуры (рис. 5.).

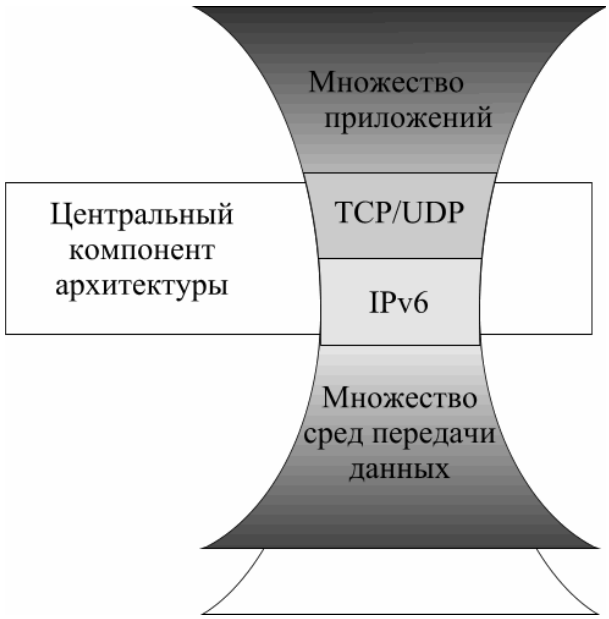

Рис. 5. Современная концепция сетевой архитектуры.

1.7.4.2.1. Сетевой протокол модели TCP/IP: протокол IP

Протокол IP является единственным протоколом сетевого уровня семейства TCP/IP, поэтому все транспортные протоколы стека TCP/IP используют сервисы протокола IP. Т.е. сервис, представляемый IP, является средой, в которой исполняется протокол TCP.

Протокол IP предоставляет пользователю ненадежный сервис по передаче пакетов. В дополнение к функциям маршрутизации, т.е. определения оптимального маршрута пакетов от отправителя до получателя на основании информации о топологии сети, IP также ответственен за фрагментацию и сборку пакетов и уведомление о сбойных ситуациях.

1.7.4.2.2. Транспортные протоколы модели TCP/IP

Транспортный уровень набора протоколов интернета состоит из двух протоколов: TCP (Transmission Control Protocol - протокол контроля передачи) и UDP (User Datagram Protocol - протокол пользовательских датаграмм). TCP предоставляет надежный транспорт с установкой логического соединения

TCP является наиболее важным транспортным протоколом модели TCP/IP он обеспечивает полностью дуплексную связь с кумулятивным подтверждением приема. Он перемещает информацию в виде непрерывного неструктурированного потока, где байты идентифицируются порядковым номером. Объект протокола TCP может одновременно поддерживать множество сеансов информационного обмена для протоколов высших уровней, осуществляя мультиплексирование потоков.

1.7.5. Эволюция коммуникационных протоколов

Рассмотрим процесс развития сетевых протоколов на примере стека TCP/IP. Нас интересует вопрос преемственности в развитии протоколов, и их обратной совместимости. На развитии реальных протоколов отражается множество факторов: это степень сложности протокола, качество его спецификации, верифицируемость данного протокола и результаты его тестирования на соответствие стандартам. Кроме того, ряд практических характеристик протокола может существенно повлиять на его реальное использование и одним из важнейших факторов здесь является обратная совместимость нового протокола с предыдущими версиями или реализациями.

Обратная совместимость новой версии протокола означает, что его реализация сможет взаимодействовать со старыми версиями без потери производительности, причем улучшение характеристик работы системы будет происходить при взаимодействии новых версий протоколов или в некоторых случаях уже при взаимодействии старой версии с новой. Целесообразность требования обратной совместимости вполне оправдана, с другой стороны, диалектическое развитие протоколов коммуникационных систем приводит к необходимости смены одного протокола на другой, несовместимым с прежним на определенном этапе развития системы. При этом обратная совместимость либо не сохраняется вовсе, либо обеспечивается за счет временного применения дополнительных механизмов не являющихся частью системы протоколов. Такая ситуация наблюдается в настоящее время, когда сетевой протокол в сети Интернет IP версии 4 заменяется на новую версию IPv6 [41], которая не предусматривает обратной совместимости со старым протоколом Интернет. Такой подход вполне оправдан, поскольку задача обеспечения обратной совместимости требует усложнения многих компонентов протокола – его словаря и процедурных правил, существенно затрудняет его анализ и верификацию. В случае IPv6 было решено пожертвовать обратной совместимостью для обеспечения минимальности и простоты множества процедурных правил протокола.

Первая спецификация транспортного протокола TCP была дана в работе [4] в 1980 году. За прошедшие 20 лет протокол TCP подвергался большому количеству оптимизаций и дополнений, которые либо решали очевидные проблемы выявляющиеся по ходу применения протокола, либо улучшали его характеристики для систем узкой специализации. Все эти изменения оставляли протокол совместимым со старыми версиями. В результате сложность протокола TCP возросла настолько, что полный перебор достижимых состояний конечного автомата моделирующего протокол и даже контролируемый выборочный перебор не являются возможными для автоматизированной верификации. Вследствие этого затруднена

не только автоматизированная верификация протокола TCP, но и ручной анализ даже избранных наборов его состояний.

Основное новшество предлагаемого в данной работе протокола транспортного уровня ARTCP заключается в заново созданном алгоритме управления скоростью потока, который использует совершенно отличные от TCP принципы. Реализация протокола ARTCP может обеспечивать совместимость со стандартным TCP.

1.7.6. Транспортный уровень: роль и компоненты

Транспортный уровень является центральным для всей иерархии протоколов. Задача этого уровня – обеспечивать надежную и эффективную транспортировку информации от исходного к конечному устройству вне зависимости от физических особенностей сетей или ограничений накладываемых протоколами БС.

1.7.6.1. Сервис транспортного уровня

1.7.6.1.1. Сервисы, предоставляемые пользователю

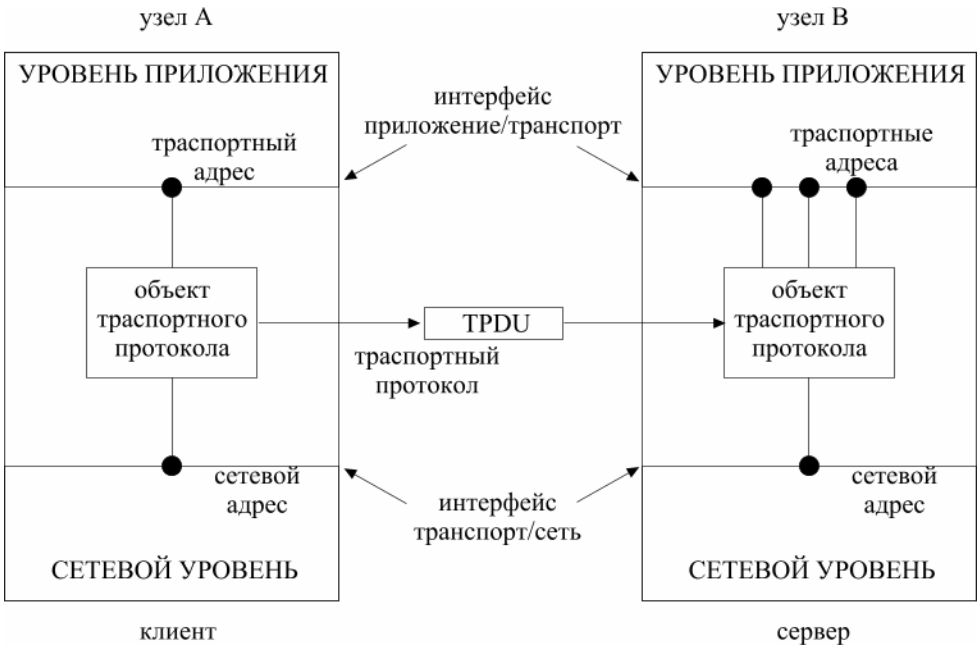

Рис. 6. Обмен данными на уровне транспортного протокола. Формат данных транспортного протокола: TPDU.

Главная задача транспортного уровня заключается в предоставлении эффективного и надежного сервиса для соответствующих пользователей – главным образом пользовательским процессам уровня приложения. Для выполнения этих задач транспортный уровень полагается на сервисы сетевого уровня. Аппаратное и/или программное обеспечение, выполняющее функции транспортного уровня называется транспортным

объектом. Этот объект может быть частью ядра операционной системы, отдельным процессом в пользовательском пространстве, в виде библиотечных объектов скомпонованных с пользовательскими программами или функцией отдельной интерфейсной платы. Блок данных, посредством которого происходит обмен между объектами транспортного протокола, называется TPDU (сообщение транспортного протокола) (рис. 6.).

В отличие от всех нижних уровней, транспортный уровень функционирует непосредственно между участниками обмена информацией, напрямую и позволяет этим системам обмениваться информацией вне зависимости от промежуточных сетей и систем. Пользователи, как правило, не имеют контроля над БС, поэтому единственный способ, посредством которого они могут влиять на качество услуг БС, это использование дополнительного уровня над сетевым.

Транспортные протоколы делают возможным предоставлять пользователю транспортный сервис гораздо более надежный, чем используемый сетевой сервис. Наличие потерянных и поврежденных пакетов контролируется и компенсируется транспортным уровнем. Кроме того, сервисные примитивы транспортного уровня могут быть разработаны таким образом, чтобы быть полностью независимыми от сетевых сервис примитивов, которые радикально различаются для разных типов сетей. Благодаря наличию транспортного уровня, пользовательские приложения могут разрабатываться с применением стандартного интерфейса (стандартного набора вызовов сервис примитивов) и использоваться без изменений на самых разных типах сетевых технологий, что и имеет место на практике. Транспортный уровень изолирует прикладные программы от особенностей технологии, разработки, ненадежности и разнородности сетей. Именно поэтому существует разделение уровней модели OSI RM на две группы с 1-го по 4-й и выше 4-й. Первая называется поставщиком транспортных услуг, вторая - пользователем транспортных услуг. Проведение такого различия, во-первых, влияет на специфику реализации уровней и протоколов, а во-вторых, ставит транспортный уровень на место ключевого звена в иерархической модели, поскольку он находится на интерфейсе между поставщиком и пользователем надежного транспортного сервиса. Основная задача транспортного уровня в том, чтобы максимально улучшить качество услуг сетевого уровня.

Рассмотрим, каким образом протокол TCP улучшает параметры сервиса по сравнению с сервисом сетевого уровня. Как было сказано ранее совокупное влияние физического, канального и сетевого уровней на передаваемые данные выражается в появлении трех типов ошибок данных:

- Потеря

- Дублирование
- Искажение порядка отправки

Протокол TCP полностью устраняет эти ошибки. С точки зрения пользователя транспортного протокола, сеть выглядит как полностью дуплексный канал, который перемещает поток байтов пользователя без потерь, причем байты появляются из канала в том самом порядке, в котором они поступили в него на передающей стороне и только однократно. Выполнение всех этих функций никак не зависит от используемых технологий сетевого, канального и физического уровней, каковых может быть много на маршруте от отправителя до получателя.

Приложению, использующему сервис TCP, достаточно лишь запросить связь с другим приложением, а затем начать передавать свою последовательность данных после установки логического соединения, не заботясь о возможных потерях, ошибках, управления скоростью передачи и т.д.

1.7.6.1.3. Примитивы транспортного сервиса

Примитивы транспортного сервиса уже являются достаточной формализацией для описания сервисного компонента протокола. Через эти примитивы пользователи (приложения) получают доступ к услугам транспортного уровня. Каждый тип транспортного сервиса имеет свой набор примитивов. В отличие от ненадежной связи без установки ВК на сетевом уровне, транспортный уровень предлагает надежную связь с построением логического канала.

Поскольку реальные сети не могут гарантировать отсутствия ошибок, то задача транспортного уровня как раз в том, чтобы обеспечить надежную связь, пользуясь ненадежной. Например, два процесса использующие именованный коммуникационный канал (pipe) [86] в среде ОС UNIX [64], в своей работе предполагают, что соединение является идеальным. Разработчику программы реализующей эти процессы нет необходимости заботиться об отслеживании подтверждений, потерь сообщений, перегрузок. Соединение для него выглядит как абсолютно надежное. Аналогично, транспортные протоколы представляют ненадежный канал как битовый поток свободный от ошибок для пользователя.

В качестве дополнительного сервиса транспортный уровень может предлагать и ненадежную связь без подтверждений приема и управления потоком, в среде TCP/IP такой сервис реализуется протоколом UDP.

Различны также и пользователи транспортного и сетевого уровня. Сетевой сервис используется только объектами транспортного уровня. Крайне редко программы

разрабатываются так, чтобы использовать сервисы сетевого уровня напрямую. Основная масса приложений разрабатывается именно в расчете на использование сервисов, а, следовательно, и примитивов транспортного уровня. Поэтому спецификация сервиса транспортного уровня должна быть универсальной и простой в использовании, как, например, интерфейс *Berkeley Sockets*.

Для того чтобы составить представление о примитивах транспортного сервиса рассмотрим реальный интерфейс Berkeley Sockets [84, 85]

Это интерфейс для взаимодействия приложений с объектами транспортных протоколов, включая протокол TCP. Изначально был разработан для работы с TCP в операционной системе семейства BSD UNIX [64] в 1980 году. Сейчас интерфейс Berkeley Sockets является промышленным стандартом и используется в большинстве операционных систем.

Примитив	Пояснение
SOCKET	Создать новую точку доступа к системе связи (ТДС)
BIND	Присвоить локальный адрес ТДС
LISTEN	Объявить о готовности к приему соединений, установить размер очереди
ACCEPT	Блокировать инициатора до поступления запроса на соединение
CONNECT	Активная попытка установки соединения
SEND	Передача данных
RECEIVE	Получение данных
CLOSE	Завершить сеанс связи

Пользователь получает доступ к данным примитивам на UNIX системе в виде системных вызовов, поскольку объект протокола TCP в системе UNIX является частью ядра операционной системы [86].

Вызов SOCKET в случае успешного завершения возвращает дескриптор[4] файла и выделяет место под таблицы в контрольных блоках на транспортном уровне. Параметры вызова указывают на тип и характеристики требуемого сервиса.

После получения дескриптора, характеризующего одну сторону соединения, ему присваивается локальный адрес с помощью вызова BIND. Причина использования отдельного вызова в том, что некоторые процессы требуют использования стандартного адреса, для других же этого не требуется.

LISTEN объявляя о готовности принимать соединения, выделяет место под очередь принимаемых сегментов. LISTEN не является блокирующим вызовом. Вызов ACCEPT блокирует процесс вызвавший его в ожидании соединения. Примитив CONNECT блокирует вызвавший его процесс и начинает активную попытку открыть соединение. Когда этот вызов

[4] указатель по которому пользовательские программы могут иметь доступ к файлам или другим объектам операционной системы

возвращается, процесс деблокируется, и может начинать обмен информацией по уже установленному соединению. Разрыв соединения происходит при выполнении примитива CLOSE и является симметричным.

1.7.6.2. Характеристика среды исполнения

Для того, что бы определить среду исполнения протокола TCP или его предложенной модификации ARTCP необходимо рассмотреть принципы функционирования уровней иерархии, сервисами которых пользуется транспортный протокол. Согласно эталонной модели OSI RM транспортному уровню предшествуют сетевой, канальный и физический уровни иерархии. Поскольку задача каждого из уровней в том, чтобы максимально изолировать своего от проблем нижних уровней, достаточно рассмотреть лишь сервис сетевого уровня.

1.7.6.2.1. Сервисы сетевого уровня

Доступ к сервисам сетевого уровня возможен на интерфейсе между сетевым и транспортным уровнями. Важность данного интерфейса определяется тем, что часто этот интерфейс отделяет потребителя коммуникационных услуг от их поставщика, т.е. оператора базовой сети. Оператор базовой сети имеет полный контроль над протоколами предшествующими транспортному уровню. Именно по этой причине интерфейс должен прорабатываться особенно тщательно. Сервисы сетевого уровня любой коммуникационной системы должны обладать следующими основными характеристиками:

- сервисы не зависят от канальной технологии базовой сети.
- транспортный уровень должен быть экранирован от количества, типов и топологий различных базовых сетей.
- если сетевые адреса являются доступными для транспортного уровня, то они должны укладываться в пределы стандартной схемы адресации, в которой каждый адрес уникален.

Рассмотрим вкратце функционирование протокола IP, и проанализируем его с точки зрения среды исполнения транспортного протокола.

Получая данные от вышележащего уровня в виде блоков конечного размера, протокол IP инкапсулирует их в пакет, добавляя к данным пользователя свой заголовок. На рис. 7. приведены поля заголовка протокола IPv6. Для нас представляет интерес то, что в заголовке отсутствует поле кода циркулярного контроля. Это связано с тем, что задача проверки и коррекции ошибок передачи возложена на канальный уровень, который всегда выполняет эту проверку. Протокол IPv4 осуществлял проверку целостности данных при помощи кода

контроля четности, но из функциональности IPv6 данная операция была изъята, поскольку она дублирует аналогичную или более мощную систему канального уровня.

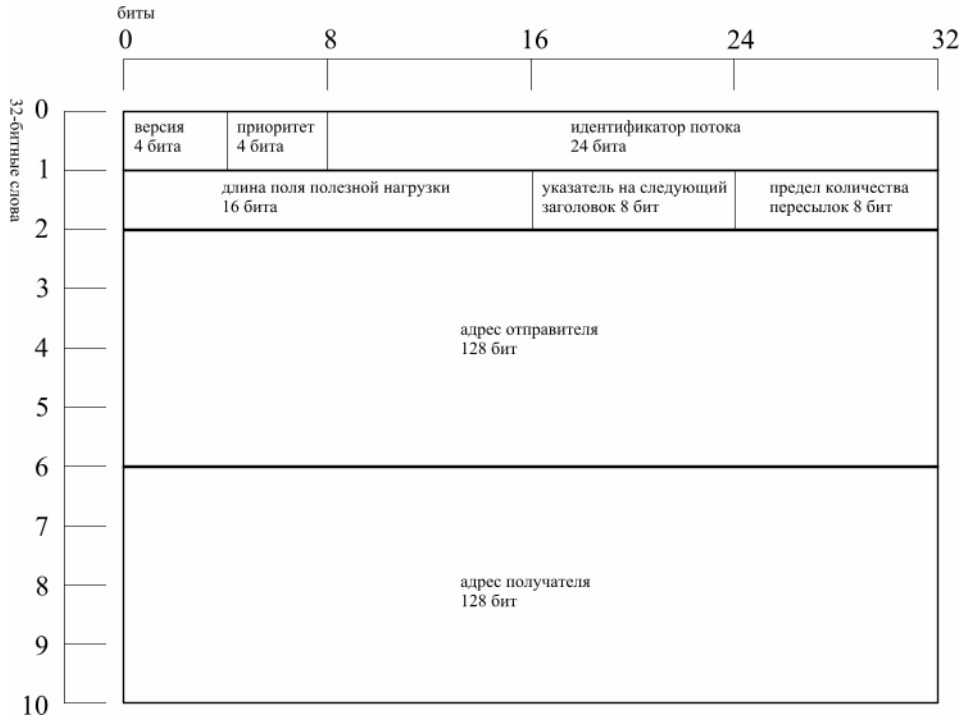

Рис. 7. Формат заголовка пакета протокола IP (изображен для IP версии 6).

Закончив компоновку пакета, IP передает его в сеть на маршрутизатор топологически более близкий к узлу которому адресовано сообщение. Каждый IP пакет в заголовке содержит полный адрес отправителя и получателя. Маршрутизаторы в составе сети обмениваются топологической информацией с помощью протоколов маршрутизации и благодаря этому обладают знанием топологии сети. Получая IP пакет, каждый маршрутизатор сравнивает адрес узла назначения с имеющейся у него таблицей топологической информацией и направляет пакет на соответствующий выходной порт. Поскольку разные маршрутизаторы могут обладать различным представлением о топологии сети, то и маршрутов между двумя узлами в сети может быть несколько. Если на выходной порт маршрутизатора поступает больше пакетов в единицу времени, чем может его покинуть, то пакеты организуются в очередь. Поскольку ресурсы буферного пространства маршрутизаторов конечны, то переполнение очереди приводит к потерям пакетов.

1.7.6.2.2. Возникновение и коррекция ошибок

Низкое соотношение энергии сигнала к энергии шума на линии приводит к искажениям принимаемого сигнала и как следствие к высокой вероятности битовых ошибок на приемнике. Ошибки канала передачи данных проявляются как:

1. Вставленные данные: данные, полученные приемником, но никогда не передававшиеся отправителем.
2. Потерянные данные: данные отправленные, но не дошедшие до получателя (исчезнувшие на физическом уровне).
3. Дублированные данные: данные, переданные единожды и полученные в нескольких экземплярах
4. Искаженные данные: данные, поврежденные в транзите
5. Разупорядоченные данные: данные, последовательность получения которых не совпадает с последовательностью передачи

Применяемые на канальном уровне алгоритмы распознавания и коррекции ошибок с достаточной вероятностью позволяют отобразить ошибки типа вставленных и искаженных данных на оставшиеся три типа ошибок, а именно потеря, дублирование и разупорядочивание данных.

Сетевой уровень стека TCP/IP предоставляет пользователю ненадежный сервис без установки логического соединения. Сказанное означает, что IP пакеты могут начать поступать в сеть без задержки, как только информация готова к отправке (пакет сформирован). Отправленные пакеты не подтверждаются получателем. Пакет может быть потерян, и никогда не дойти до получателя, по нескольким причинам: он может быть отброшен из переполненного буфера маршрутизатора вследствие перегрузки последнего, либо данные в составе пакета могут быть разрушены в процессе передачи по ненадежному каналу. Таким образом, сетевой уровень может привнести дополнительные ошибки типа потерь данных. Также вследствие возможности наличия нескольких маршрутов к получателю порядок прибытия пакетов может не совпадать с порядком их отправки, если поток пакетов будут доставляться по маршрутам с различной задержкой. Более того, существует возможность доставки получателю нескольких копий одного пакета.

Протокол TCP, таким образом, должен самостоятельно отслеживать возникновение трех типов ошибок данных и осуществлять самостоятельное восстановление в ситуациях, когда сегмент не приходит вообще, когда сегменты доставляются сетью не в том порядке, в котором были отправлены и когда сеть доставляет несколько копий одного сегмента.

1.7.6.2.3. Управление потоком

Каждый из нижележащих уровней осуществляет управление скоростью передачи данных. Физический уровень ответственен за синхронизацию записи и сканирования среды передачи. Канальный уровень управляет скоростью передачи пакетов между двумя узлами, применяя схему управления потоком той или иной сложности, в зависимости от назначения

– Xon/Xoff, Alternating bit [42], sliding window [43] и т.п. Сетевой уровень также имеет примитивную схему управления потоком. Маршрутизатор в состоянии перегрузки может отправить своим топологическим соседям сообщение о наличии перегрузки при помощи протокола ICMP. Однако все алгоритмы канального и сетевого уровней осуществляют управление скоростью потока лишь локально. TCP, как протокол транспортного уровня, осуществляет управление потоком по всей длине логического канала между передающей и принимающей системами.

В протоколе TCP существует механизм, ограничивающий скорость отправки сегментов в сеть, для того чтобы избежать переполнения буферов промежуточных маршрутизаторов и буфера приемника.

1.7.6.3. Процедурные правила

Сервис транспортного уровня реализуется транспортным протоколом между двумя объектами транспортного уровня. Процедурные правила, требуемые для реализации этого сервиса слишком громоздки, поэтому их формальное описание например в виде конечных автоматов было бы слишком громоздким. Далее мы дадим описание процедурных правил TCP в виде отдельных алгоритмов. Отдельными важными аспектами транспортного протокола, которые должны быть учтены в его процедурных правилах являются:

- Адресация и мультиплексирование
- Инициализация и закрытие связи
- Буферизация и управление потоком

1.7.6.4. Словарь транспортного протокола

Словарь простейшего транспортного протокола, обеспечивающего надежный канал связи и контроль скорости передачи, должен позволять передавать следующие типы сообщений [1, 3]:

Данные (DATA)

Подтверждения (ACK)

Размер приемного окна (WND)

Таким образом, минимальный словарь транспортного протокола обеспечивающего надежную связь таков: $V=\{DATA, ACK, WND\}$. Если рассматривать каждое направление полнодуплексного транспортного соединения раздельно, то данные передаются в направлении от отправителя к получателю, а подтверждения и обновления окна в противоположном направлении.

Поскольку каждое индивидуальное сообщение транспортного протокола инкапсулируется сетевым уровнем в отдельный пакет, то выгодно объединить как можно

больше сообщений в каждом TPDU. Реально в каждом сообщении объединяются все эти типы. Формат сообщений транспортного протокола таков:

{DATA, SEQ}

{ACK, SEQ, WND}, где SEQ и WND – целые числа.

Поскольку транспортные соединения рассчитаны на двусторонний обмен сообщениями, то объединяются все поля в едином формате сообщения:

{DATA, SEQ, ACK, SEQ, WND}

Число SEQ на второй позиции нумерует передаваемые данные, а в третьей позиции – подтверждаемые данные.

1.7.6.5. Кодировка сообщений на транспортном уровне

Итак, словарь транспортных протоколов состоит из сообщений – так называемых TPDU, которые инкапсулируют передаваемые данные. Сам TPDU в свою очередь заключен в пакет сетевого уровня.

Формат сегмента, как правило, следующий: заголовок фиксированной длины выровненный на 32 битной границе предшествует полю данных переменной длины. Более конкретная схема кодировки сегмента и детализация дополнительных полей приведена в схеме заголовка TCP сегмента (рис. 8.).

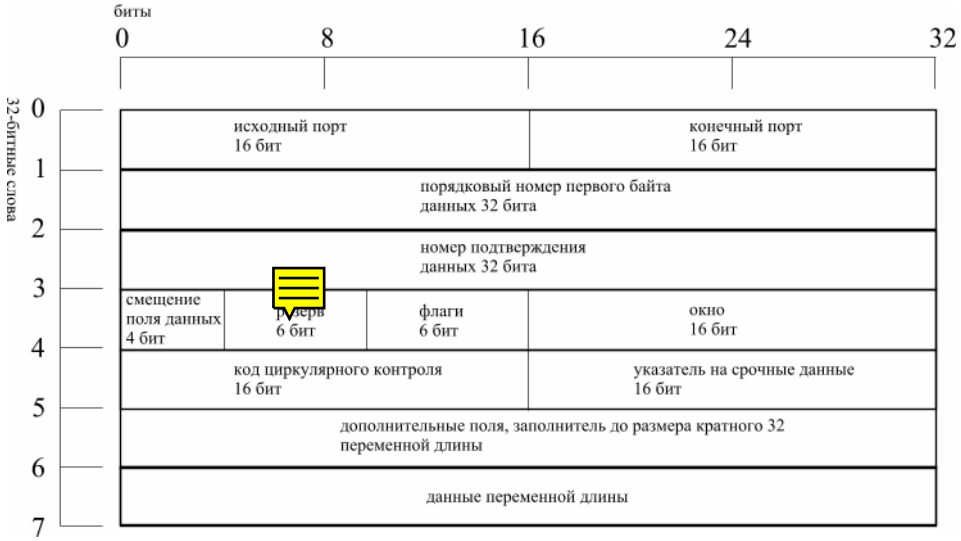

Рис. 8. Формат заголовка сегмента TCP.

Заголовок сегмента транспортного протокола содержит поле, позволяющее проверить целостность данных и заголовка принятого сегмента. В протоколе TCP это 16-ти битное поле несет проверочную сумму, полученную суммированием по модулю 2 всех 16-ти битных слов части заголовка и поля данных. При получении сегмента проверочная сумма вычисляется заново и если результат не равен нулю, то сегмент содержащий ошибку отбрасывается.

1.7.6.6. Адресация

Для установки связи с удаленным приложением требуется указать его адрес. Адресация необходима для всех видов сетей. Существует метод, позволяющий устанавливать транспортный адрес, по которому приложение-сервер ожидает прибытия запроса на соединение. Для архитектуры TCP/IP это пара: IP адреса и номер локального порта. Объекты транспортного уровня допускают использование нескольких транспортных адресов (рис. 9), при этом мультиплексирование потоков в пределах узла осуществляется самим транспортным протоколом. Постоянно существующие серверные приложения адресуются напрямую, для работы с приложениями, запускаемыми лишь на время существования сессии, используется так называемый «протокол первоначального соединения» или интернет сервер (UNIX inetd) [75].

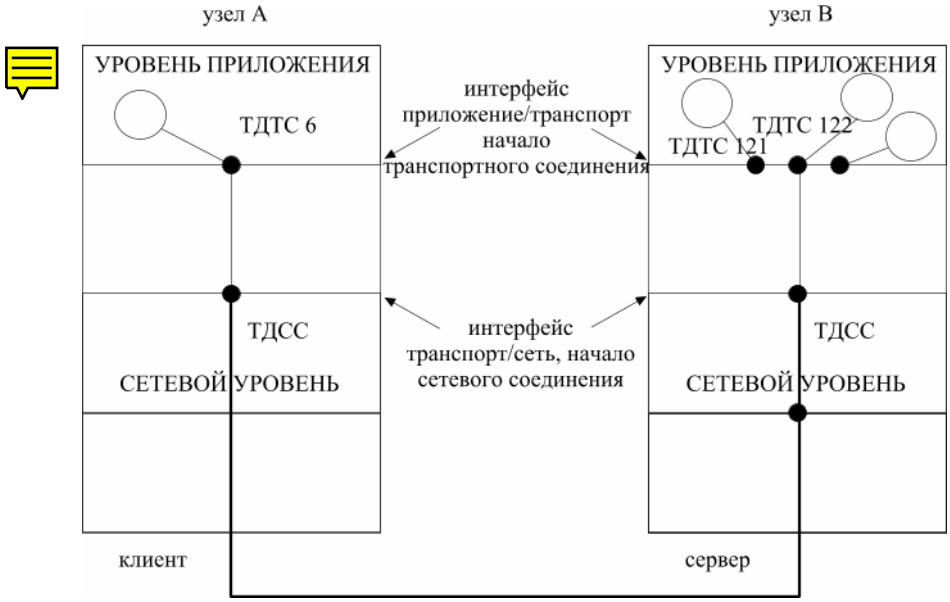

Рис. 9. Точки доступа к сервису транспортного уровня (ТДТС). Изображены интерфейсы транспортного уровня с уровнем приложения и сетевым уровнем.

1.7.6.7. Установка соединения

Без способности БС к накоплению пакетов задача установления соединений свелась бы к двум действиям – послать запрос на соединение – дождаться положительного ответа на него. На самом же деле проблема значительно более сложна. Если БС перегружена, и подтверждения не успевают прибыть к отправителю вовремя, то и запросы на установление соединения могут быть отправлены несколько раз. Поскольку маршрутизация каждого пакета происходит независимо, то существует вероятность задержки некоторых пакетов в течение более длительного времени, чем других и, соответственно, сбоев при установлении

соединения. Единичная транзакция может, таким образом, случайно произойти два и более раза.

Используемый в архитектуре TCP/IP метод сочетает с идентификацию каждого пакета с требованием ограничения времени жизни пакетов в сети. Причем идентификация каждого пакета подразумевает наличие у него порядкового номера.

В реальности необходимо гарантировать, что не только сам пакет исчез из сети, но и все его подтверждения, поэтому используется промежуток времени T, кратный максимальному времени жизни пакета. Если прошло время T с момента отправки пакета, то мы можем быть уверены, что ни сам пакет, ни его подтверждения не существуют на сети. Если опираться на такую предпосылку, то можно разработать надежный способ безопасной установки соединения. Метод был впервые предложен [37] в 1975 и усовершенствован [38] в 1978. Метод называется «трехсторонний обмен» (Three-way handshake). Этот метод не требует использования обеими сторонами одних и тех же номеров. Нормальная процедура установки соединения иллюстрирована на рис. 10 (часть A.) Устройство A посылает запрос на соединение (CR) устройству B, указывая при этом начальный порядковый номер который будет использоваться транспортным протоколом устройства A для передачи данных. Устройство B, получив запрос, реагирует отправкой пакета, уведомляющего о согласии установить соединение, указывает свой собственный начальный номер *y* и подтверждает номер *x*. После этого в первом пакете с данными, который имеет номер *x,* машина A подтверждает прием сообщения с номером Варианты B. и C. иллюстрируют поведение транспортных объектов при получении дубликата запроса на соединение и дубликата подтверждения.

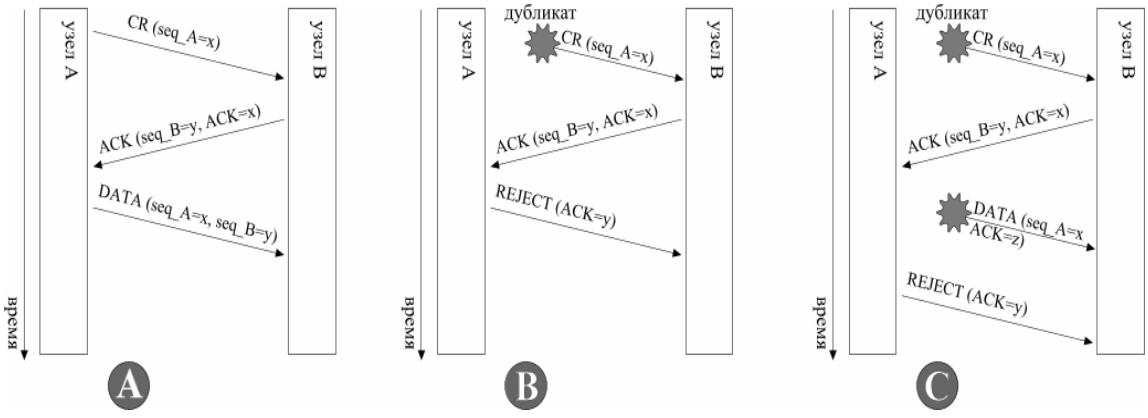

Рис. 10. Установка соединения по методу тройного обмена. Часть A. - нормальная установка соединения. B. - реакция на дубликат запроса. C. - реакция на дубликат запроса и данных.

Схема установки соединения приведенная выше применяется транспортным протоколом TCP. Альтернативная схема надежной установки соединения в условиях наличия задержанных копий сообщений описана [39].

1.7.6.8. Разрыв соединения

Для закрытия транспортного соединения используется так называемый симметричный разрыв, при котором каждое направление соединения считается независимым и закрывается по отдельности.

Рис. 11 иллюстрирует процесс закрытия соединения с использованием таймеров. Несмотря на относительную надежность такого протокола, он может не сработать в случае потери начального DR и всех N повторных передач. В этом случае инициирующая сторона закроет соединение, противоположная же не получив информации о закрытии будет активной. Такое состояние называется полуоткрытое соединение. Для предотвращения возникновения таких состояний вводится правило, по которому соединение будет автоматически закрываться, если в течение определенного времени на нем не зарегистрировано никакой активности.

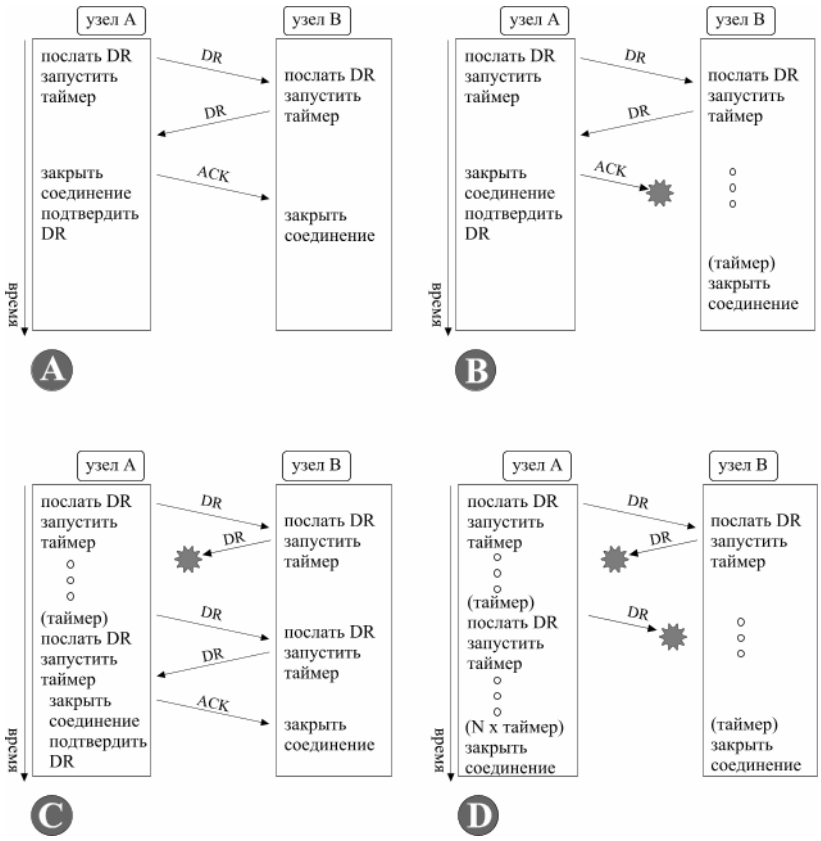

Рис. 11. Сценарии разрыва соединения с использованием подтверждений и таймеров. Часть A. - сценарий без ошибочных ситуаций. B, C и D - различные сценарии возникновения ошибок. Потерянный сегмент изображается звездочкой.

Если же соединение должно оставаться открытым в течение длительного времени, ожидая появления данных для передачи, то сторонам придется периодически обмениваться пустыми сообщениями, удерживающими противоположную сторону от закрытия.

1.7.6.9. Управление потоком и буферизация данных

Управление потоком непосредственно связано с буферизацией. Задачей процесса управления потоком является не допустить переполнения получателя и промежуточных узлов информацией, которую те не успевали бы обрабатывать. В том случае, если БС предоставляет ненадежный датаграммный сервис, то отправитель должен буферизовывать отправленные сообщения на случай возникновения необходимости повторной отправки. Получатель, зная, что сообщения остаются в памяти отправителя пока не поступит подтверждение их приема, может выделять или не выделять буферное пространство для каждого соединения. В случае буферизации сообщений у получателя, что позволяет существенно повысить производительность системы, возникает вопрос о схеме выделения памяти под буферы. Это может быть и система буферов равных размеров (если все сообщения примерно одного размера) или большой циркулярный буфер или набор буферов разных размеров.

Оптимальное решение относительно буферизации у отправителя и получателя зависит от характеристик потока. Например, для трафика порожденного интерактивной работой с удаленным терминалом, характеризуемого небольшой скоростью, лучше вообще не выделять буферов, а получать и передавать информацию приложениям по мере ее поступления (естественно, что переданное сообщение должно оставаться в памяти отправителя до получения подтверждения). С другой стороны для потока созданного не интерактивной передачей данных, например – загрузкой удаленного файла, эффективность может быть существенно повышена, если получатель выделит максимальное количество буферного пространства, чтобы как можно более повысить скорость передачи.

Таким образом, с течением времени отправитель и получатель должны согласовывать свои схемы выделения буферного пространства. Отправитель должен иметь возможность запрашивать получателя о выделении определенного количества буферного пространства или же получатель (не имеющий информации о потребностях отправителя) должен сообщать, что для соединения зарезервировано X буферов.

Выделение буферов должно происходить независимо от прихода подтверждений (в отличие от большинства протоколов канального уровня). Эта схема подразумевает наличие «окна» переменного размера. Получатель выделяет такое количество буферного пространства в ответ на запрос отправителя, какое позволяют его ресурсы. Каждый раз,

посылая сообщение, отправитель должен уменьшить указанное количество и временно прекратить передачу, если размер свободного буферного пространства у получателя уменьшится до нуля. Получатель, в свою очередь должен указывать текущий размер окна и номер подтверждения в потоке идущем в обратном направлении. При увеличении количества памяти доступной для буферизации сетевых соединений уже не недостаток свободного буферного пространства будет ограничивать максимальную скорость передачи данных, а доступная пропускная способность БС. Таким образом, необходим механизм управления потоком, который учитывает также и несущую способность БС, а не только количество буферного пространства у конечных участников обмена. Если используется механизм ограничивающего окна, то его размер должен отражать текущую пропускную способность БС.

1.7.7. Протокол TCP

Семейство протоколов TCP/IP включает в себя два протокола транспортного уровня это TCP – Transmission Control Protocol - и UDP User Datagram Protocol. UDP достаточно прост, по существу это небольшое дополнение к IP, и не использует виртуальный канал. TCP это надежный протокол, который использует предварительную установку виртуального канала и доставляет данные пользователя без ошибок. TCP был специально разработан, чтобы обеспечивать надежную передачу байтового потока от отправителя к получателю через ненадежную БС. БС может состоять из различных сетевых компонентов, обладающих разными топологическими схемами, технологиями, пропускной способностью, задержками.

Формально протокол TCP был определен в RFC 793 [4]. Со временем накопившиеся ошибки и недостатки были учтены и нашли выражение в требованиях к устройствам интернет в RFC 1122 [5]. В дальнейшем было проведено большое количество исследований с протоколом. Часть модификаций стала стандартом и новые спецификации протокола содержится в RFC 1323 [6].

Каждое устройство, поддерживающее TCP, содержит в себе объект протокола, который управляет TCP потоками и взаимодействует с уровнем IP. Этот объект принимает потоки данных от локальных процессов, разбивает их на сообщения определенной длины (сегменты) и передает сегмент на уровень IP для его передачи в виде отдельного IP пакета. По прибытии на адресуемую машину информация из этих пакетов передается TCP объекту, который реконструирует оригинальный байтовый поток и передает ее пользовательской программе. Уровень IP не гарантирует доставку пакетов, поэтому TCP должен отслеживать потерянные пакеты и осуществлять повторную отправку. Пакеты могут прибывать и в порядке, отличающемся от порядка, в котором они были отправлены, поэтому задачей TCP

также является реконструкция порядка в битовом потоке. Для управления потоком и предотвращения перегрузок БС TCP применяет механизм скользящего окна [43].

1.7.7.1. Модель обслуживания TCP

Доступ к сервису TCP можно получить путем создания на конечных машинах точек доступа (Sockets). Каждая такая точка имеет адрес, состоящий из IP адреса машины и 16-ти битного номера, уникального в пределах устройства, идентифицирующего порт. TCP соединение устанавливается между двумя точками доступа и пара транспортных адресов однозначно идентифицирует соединение. Несколько соединений могут оканчиваться на одной точке доступа. Порты с номерами ниже 256 однозначно соответствуют набору конкретных приложений. Порты с другими номерами выделяются приложениям динамически. Все TCP соединения полнодуплексные и имеют тип точка-точка. Соединение представляет собой неструктурированный байтовый поток, т.е. границы между сообщениями не сохраняются.

Когда приложение передает данные TCP, протокол может либо сразу осуществить их отправку, либо накопить достаточное их количество в буфере. Если же приложению необходимо послать информацию мгновенно, то оно может использовать флаг PUSH. Кроме того, существует возможность пересылать срочную информацию с использованием флага URGENT. Когда приложение устанавливает этот флаг, то TCP передает все имеющиеся данные без промедления, кроме того, TCP получателя генерирует прерывание для приложения которому предназначается срочная информация и указывает на положение ее в полученном потоке.

Каждый байт TCP соединения имеет уникальный 32-х битный порядковый номер. Эти номера используются не только для подтверждений, но и для механизма управления окном. Объекты TCP обмениваются информацией в виде сегментов. Сегмент состоит из 20-ти байтного заголовка (+ необязательная часть) и нуля или более байтов данных (рис. 8). Протокол сам принимает решение относительно размера сегмента. Существуют два ограничения размера сегмента. Первое: сегмент должен умещаться в пределах максимального размера поля полезной нагрузки IP – 2^{16} байт. Второе: каждая сетевая технология имеет так называемую максимальную единицу передачи (Maximum Transfer Unit - MTU) в которую и должен уместиться полный IP пакет, включающий в себя IP и TCP заголовки и данные.

Одновременно с передачей сегмента запускается таймер повторной передачи. Когда сегмент прибывает к получателю, TCP объект последнего отправляет сегмент, содержащий

подтверждение приема этого сегмента. Если таймер повторной передачи для сегмента срабатывает раньше времени прихода подтверждения, то данный сегмент ретранслируется.

1.7.7.3. Формат заголовка TCP

Заголовок сегмента TCP содержит следующие поля (рис. 8):

Порядковый номер идентифицирует первый байт данных в этом пакете, может также использоваться для указания первого номера в серии который будет использоваться в последующих передачах.

Номер подтверждения - содержит порядковый номер следующего байта данных, который ожидается на принимающем устройстве.

Флаги - разнообразная контрольная информация:

URG – срочные данные, указатель на срочные данные в этом случае указывает на их положение в сегменте

ACK – 1 значит, что поле подтверждения содержит правильную информацию

PSH – получатель должен доставить данные приложению как можно скорее

RST – соединение должно быть закрыто

SYN – используется для установки соединения, запрос на установку содержит SYN=1, ACK=0, ответ на этот сегмент содержит SYN=1, ACK=1.

FIN – отправитель сигнализирует о желании закрыть соединение. После отправки FIN машина может еще неограниченно долго продолжать получать информацию. SYN и FIN сегменты имеют порядковые номера, что гарантирует их обработку в правильном порядке.

Размер окна сообщает о том, как много байтов можно отправить, начиная с последнего подтвержденного байта. Этот размер может быть равен нулю.

1.7.7.4. Управление соединением

Установка соединения в TCP использует процедуру трехстороннего обмена. Для открытия соединения пассивная сторона исполняет примитивы LISTEN и ACCEPT, а другая открывает соединение в активном режиме, исполняя примитив CONNECT. Когда запрос на открытие соединения, прибывает к получателю, тот проверяет, есть ли процесс зарегистрированный для приема соединений по указанному порту. Если такой процесс существует, то соединение устанавливается (рис. 12 часть А).

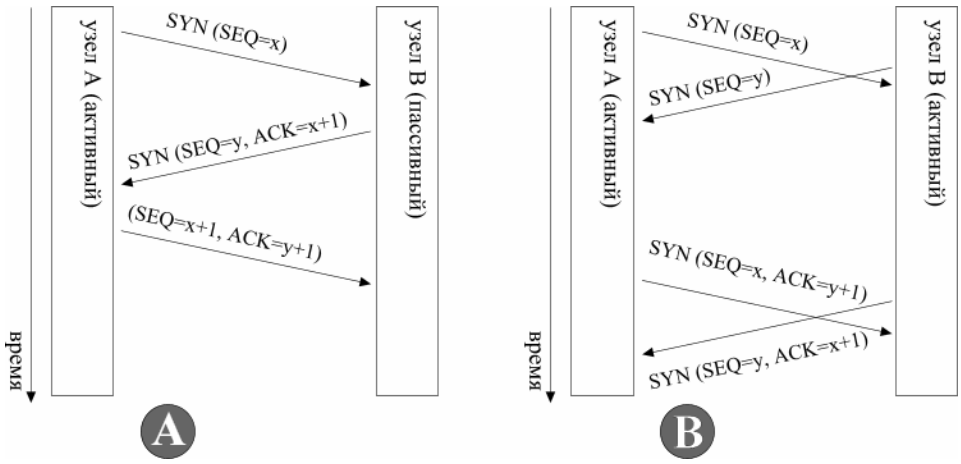

Рис. 12. Сценарии установки соединения TCP. Часть A.- нормальная процедура, часть B.- установка соединения при столкновении запросов.

Хотя TCP соединения полнодуплексные, закрываются соединения в каждом направлении по отдельности. В нормальной ситуации требуется четыре сегмента, чтобы полностью закрыть соединение (два FIN и два ACK). По каждому из симплексных соединений данные могут передаваться неограниченно долго. Для предотвращения появления полуоткрытых соединений используются несколько таймеров. Если ответ не сегмент FIN не приходит в течение двух максимальных периодов жизни пакета, отправитель сегмента FIN разрывает соединение.

1.7.7.5. Управление передачей

Управление окном в TCP не привязано напрямую к подтверждениям (как во многих протокола канального уровня). Упрощенный вариант схемы управления передачей приведен в примере на рис. 13.

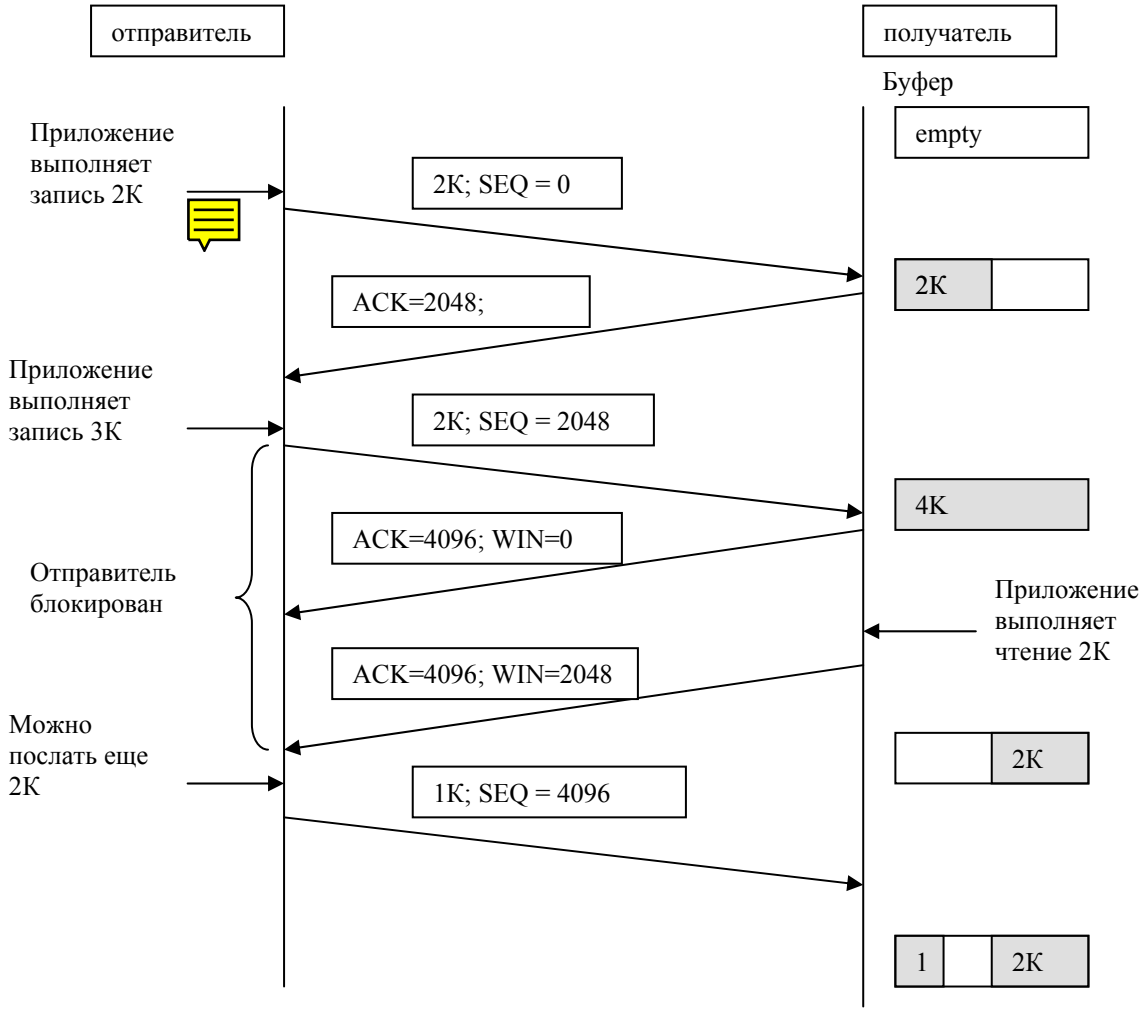

Рис. 13. Управление передачей данных и окном в TCP. Ось времени - сверху вниз. Закрашенный элемент буфера означает заполненный. Слева размещены комментарии.

Подтверждения, используемые протоколом TCP, являются кумулятивными, т.е. каждое подтверждение соответствует последнему полученному в правильном порядке байту потока. Приход подтверждения с номером N означает, что все байты [0…N-1] были успешно доставлены получателю.

1.7.7.6. Управление окном и тактика подтверждений

На ранних стадиях эксплуатации протокола было замечено, что часто в сетях возникали перегрузки при очень низком эффективном использовании ресурсов [8]. Были

выявлены три независимые причины этого: приложение вызывающие протокол для отправки незначительного количества информации, обновление окна TCP очень маленькими порциями и вызывающие ответные сегменты с малой полезной нагрузкой, и сам протокол посылающий слишком много подтверждений. Было предложено использовать так называемый алгоритм SWS [5] для решения проблемы с обновлением окна.

А. Алгоритм SWS у получателя, совмещен с задержкой подтверждения предписывает: обновление окна вместе с подтверждением отправляется только если освободившийся объем буфера превосходит определенную долю общего буферного пространства (30-50%) или 2-3 максимальных размера сегмента.

Б. Алгоритм SWS у отправителя предписывает: отправитель пытается приблизительно оценить размер буфера получателя, отслеживая максимальный размер окна открываемого получателем на данном соединении. *max(SND.WND[5])*. Используемое окно, данные в пределах которого могут быть отправлены, вычисляется *U=SND.UNA[6] + SND.WND – SND.NXT[7]*. Если D количество данных буферизованных протоколом, но еще не отправленных, то TCP отправляет данные если:

- Есть возможность послать сегмент максимального размера *min(D,U) >= Eff.snd.MSS[8]*
- установлен флаг PUSH и *[SND.NXT = SND.UNA] && PUSHED && D <= U* [*SND.NXT = SND.UNA*] условие налагаемое алгоритмом Nagle. [8].
- как минимум доля *Fs* от максимального окна может быть отправлена *[SND.NXT = SND.UNA] && min(D, U) >= Fs * Max(SND.WND)*
- установлен флаг PUSH и сработал таймер

В. Алгоритм Nagle: эффективен когда приложение передает малые объемы информации. Согласно этому алгоритму, если есть неподтвержденные данные *(SND.NXT > SND.UNA),* то TCP отправителя буферизует данные без отправки, несмотря на наличие флага PUSH, до момента, когда либо приходит подтверждение, либо становится возможным отправить сегмент максимального размера. Существует также таймер, предотвращающий ситуацию взаимной блокировки.

---

[5] SND.WND переменная в контрольном блоке каждого соединения. SND.WND указывает на «правую» границу окна в потоке данных.

[6] SND.UNA – переменная указывающая на последний неподтвержденный байт.

[7] SND.NXT – переменная содержащая указатель на следующий готовый к отправке байт данных.

[8] Eff.SND.MSS – реальный размер сегмента TCP.

Г. Алгоритм задержки подтверждений требует:

Подтверждение должно задерживаться не менее чем на *0.5 с.* Подтверждения задерживаются до тех пор, пока не возникнет возможность указать новое окно или данные отправляются в противоположном направлении. Таймер генерирует безусловное подтверждение каждые 200 мс. Кроме того, каждый второй сегмент должен подтверждаться. Также сегменты, буферизованные у получателя (прибывшие в неправильной последовательности) генерируют подтверждение мгновенно для включения механизма быстрой ретрансляции.

1.7.7.7. Управление потоком и предотвращение перегрузок

Если суммарная скорость поступления данных в на сеть превышает максимальную скорость, с которой сеть передает эти данные, то возникает перегрузка сети. Если состояние перегрузки длится в течении времени, достаточного для переполнения буферов в сети, то происходят потери данных в сети. Потерянные сегменты ретранслируются протоколом TCP и способствуют продолжению роста перегрузки. Алгоритм управления потоком транспортного протокола должен предотвращать возникновение перегрузки и способствовать выходу сети из состояний перегрузки, в случае его возникновения.

Перегрузка сети может быть предотвращена с помощью использования принципа сохранения пакетов[9] [2] – если сеть загружена полностью, то не передавать следующий пакет раньше, чем предыдущий покинет сеть (т.е. будет доставлен по назначению). TCP стремится следовать этому принципу, изменяя размер окна.

Схема определения наступления состояния перегрузки в TCP основывается на предположении, что причина всех потерь пакетов заключается в переполнении очередей маршрутизаторов, т.е. перегрузке сети. Таким образом, алгоритм управления потоком в TCP считает потери связанные с разрушением данных из-за шума на линии несущественными и реагирует на все потери как на признак перегрузки. Такой механизм является неэффективным, особенно в случае беспроводных соединений, таких как радио и спутниковые каналы, ИК приемопередатчики.

TCP оперирует двумя окнами, одно определяет размер буфера отправителя, другое (*CWND*) является результатом попыток протокола определить пропускную способность сети на данном соединении. Для отправки данных используется минимальное из двух окон.

[9] Принцип сохранения здесь применим к равновесному состоянию системы, когда в транзите находится количество данных равное полномстью открытому окну. В таком случе, в любой момент времени интеграл по плотности пакетов по закрытой петле ВК является константой.

Механизмы Slow start, congestion avoidance, fast retransmit, fast recovery [6] являются стандартом для современных реализаций TCP.

Slow start (замедленный запуск) применяется для постепенного увеличения скорости передачи информации после открытия соединения или потери пакета. Изначально размер окна CWND устанавливается равным одному сегменту. После получения каждого подтверждения *CWND* увеличивается на размер одного сегмента. Алгоритм медленного запуска отключается когда происходит потеря сегмента или достигается верхняя граница окна получателя. Время, затрачиваемое для полного открытия окна

$$t = RTT \times \log_2 W \qquad (1)$$

где W – размер окна в сегментах.

Congestion avoidance (предотвращение перегрузки) используется для определения наличия дополнительной пропускной способности. Обычно этот алгоритм действует совместно с алгоритмом медленного старта. Для фиксации момента перехода с медленного старта на предотвращение перегрузки используют переменную *SSTHRESH. SSTHRESH* инициализируется размером максимально возможного окна при открытии нового соединения. Медленный старт продолжается пока значение *CWND* не достигнет *SSTHRESH,* после этого режим роста *CWND* изменяется на линейный, а именно, для каждого подтверждения CWND увеличивается на значение (*1/ CWND*).

Fast Retransmit (быстрая ретрансляция). Использует последовательное получение подтверждения одного и того же сегмента, как признак потери, ретранслирует этот сегмент до срабатывания таймера ретрансляции.

Fast recovery (быстрое восстановление). После срабатывания механизма быстрой ретрансляции TCP переходит в режим предотвращения перегрузки, а не медленного старта. После прихода третьего дубликата подтверждения *SSTHRESH* устанавливается равной ½ *CWND,* и отсутствующий сегмент ретранслируется, *CWND* устанавливается равным *SSTHRESH+3* (количество сегментов покинувших сеть). Затем на каждое подтверждение *CWND* увеличивается на размер одного сегмента. Если размер *CWND* позволяет, передаются дополнительные сегменты. Когда приходит подтверждение, включающее в себя ретранслированный сегмент, *CWND* устанавливается равным *SSTHRESH* и TCP переходит в режим предотвращения перегрузки.

Описанные выше механизмы являются стандартной частью протокола TCP, их применение регламентируется RFC 1323 [6].

1.7.7.8. Управление таймерами

Объекты TCP концептуально используют несколько таймеров для выполнения своих функций. Наиболее важным из них является таймер повторной передачи (ТПП). ТПП перезапускается каждый раз при отправке сегмента. Если подтверждение приема сегмента не приходит до срабатывания таймера, то этот сегмент ретранслируется.

Определение суммарного времени затрачиваемого сегментом на достижение получателя и времени затрачиваемого подтверждением, чтобы вернуться к отправителю (*RTT*) само по себе достаточно сложно в реализации, еще сложнее даже зная это время правильно установить интервал таймера. Если ТПП установлен на слишком короткий интервал, то повторная передача произойдет еще до того как может прибыть подтверждение и это еще больше увеличит степень загрузки БС. Если же таймер установлен со слишком длительную задержку, то некоторую часть времени канал будет использоваться неэффективно. Кроме того, среднее значение измеряемого RTT и его дисперсии крайне динамично меняется с течением времени в зависимости от степени загрузки БС.

В протоколе TCP используется высоко динамичный алгоритм, который непрерывно корректирует значение ТПП, отслеживая время обращения сегментов в сети. Этот алгоритм функционирует следующим образом [2]: для каждого соединения TCP сохраняет значение наилучшей текущей оценки времени обращения сегментов в переменной *RTT*. При отправке сегмента устанавливается таймер для контроля повторного отправления и для измерения *RTT*. Если подтверждение прибывает до срабатывания таймера, то, измеряя время которое прошло до прихода подтверждения (*M*), TCP обновляет среднее значение RTT по формуле:

$$RTT = \alpha \times RTT + (1 - \alpha) \times M \qquad (2)$$

где α весовой коэффициент применимый к старому значению (с типичным значением 7/8).

Для того, чтобы значение *RTT* не искажалось из-за учета ретранслированных сегментов, Карном [7] был предложен алгоритм, который добавил к TCP следующее правило: не обновлять значение *RTT* по данным, относящимся к ретранслированным сегментам, вместо этого значение *ТПП* удваивается с каждой ретрансляцией. Этот принцип получил название “Exponential Backoff”.

1.8 Свойство самоподобия сетевого трафика

1.8.1. Традиционная методология: системы массового обслуживания

Традиционной методологией применяемой для изучения процессов происходящих в территориально распределенных сетях с большим количеством пользователей до недавнего времени являлась теория массового обслуживания [9].

Применение теории массового обслуживания к исследованию процессов, происходящих в сетях передачи данных, основывается на предположении о том, что моменты поступления требований в систему (моменты прихода пакетов в маршрутизатор) образуют пуассоновский поток с интенсивностью λ. Если распределение длительностей времени обслуживания также является пуассоновским, то применение методологии системы массового обслуживания M/M/1 позволяет определить среднее число требований в системе и с помощью теоремы Литтла связать это число со средней задержкой в системе в равновесном состоянии. Анализ систем массового обслуживания предполагает, что процесс, считающий число требований находящихся в системе является марковским процессом с непрерывным или дискретным временем, исследование которого и дает среднее число требований в системе в стационарном режиме. Таким образом, получают формулы для среднего числа требований и среднего времени пребывания требований в системе.

$$N = \frac{\lambda}{\mu - \lambda}, \text{ где } \mu \text{ - скорость обслуживающего прибора.} \qquad (33)$$

$$T = \frac{1}{\mu - \lambda} \qquad (34)$$

Если же предполагается произвольное распределение длительностей обслуживания требований, то такая система массового обслуживания обозначается M/G/1 и среднее время пребывания требования в обслуживающем приборе выражается формулой Поллачека-Хинчина:

$$W = \frac{\lambda \overline{X}^2}{2(1-\rho)}, \text{ где W – математическое ожидание времени пребывания требования в очереди, } \rho = \lambda / \mu = \lambda \overline{X}, \text{ если } X_i \text{ - длительность обслуживания i-го требования, а } \overline{X}^2 = E(X^2) \text{ момент второго порядка распределения длительностей обслуживания.}$$

(35)

$$T = \overline{X} + \frac{\lambda \overline{X}^2}{2(1-\rho)} \qquad (36)$$

Для сетей с коммутацией пакетов вообще и сетей передачи данных в частности долгое время применялись методы, основывающиеся на предположении о том, моментов прибытия пакетов или инициализации соединений имеют ограниченную дисперсию, поскольку аналитическая модель для таких систем хорошо известна [88, 9]. На основании этой модели рассчитывались основные характеристики сети, и производилось планирование ее ресурсов, в частности, объема буферного пространства.

1.8.2. Критика традиционных моделей трафика

С развитием сети Интернет и появлением возможности исследования большого количества экспериментального материала по трафику начали появляться свидетельства в пользу того, что трафик в сетях с коммутацией пакетов с большей точностью моделируется статистически самоподобными процессами [95] или, по крайней мере, не имеет экспоненциального распределения [90, 91]. Для самоподобного процесса представляющего пакетный трафик не существует естественного ограничения длительности всплеска. Всплески могут происходить в любом временном масштабе. Кроме того, множество типов обменов в современной сети инициируются и управляются операционной системой сетевых узлов автоматически, например обновления маршрутной информации, трафик протоколов SMTP, NNTP, что может приводить в глобальной синхронизации потоков в сети [92]. Такая ситуация является невозможной для пуассоновской модели, однако имеет место в реальности.

Итак, традиционный подход к моделированию трафика сети с коммутацией пакетов базируется полностью на предположении о неограниченности дисперсии процесса поступления пакетов, что позволяет применять теоретический аппарат марковских цепей для анализа таких систем. Однако развитие современных скоростных сетей передачи данных и мультимедиа информации с выраженной гетерогенной физической и логической инфраструктурой и большим набором разнообразных сервисов и генерируемых ими потоков, приводит к появлению существенно более сложных характеристик потоков. Трафик в таких сетях имеет явно выраженный всплесковый характер, не сглаживаемый усреднением по длительным временным промежуткам и большому числу потоков. Традиционный подход к моделированию трафика изменялся путем разработки все более сложных стохастических моделей: fluid flow models, Markov modulated Poisson processes, Markovian arrival processes, Batched Markovian arival processes, и т.д. Основной целью разработки всех этих моделей являлось сохранение возможности аналитического подхода к исследованию задач управления очередями и эффективностью системы. Однако, авторы обзора [93] отмечают, что статистические свойства стохастических моделей не совпадают со свойствами

экспериментальных данных по сетевому трафику. Причиной того, что стохастический анализ применялся в течении длительного времени, был недостаток эмпирических данных. Сейчас в экспериментальных данных и модельных экспериментах нет недостатка - получено большое количество наборов данных по трафику в территориально распределенных сетях, в ЛВС, для различных технологий канального уровня и для разнообразных протоколов прикладного уровня, для TCP/IP и ATM сетей.

В связи с этим, усилия ученых сейчас направлены на разработку новых способов анализа коммуникационных систем.

1.8.3. Исследования экспериментальных данных

Исследования большого объема данных по трафику с высоким разрешением по времени было проведено в целом ряде работ, перечисленных в [93]. Эти работы показали, что трафик реальных сетей с коммутацией пакетов обладает характеристиками самоподобия или мультифрактальности. Более того, в цитируемых работах показывается существенное отличие статистических характеристик трафика генерируемого традиционными стохастическими моделями и эмпирическими данными.

Подробный библиографический обзор на тему самоподобия и фрактальных свойств процессов в телекоммуникационных сетях, а также моделирования современных высокоскоростных сетей представлен в работе [93] и включает в себя аннотации 420 работ.

Большое число работ, упоминаемых в [93] посвящено нахождению физического или феноменологического объяснения фрактальной природы эмпирически изучаемого трафика. В этих исследованиях приводится связь фрактальной природы наблюдаемого трафика с синдромом бесконечной дисперсии или так называемыми распределениями с тяжелыми хвостами.

На настоящий момент, согласно авторам работы [93], не существует аналитического аппарата применимого процессам, характеризующимся свойством самоподобия.

Для того чтобы определить наличие характеристик самоподобия у больших последовательностей результатов измерения трафика применяются несколько статистических методов: R/S статистика, дисперсионно-временной анализ, метод спектральных областей и другие [94].

1.8.4. Определения моно- и мультифрактальных процессов

В работе [95] даются принятые на сегодня определения и признаки самоподобных (монофрактальных) и мультифрактальных процессов следующим образом:

Определение: процесс с непрерывным временем $Y = \{Y(t), t \in T\}$ является самоподобным (с параметром самоподобия H), если удовлетворяет условию:

$$Y(t) \overset{d}{=} a^{-H} Y(at), \forall t \in T, \forall a > 0, 0 \le H < 1 \quad (37)$$

(равенство в том смысле, что два процесса имеют одинаковые распределения). Коэффициент H, называется коэффициентом или параметром Хёрста и является мерой самоподобия.

Существуют также несколько признаков самоподобия.

Признак 1. Дана стационарная последовательность $X = \{X(i)\ , i \ge 1\}$, пусть

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X(i)\ \text{, k = 1, 2, …,} \quad (38)$$

является соответствующей X агрегированной последовательностью с уровнем агрегирования m. Последовательность $X^{(m)}(k)$ получена путем разбиения исходной последовательности X на непересекающиеся блоки размера m и усреднением по каждому блоку. Индекс k нумерует блоки. Если X является процессом приращения самоподобного процесса Y определенного в (37), т.е. X(i)=Y(i+1)-Y(i), то для всех целочисленных m:

$$X \overset{d}{=} m^{1-H} X^{(m)} \quad (39)$$

Стационарная последовательность $X = \{X_i, i \ge 1\}$ является самоподобной в точности, если она удовлетворяет выражению (38) для всех уровней агрегирования m. Также стационарная последовательность $X = \{X_i, i \ge 1\}$ считается асимптотически самоподобной, если условие (39) выполняется при $m \to \infty$.

Еще один признак самоподобия [95] дается через поведение абсолютных моментов.
Признак 2. Если определить:

$$\mu^{(m)}(q) = E \mid X^{(m)} \mid^q = E\left| \frac{1}{m} \sum_{i=1}^{m} X(i) \right|^q \quad (40)$$

Если X обладает свойством самоподобия, то $\mu^{(m)}(q)$ пропорционален $m^{\beta(q)}$, т.е. $\log[\mu^{(m)}(q)]$ линеен относительно $\log[m]$ для фиксированных значений q:

$$\log[\mu^{(m)}(q)] = \beta(q) \log[m] + C(q) \quad (41)$$

также $\beta(q)$ линейна по отношению к q, действительно, поскольку $X^{(m)}(i) \overset{d}{=} m^{H-1} X(i)$, то

$$\beta(q) = q(H-1) \quad (42)$$

таким образом, второй признак самоподобия в том, что моменты должны масштабироваться согласно (41) и $\beta(q)$ удовлетворять условию (42). Стационарная последовательность не может быть точно или асимптотически самоподобной, если ее среднее значение отлично от нуля.

Признак 3. самоподобия может быть обобщен для так называемых мультифрактальных процессов. Неотрицательный процесс X(t) является мультифрактальным, если логарифмы абсолютных моментов масштабируются линейно с логарифмом значения уровня агрегирования m. В отличие от самоподобного процесса здесь не налагается требований на линейность $\beta(q)$ относительно q. Самоподобные процессы, таким образом, являются частным случаем мультифрактальных.

1.8.5. Причина свойств самоподобия Интернет трафика

Причиной наличия свойства самоподобия, как показано в работах [98, 97], является распределение ON периодов имеющее тяжелый хвост (убывающий гиперболически).

Авторы [97] предлагают разделять объяснения причин самоподобия трафик в больших и малых временных диапазонах. Так самоподобие LAN и WAN трафика в масштабах порядка секунды и более объясняется и математически доказывается в [98]. Однако для полного понимания динамики сетевого трафика необходимо исследовать причины мультифрактальных характеристик трафика в масштабах менее одной секунды, в котором поведение трафика особенно динамично. В этих масштабах локальное поведение каждого из составляющих трафик потоков определяется используемым данным протоколом алгоритмом управления потоком. В общем же поведение суммарного трафика определяется взаимодействием нескольких реализаций одного или нескольких протоколов в совокупности с взаимодействием потоков в обслуживающем приборе.

1.8.5.1. Трафик с малым разрешением по времени

Предположим, что каждое индивидуальное соединение i отправляет пакеты с постоянной скоростью в период ON и не отправляет пакетов вообще в период OFF. Такое поведение ЛВС было показано в [100] как обмены типа "запрос-ответ". С другой стороны для территориально распределенных сетей, например Интернет, индивидуальные соединения ассоциируются с сессиями, устанавливаемые транспортным протоколом типа TCP. Каждая такая сессия инициализируется в некоторый случайный момент времени, осуществляет передачу в течение некоторого времени с постоянной скоростью, а затем отключается. Каждая сессия обслуживает то или иное приложение, реализующее свой протокол прикладного уровня, например FTP[DATA], TELNET, NNTP, SMTP, HTTP. Для периодов

длиной до 1 часа было показано, что процесс инициализации сессий в сетях является пуассоновским [91]. Для анализа трафика в большом временном масштабе только характеристики самих сессий важны для построения модели, в то время как индивидуальные процессы прибытия пакетов в рамках каждой сессии не рассматриваются. Для полного описания процесса счетчика общего трафика необходимо рассмотреть распределение длительностей сессий или ON/OFF периодов. Данные реальных наблюдений за потоками в LAN и WAN [91, 100] дают основание утверждать, что распределение длин ON/OFF периодов имеет бесконечную дисперсию и тяжелый хвост.

Положительная случайная величина U с функцией распределения F имеет распределение с тяжелым хвостом с индексом хвоста $\alpha > 0$, если выполняется:

$$P[U > y] = 1 - F(y) \approx cy^{-\alpha} \quad \text{при} \quad y \to \infty \qquad (45)$$

c - положительная конечная константа не зависящая от y. Выражение 45 описывает гиперболическое или степенное распределение, частным случаем которого является хорошо известное распределение Парето [91, 97]. В работе [98] доказывается тот факт, что распределение длительностей периодов ON/OFF спадающее гиперболически или по степенному закону (выражение 13) при больших значениях количества потоков и длительном времени наблюдения определяют наличие свойства LRD суммарного потока с параметром Хёрста

$$H = (3 - \alpha)/2 . \qquad (46)$$

Авторы доказали, что отклонение от среднего значения представляет собой дробный гауссовский шум (Fractional Gaussian Noise) - классический пример в точности самоподобного процесса. Таким образом, в больших временных масштабах (асимптотическое) самоподобие сетевого трафика является следствием свойств сессий обмена (ON/OFF периодов). А именно того, что длительности этих периодов имеют распределение с тяжелым хвостом. Поэтому в больших масштабах самоподобие трафика не зависит от инфраструктуры сети или ее протоколов, а определяется набором информационных объектов передаваемых по этой сети. Дальнейшее развитие этой гипотезы дано в [99], где показано, что причиной наличия тяжелых хвостов в распределении длительностей ON периодов является соответствующее распределение размеров объектов в WWW - доминантной информационной системы в сети Интернет. В [99] наблюдения за реальным HTTP трафиком и файлами отчета крупнейших WWW серверов показывают наличие связи между гиперболическим распределением размеров информационных объектов WWW и гиперболическим распределением длительностей HTTP сессий, трафик которых составляет более половины всего Интернет трафика (по результатам множества

исследований). Таким образом, трафик с малым разрешением по времени адекватно моделируется самоподобными или асимптотически самоподобными процессами согласно результатам [95, 97, 100].

1.8.5.2. Трафик с высоким разрешением по времени

Возвращаясь к высокодинамичному поведению сетевого трафика в малых временных масштабах, авторы [97] указывают на то, что причинами появления свойства LRD трафика в этих условиях являются свойства самих транспортных протоколов, управляющих потоками. Поскольку эти алгоритмы отвечают непосредственно за диспетчеризацию индивидуальных пакетов в рамках отдельных соединений, то вызываемые ими выраженные локальные нерегулярности трафика не являются связанными с самоподобными свойствами трафика в более грубом временном масштабе. Так при малом временном разрешении интернет трафик наиболее качественно описывается именно мультифрактальными процессами [96].

Надо отметить, что до настоящего времени неизвестна причина вызывающая такое поведение трафика.

1.8.6. Методы определения меры самоподобия

Для оценки меры самоподобия трафика ARTCP нами применены два метода - метод R/S статистики и метод Aggregated Variance. Оба метода относятся к числу хорошо изученных и успешно применяются в целом ряде работ [94].

1.8.6.1. Метод R/S статистики

Для последовательности $X = \{X_i, i \geq 1\}$, с частными суммами $Y(n) = \sum_{i=1}^{n} X_i$ и дисперсией $S^2(n) = (1/n)\sum_{i=1}^{n} X_i^2 - (1/n)^2 Y(n)^2$ R/S статистика (Rescaled adjusted range) определяется выражением:

$$\frac{R}{S}(n) = \frac{1}{S(n)}\left[\max_{0\leq t\leq n}(Y(t) - \frac{t}{n}Y(n)) - \min_{0\leq t\leq n}(Y(t) - \frac{t}{n}Y(n))\right] \qquad (47)$$

поскольку для FGN выполняется свойство:

$$E[R/S(n)] \sim C_H n^H \text{ при } n \to \infty \qquad (48)$$

то для определения коэффициента H по методу R/S применяется такой алгоритм:

Последовательность длиной N разбивается на K блоков размером N/K. Затем для каждого значения n вычисляется $R(k_i, n)/S(k_i, n)$. k_i начальный номер элемента

последовательности $k_i = iN/K + 1$, i = 1, 2, …, но таких, что $k_i + n \le N$. Таким образом получаем для значений n < N/K K оценок $R(n)/S(n)$, по мере роста n число получаемых оценок для каждого n уменьшается.

Если построить график зависимости $\log[R(k_i,n)/S(k_i,n)]$ от $\log[n]$, то параметр H вычисляется как коэффициент наклона прямой, аппроксимированной по точкам полученной зависимости, после отбрасывания точек соответствующих большим и малым n.

1.8.6.2. Метод Aggregated Variance

Если исходную последовательность $X = \{X_i, i \ge 1\}$ разбить на блоки размера m и произвести усреднение по каждому блоку k, то получим агрегатную последовательность:

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X(i) \qquad (49)$$

для последовательности значений m. Вычислим затем дисперсию $X^{(m)}(k)$ по всем блокам. Поскольку для дробного гауссовского шума:

$VarX^{(m)} \sim m^{(2H-2)}$, то для определения H пользуются следующим алгоритмом. Для определенного значения m разбить данные на N/m блоков размера m. Вычислить $X^{(m)}(k)$ для k = 1, 2, … , N/m и дисперсию:

$$VarX^{(m)} = \frac{1}{N/m} \sum_{k=1}^{N/m} [X^{(m)}(k)]^2 - \left[\frac{1}{N/m} \sum_{k=1}^{N/m} X^{(m)}(k) \right]^2 \qquad (50)$$

Повторяя эту процедуру для различных значений m (m << N) можно построить график зависимости $\log[VarX^{(m)}]$ от log[m]. По полученным точкам аппроксимируем прямую, и по углу ее наклона определяем коэффициент H.

1.9 Управление потоками в коммуникационных системах

1.9.1. Общие принципы

Управление потоками в коммуникационных сетях обозначает регулировку скорости отправки данных в сеть с целью достижения максимального использования ресурса сети и минимизации потерь данных.

В состав протокола TCP входит большое число разнообразных механизмов, которые отвечают за различную функциональность и находятся в определенном взаимодействии между собой в силу концептуально-необходимой связи или связи, возникшей при реализации. На рис. 14 приведена схема основных механизмов, входящих в состав самой распространенной реализации протокола TCP - Reno в составе 4.4 BSD UNIX [64].

Рис. 14. Набор функций и алгоритмов в составе стандартного протокола TCP.

Задачу управления скоростью передачи данных можно условно разбить на два компонента: не допущение переполнения принимающей стороны, и не допущение переполнения сети. Иными словами, целью системы управления потоком является выравнивание скорости передачи данных со скоростью их приема. Механизм контроля перегрузки отправляет данные в сеть не быстрее, чем сеть может их доставлять в место назначения и не быстрее, чем получатель может их обрабатывать.

Перегрузка в сети с коммутацией пакетов это такое состояние, при котором производительность системы падает в связи с переполнением ресурсов сети - коммуникационных каналов, процессорного времени и буферного пространства. Последствия перегрузки проявляются в увеличении времени доставки данных, падении эффективности использования сетевых ресурсов и так называемом сетевом коллапсе, когда процесс передачи полезных данных в сети полностью прекращается.

Изучение систем управления потоками и предотвращения перегрузок, очень важно для развития сетей. Из множества предложенных схем лишь несколько стали стандартами и получили широкое распространение: IBM Systems Networking Architecture (SNA) [10], Digital's Networking Architecture (DNA) [11], и модель TCP/IP [12]. Наиболее полная классификация различных методик управления потоком приводится в [13].

1.9.1.1. Перегрузка и методы ее контроля в сетях с коммутацией пакетов

Перегрузка является проблемой неэффективного совместного использования разделяемых ресурсов. В сети ресурсы распределены между всеми узлами, коммутаторами и каналами передачи данных. Любой из этих трех компонентов может стать узким местом в сети и вызвать ее перегрузку. С одной стороны сеть должна обслуживать все пользовательские запросы на передачу данных, которые, как правило, не могут быть предсказаны детерминистически и выражаются в виде всплесков по отношению к времени возникновения, скорости и объему передаваемой информации. С другой стороны все физические ресурсы сети имеют ограниченную мощность и должны быть управляемыми для достижения оптимального общего использования многими сессиями обмена данными. Более формальное описание состояния перегрузки сети можно получить рассмотрев производительность ее работы в зависимости от нагрузки (рис. 15). Мерой производительности может служить мощность, определенная как

$$P = W / RTT \qquad (4)$$

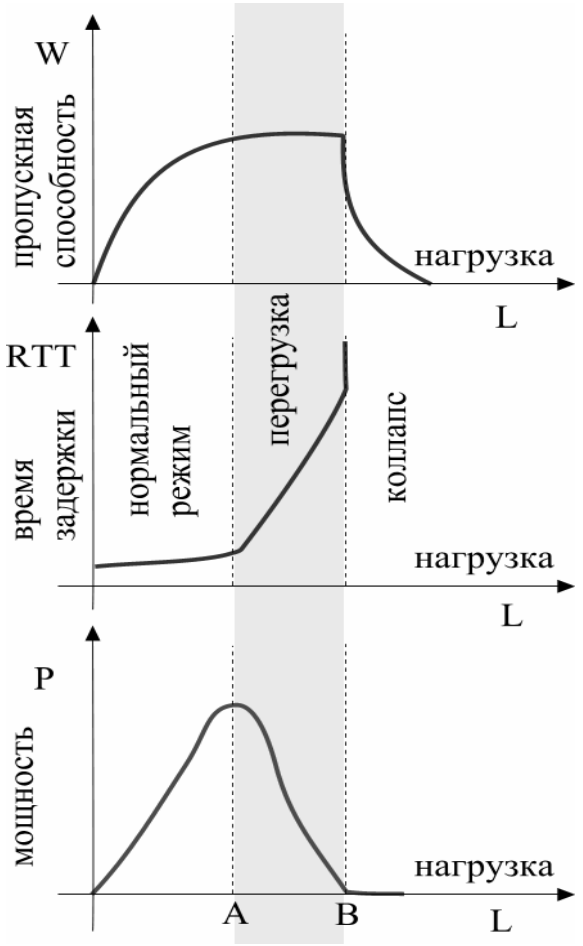

Рис. 15. Зависимость характеристик сети от нагрузки. Пропускная способность (вверху), время задержки (в середине), мощность сети (внизу).

После достижения сетью насыщенного состояния (точка *A*) пропускная способность перестает расти, а время ответа (*RTT*) продолжает, поскольку происходит заполнение буферов внутри сетевых устройств. Мощность сети достигает пика в точке *A* – это оптимальный режим работы сети.

Механизмы управления потоками должны удерживать режим работы сети на оптимальном значении нагрузки и обеспечивать возможность выхода из состояния перегрузки, если перегрузка все-таки произошла. Механизм управления потоками в сетях TCP/IP обеспечивается как самими конечными системами, в которых исполняются объекты протокола TCP, так и с помощью определенных механизмов управления очередями и диспетчеризации потоков в маршрутизаторах.

1.9.2. Управление задержкой сети

Задержка данных при их передаче по сети состоит из двух компонентов:

$$D = D_T + D_q \qquad (5)$$

D_T есть время, затрачиваемое пакетом на прохождение по элементам физической инфраструктуры линии связи зависящее лишь от скорости распространения электромагнитных волн в той или иной среде, а D_q определяется временем, которое пакет проводит в очередях маршрутизаторов, ожидая обслуживания. Например, для межконтинентального канала [6] протяженностью 5000 км задержка распространения сигнала составит 0.0167 с, в то время как задержка связанная с очередями в буферах маршрутизаторов может составлять несколько секунд в реальной системе. Для передачи мультимедиа информации очень важно чтобы значение задержки и ее дисперсия для индивидуальных пакетов были минимальными. Это требование выражается в требовании минимальности очередей в маршрутизаторах.

Поскольку TCP в процессе передачи целиком заполняет очередь маршрутизатора, то за счет компонента D_q общая задержка в сети существенно возрастает. Поэтому в сетях, где величина задержка актуальна, помимо управления скоростью средствами протокола транспортного уровня необходимо использовать механизмы, расположенные в маршрутизаторах [16].

1.9.2.1. Алгоритм RED

Поскольку средствами одного протокола TCP обеспечить минимальность средней длины очередей в сетевых устройствах невозможно, то в схеме управления трафиком должны быть задействованы сами сетевые устройства. Для реализации такого управления

используется алгоритм случайного раннего обнаружения (Random Early Detection - RED) [18].

Алгоритм RED применяется для предотвращения перегрузок в сетях передачи данных работающих по принципу коммутации пакетов. Маршрутизатор определяет близость состояния перегрузки, вычисляя среднюю длину очереди. Маршрутизатор оповещает соединения, проходящие через него, о состоянии перегрузки путем отбрасывания пакетов. Вероятность отбрасывания зависит от средней взвешенной длины очереди, пока ее значение находится в заданных пределах и становится равной единице при пересечении верхней границы.

Маршрутизатор применяющий алгоритм RED позволяет удерживать низкой среднюю длину очереди, при этом допуская принятие в очередь всплески пакетов. В режиме перегрузки вероятность оповещения определенного транспортного соединения, путем отбрасывания или отметки пакета, с целью уменьшения окна контроля потока для этого соединения, пропорциональна доле этого соединения в общей пропускной способности канала. Алгоритм RED предназначен для работы с транспортным протоколом, который управляет своей пропускной способностью, реагируя на потери пакетов, которые протокол считает признаком перегрузки, снижением своей скорости передачи. RED не допускает дискриминации неравномерных потоков и приводит к глобальной синхронизации многих соединений.

Однако применение алгоритма RED в маршрутизаторах усложняет их и увеличивает их стоимость.

1.9.3. Альтернативные схемы управления потоком

1.9.3.1. TCP VEGAS

Vegas представляет собой улучшенную реализацию протокола TCP по сравнению с распространенными реализациями Tahoe и Reno. Улучшения, предложенные в системе Vegas [14, 15], затрагивают в основном механизм быстрой ретрансляции. В алгоритме Vegas используется более точный способ определения необходимости быстрой ретрансляции потерянных сегментов. Этот алгоритм использует значения временной метки в заголовке сегментов.

Таким образом, Vegas лучше, чем TCP, реагирует на ситуации в которых происходит потеря более чем одного сегмента в пределах окна, и с которыми не может эффективно бороться стандартный механизм ускоренной ретрансляции [19, 20, 21]. Также в рамках этого механизма Vegas более точно управляет размером окна в процессе ретрансляции. Для реализации контроля перегрузки алгоритм Vegas сохраняет значение минимального RTT в

переменной *BaseRTT=min(BaseRTT, RTT)*. Ожидаемое значение пропускной способности определяется по формуле *Expected=размер_окна/BaseRTT*. Далее, реальная пропускная способность периодически вычисляется как количество байт переданных с момента отправки определенного сегмента и до прихода его подтверждения. Разность реальной и ожидаемой пропускной способности определяет выбор режима линейного увеличения или уменьшения окна.

Поскольку основной алгоритм Vegas полностью взят от TCP, то все основные недостатки, присущие TCP, характерны и для TCP Vegas.

1.9.3.2. TRUMP

Данный механизм разработан в докторской диссертации В.К. Тумей [22] и в него входит не только спецификация транспортного протокола, но и целая система управления потоком на сетевом уровне. Эта система включает в себя механизм измерения загрузки узлов с явной реверсивной обратной связью и собственный формат данных. Интересным аспектом системы TRUMP является то, что скорость отправки данных в сеть регулируется не скользящим окном, а значением разрешенной скорости, задаваемым сетевыми устройствами. К недостаткам системы нужно отнести тот факт, что системы с явной передачей контрольной информации существовали и ранее [23, 24, 25]. Попытки реально внедрить такие системы для сети Интернет не имели успеха, поскольку это потребовало бы введение дополнительных функций маршрутизаторов, в то время как TRUMP предусматривает не только существенное изменение сетевых устройств, но и полное отсутствие совместимости с архитектурой TCP/IP.

1.9.3.3. PP (Packet Pair Flow Control)

Управление потоком по методу Packet Pair Flow Control [26] представляет собой оригинальную идею измерения доступной пропускной способности сети. Схема предназначена для сетей с виртуальными каналами, которые изолируют транспортные потоки при помощи диспетчера квотированного обслуживания (ДКО) очередей [27, 28, 29]. Мотивация для создания такой схемы управления потоками в том, что предоставление интегрированных сервисов становится доминирующим направлением в современной сетевой парадигме. Согласно ей сети должны одновременно обслуживать стабильные потоки с постоянной скоростью передачи данных одновременно с потоками, работа которых характеризуется всплесками и неравномерностью. Первый тип обменов должен получать обслуживание от сети с гарантированным качеством, а второй тип трафика использовать оставшиеся ресурсы. Это основная концепция построения сетей с интеграцией сервисов [30]. Для ее реализации, межсетевые устройства должны применять один из алгоритмов

равноправного обслуживания очередей. Как правило это алгоритм взвешенной равноправной диспетчеризации WFQ [28].

Работа по РР в дополнение к механизму определения пропускной способности, описывает оценочную функцию для сглаживания полученных измерений, функцию управления, метод управления соединением, методы управления потоком и передачей. Устройство, где исполняется алгоритм ДКО, называют сервером выделения пропускной способности (СПС). В таком устройстве пакеты каждого отдельного транспортного потока помещаются в отдельную очередь на выходном интерфейсе. Каждая из этих очередей имеет тип FIFO и обслуживается диспетчером со скоростью

$$R = R_b / N \tag{6}$$

Где R_b - скорость самого интерфейса, а N – количество очередей на нем. Некоторые очереди могут иметь приоритет по отношению к другим. На каждом таком сервере соединение обслуживается с некоторой скоростью независимо от других соединений. Все СПС на пути соединения пронумерованы 1,2,3,…,n. Получатель подтверждает каждый пакет. Время обслуживания на i-том сервере $s_i(t)$, а мгновенная скорость обслуживания тогда

$$\mu_i = 1/s_i(t) \tag{7}$$

Время $s_i(t)$ есть интервал обслуживания всех остальных потоков, проходящих через данный интерфейс – период цикла обслуживания. Скорость отправки сегментов в сеть обозначена λ. Скорость наиболее узкого места в сети определена как

$$s_b(t) = \max(s_i(t) \mid 0 \le i \le n) \tag{8}$$

Где b – индекс ограничивающего СПС. $\mu_b(t) = 1/s_b(t)$ есть скорость ограничивающего СПС, которая изменяется с изменением числа активных потоков в сети. Авторы моделируют эти изменения как происходящие через дискретные интервалы 1, 2, …, k, k+1, …. Если число активных потоков N_{ac} велико, то можно ожидать, что $\Delta N_{ac} << N_{ac}$ за один интервал, т.е. значение $\mu_b(k)$ близко к $\mu_b(k+1)$. За счет этого определяется

$$\mu_b(k+1) = \max(\mu_b(k) + \omega(k), 0) \tag{9}$$

Где $\omega(k)$ - случайная переменная.

Отправляя данные в виде пар пакетов, в которых второй пакет мгновенно следует за первым и, измеряя временное расхождение в прибытии подтверждений, система РР определяет минимальную скорость на протяжении всего маршрута: μ_b и вычисляет необходимую на новом шаге скорость передачи $\lambda(k+1)$ в соответствии с принципом

сохранения длины очереди в ограничивающем СПС. В работе также доказывается независимость полученных измерений от всех СПС кроме ограничивающего.

Алгоритм PP не подходит для использования в сетях TCP/IP, поскольку в таких сетях отсутствует логическая изоляция потоков на сетевом уровне. Однако PP вполне может найти применение в сетях ATM, при наличии раздельной буферизации каждого из виртуальных каналов.

1.9.3.4. NETBLT

Протокол NETBLT [33, 34] разработан с цель предоставления пользователю сервиса аналогичного сервису протокола TCP, а именно надежной передачи большого объема информации с подтверждениями и контролем скорости потока. В протоколе NETBLT представлено несколько принципиально новых идей.

Во-первых, скорость потока регулируется заданной получателем величиной скорости. Передача осуществляется из предварительно сформированных банков информации. После передачи каждого банка (которые могут передаваться независимо), получатель подтверждает корректно полученные данные или запрашивает выборочную ретрансляцию потерянных пакетов. Также в зависимости от наблюдаемого уровня потерь NETBLT устанавливает новую скорость передачи.

Во-вторых, сама передача регулируется несколькими параметрами – частотой и длительностью всплесков, каждый из которых может состоять из нескольких пакетов. За счет этого повышается эффективность использования системных ресурсов, в частности уменьшается число прерываний используемых для диспетчеризации передачи. Этот подход можно применить и для протокола ARTCP.

Таким образом, протокол NETBLT способен обеспечить хорошие результаты при работе по каналам, характеризующимся большим значением RTT*BW. Однако NETBLT не свободен и от недостатков присущих TCP, например оба протокола интерпретируют потерю пакета как признак перегрузки, что неверно в общем случае.

1.9.3.5. Tri-S

Алгоритм Slow Start and Search (Tri-S) [35] основан на механизме замедленного старта протокола TCP. Авторы проанализировали недостатки замедленного старта и попытались устранить неприемлемо высокую амплитуду осцилляций размера окна, которая приводит к большому числу потерь и значительному увеличению компонента задержки связанного с буферизацией пакетов в сети D_q и к высокой дисперсии измерений времени *RTT*. Кроме того, авторы работы [35] показали, что схема управления потоком TCP позволяет обменам с

меньшим числом промежуточных узлов (и соответственно меньшим средним значением RTT) в среднем использовать большую долю пропускной способности, чем потокам с большим RTT.

В [35] предложено использовать так называемый нормализованный градиент пропускной способности (НГПС) и интерпретировать изменение в пропускной способности (ПС) как сигнал о перегрузке или наличии резерва ПС в сети. Алгоритм Tri-S вычисляет НГПС по формулам:

$$TG(W_N) = \frac{T(W_N) - T(W_{N-1})}{W_N - W_{N-1}} \quad (10)$$

$$NTG(W_N) = \frac{TG(W_N)}{TG(W_1)} \quad (11)$$

где $TG(W_N)$ - ненормированный ГПС, W_N - размер окна, $NTG(W_N)$ - НГПС при значении окна W_N, а $T(W_N)$-скорость, когда размер окна равен W_N. Интервал значений НГПС: [0,1]. В сети с небольшой нагрузкой $NTG \approx 1$, по мере приближения нагрузки к суммарной ПС сети, НГПС приближается к 0. Средняя скорость передачи данных вычисляется по времени обращения n-ного пакета:

$$T_N(W_N) = \frac{W_N}{RTT_N} \quad (12)$$

Сравнивая значения НГПС с двумя предельными значениями, алгоритм Tri-S принимает решение об уменьшении или увеличении размеров окна.

По результатам моделирования схема Tri-S позволяет улучшить показатель справедливости деления ПС по сравнению с TCP. Однако выбор предельных значений оказывает сильнейшее влияние на стабильность системы. Кроме того, как и стандартный TCP, схема Tri-S интерпретирует потерю пакета как признак перегрузки.

1.9.3.6. DUAL

Данный метод был предложен как способ устранения проблем с осцилляцией окна, наблюдаемых с алгоритмом замедленного старта [36]. DUAL использует изменение измеряемого RTT вместе с отслеживанием потерь пакетов для определения перегрузки.

Поскольку $D = D_T + D_q$, то $RTT_{\min} = 2D_T$.[10] В о время как

$$RTT_{\max} = D_T + D_q = D_T + Q_{\max} / R \quad (13)$$

[10] Минимальное RTT пропорционально величине задержки. Поскольку RTT равно сумме времени доставки

Где $Q_{\max}$ – максимальная длина очереди, а R – скорость обслуживания очереди. Алгоритм DUAL постоянно вычисляет минимальное и максимальное значение RTT и, сравнивая их с предельным значением RTT_i увеличивает или уменьшает скорость передачи.

$$RTT_i = (1-\alpha) \times RTT_{\min} + \alpha \times RTT_{\max} \quad (14)$$

α<1. Предлагается использовать значение 0.5. От алгоритма медленного старта DUAL отличается тем, что с периодичностью 2RTT он сокращает значение окна на 7/8, если $RTT > RTT_i$. Значение RTT_i обновляется с каждым измерением RTT. Кроме того, минимальное и максимальное RTT устанавливаются в ∞ и 0, соответственно, при срабатывании ТПП.

Проблема алгоритма DUAL в том, что если начальные значения $RTT_{\min}$ различны для различных соединений, то они никогда не достигнут равного деления ПС канала между собой, поскольку соединение имеющее наименьшее $RTT_{\min}$ будет позже остальных начинать сброс значения окна.

1.9.3.7. Выводы

Каждый из этих методов обладает определенными недостатками и не применяется в стандартных протоколах. Главный недостаток Vegas, Tri-S, DUAL в том, что в них используется протокол TCP, который реагирует на потерю сегмента снижением скорости, а в отсутствии потерь линейно увеличивает нагрузку на сеть, приводя к переполнению буферов. Схема TRUMP предполагает расширение функциональности маршрутизаторов механизмом, уведомляющим источники о перегрузке в явном виде, что означает отход от парадигмы сетей с пакетной коммутацией. PP предлагает эффективный способ измерения загрузки сети, однако неприменимый для TCP/IP, поскольку предполагает работу в сети с виртуальными каналами на сетевом уровне.

1.9.4. Принципы AIMD и STA

Большинство алгоритмов управления потоками различаются способом определения наступления состояния перегрузки и реагируют на изменения состояния сети путем аддитивного (линейного) увеличения нагрузки и мультипликативного сброса, в случае определения наступления перегрузки. В литературе такой подход получил название Addititve Increase and Multiplicative Decrease (AIMD). Обоснование применения принципа AIMD изложены в работе [2].

данных и времени возврата подтверждения, то коэффициент пропорциональности равен 2.

Пусть загрузка сети измеряется через некоторые интервалы фиксированной длины, например среднее время D_T. $D_T \approx RTT/2$, когда сеть не перегружена. В этом случае модель сети проста: $L_i = N$, где L – показатель загрузки (например, средняя длина очередей), а N – константа. Если же сеть подвержена перегрузке, то модель изменяется. Теперь

$$L_i = N + \gamma L_{i-1} \quad (15)$$

где γ определяет влияние трафика за прошлый период, а также последствия потерь, т.е. наличие в сети ретранслированных данных. Когда сеть находится в состоянии перегрузки γ велико и длина очередей растет экспоненциально, поскольку из $L_i \approx \gamma L_{i-1}$ следует $L_n = \gamma^n L_0$. Вследствие этого система может стабилизоваться, только если источники трафика в сети могут уменьшать скорость его генерации также экспоненциально. Таким образом, получается, что при перегрузке размер окна *W* должен изменяться по закону:

$$W_i = dW_{i-1} \quad (16)$$

где 0 < d < 1. Это и есть мультипликативный сброс, который превращается в экспоненциальный, если перегрузка продолжается в течение нескольких периодов обновления окна D_T.

В случае отсутствия перегрузки, сеть никак не сообщает соединениям о наличии дополнительных ресурсов, в связи с этим, в отсутствие сигналов о перегрузке, соединения должны увеличивать нагрузку, до тех пор, пока потеря пакет не сообщит о начале перегрузки. Экспоненциальный рост всегда приводит к новой перегрузке, поэтому автор [2, 44, 45] рекомендует линейный рост или аддитивное увеличение:

$$W_i = W_{i-1} + u,\ u << W_{\max} \quad (17)$$

где $W_{\max} = D_T \times BW$ - произведение задержки передачи и пропускной способности канала.

Глава 1. Постановка задачи

1.1. Недостатки протокола TCP

К наиболее существенным недостаткам протокола TCP в области управления потоками относится следующее:

1. К основному недостатку протокола TCP следует отнести проблему не столько реализации протокола, сколько самой логики функционирования. Так механизмы коррекции ошибок и управления потоком в TCP, реализующие различные функции, оказались связанными. Причиной этого является интерпретация протоколом факта потери сегмента (и, соответственно, факта срабатывания механизма коррекции ошибок) как признака перегрузки сети. Вследствие этого TCP реагирует на любую потерю данных снижением скорости передачи. Результатом является крайне низкая эффективность применения протокола TCP для систем, где потери данных происходят не только вследствие перегрузки – всех систем, где существует ненулевая вероятность потери данных на физическом-канальном уровнях. Это, в частности, все виды беспроводных сетей: спутниковые, сотовые, оптические.
2. Применяемый в TCP метод AIMD предписывает постоянное линейное увеличение нагрузки на сеть с целью определения момента начала перегрузки. Вследствие этого сеть постоянно находится либо в состоянии перегрузки, либо в состоянии выхода из нее. Это отрицательно сказывается на соединениях в виде увеличенного среднего RTT, большой дисперсии измеряемых значений RTT, постоянном наличии потерь, с помощью которых сеть сигнализирует о начале перегрузки.
3. В большинстве условий протокол TCP осуществляет отправку данных очень неравномерно. Фактически все данные в пределах окна отправляются за небольшое время в виде всплеска пакетов [46]. Неравномерный режим отправки пакетов приводит к повышению числа потерь. Поскольку средняя длина очередей в сетевых устройствах близка к максимальной, то вероятность потери сегментов в пределах всплеска повышается.
4. Механизм управления передачей TCP зависит от потока подтверждений в обратном направлении, прибытие которых заставляет перемещаться окно и разрешает отправку новых сегментов. Такой режим называют синхронизированным по подтверждениям. Его эффект не заметен для симметричных каналов, однако для каналов, чьи свойства в различных направлениях различны синхронизация по подтверждениям ведет к ограничению эффективности использования ресурсов. Это относится к таким

асимметричным системам, как «спутниковый/наземный каналы», «кабельный модем/коммутируемое соединение».

Поскольку недостатки TCP широко известны, то множество работ было посвящено созданию отдельного транспортного протокола для асимметричных инфраструктур [51] или беспроводных сетей [47, 48, 49, 50]. Большинство предлагаемых протоколов не являются совместимыми с TCP и требуют наличия агентов-посредников транспортного уровня. Тем самым нарушается основной постулат Интернет заключающийся в том, что сеть должна обеспечивать беспрепятственную связь на транспортном уровне между непосредственными участниками обмена. Также излишне усложняется вся структура сети.

Наша работа была направлена на создание нового протокола, названного ARTCP, который мог бы стать эффективной заменой существующему TCP, устранив важнейшие недостатки последнего.

1.2. Цель работы

Итак, задача данной работы в создании нового механизма управления потоком для транспортного протокола в архитектуре сети с коммутацией пакетов (TCP/IP). Новый алгоритм управления потоком должен исправить основные недостатки, присущие протоколу TCP приведенные выше. При этом новый протокол должен сохранить все внешние свойства и интерфейсы существующего транспортного протокола. Новый протокол должен существовать в типичной среде исполнения транспортного протокола Интернет и предоставлять пользователю тот же сервис.

Для изучения характеристик протокола ARTCP и сравнения их с TCP, в рамках данной работы необходимо разработать программную имитационную модель, воспроизводящую основные характеристики сети, которые и определяют функционирование в ней транспортного протокола: задержку, мультиплексирование, потери и ошибки передачи.

На созданной имитационной программной модели предполагается провести ряд модельных экспериментов для определения основных параметров ARTCP в различных условиях и сравнения их с TCP. Поскольку, как показывает множество исследований, трафик в TCP/IP сетях обладает свойством самоподобия, то проверка ARTCP трафика на наличие у него свойства самоподобия также является целью модельного эксперимента.

1.3. Формальная модель системы

Формализуем модель на которой будем изучать протокол ARTCP. Имеется сеть, в топологическую схему которой входят несколько узлов, два маршрутизатора и набор каналов соединяющих узлы (рис. 16). На каждом узле исполняется объект протокола ARTCP. Узлы объединены в две локальные вычислительные сети (ЛВС), каждая из которых

подключена к одному маршрутизатору. Маршрутизаторы связывают локальные сети, передавая трафик по каналу с малой ПС и большим значением задержки.

Каждый из узлов в одной из ЛВС отправляет сегменты с заданным межсегментным интервалом, который и определяет скорость передачи. Эта скорость задается алгоритмом управления потоком ARTCP. Предполагается, что источники потоков всегда имеют информацию для отправки. Источники и приемники трафика находятся в разных ЛВС. Приемники трафика ARTCP отправляют подтверждения в противоположном направлении в виде сегментов, не содержащих данных.

Задача маршрутизатора в том, чтобы осуществлять передачу сегмента по адресу получателя в его заголовке. В буфере маршрутизатора R1 организуется FIFO очередь сегментов, которые отправляются далее к маршрутизатору R2. Очередь имеет конечную длину. Сегмент, поступающий на выходной интерфейс, помещается в очередь, если она может вместить этот сегмент, иначе сегмент теряется. Очередь маршрутизатора R1 обслуживается со скоростью канала, соединяющего маршрутизаторы.

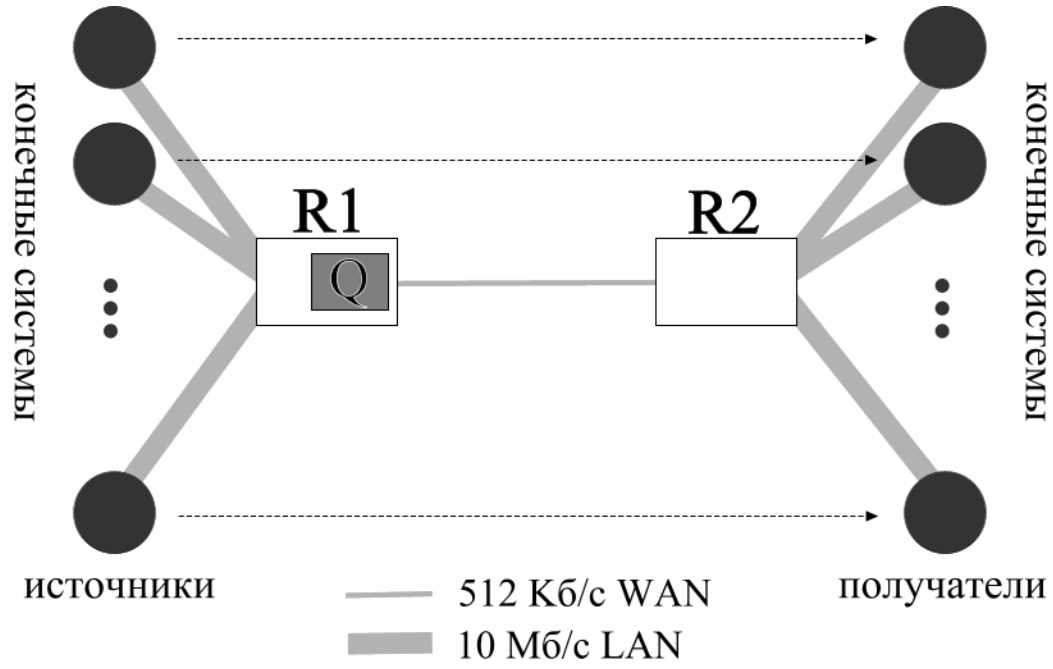

Рис. 16. Формальная модель исследуемой сети. Стрелками указано направление передачи данных.

Мы рассматриваем следующие характеристики каналов: ПС, задержку передачи и вероятность битовой ошибки. Пропускная способность канала определяет скорость поступления битов в канал, задержка передачи характеризует длительность интервала между поступлением определенного бита в канал и появлением его из канала. Вероятность битовой

ошибки BER определяет вероятность потери сегмента как $1-(1-BER)^S$ в зависимости от вероятности ошибки при передаче.

1.4. Основные характеристики протокола

В работе рассматриваются следующие основные характеристики протокола:

1. Относительное число потерь сегментов (отношение числа потерянных к общему числу отправленных сегментов)
2. Коэффициент использования пропускной способности каналов $U = \frac{число\_принятых\_битов}{(скорость\_канала) \times время}$ (отношение числа успешно принятых битов к максимально возможному их числу).
3. Коэффициент равноправия разделения ресурсов $F = (\sum_{i=1}^{n} b_i)^2 \Big/ n \times (\sum_{i=1}^{n} b_i^2)$, где b_i есть доля пропускной способности, занятая i-м соединением.
4. Средняя длина очереди Q в маршрутизаторе R1.

Сравнение ARTCP и TCP производится по этим основным характеристикам.

Приведенные выше характеристики протокола являются стандартными и используются во многих исследованиях протоколов [73].

1.5. Сеть как самоорганизующаяся система

Определение самоорганизующейся системы в смысле Г. Хакена следующее: система с большим числом степеней свободы, под воздействием сложных внутренних сил, переходящая в состояние с существенно меньшим числом степеней свободы: несколькими параметрами порядка, называется самоорганизующейся [101]. Хаотическое поведение системы на уровне отдельных компонентов приводит к детерминированному поведению ее средних величин.

Рассматриваемая нами модель сети представляет собой набор объектов транспортного протокола ARTCP и объектов сети. Каждый объект протокола может управлять собственной скоростью передачи потока, как система с обратной связью. Взаимодействие между всеми потоками осуществляется через общие для этих потоков объекты топологии. Использование случайной переменной алгоритмом каждого из протоколов, вместо увеличения числа степеней свободы, приводит к упорядочению суммарного действия всех элементов системы. Это упорядочение выражается в том, что суммарная скорость всех потоков протокола ARTCP выравнивается со скоростью совместно используемого этими потоками канала. Таким образом, сеть с функционирующими в ней соединениями транспортного протокола ARTCP, является самоорганизующейся системой.

Все вышесказанное позволяет сформулировать это утверждение в следующем виде:

Теорема 1: Рассматриваемая сеть с соединениями протокола ARTCP, является самоорганизующейся системой в смысле Г. Хакена.

Глава 2. Алгоритм ARTCP

Предлагаемый в данной работе новый протокол Adaptive Rate Transmission Control Protocol (ARTCP) заимствует некоторые механизмы от протокола TCP. В ARTCP полностью пересмотрен алгоритм управления потоком, который и отличает его от TCP. Вместе с тем, предлагаемый протокол может обеспечивать совместимость с TCP.

2.1. Аспекты новизны протокола ARTCP

ARTCP отличается от стандартного TCP тем, что сегменты отправляются в сеть не в виде всплеска в пределах окна, а разделенные временными промежутками, длительность которых определяется текущим значением скорости. Скорость потока регулируется не размером переменного окна, а значением скорости, изменением которой осуществляется адаптация алгоритма в соответствии с условиями. Механизм скользящего окна в ARTCP применяется только для предотвращения переполнения буферов получателя. На размер окна в ARTCP не наложено никаких ограничений ни в одном из режимов работы. Окно ограничено лишь наличием буферного пространства у получателя, поэтому для получателя рекомендуется устанавливать окно на максимальный размер.

В случае, когда получатель ограничен в буферном пространстве, ARTCP будет вести себя как обычный протокол TCP, ограниченный окном. Таким образом, протокол ARTCP сочетает в себе метод скользящего окна для регулирования управления потоком из конца в конец и метод контроля скорости для подстройки под ПС промежуточных узлов соединения.

Вторым важнейшим отличием ARTCP является то, что в качестве сигнала о состоянии перегрузки или наличия дополнительных ресурсов в сети используются темпоральные характеристики потока – измерение скважности[11] потока у получателя и изменения времени RTT. Потеря пакета никак не отражается на работе ARTCP кроме осуществления ретрансляции потерянного пакета механизмом коррекции ошибок. Как и в TCP о потери пакета сообщают два возможных события: срабатывание ТПП или последовательное получение двух[12] подтверждений одних и тех же данных.

Таким образом, по сравнению со своим предшественником, ARTCP обладает следующими преимуществами:

1. ARTCP не требуется доводить сеть до состояния перегрузки, чтобы определить доступную долю ПС, поэтому исключены потери пакетов связанные с этим процессом.

---

[11] Скважность – длительность межпакетных (сегментных) временных интервалов.

[12] В TCP механизм ускоренной ретрансляции активируется получением трех дубликатов подтверждения.

2. ARTCP существенно снижает требования к межсетевым устройствам. Во-первых, для нормального функционирования данного протокола требуется меньший объем буферного пространства, чем для TCP, поскольку режим передачи является сглаженным. Во-вторых, ARTCP не требует и не зависит от наличия каких либо механизмов диспетчеризации или управления очередями, таких как RED или WFQ.
3. ARTCP не интерпретирует потерю пакета как признак перегрузки сети, используя вместо этого темпоральные характеристики потока. Поэтому ARTCP должен особенно эффективно работать в системах беспроводной связи, там где использование TCP неэффективно.
4. В отличие от TCP новый протокол не полагается целиком на поток подтверждений в обратном направлении для синхронизации процесса передачи. В связи с этим возможна реализация ARTCP с меньшей частотой подтверждений, которая не ограничивала бы скорость в асимметричных системах.

2.2. Эвристика в основе алгоритма ARTCP

Анализ работ в области транспортных протоколов и в частности механизма PP (см. часть 1.8 введения) позволил заключить, что недостатки протокола TCP весьма существенны и являются следствием самого алгоритма, лежащего в основе протокола TCP. Поэтому модификация TCP без замены его основных алгоритмов не может привести к существенному улучшению характеристик протокола.

Поэтому в этой работе было решено создать новый алгоритм транспортного протокола, остающийся, однако, полностью совместимым с архитектурой TCP/IP.

Для того чтобы устранить недостатки, свойственные TCP, необходимо было найти способ получения информации о состоянии сети, отличный от применения в этих целях потерь сегментов. Наиболее хорошо на роль индикатора состояния сети подходят временные характеристики потока: время RTT и межсегментные интервалы. С использованием межсегментных интервалов можно также определить долю ПС канала. Для этого требуется запоминать межсегментные интервалы потока у отправителя и измерять их у получателя. Сравнение значений интервалов характеризует состояние сети, а минимальное значение измеряемых интервалов у получателя позволяет определить доступную долю ПС.

Таким образом, получается следующая схема: установка скорости потока отправителем посредством тщательной диспетчеризации сегментов, измерение скорости прибытия потока у получателя и передача этой информации отправителю вместе с остальной контрольной информацией. Разность старого и нового значений скорости отправки потока ARTCP на каждом шаге задается случайной переменной, однако, при наличии сигнала о

перегрузке сети вероятность снижения скорости превышает вероятность ее увеличения на каждом новом шаге.

2.3. Параметры и переменные

Пусть τ временной интервал между последовательными трансляциями пакетов. Задача функции диспетчеризации сегментов в том, чтобы задерживать отправку очередного сегмента на время τ_s после начала передачи предыдущего сегмента. Обозначим все переменные, относящиеся к отправителя индексом s, и r – относящиеся к получателю. Итак, τ_s временной интервал между моментами начала отправки в сеть сегмента i+1 и i-го, а τ_r интервал между последовательно прибывшими к получателю сегментами.

Пусть скорость канала связи, непосредственно к которому подключен отправитель R_{ls}, тогда время уходящее на отправку одного сегмента (с момента начала передачи до момента ее окончания) $t_{ls} = S / R_{ls}$, где S – размер передаваемого сегмента. Очевидно, что максимально возможная скорость потока

$$R^{\max}{}_s = R_{ls} = S / \tau^{\min}{}_s \qquad (18)$$

Минимальное значение межсегментного интервала в этом случае будет $\tau^{\min}{}_s = t_{ls}$, когда пакеты отправляются в сеть без задержек с максимальной скоростью канального уровня. Путем изменения τ_s в пределах $[\tau_s{}^{\min}, \infty)$, ARTCP может контролировать скорость потока в пределах $[R_{ls}, 0)$. Иллюстрация принципа управления скорость приведена на рис. 18.

Задержка отправки готовых сегментов производится с помощью системного таймера. Например, для полного использования ПС канала в 512 Кб/с при размере сегмента в 1000 байт каждый сегмент необходимо отправлять с задержкой τ_s=0.015625 с, чтобы скорость потока составила 64 пакета/с.

В таблице 1. приведен список параметров и переменных используемых алгоритмом управления потоком протокола ARTCP.

S	Размер кадра канального уровня содержащего сегмент
τ_S	Временной промежуток между последовательными трансляциями сегментов (от первого бита до первого бита)
τ_R	Временной промежуток, измеренный между последовательными моментами прибытиями сегментов (от последнего бита до последнего бита)
$R_S(t)$	Скорость потока, устанавливаемая отправителем
$R_R(t)$	Скорость потока, вычисляемая получателем
$R_e(t)$	Оценка доступной пропускной способности в момент t-RTT
R'_S	Первая производная скорости по времени, устанавливается отправителем

R_{pe}	Значение оценки доступной пропускной способности в момент i-1
A_C	Площадь области компенсации
SSGR	Параметр экспоненциального роста R'_S в режиме SS
RTT	Временной промежуток между моментом отправки сегмента и моментом прихода его подтверждения
ERTT	Взвешенное скользящее среднее RTT
SmoothedRe	Взвешенное скользящее среднее Re
MaxERTT	Максимальное значение сглаженного RTT
MinERTT	Минимальное значение сглаженного RTT
MDFACTOR	Коэффициент, используемый при мультипликативном снижении $R_S(t)$
R_s^{amd}	Значение скорости отправки данных непосредственно на выходе режима MD1
R_s^{bmd}	Значение скорости отправки данных непосредственно перед входом в режим MD1
R_e^{amd}	Значение оценки доступной пропускной способности в режиме MD1
speedup	Коэффициент, определяющий вероятность увеличения скорости в режиме FT
Slowdown	Коэффициент, определяющий вероятность снижения скорости в режиме FT
Midpoint	Точка отсчета скорости на каждом шаге в режиме FT
sR_s	Взвешенное скользящее среднее значение R_s
K	Коэффициент критерия осуществления перехода 6
BER	Bit error ratio - резидентная вероятность инвертирования бита на линии
$R_s^{\min}$	Начальное минимальное значение скорости, с которого начинается рост в состоянии SS

Таблица 1. Переменные и параметры модели ARTCP.

2.4. Формат сообщения

Формат сообщений используемых ARTCP может в точности совпадать с форматом пакета TCP (рис. 8). Стандарт TCP [4] предусматривает наличие дополнительных полей в заголовке сегмента между стандартным заголовком и полем данных. ARTCP может передавать дополнительную информацию в этих полях, что будет гарантировать совместимость с TCP. Всего протокол ARTCP требует использования лишь двух новых полей: значения предыдущего порядкового номера “PS” в направлении от отправителя к получателю и значения скважности “TI” в направлении от получателя к отправителю. Значение “TI” можно передавать в виде опции временной метки [6], а значение “TI” требует поля, позволяющего поместить порядковый номер сегмента.

2.5. Структурная схема ARTCP

В протоколе ARTCP полностью переработаны все механизмы управления потоком. Механизм коррекции ошибок передачи в ARTCP не влияет на скорость передачи. От TCP сохранены оконный механизм для управления загрузкой получателя, алгоритмы определения RTT и установки таймера ТПП. Признаком потери сегмента служит срабатывание ТПП или приход двух последовательных подтверждений одного сегмента. Алгоритм управления

скоростью включает в себя: функции диспетчеризации сегментов, измерения скорости и адаптации скорости (рис. 17). Далее рассмотрим эти функции подробно.

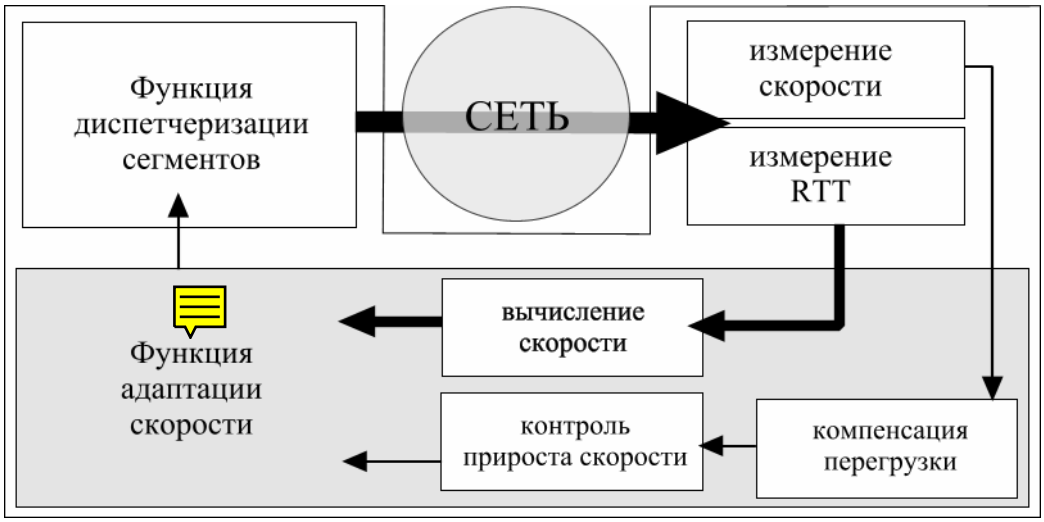

Рис. 17. Функциональная схема механизма управления потоком ARTCP. Жирная стрелка вверху обозначает направление передачи данных. Более тонкие стрелки: направление передачи управляющей информации.

В задачи функции диспетчеризации сегментов входит отправка сегментов в сеть со строго заданной скоростью, которая выражается в значениях межсегментных временных интервалов.

Функция измерения скорости определяет скважность потока поступающего к получателю и информирует отправителя о темпоральных параметрах прибывающего потока. Кроме того, отправитель сам производит измерение времени RTT (в рамках стандартной функциональности протокола TCP).

Значения скважности потока измеренной получателем и времени RTT измеренного отправителем поступают на вход функции адаптации, которая определяет новое значение скорости отправки потока в соответствии с полученными на вход значениями и своим состоянием в этот момент времени.

ARTCP использует два признака начала перегрузки сети, когда средняя скорость прибытия запросов сравнивается со средней скоростью обслуживания и началом роста очереди: начало роста RTT и стабилизацию $R_r(t)$ при увеличении $R_s(t)$. Получатель ARTCP в сегментах с подтверждениями указывает значение скорости прибытия потока. Получая подтверждение сегмента спустя время RTT после его отправки, источник ARTCP получает информацию о значении скорости, с которой поток, содержащий этот сегмент, прибыл к получателю и использует $R_r(t)$ в качестве оценки $R_e(t)$ ПС сети.

2.5.1. Диспетчеризация сегментов

Диспетчер сегментов отправляет сегменты на линию через строго заданные межсегментные временные интервалы. Значение интервалов определяются скоростью отправки потока, которая задается функцией адаптации.

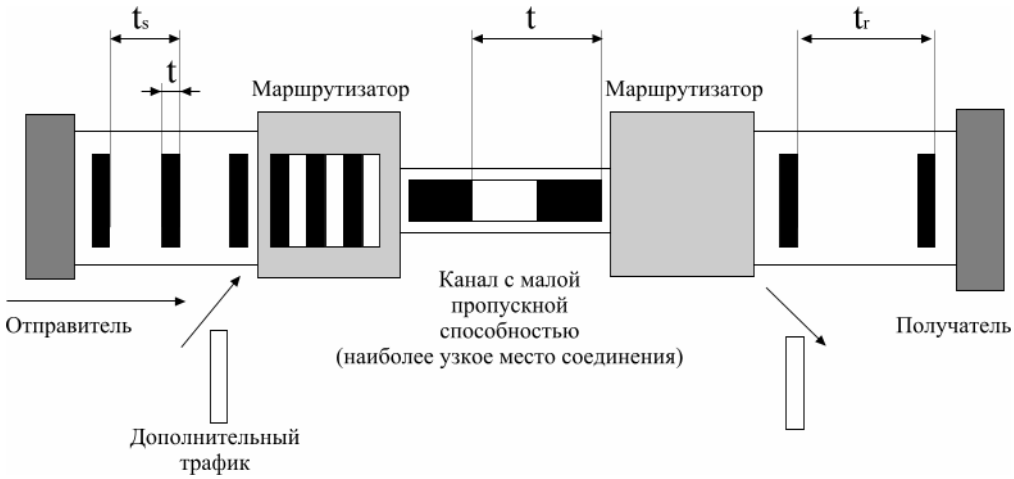

Рис. 18. Диспетчеризация сегментов и измерение скорости потока алгоритмом ARTCP. Черные прямоугольники обозначают сегменты одного соединения. Различие ширины прямоугольников отражает различие скоростей каналов. Меньшей скорости соответствует более длительное время передачи.

2.5.2. Измерение скорости

Очевидно, что скорость приема потока получателем не может быть выше скорости обслуживания потока на участке с наименьшей ПС, через который проходит соединение. Таким образом, зная скорость прибытия потока к получателю, можно определить доступную пропускную способность сети. Для корректного измерения скорости необходимо не учитывать выпавшие из потока, т.е. потерянные сегменты, а также сегменты, доставляемые сетью в измененном порядке. Для выполнения этого условия в поле “PS” каждого отправляемого сегмента записывается порядковый номер (или смещение) от предыдущего сегмента.

Получив сегмент i, получатель вычисляет разницу текущего времени и времени прибытия предыдущего (j) сегмента τ_r и в случае, если поле “PS” i-го сегмента содержит значение j, помещает $R_r = S / \tau_r$ в поле “TI” подтверждения следующего в противоположном направлении (рис. 18). Получатель извлекает значение поля “TI” из получаемых подтверждений и использует его для управления скоростью передачи.

2.5.3. Функция адаптации

Описание алгоритма работы функции адаптации, непосредственно осуществляющего управление потоком ARTCP, было дано в работах [52, 53, 54, 76].

Способ управления потоком должен достичь, во-первых, быстрой реакции потока на изменяющиеся условия соединения и, во-вторых, стабилизировать скорость передачи, когда она равна максимальной скорости сети. Алгоритм управления потоками ARTCP функционирует в нескольких режимах.

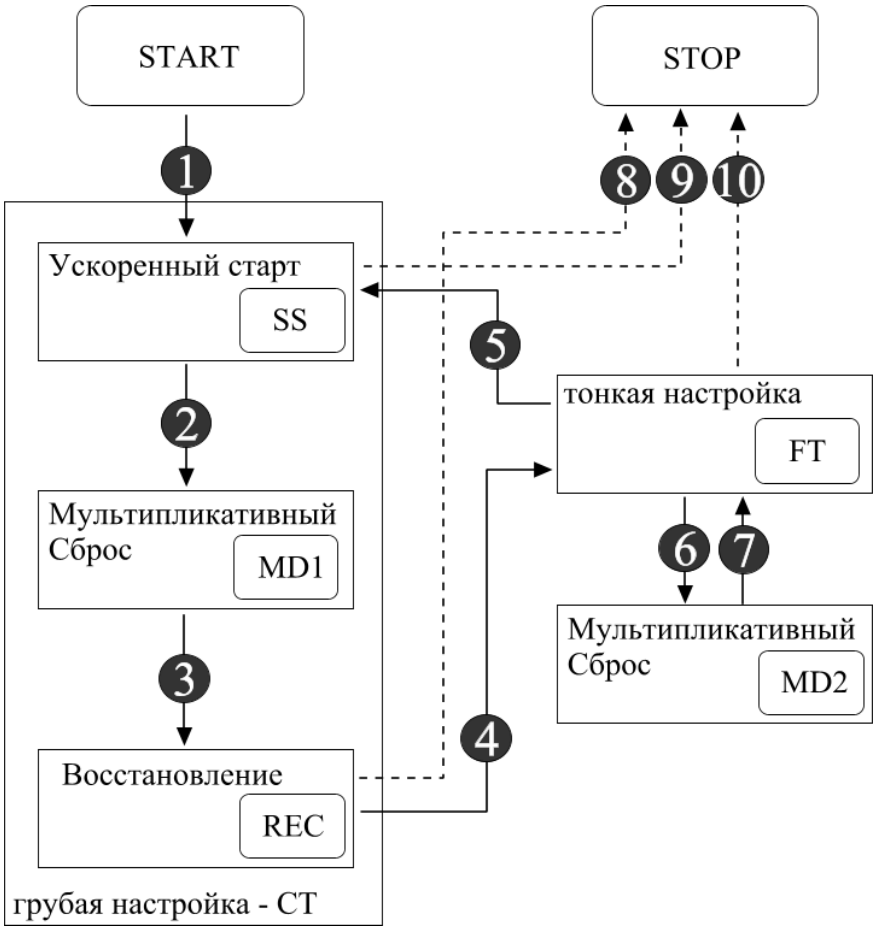

Рис. 19. Диаграмма режимов алгоритма адаптации протокола ARTCP. Штрихованные линии обозначают возможные переходы в состояние остановки системы.

Работа ARTCP начинается с режима быстрого увеличения скорости, аналогичной механизму замедленного старта стандартного TCP, для максимально быстрого достижения соединением верхнего предела доступной пропускной способности. После того, как верхний предел достигнут, алгоритм ARTCP переходит в режим точной настройки, в течение которой удерживает скорость на уровне доступной ПС. В случае определения уменьшения доступной ПС, ARTCP совершает мультипликативное снижение скорости, которое в случае

продолжительного состояния перегрузки продолжается экспоненциально. Итак, адаптация скорости передачи потока протоколом ARTCP происходит в пяти режимах (рис. 19).

Режим ускоренного старта (SS) имеет цель максимально быстро увеличить скорость потока от минимального значения до значения, равного или превосходящего ПС канала сразу после инициализации соединения. Для этого скорость увеличивается экспоненциально:

$$R'_S(t+RTT) = R'_S(t) \times SSGR \quad (19)$$

$$R_S(t_i) = R_S(t_{i-1}) + R'_S(t_i) \times (t_i - t_{i-1}) \quad (19а)$$

Начальное значение скорости устанавливается после синхронизации соединения как:

$$R_s^{\min} = \frac{SEGSIZE}{\text{minRTT}} \quad (19б)$$

Выход из режима SS происходит, когда $R_e(t_i) < (1-\varepsilon) \times R_S(t_i - RTT)$. После реализации перехода 2, алгоритм переходит в состояние мультипликативного сброса MD1.

Режим мультипликативного сброса (MD1) следует за режимом SS. После выхода из SS значение $R_s(t)$ будет превышать $R_e(t)$, поэтому в режиме MD1 скорость потока скачкообразно устанавливается заведомо ниже $R_e(t)$:

$$R_s(t_i) = R_e(t_i) - MDFACTOR \times (R_s(t_{i-1}) - R_e(t_i)) \quad (20)$$

После снижения скорости алгоритм переходит в режим восстановления.

Режим восстановления (REC) имеет целью, линейно увеличивая скорость, довести ее до уже известного значения ПС канала: $R_e(t)$, компенсируя возникшую в режиме SS перегрузку. В режиме REC вычисляется значение площади области компенсации $A_c(t_i)$ как площади фигуры, образованной значениями $R_s(t)$ над прямой $R_e(t_i)$ за время, пока $R_s(t_i) > R_e(t_i)$ в режиме SS:

Значение площади области компенсации равно сумме площадей набора трапеций образованных значениями $R_s(t)$ над прямой $R_e(t_i)$. Площадь каждой трапеции

$$S_T = \Delta t_i / 2 \, (2R_s(t_i) - R'_s(t_i) \times \Delta t_i - 2R_e(t_i)) \quad (21)$$

где Δt_i время, в течение которого не происходило изменений значения $R'_s(t_i)$.

$$A_c(t_i) = \frac{1}{2}\Delta t_0 [2R_s(t_i) - \Delta t_0 R'_s(t_i) - 2R_e(t_i)] + \quad (22)$$

$$+\frac{1}{2} RTT [2R_s(t_i) - \frac{R'_s(t_i)}{SSGR} - RTT \frac{R'_s(t_i)}{SSGR} - R_e(t_i)] +$$

$$+\frac{1}{2} RTT [2R_s(t_i) - \frac{R'_s(t_i)}{SSGR^2} - RTT \frac{R'_s(t_i)}{SSGR^2} - R_e(t_i)] + \cdots$$

$$+\frac{1}{2}\frac{[R_s(t_i)-\frac{R'_s(t_i)}{SSGR^N}-R_e(t_i)]^2}{\frac{R'_s(t_i)}{SSGR^N}},$$

где Δt_0 промежуток времени между моментом t_i и моментом предыдущего изменения $R'_s(t)$ и N - индекс слагаемого, при котором

$$R_s(t_i)-\frac{R'_s(t_i)}{SSGR^N} > R_e(t_i) \qquad (23)$$

Итак, первое слагаемое приведенной формулы есть площадь трапеции с высотой меньшей *RTT*, последнее слагаемое - площадь треугольника, слагаемые от 2 до N-1 площади трапеций с высотой равной *RTT*.

Например, в случае, приведенном на рис. 20, $A_c(t_i)$ равна сумме площадей трапеции DCBE и треугольника ABE. Δt_0 равно отрезку ED.

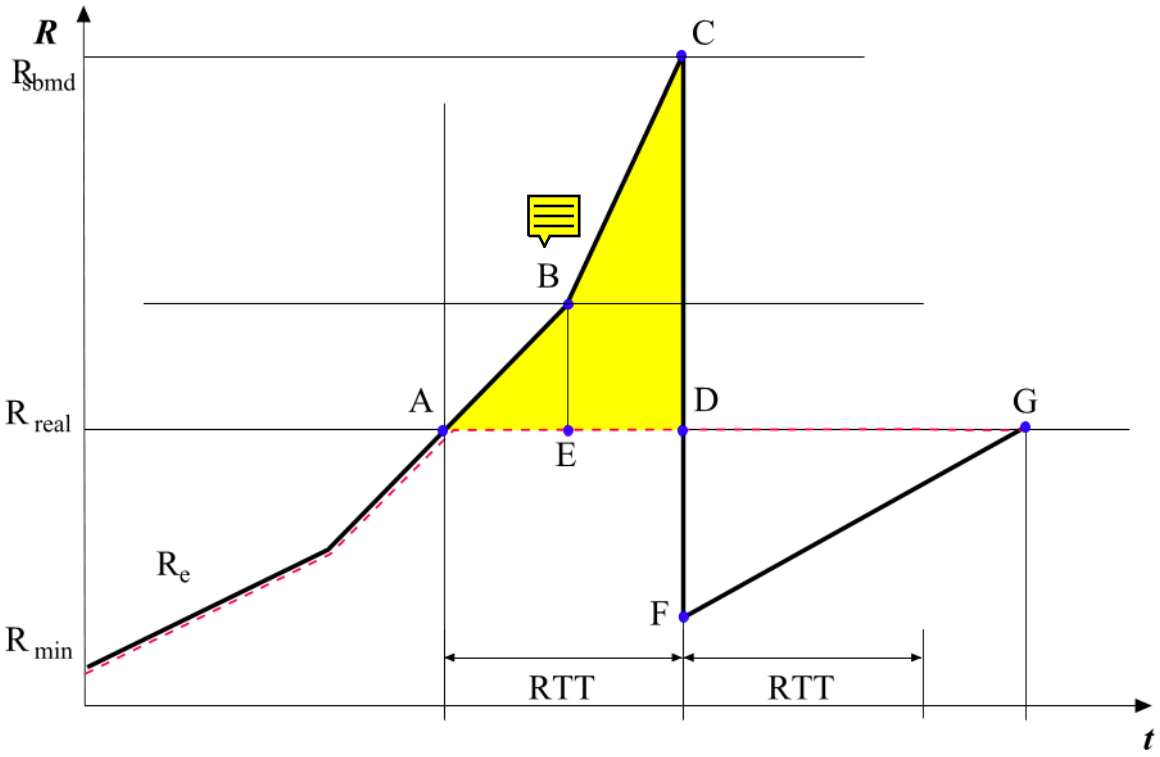

Рис. 20. Зависимость скорости от времени в фазе грубой настройки ARTCP. Закрашенная область, состоящая из ABE и BCDE есть площадь области компенсации и выражает объем данных, накопившихся в буфере. Штриховая линия обозначает текущую оценку ПС сети протоколом ARTCP.

Значение $A_c(t_i)$ применяется для определения величины $R'_s(t_i)$ в состоянии восстановления REC. Идея в том, что из за задержки информации о состоянии сети на время *RTT*, в состоянии быстрого увеличения скорости отправки сегментов поток вызовет

наполнение буферов сети. Пакеты будут накапливаться в сети в течение времени, когда скорость отправки сегментов превышает ПС сети (отрезок AD на рис. 20). Пребывание соединения в состоянии восстановления необходимо для того, чтобы сеть справилась с возникшей до уменьшения скорости отправки перегрузкой. Очевидно, что количество данных, накопившихся в буферах сети, определяется площадью области $A_c(t_i)$, поэтому скорость отправки сегментов в состоянии REC должна быть снижена таким образом, чтобы площадь фигуры DFG была равна $A_c(t_i)$. Скорость в состоянии REC меняется линейно, определяемая значением

$$R'_s(t_i) = \frac{[R_{pe}(t_i) - R_s(t_i)]^2}{2A_c(t_i)} \quad (24)$$

В состоянии REC скорость отправки данных возрастает линейно от значения полученного в предшествующей стадии MD1 по закону

$$R_s(t_i) = R_s^{amd} + t \times R'_s(t_i) \quad (25)$$

Выход из состояния REC (переход 4) осуществляется в том случае, когда

$$R_s(t) \geq R_e^{amd} \quad (26)$$

Режим тонкой настройки (FT) следует за режимом REC, в режиме FT скорость отправки данных медленно подстраивается под ПС канала. Отношение коэффициентов *speedup* и *slowdown* в состоянии FT определяет вероятность снижения или повышения скорости на каждом шаге. Коэффициент *speedup*, отвечающий за повышение скорости обратно пропорционален скорости данного соединения. Коэффициент *slowdown*, отвечающий за снижение скорости, пропорционален отношению измеряемого *RTT* к минимальному значению *RTT*. Значение *speedup* больше при меньших значениях $R_s(t)$, что дает медленным соединениям преимущество для получения доступа к большей относительной доле ПС. Значение *slowdown* одинаково для всех соединений и растет при росте *RTT*. Таким образом, вероятность повышения скорости для медленных соединений больше, а вероятность снижения скорости одинакова для всех соединений. Выход из режима FT происходит в случае скачкообразного изменения измеряемого *RTT*.

В состоянии FT значения скорости передачи определяются по закону:

$$R_s(t_i) = \text{midpoint} + [2 \times Rand - \frac{slowdown}{speedup}] \times \frac{speedup}{slowdown} \times INTERVAL \times \text{midpoint} \quad (27)$$

где *rand* – равномерно распределенная случайная величина, генерируемая функцией `drand48()` с областью значений [0; 1]. После попадания в состояние FT (реализации перехода 4) значение midpoint устанавливается равным R_e^{amd}, в дальнейшем значение

midpoint устанавливается равным sR_s. Переменные speedup и slowdown определяют направление изменения скорости отправки данных в зависимости от изменения времени RTT.

Коэффициенты slowdown и speedup для использования в формуле (27) определяются по следующим формулам:

$$speedup = [1 + \frac{S}{\min \mathrm{RTT} \times sR_s}]^2 \quad (30)$$

где второе слагаемое представляет собой отношение минимального количества данных в транзите по соединению (S байт за время RTT) к реальному их количеству (произведение минимального времени RTT на среднюю скорость отправки потока). Соответственно рост реальной скорости потока выражается в уменьшении значения коэффициента speedup, таким образом, при прочих равных условиях вероятность роста R_S для соединения с меньшей скоростью будет больше. За счет этого устраняется неравномерность использования ресурсов разными соединениями.

Коэффициент slowdown вычисляется следующим образом:

Если *RTT>minERTT*(1+PRECISION)*, (31)

то $slowdown = 2 \times (RTT / \min ERTT) \times speedup$ иначе

slowdown=1

Отношение speedup/slowdown определяет знак отклонения мгновенного значения скорости от среднего. Если *speedup>slowdown* то отклонение от среднего значения для мгновенного значения скорости будет положительным, т.е. скорость потока будет увеличиваться. В случае *speedup<slowdown* скорость потока будет снижаться. Также, в состоянии FT максимальное отклонение мгновенного значения скорости отправки пакетов от среднего за предыдущий период, согласно формуле (27), пропорционально среднему значению скорости. В связи с этим поток, совершая переход 4 в состоянии FT при большем значении оценки доступной ПС, приспосабливается к небольшим изменениям ПС более интенсивно.

Условием перехода из FT в режим мультипликативного сброса MD2 (переход 6) являтся:

$$ERTT > K \times (FT \max RTT - \min ERTT) \quad (28)$$

Режим мультипликативного сброса (MD2) необходим для быстрого снижения скорости при условии резкого роста *RTT*:

$$midpoint_i = midpoint_{i-1} \times MDFACTOR \quad (29)$$

После этого протокол переходит в состояние FT, реализуя переход 7. В том случае, если условие (28) продолжает оставаться истинным, то мультипликативное уменьшение

продолжается, поскольку последовательность переходов 6 - 7 реализуется неоднократно, выражаясь в экспоненциальном уменьшении скорости передачи данных.

Завершение работы протокола может произойти из любого состояния {SS, REC, FT} - переходы (8, 9, 10).

Ожидаемое поведение алгоритма управления скоростью потока в различных режимах изображено на рис. 21.

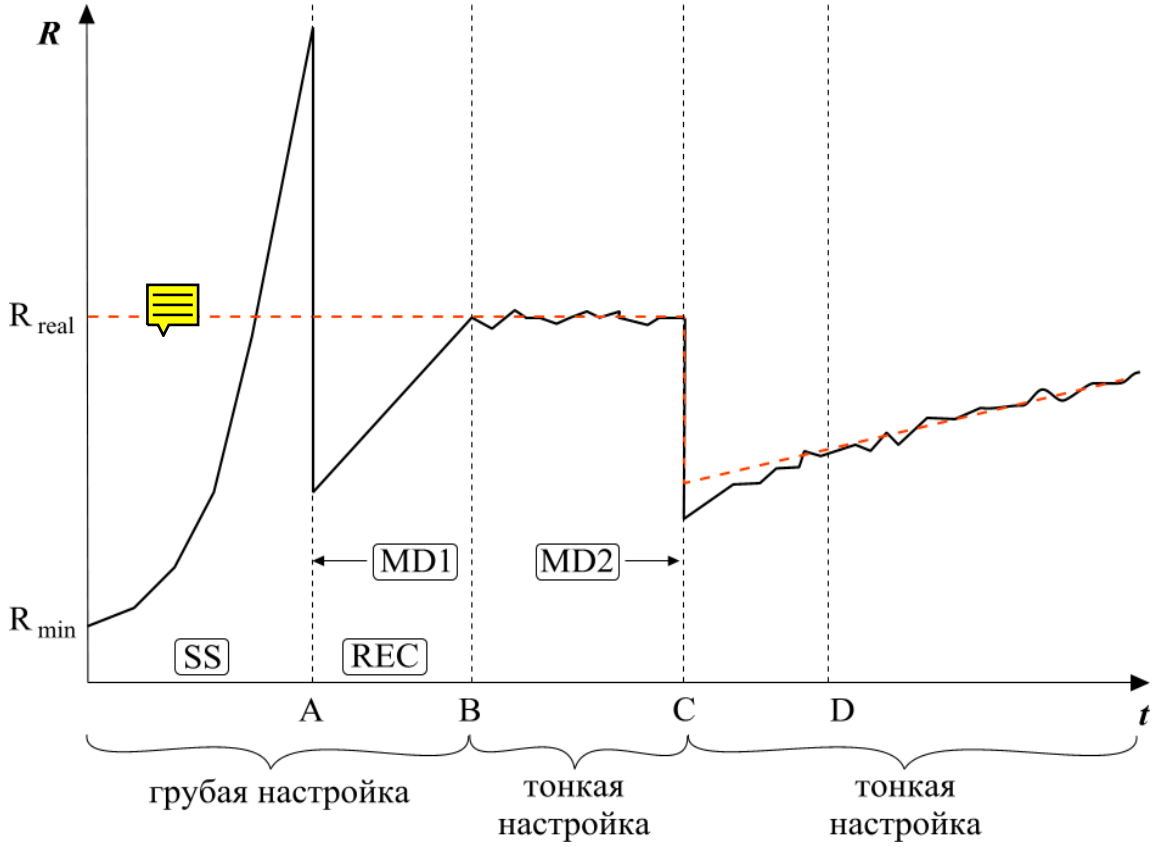

Рис. 21. Ожидаемое поведение алгоритма управления скоростью потока (зависимость скорости от времени). Значения *t* в точках A,B,C обозначают моменты перехода в новый режим.

2.6. Совместимость с TCP

Система, поддерживающая ARTCP, может быть также совместима с TCP. Для этого, инициатор соединения, поддерживающий ARTCP, помещает в заголовке синхронизирующего пакета опцию “PS”. Наличие поля “PS” в заголовке SYN пакета должно включать механизм ARTCP получателя. Если такая возможность имеется, то получатель включает в сегмент SYN-ACK поле “TI”, если нет, то отсутствие опции “TI” в ответе получателя отключает механизм ARTCP у инициатора и дальнейший обмен происходит по протоколу TCP. Если же опция “TI” присутствует в ответе, то инициатор уверен, что и получатель задействовал механизм ARTCP.

2.7. Сравнение ARTCP и TCP на основе анализа алгоритма

Протокол TCP не может обходиться без потерь сегментов, которые создаются самим протоколом путем переполнения очереди, и которые ограничивают дальнейший рост скорости TCP.

Теорема 2: Протокол ARTCP не создает потерь сегментов, если $\frac{Q_{\max}}{R} > \varepsilon$, где $Q_{\max}$ максимальная длина очереди, ε - предельное значение, используемое алгоритмом ARTCP, а R - скорость канала.

Доказательство:

Потеря сегмента может произойти лишь в случае поступлении очередного сегмента в очередь обслуживающего устройства, когда размер сегмента превышает разность $Q_{\max}$ - Q. Если D сумма задержек передачи в каналах на пути от отправителя к получателю, то в предположении, что каналы в системе симметричны и трафик передается лишь в одном направлении, минимальное время *minRTT* определяется как *2D*. В случае если средняя скорость отправки потока R_S превышает скорость R канала, обслуживающего очередь, возникает очередь сегментов в маршрутизаторе. Появление очереди длины Q приведет к увеличению времени RTT на величину *Q/R*. Однако для ARTCP, если разность *RTT-minRTT* превышает некоторое предельное значение ε, вероятность снижения скорости отправки потока R_S станет отличной от нуля и скорость потока будет снижаться, пока значение разрешенного превышения *RTT* над *minRTT* не станет ниже предельного значения. Таким образом, выполняется неравенство: $Q < \varepsilon \times R$. Выбирая достаточно малое значение ε, алгоритм ARTCP гарантирует, что длина очереди не превысит определенного значения, меньшего, чем максимальная длина очереди $Q < Q_{\max}$. Для выполнения этого, необходимо выбрать ε, так, чтобы $\varepsilon < \frac{Q_{\max}}{R}$. Следовательно, при таком выборе предельного значения, отсутствие потерь сегментов при работе ARTCP гарантировано.

Следствие из теоремы 2: при числе потоков большем 1, протокол ARTCP в отличие от TCP, может быть настроен так, чтобы вообще не создавать потерь сегментов.

Для протокола TCP факт потери сегмента служит индикатором возникновения перегрузки сети. Значение вероятности потери сегмента определяет развиваемую TCP соединением скорость передачи. Любая потеря сегмента вызывает скачкообразное уменьшение размера окна и, следовательно, снижение скорости передачи TCP. В условиях, когда причиной потерь является исключительно переполнение очереди, снижение скорости TCP при возникновении потерь приводит к тому, что выполняется равенство средней

скорости TCP и скорости обслуживания канала. Очевидно, что снижение скорости вследствие дополнительных потерь, вызванных ошибками передачи, приведет к тому, что средняя скорость TCP потока станет меньше, чем скорость канала. Поскольку TCP не может определить причину потери сегмента и реагирует снижением скорости на любую потерю, то его эффективность в сетях, где потери сегментов могут являться следствием ошибок передачи, будет тем меньше, чем выше вероятность потери сегмента.

Протокол ARTCP в отличие от TCP не снижает скорость передачи потока при возникновении потери сегмента. Потерянные данные ретранслируются, не оказывая влияния на скорость передачи. Вследствие этого, потери сегментов не оказывают влияния на скорость потока ARTCP.

Сказанное выше можно сформулировать в следующем виде:

Свойство: В отличие от TCP, протокол ARTCP не чувствителен к потерям сегментов.

2.8. Направления дальнейшего развития ARTCP

Протокол ARTCP, предложенный в этой работе, способен работать более эффективно и качественно, чем TCP, однако можно выделить несколько направлений дальнейших исследований нового протокола, которые могут, во-первых, дать возможность эффективно использовать его в асимметричных системах, а во-вторых, достичь равноправия между потоками с разной длиной маршрута.

2.8.1. Асимметричные системы

Поскольку в протоколе ARTCP устранена ACK-синхронизация, присущая TCP, то отправка сегментов происходит независимо от прибытия подтверждений вплоть до исчерпания максимального окна, то в отличие от TCP, ARTCP может быть усовершенствован так, чтобы эффективно работать в системах с асимметричными каналами.

Для использования ARTCP в таких системах необходимо уменьшить частоту подтверждений. Поскольку искусственная задержка подтверждений вызовет увеличение задержки в петле обратной связи, то, измерение задержки передачи сегментов нужно также связать с получателем. Поскольку трудно добиться хорошей синхронизации системных часов получателя и отправителя, то получатель может лишь замечать изменение времени передачи сегментов, если отправитель использует стандартное поле временной метки [6]. Если разность значений метки в потоке и системных часов получателя изменяется, значит изменяется и абсолютное значение задержки. В этом случае получатель должен увеличить частоту подтверждений, чтобы отправитель мог среагировать на изменение нагрузки в сети. Когда значения скорости прибытия потока и задержки передачи не меняются, частота подтверждений может быть снова уменьшена.

Алгоритм ARTCP не содержит препятствий для этого усовершенствования, кроме того, модифицированный таким способом ARTCP для асимметричных каналов сохранит совместимость с представленным здесь протоколом.

2.8.2. Соединения с различными RTT

Для соединений, обладающих различным временем RTT, из-за различий длин их маршрутов, среднее значение коэффициента F ограничивается некоторым числом, меньшим 1, как показано в работах [80, 81, 82, 83]. Это верно как для ARTCP, так и для TCP. Чтобы достичь равноправия разделения ПС между ARTCP потоками с разной длиной маршрута, необходимо устранить зависимость коэффициента speedup от минимального времени RTT. Этого результата можно достичь путем модификации алгоритма адаптации таким образом, чтобы в режиме FT полностью отказаться от использования измеренного RTT как индикатора перегрузки, используя лишь значения межсегментных интервалов потока.

Глава 3. Имитационная модель

Для исследования возможностей протокола и отработки его механизмов была разработана программная имитационная модель самого протокола и сетевых компонентов, в среде которых должен функционировать протокол ARTCP. Модель представляет собой набор компонентов имитирующих реальные объекты в составе сети и объекты протоколов ARTCP и CBR (Constant Bit Rate). Изучаемая сеть может иметь топологическую схему достаточно большой сложности, которая строится из необходимого числа экземпляров имеющихся классов.

Основными способами позволяющими исследовать и верифицировать сложные сетевые протоколы являются стенды для тестирования (экспериментальные сети), программные эмуляторы и системы автоматизированной верификации. Верификацию набора процедурных правил протокола можно осуществлять с помощью программного перебора состояний конечного автомата представляющего протокол [56, 57, 58], например, в интерпретаторе языка PROMELA, таком как SPIN [3]. Однако верификация набора процедурных правил, и изучение эффективности протокола преследуют различные цели и, соответственно, для них применяются различные инструменты. Верификация протокола означает применение к его набору процедурных правил формального метода, который позволяет доказать, что этот набор или моделирующая его система конечных автоматов (с обменом данными) полна, не содержит недостижимых состояний, свободна от статических и динамических блокировок. Верификация протокола не ставит задачей даже определение количественных характеристик его эффективности, с точки зрения верификации важно лишь то, что прогрессивный обмен данными вообще происходит. Вследствие этого методы верификации построены на полном или частичном анализе доступных состояний КА, представляющего протокол. Для систем с большим числом состояний ($>10^5$) верификация основанная на полном анализе затруднена на практике.

Моделирование протокола не дает гарантии полного анализа достижимых состояний, но зато позволяет исследовать количественные характеристики системы. Вместе с тем моделирование позволяет сделать статистически обоснованный вывод о надежности протокола, по крайней мере, относительно отсутствия в нем блокировок. Сложность имитационного моделирования в том, что помимо реализации процедурных правил самого исследуемого протокола в состав модели должны входить все его компоненты: среда функционирования, словарь и способы кодировки сообщений, модель сервиса протокола. Однако при этом моделирование требует все же меньших затрат, чем разворачивание экспериментальной сети для исследования свойств протокола. В данном случае наша задача

состоит не в верификации протокола ARTCP, а в определении численных значений его характеристик в различных условиях.

Одним из наиболее мощных эмуляторов протоколов по своим функциональным возможностям является программный комплекс NS (Network Simulator) входящий в состав системы VINT[13] (Virtual Internetwork Testbed) [59, 60, 61]. Однако эта система вследствие крайней обширности области применения является также очень требовательной к вычислительным ресурсам и не дает возможности эффективно работать с диспетчеризацией на уровне сегментов, поэтому для моделирования протокола ARTCP использовалось программное обеспечение собственной разработки. Для построения данной программной модели (ПМ) были использованы методы объектной разработки и реализация на языке C++ [62, 63] в среде Digital UNIX[14].

Далее мы рассматриваем формат сообщения, используемый в модели, иерархию и реализацию ее основных классов.

3.1. Формат сообщения

Сегменты протоколов моделируется следующей структурой данных.

```
struct artcp_segment_s {
        int src, dst, port;
// адрес отправителя, получателя, порт назначения
        long int seq, psn;
// порядковый номер сегмента, смещение от предыдущего (поле PS)
        double psk;
// скорость потока, при получении этого сегмента (поле TI)
        long int rcv_wnd;
// окно объявленное получателем
        long int ack_trig, ack;
// порядковый номер сегмента вызвавшего это подтверждение, номер
// подтверждения
        long int syn_ack;
// подтверждаемый порядковый номер SYN-сегмента
        unsigned char flags;
// управляющие флаги
        int hdr_size, payload_size;
// размер заголовка, размер поля данных
```

[13] Программное обеспечение комплекса VINT: Network Simulator и Network Animator является проектом университета Беркли США и его можно получить по адресу: http://www-mash.cs.berkeley.edu/ns/ns.html

[14] для разработки, отладки и использования модели применялся региональный кластер научных вычислений

```
        int link;
// число ссылок на этот сегмент (для корректной работы с памятью)
        long int id;
// уникальный в модели номер сегмента
};
```

Новые сегменты создаются внутри объектов протоколов ARTCP или CBR. Все объекты топологии передают друг другу указатель на область памяти, содержащую соответствующий сегмент, вместо копирования реальных данных. Кроме того, размеры заголовка сегмента и поля полезной нагрузки задаются в самой структуре данных, и все объекты определяют размер сегмента, интерпретируя значения этих полей. Поскольку ссылка на объект может одновременно содержаться во всех объектах, где реализована буферизация: ARTCP, link, interface, то во избежание конфликтов счетчик ссылок хранит количество объектов, в буферах которых находится ссылка на данный сегмент. Освобождение области памяти, содержащей сегмент, разрешено только в случае обнуления счетчика. За исключением полей “TI” и “PS” и служебных полей link, id, все остальные поля сегмента соответствуют полям стандартного TCP заголовка.

Поле заголовка ARTCP в модели не содержит проверочной суммы, повреждения сегментов в транзите могут моделироваться путем реализации случайного отбрасывания сегментов на сетевых узлах или в конечной системе с вероятностью, соответствующей вероятности битовых ошибок среды передачи, которую необходимо смоделировать.

3.2. Объектная структура ПМ

Для моделирования самого протокола ARTCP и среды его исполнения используются организованные в топологическую модель сети экземпляры следующих основных классов:

- Объект протокола ARTCP
- Объект протокола CBR
- Конечная система (host)
- Симплексный канал (link)
- Маршрутизатор (router)
- Интерфейс (interface)

состоящий из Alpha-станций под управление OS Digital UNIX

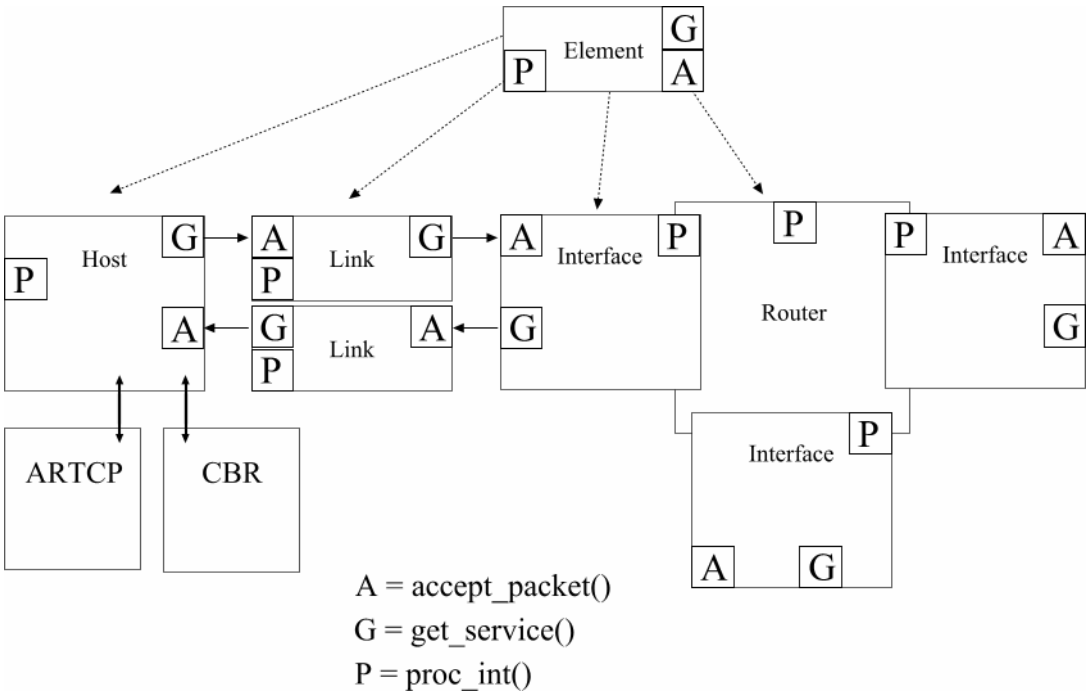

Рис. 22. Иерархия объектов модели сети в ПМ. Буквами A, G, P обозначены основные компоненты интерфейсов объектов. Штриховые линии указывают отношение наследования. Сплошные линии – направления вызовов методов для передачи сегментов.

Все классы, описывающие топологические элементы сети: конечная система (host), канал (link), маршрутизатор (router), интерфейс (interface), являются наследниками базового класса element. Интерфейс каждого из этих классов определяется тремя важнейшими компонентами: прием сегмента - accept_packet(), запрос сервиса - get_service(), обработка прерывания - proc_int(). Эти методы являются переопределением виртуальных функций базового класса. Кроме того, каждый производный класс расширен дополнительными специфическими для него методами. Посредством этих элементов интерфейса моделируется передача данных в системе (рис. 22). Каждый из наследников класса element ссылается на элемент, с которым он связан топологически, по указателю на базовый класс.

Экземпляр каждого из приведенных классов использует вызов accept_packet() топологически привязанного к нему объекта для передачи ему сегмента. В зависимости от наличия ресурсов метод вызываемого объекта может принять или не принять сегмент на обслуживание. Метод proc_int() каждого экземпляра вызывается из основной программы для обработки очереди и принятия решения об обслуживании очередного сегмента.

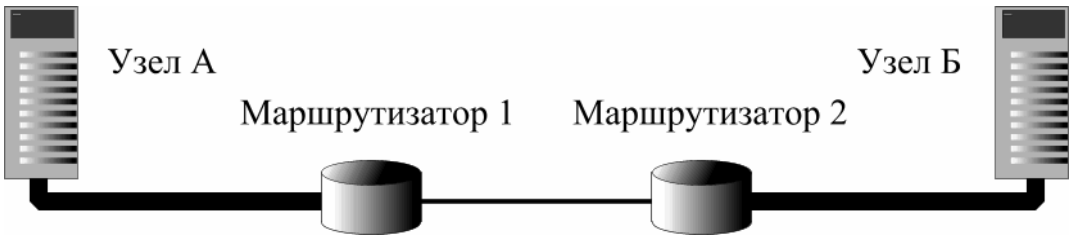

Рис. 23. Пример простейшей топологической схемы 1, состоящей из двух узлов и двух маршртутизаторов.

Объединение элементов сети в топологическую схему осуществляется с помощью вызова специального метода attach_next() для каждого экземпляра всех классов. Например, для организации модели сети по простейшей схеме (рис. 23) требуется набор из 14 объектов (рис. 24).

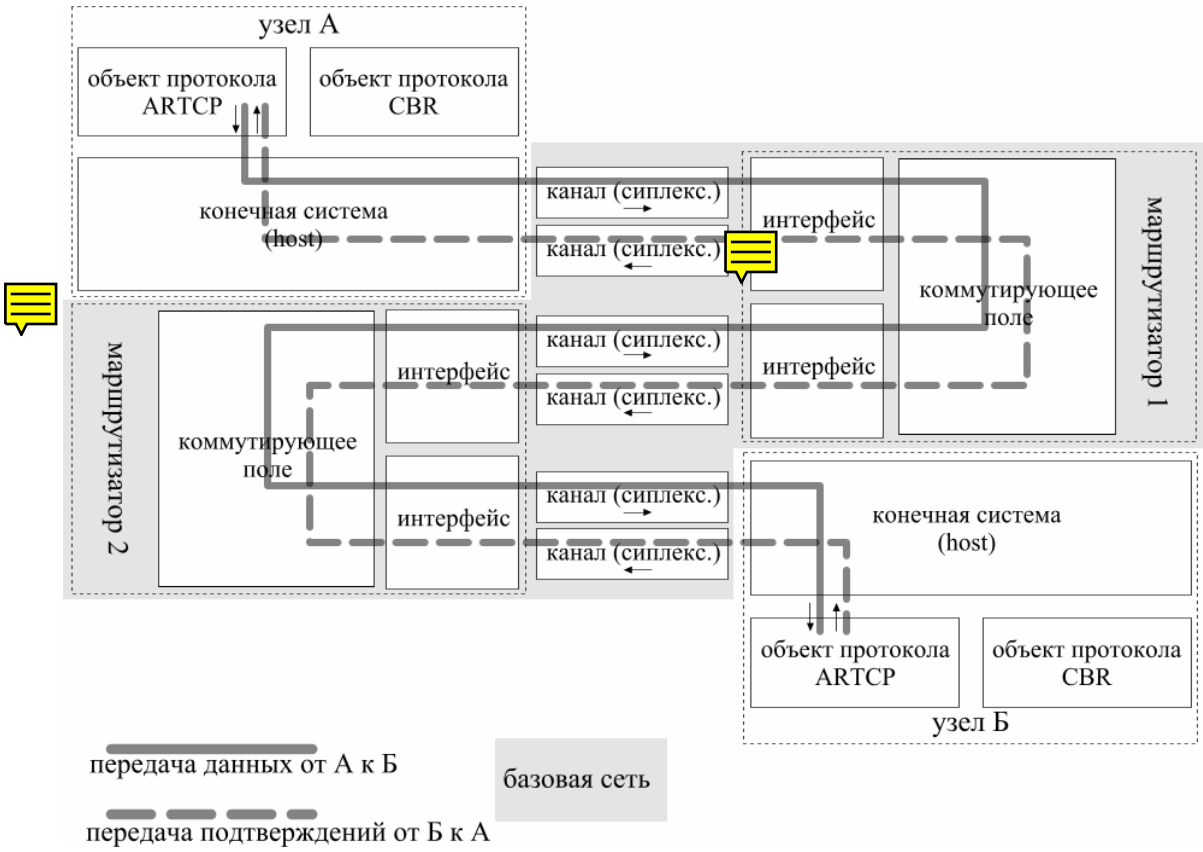

Рис. 24. Набор взаимодействующих объектов для топологической схемы 1. Закрашенная область обозначает топологические элементы сети (каналы и маршрутизаторы)

3.2.1. Класс link: аспекты реализации

Класс, моделирующий канал, получает значения пропускной способности и задержки передачи при инициализации. Структура данных класса реализуется односвязной

динамической очередью, в которую помещаются сегменты, принятые на обслуживание. После начала обслуживания сегмента канал блокирует последующие запросы в течение времени t=S/R, где S – размер сегмента, а R – скорость канала, т.е. времени приема сегмента размера S в канал. В очереди канала сегмент задерживается в течение установленного времени задержки передачи. По истечении этого времени, объект канала вызывает метод accept_packet() следующего объекта в топологической схеме сети. Если следующий объект отказывает в обслуживании сегмента, то этот сегмент отбрасывается. Для моделирования дуплексной связи применяются 2 экземпляра канала. Параметры каналов могут быть различными для моделирования асимметричных систем.

Метод proc_int() всех экземпляров класса link вызывается из главного цикла. Обработка прерывания переводит внутренний счетчик времени и вызывает передачу готового сегмента из канала следующему объекту топологии.

Для моделирования ошибок среды передачи экземпляр класса link может быть инициализирован перегружаемым инициализатором, который помимо параметров скорости и задержки канала получает также параметр соответствующий резидентному[15] значению ошибки передачи BER. С вероятностью $1-(1-BER)^S$ сегмент отбрасывается вместо передачи следующему элементу топологии. Таким образом, можно моделировать спутниковые, радио, сотовые каналы.

3.2.2. Класс router: аспекты реализации

Структура класса router является сложной. В его состав входят несколько экземпляров класса interface. При инициализации класса router ему передаются два параметра: количество интерфейсов и объем буферного пространства (максимальная длина очереди), одинаковый для всех интерфейсов. Объект маршрутизатора создает заданное количество экземпляров класса interface. Указатели на каждый интерфейс располагаются в специальном массиве, проиндексированном по порядку создания интерфейсов. После этого на конкретный интерфейс можно ссылаться как router.interface(X), где X – номер нужного интерфейса. Каждый интерфейс при инициализации получает указатель на создавший его объект класса router, по которому он может ссылаться на методы класса router для передачи сегментов в матрицу коммутации (обозначена в схемах как "поле коммутации").

При приеме сегмента интерфейс незамедлительно передает его объекту маршрутизатора. Блокировки при этом произойти не может, т.е. моделируется не

[15] резидентной вероятностью ошибки называется ее вероятность после применения алгоритмов коррекции ошибок.

блокирующая коммутационная матрица. Объект маршрутизатора вносит запись, состоящую из двух полей: {*номер интерфейса, адрес отправителя*} в таблицу маршрутизации. После этого сопоставление адреса назначения сегмента со вторым полем таблицы маршрутизации дает номер выходного интерфейса для данного сегмента. При отсутствии совпадения с записями таблицы маршрутизации, сегмент передается для рассылки всем интерфейсам маршрутизатора, кроме того, с которого он был получен. Полученный от коммутирующей матрицы, сегмент помещается в выходную очередь интерфейса, если свободное пространство в ней $Q_{\max} - Q \geq S$, в противном случае сегмент отбрасывается. Очередь моделируется списком двойной связности, динамически выстраиваемом в памяти [63]. Очередь обслуживается со скоростью канала, к которому подключен интерфейс.

Метод обработки прерывания каждого из интерфейсов вызывается методом обработки прерывания объекта маршрутизатора. Также объекты маршрутизатора и интерфейса снабжены методами для выдачи отчета по текущим параметрам трафика: средняя скорость, счетчики пропущенных/отброшенных сегментов, таблица маршрутизации, средняя длина очереди.

Таким образом, объект маршрутизатора моделирует современное межсетевое устройство с не блокирующей коммутационной матрицей и буферизацией на выходе [67]. Возможно также расширение класса маршрутизатора алгоритмами RED и WFQ или CBQ [65, 66].

Поведение предложенной модели маршрутизатора относительно построения таблицы маршрутизации несущественно и совпадает со схемой функционирования коммутатора ЛВС. Таблицы маршрутизации заполняются на основании наблюдения за трафиком, а не в результате работы алгоритмов маршрутизации, поэтому не требуется никакой конфигурации маршрутизатора, в процессе работы он обучается топологии сети.

3.2.3. Класс host: аспекты реализации

Класс host имеет в своем составе экземпляр класса ARTCP, моделирующего сам протокол и экземпляр класса CBR, моделирующий протокол передачи данных без подтверждений и управления потоком с постоянной битовой скорости. В каждый момент времени только один из протоколов может быть в активном состоянии.

При отправке сегмента метод get_service() экземпляра класса host вызывается одним из протоколов. В случае приема сегмента на обслуживание объектом канала передачи, положительное подтверждение возвращается вызвавшему метод объекту протокола и указатель на область памяти, хранящую сегмент передается объекту канального уровня. В случае отсутствия возможности передать сегмент на канальном уровне, метод get_service()

объекта host возвращает отрицательное подтверждение. При инициализации экземпляра класса host он получает сетевой адрес. Отдельно устанавливается сетевой адрес назначения.

В случае приема сегмента от объекта канала, метод accept_packet() передает указатель протоколу определяемому по значению поля port заголовка сегмента. Сегменты, чей адрес назначения не совпадает с адресом данного экземпляра узла, отбрасываются.

Метод proc_int() класса host используется для передачи запроса на обработку прерывания объекту активного протокола.

3.2.4. Объект протокола CBR

Протокол постоянной скорости передачи напрямую соответствует реальному режиму передачи данных с постоянной битовой скоростью, например CBR в ATM [68] или (Circuit emulation services) CES [69] в IP сетях. Корректное взаимодействие с таким режимом передачи данных является актуальной задачей любого адаптивного транспортного протокола в современных сетях с интеграцией сервисов [30]. Именно поэтому протокол CBR включен в модель среды исполнения ARTCP. Протокол CBR генерирует сегменты и отправляет их в сеть с предустановленной скоростью R_{CBR}, т.е. временные промежутки между моментами начала последовательных трансляций составляют $\tau_{CBR} = S / R_{CBR}$. Подтверждения и, следовательно, ретрансляция сегментов и контроль скорости не реализуется протоколом CBR.

3.2.5. Объект протокола ARTCP

Модель ARTCP реализует управление скоростью отправки сегментов посредством механизма скользящего окна и собственно алгоритма ARTCP, коррекцию ошибок передачи не связанную с управлением передачи, быструю ретрансляцию сегментов и стандартную ретрансляцию по срабатыванию ТПП. Основные методы класса ARTCP таковы: Establish() – применяется для начальной синхронизации соединения, getarea() – вычисляет значение области компенсации, accept_packet(), proc_int(), status() – запрашивает результат отправки сегмента узлом. Из них три последние являются открытыми и составляют интерфейс класса. В состав класса ARTCP входят два экземпляра класса Queue, которые реализуют приемный и передающий буфер и методы для манипуляции с ними.

3.2.5.1. Класс Queue

Данный класс реализует схему стандартного управления потоком по методу скользящего окна. Класс содержит динамический список двойной связности, в который записываются указатели на сегменты, поставленные в очередь вместе с экземпляром класса таймера, реализующего ТПП. Методы класса Queue позволяют осуществлять вставку

сегмента с сортировкой, сканирование списка, нахождение очередного сегмента готового к передаче. Ретрансляция реализована посредством изменения очередности готовых к отправке сегментов. Взаимодействие очередей с объектом протокола и схема обработки сегментов приведены на рис. 25.

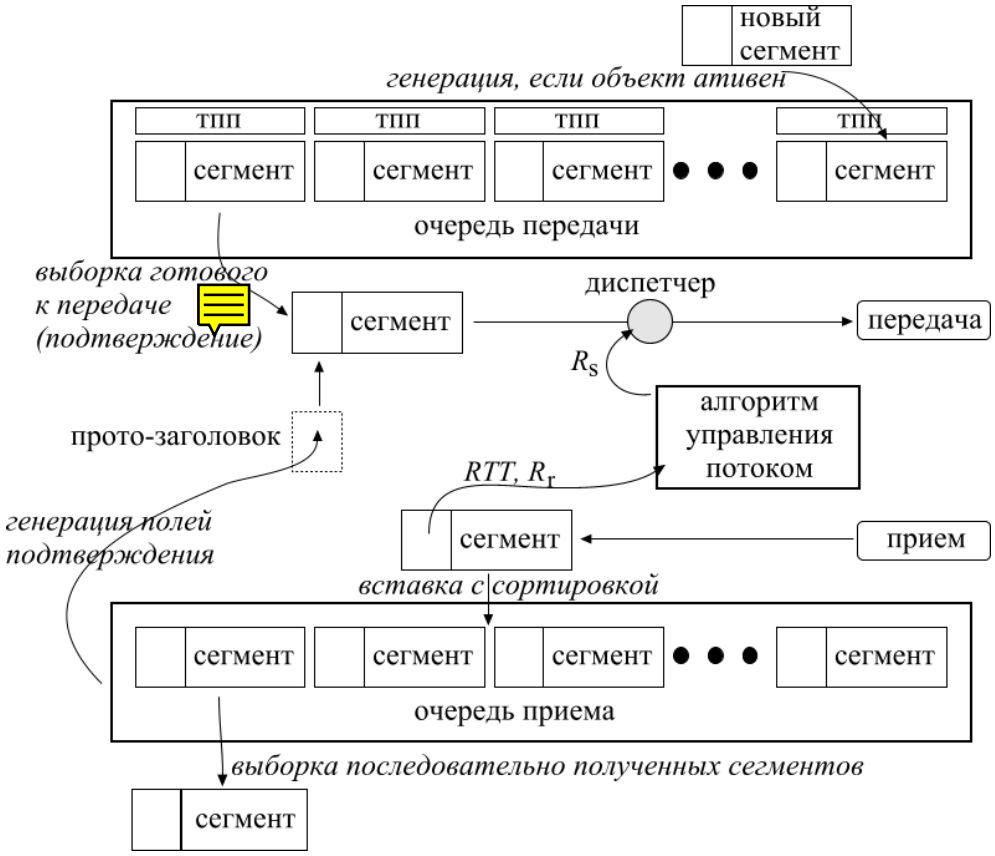

Рис. 25. Концептуальная схема взаимодействия очередей приема и передачи с алгоритмом управления потоком в объекте протокола ARTCP. Прямые линии указывают направление потоков данных, кривые - управляющей информации.

3.2.5.2. Обработка приема

Сегмент, принятый узлом, передается объекту ARTCP. Метод accept_packet() принимает сегмент и обрабатывает его как подтверждение, если сегмент содержит поле подтверждения. Параметры ARTCP вычисляются, если установлено поле “TI” сегмента. Если номер подтверждаемого сегмента образует непрерывную последовательность с порядковыми номерами сегментов находящимися в очереди передачи, то вся последовательность, кроме подтверждаемого сегмента удаляется из очереди. Например, сегменты в очереди передачи: {k, k+1, k+2, k+3, k+4, …} и узел получает подтверждение {k+3}, из очереди удаляются сегменты k, k+1, k+2.

Если сегмент содержит данные, то он помещается в приемную очередь, после чего приемная очередь сканируется, из нее убираются (передаются пользователю) сегменты,

полученные в непрерывной последовательности порядковых номеров. Затем генерируется подтверждение следующего ожидаемого получателем сегмента, в поля которого заносятся следующие значения: ack – кумулятивное подтверждение, ack-trig – номер последнего полученного сегмента, adv_wnd – свободное пространство в очереди приема. Например, в очереди приема находятся сегменты {i+1, i+2, i+3}, после получения i-го сегмента и его записи в очередь, из нее удаляются сегменты i, i+1, i+2, i+3 и значение кумулятивного подтверждения становится равным i+4. Прототип контрольного сегмента с подтверждением ставится на очередь к передаче.

3.2.5.3. Обработка прерывания

Метод proc_int() активного протокола вызывается из метода proc_int() содержащего его экземпляра класса узла. Данный метод обновляет значение внутреннего счетчика времени экземпляра объекта протокола, вычисляет текущую скорость передачи потока по формулам (19), (19a) в состоянии SS; (24) и (25) в состоянии REC; (27) в состоянии FT.

Вычислив значение межсегментного интервала, метод proc_int() класса протокола ARTCP определяет, прошло ли это время с момента отправки предыдущего сегмента. Если да, то запрашивается экземпляр очереди передачи на предмет наличия в ней готового к отправке сегмента. Сегмент для отправки берется из очереди или создается новый (в случае отсутствия готового сегмента в очереди), после чего в поля ack, ack_trig, adv_wnd заносятся значения из прототипа контрольного сегмента и сегмент передается объекту узла для передачи. Если передача успешна, то узел, вызывая метод status(), уведомляет об этом объект протокола, который отмечает соответствующий сегмент как отправленный и запускает ТПП для него.

3.2.5.4. Ускоренная ретрансляция

Поддержка ускоренной ретрансляции сегментов также включена в ПМ. Для этого объект протокола ARTCP отслеживает поступление дублирующих подтверждений. Если последовательно приходят два подтверждения одного и того же i-го сегмента, то ARTCP осуществляет ретрансляцию данного сегмента вне зависимости от значения его ТПП. После осуществления ретрансляции алгоритм переходит в состояние, в котором ускоренная ретрансляция не разрешена и остается в этом состоянии пока не придет хотя бы одно подтверждение следующего: (i+1)-го сегмента.

3.2.5.5. Начальная синхронизация

Начальное значение RTT необходимо для вычисления начальной (минимальной) скорости работы соединения, равной S байт за время RTT. Для осуществления этого

измерения при открытии нового соединения объект протокола ARTCP реализует обмен сегментом специального типа с противоположной стороной. При активации объект ARTCP попадает в "не синхронизированное" состояние. Сегмент, обозначенный как SYN (по наличию соответствующего флага), отправляется от инициирующей стороны обмена к получателю. Сегмент SYN несет порядковый номер, не влияющий на порядковые номера при обмене данными. Получатель сегмента SYN реагирует отправкой сегмента с установленными флагами SYN, ACK, где поле ack несет значение порядкового номера SYN сегмента. Получение SYN, ACK сегмента дает возможность отправителю измерить время RTT и переводит соединение в состояние "синхронизации", после чего начинается обмен данными. Наличие идентифицирующего SYN сегмент порядкового номера дает возможность различить их возможные копии и правильно измерить время RTT. Синхронизация соединения может быть инициирована одновременно обеими сторонами. Для протокола TCP, например начальная синхронизация имеет место при открытии соединения и имеет целью установить идентичные начальные значения переменных в контрольных блока обеих сторон соединения.

3.3. Главный цикл

Главный цикл программы вызывает метод обработки прерывания proc_int() всех элементов топологии модели (host, link, router). В результате эмулируется ход внутренних часов каждого из объектов, и моделируются процессы передачи данных. Также из главного цикла с конфигурируемой периодичностью вызываются процедуры генерирующие отчеты.

3.4. Дуплексный режим

Протокол ARTCP, как и TCP способен осуществлять симметричный обмен информацией по одному соединению. В таком режиме контрольная информация получателя транслируется не отдельными сегментами, а записывается в соответствующие поля сегментов с данными, следующими в противоположном направлении. Программная модель предусматривает наличие буфера под прототип заголовка контрольного сегмента. Этот заголовок, содержащий номер подтверждения, размер окна и другую управляющую информацию, формируется после получения очередного сегмента с данными. Однако вместо немедленной отправки пустого сегмента с подтверждением, созданным из прототипа, модель проверяет наличие сегмента с данными ожидающего передачи и если сегмент такой сегмент имеется, то подтверждение передается вместе с ним[16]. Иначе из прототипа контрольного

[16] в англоязычной литературе такой способ передачи называется piggy back

сегмента создается отдельный сегмент не несущий данных, а лишь передающий подтверждение.

3.5. Трассировка модели

Каждый из объектов топологии, таких как узел, канал и интерфейс маршрутизатора, при каждой операции с сегментом выводит запись в файл отчета. Формат этой записи таков:

```
"{+|-|d|s|r}" time elem src"->"dst "{-|D}{-|A}{-|S}" size seq"/"ack " … " psn
                psk ack_trig adv_wnd link seq/syn_ack id
```

где:

- Действие: + прием в очередь, - из очереди, d отбрасывание, s отправка протоколом, r прием протоколом
- *time* время в секундах
- *elem* идентификатор объекта совершившего действие
- *src* адрес отправителя сегмента
- *dst* адрес получателя сегмента
- флаги: D данные, A подтверждение, S синхронизация
- *size* размер сегмента в байтах
- *seq* порядковый номер
- *ack* номер кумулятивного подтверждения
- *psn* поле "PS"
- *psk* поле "TI"
- *ack_trig* номер сегмента вызвавшего отправку подтверждения
- *adv_wnd* размер окна получателя в байтах
- *link* счетчик ссылок на область памяти
- *syn_ack* номер подтверждаемого SYN сегмента
- *id* уникальный идентификатор каждого сегмента

например, следующий фрагмент отчета представляет передачу одного сегмента с данными и его подтверждения:

```
s 0.6251 0   0->1 D-- 1000 0/-1 ... 1 -1 -1 65536 2 0/-1 2
+ 0.6357 6   0->1 D-- 1000 0/-1 ... 1 -1 -1 65536 2 0/-1 2
+ 0.6358 22  0->1 D-- 1000 0/-1 ... 1 -1 -1 65536 3 0/-1 2
+ 0.8608 11  0->1 D-- 1000 0/-1 ... 1 -1 -1 65536 2 0/-1 2
+ 0.8609 24  0->1 D-- 1000 0/-1 ... 1 -1 -1 65536 3 0/-1 2
r 0.8717 1   0->1 D-- 1000 0/-1 ... 1 -1 -1 65536 1 0/-1 2
+ 0.8717 25  1->0 -A- 40   0/1  ... -1 -1 0 65536 1 0/1  4
s 0.8718 1   1->0 -A- 40   0/1  ... -1 -1 0 65536 1 0/1  4
+ 0.8817 11  1->0 -A- 40   0/1  ... -1 -1 0 65536 1 0/1  4
+ 0.8818 23  1->0 -A- 40   0/1  ... -1 -1 0 65536 2 0/1  4
+ 0.9868 6   1->0 -A- 40   0/1  ... -1 -1 0 65536 1 0/1  4
+ 0.9869 17  1->0 -A- 40   0/1  ... -1 -1 0 65536 2 0/1  4
r 0.9970 0   1->0 -A- 40   0/1  ... -1 -1 0 65536 0 0/1  4
```

С помощью информации отчета можно проследить путь каждого сегмента, т.е. до какого узла он дошел, был ли он подтвержден, потерян и ретранслирован. Исследование моделируемой системы производится при помощи визуализации информации отчета.

Помимо событий, происходящих с сегментом, в отчет записываются также значения переменных протокола ARTCP и характеристики, снимаемые с сетевых устройств (с определенных интерфейсов). В отличие от событий с сегментами, переменные ARTCP и данные с интерфейсов маршрутизаторов снимаются с заданной периодичностью. По каждому из событий отчет может выводиться как в краткой форме (приведенной выше для операций с сегментами) так и в развернутом виде в целях отладки.

Для первичной обработки результатов моделирования применяется специально разработанный набор сценариев на языке командного интерпретатора UNIX C-shell [70] а также ряд небольших программ осуществляющих статистическую обработку результатов модельного эксперимента. Результаты работы модели фильтруются и записываются в несколько файлов в виде строк с полями, разделенными символом табуляции для последующей визуализации и статистической обработки.

3.6. Визуализация данных

Для визуализации полученных при моделировании данных применялась программа gnuplot[17] версии 3.7 для OS UNIX. Как правило, визуализация работы протокола TCP производится с помощью графиков зависимости размера окна CWND от времени, зависимости последовательности передачи сегментов от времени, позволяющей визуально определить различные фазы работы протокола. В подавляющем большинстве работ в области транспортных протоколов применяется именно такая схема [87].

[17] программа gnuplot является широко используемым средством визуализации научных данных. Программа доступна по адресу http://www.cs.dartmoth.edu/gnuplot_info.html

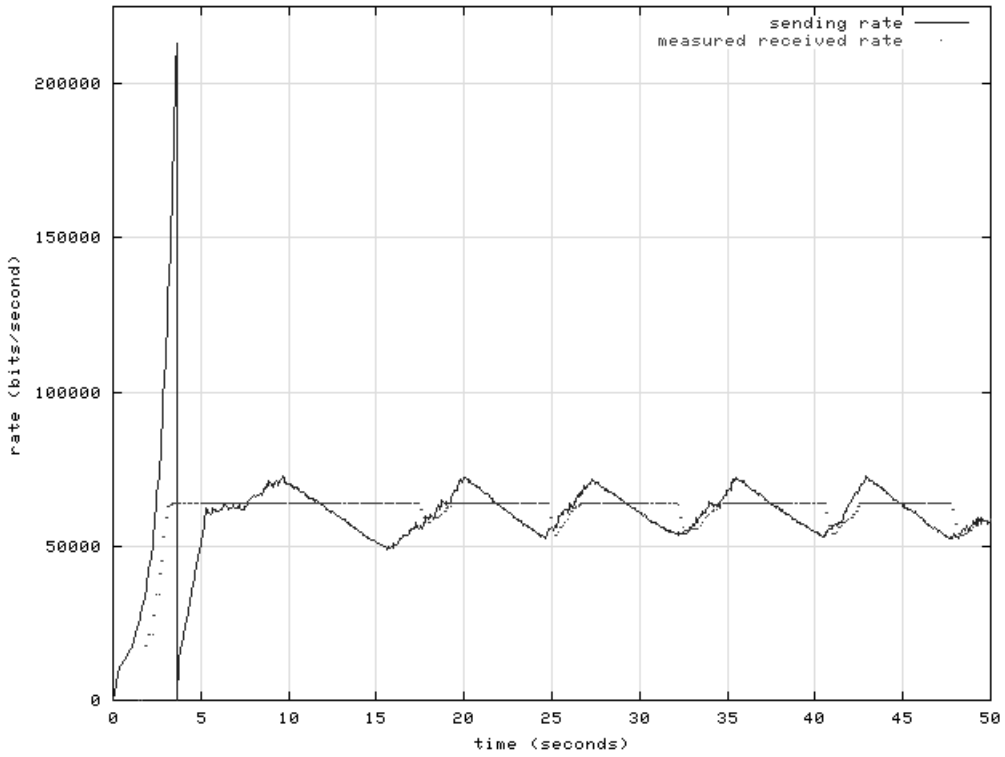

Рис. 26. График зависимости скорости отправки потока от времени. Sending rate - устанавливаемая скорость отправки, measured received rate - измеряемая скорость приема потока.

В некоторых работах приводятся более сложные системы визуализации, например в [14, 15, 19]. В настоящей работе результаты моделирования наиболее наглядно представляются следующими способами: зависимость скорости потока, порядкового номера передаваемого сегмента, времени RTT, средней длины очереди от времени (рис. 26, 27, 28, 29).

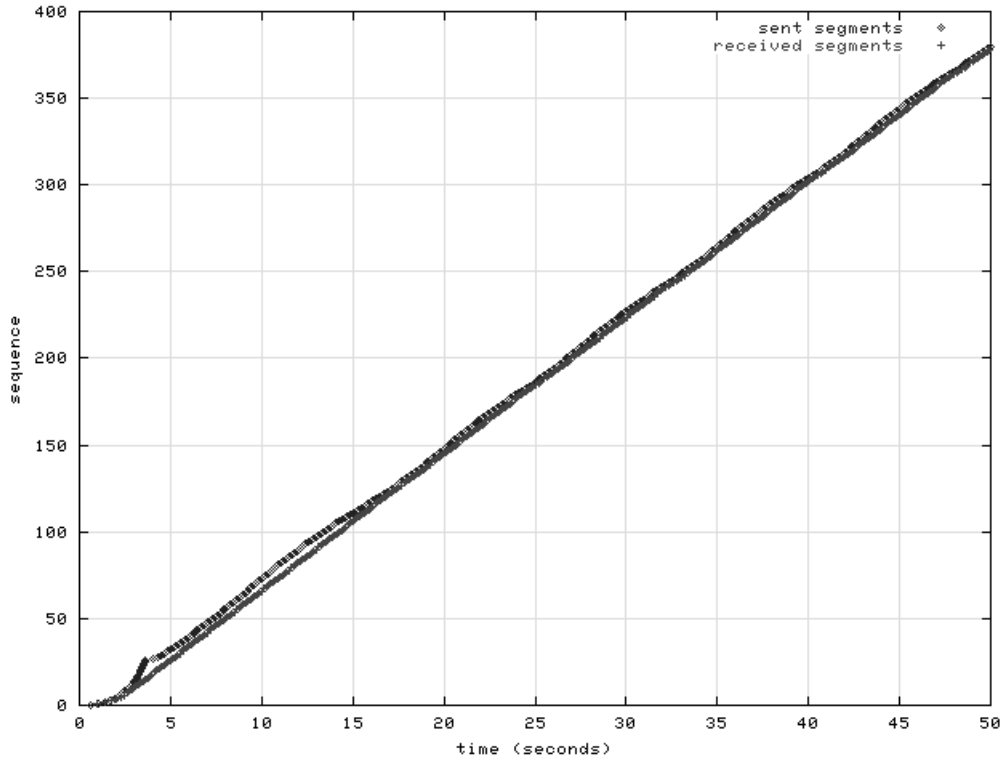

Рис. 27. Зависимость порядковых номеров отправляемых сегментов от времени. Sent segments - отправленные сегменты, received - полученные.

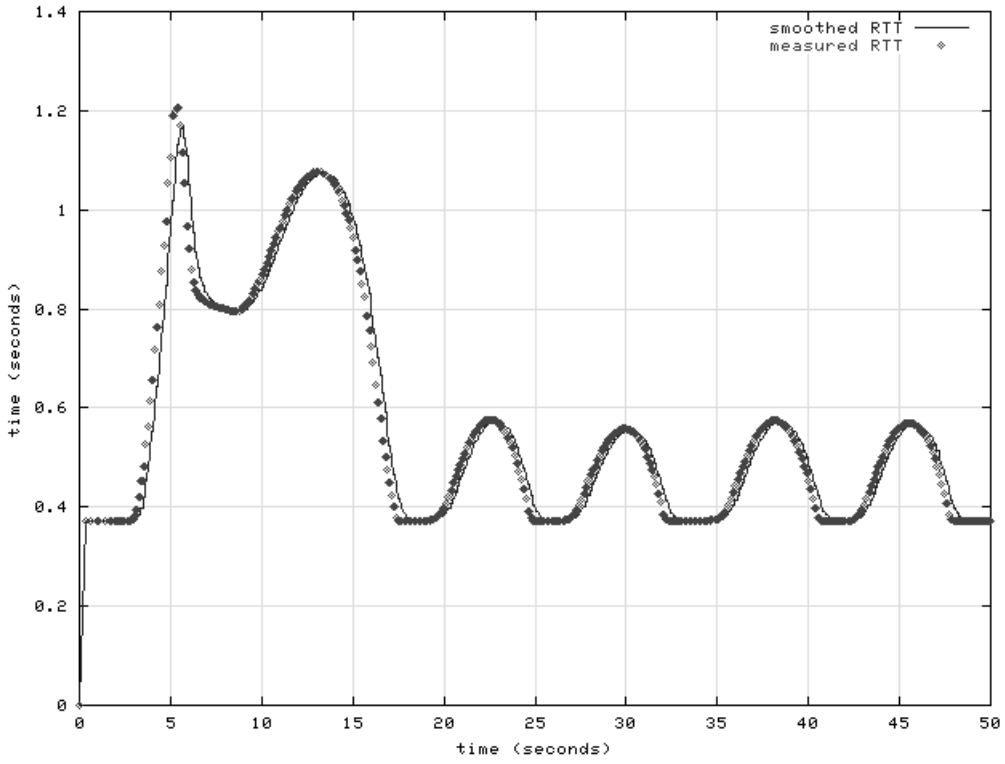

Рис. 28. Зависимость RTT от времени. Smoothed - сглаженное усреднением по окну, measured - мгновенное значение измерения RTT.

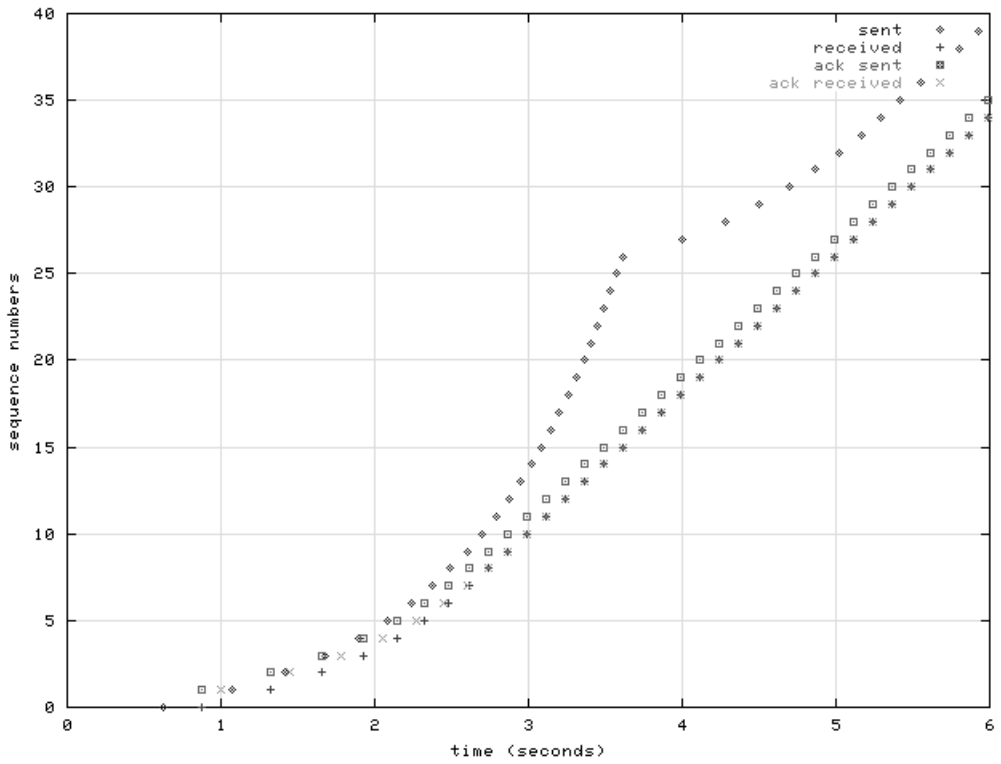

Рис. 29. Зависимость порядковых номеров передаваемых и принимаемых данных и подтверждений от времени.

На рис. 26. изображается поведение алгоритма ARTCP в виде изменения скорости в зависимости от времени. Явно просматриваются режимы SS, MD1, REC и FT. В режиме FT, переход в которое произошел примерно на 5-й секунде, имеют место спонтанные осцилляции значения скорости передачи. Сплошная линия - задаваемое значение R_s, точки - измеряемое значение R_e. Из рис. 26 с рис. 21. Хорошо заметно совпадение ожидаемого и реального поведения системы. Рис. 27. изображает эволюцию соединения, sent segments соответствуют моментам отправки сегментов, received segments - моментам приема сегментов соответствующих порядковых номеров. Оценка скорости по графику дает: между моментами t=20 и t=40 секунд было отправлено 150 1000-байтовых сегментов, т.е. средняя скорость потока составляет 7.5 сегмента/с. Это дает 93.75% использования ресурсов канала, поскольку моделируемая ПС канала равна 64 Кб/с.

Промежутки времени, когда задаваемая скорость потока превышает реальную ПС сети, выражаются в пиках значения RTT (рис. 28).

На рис. 29. динамика соединения приведена в масштабе 0-6 с., что более наглядно отражает процессы происходящие при работе ARTCP. Отмечены моменты отправки сегментов (sent), приема (received), отправки подтверждения (ack sent) и приема

подтверждения (ack received). Следует отметить, что подтверждение всегда содержит номер следующего ожидаемого получателем сегмента.

Глава 4. Результаты моделирования

4.1. Общая схема модельного эксперимента

Проведенный в рамках диссертации модельный эксперимент ставил задачей определение эффективности работы сети с алгоритмом ARTCP, а также сравнение характеристик ARTCP и TCP. Модельные эксперименты проводились в нескольких сценариях. Все сценарии укладываются в схему соединения двух ЛВС через канал с ограниченной ПС. Между ЛВС существуют один или более ARTCP потоков, а также может присутствовать CBR поток. Численными результатами экспериментов являются следующие показатели: коэффициент использования ресурсов:

$$U = \frac{число\_принятых\_битов}{(скорость\_канала) \times время}$$

коэффициент равноправия разделения пропускной способности:

$$F = \frac{(\sum_{i=1}^{n} b_i)^2}{n \times (\sum_{i=1}^{n} b_i^2)},$$

b_i доля ПС приходящаяся на i-е соединение. Это стандартная характеристика протокола, согласно [73].

среднее значение длины очереди Q, среднее число отправленных и ретранслированных сегментов.

Для получения средних значений показателей протокола в каждом из сценариев проводится большое число измерений (запусков программной модели). При визуализации эволюции соединений из всех экспериментов выбирался типичный для данного сценария.

4.1.1. Параметры моделируемой сети и диапазоны их изменения

На работу протокола транспортного уровня в сложной сети оказывает влияние огромное число факторов: это и все промежуточные системы нижних уровней со своими алгоритмами и протоколами, это и пользователь транспортного уровня. Многоуровневая архитектура позволяет нам абстрагироваться от всех свойств нижних уровней, моделируя лишь тот сервис, который предоставляет протокол IP, а именно доставку сегментов пользователя по адресу без каких либо гарантий порядка доставки, значения транзитной задержки и вероятности отсутствия потерь. Таким образом, общность модели достигается за счет того, что протокол IP, моделируемый в виде стандартного сервиса сетевого уровня, скрывает от транспортного протокола все более низкоуровневые объекты и протоколы.

Помимо нижних уровней, на работу протокола влияет поведение его пользователя, каковым является приложение, исполняемое на данном узле. Мы рассматриваем ситуацию, когда нагрузка на транспортный уровень максимальна, т.е. источник всегда имеет данные для передачи, а получатель обрабатывает данные по мере поступления. Такая ситуация наиболее распространена в реальности. Практически во всех случаях информационных потоков, кроме удаленного доступа в диалоговом режиме, соединение транспортного протокола открывается для передачи некоторого объема информации. Передача этого объема данных ведется с максимальной скоростью, после завершения обмена соединение прерывается. В процессе передачи транспортный протокол формирует сегменты максимального размера. Для разных маршрутов могут задаваться различные ограничения размера сегментов. Типичные значения, обусловленные наиболее распространенными технологиями передачи, составляют 576, 1000 и 1500 байт. В частности трафик, обусловленный обращениями пользователей Ярославской региональной сети к серверам WWW и FTP, расположенным за пределами Ярославской области, полностью состоит из сегментов размером 1000 байт (кроме последнего сегмента в соединении). Этот трафик составляет более 95% суммарного трафика региональной сети. Именно для такого трафика и предназначен ARTCP.

Основными характеристиками протокола, которые определяются посредством статистической обработки данных модельного эксперимента, являются: коэффициенты использования ПС: U, равноправия разделения ПС: F и средняя длина очереди Q. Нужно понять, зависят ли эти характеристики и если зависят, то как, от следующих параметров сети: ПС, RTT, BER, число потоков.

Диапазоны изменения этих параметров сети в модельном эксперименте выбираются таким образом, чтобы максимально адекватно соответствовать ситуации в реальных сетях. Так, например большинство каналов Ярославской региональной сети имеют ПС от 64 Кб/с до 2 Мб/с. Каналы коммутируемых соединений в этой же сети имеют ПС от 28800 б/с до 56000 б/с. Пропускная способность каналов ЛВС в подавляющем большинстве случаев равна 10 Мб/с.

Используемые в модельном эксперименте значения RTT характерны для соединений межрегионального типа, например: Ярославль-Москва. Значения BER соответствуют реальным вероятностям битовых ошибок, характерных для спутниковых каналов.

4.1.2. План модельного эксперимента

Поскольку протокол ARTCP является совершенно новым, то сначала рассмотрим в деталях поведение изолированного ARTCP потока для проверки работы его алгоритма.

Далее из множества параметров сети выделим те, которые оказывают наибольшее влияние на поведение протокола, чтобы сократить число параметров в последующих исследованиях. После этого проводим сравнение протоколов ARTCP и TCP в различных условиях. Показав превосходство ARTCP над TCP, более детально рассмотрим взаимодействие нескольких потоков ARTCP между собой и влияние на них потока CBR. По большой серии измерений исследуем трафик ARTCP на наличие свойства самоподобия.

Таким образом, план проведения экспериментов следующий:

Сценарий 1. Сравнение функционирования реализации протокола ARTCP в модели с ожидаемым поведением его алгоритма и наглядная иллюстрация работы соединения протокола ARTCP.

Сценарий 2. Изучение поведения основных характеристик протокола в зависимости от параметров моделируемой сети. Выделение набора параметров, оказывающих наибольшее влияние на поведение протокола. Дальнейшие эксперименты будут проводиться при фиксированных значениях параметров, не влияющих на функционирование протокола.

Сценарий 3. Сравнение протоколов ARTCP и TCP при различных значениях битовых ошибок передачи на канале.

Сценарий 4. Сравнение протоколов ARTCP и TCP по коэффициенту использования ПС при различных значениях числа потоков.

Сценарий 5. Сравнение ARTCP и TCP по коэффициенту равноправия разделения ПС при различных значениях числа потоков.

Сценарий 6. Произведем сравнение ARTCP и TCP по средней длине очереди при различных значениях числа потоков.

Сценарий 7. Рассмотрим детально взаимодействие одного потока ARTCP и CBR.

Сценарий 8. Рассмотрим детально взаимодействие двух ARTCP потоков и CBR потока.

Сценарий 9. Исследуем трафик протокола ARTCP на предмет наличия у него свойства самоподобия.

4.2. Сценарий 1: изолированный ARTCP

4.2.1. Задача

Задачей моделирования по сценарию 1 является детальная иллюстрация работы алгоритма протокола ARTCP в процессе адаптации к ПС канала. Проведем эксперименты при двух различных значениях ПС каналов: 32 Кб/с и 128 Кб/с.

4.2.2. Топология

Для исследования поведения протокола ARTCP в условиях свободных о влияния других транспортных потоков используется топология, изображенная на рис. 30. Стрелкой указано направление передачи данных. Узел, отмеченный знаком **S,** является отправителем, узел **R** - получателем. Набор каналов 0, 1 моделирует дуплексную коммутируемую ЛВС[18] или выделенный канал у отправителя, 2, 3 соответственно у получателя. Набор каналов 4, 5 моделирует канал типа точка-точка между двумя узлами территориальной сети.

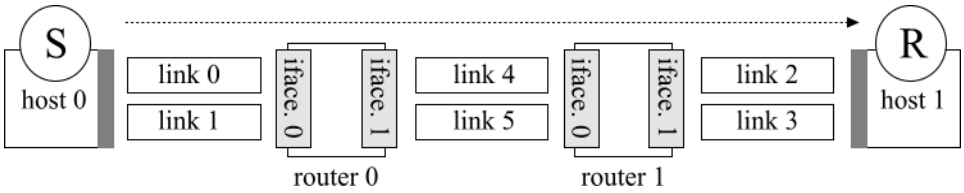

Рис. 30. Топологическая схема 1 с одним потоком и парой конечных систем.

4.2.3. Эксперимент 1: 32 Кб/с

Параметры:

Параметр	Значение
ПС каналов 0, 1, 2, 3	10 Мб/с[19]
Задержка каналов 0, 1, 2, 3	0.01 с
ПС каналов 4, 5	32 Кб/с
Задержка каналов 4, 5	0.1 с
Время моделирования	300 с
Макс. размер очереди маршрутизатора	16 Кбайт
BER	0

Результаты:

Характеристика	Значение
Коэффициент использования ПС в состоянии FT	97.81%
Средняя скорость потока	31302.64 б/с
Число потерянных сегментов	0
Число переданных сегментов	1173
$\overline{RTT}$ (усреднение за время эксперимента)	0.614 ± 0.146 с.
Минимальное RTT	0.519 с.
$\overline{Q}$ (усреднение за время эксперимента)	459.833 ± 716.51 байт

[18] обмен может независимо происходить в двух направлениях, канал использует только один узел. Это функциональный аналог сети IEEE802.3 [71].

[19] Приставка М означает: $\times 10^6$, К: $\times 10^3$

Иллюстрация функционирования протокола ARTCP в данных условиях приведена на рис. 31, 32, 33, 34.

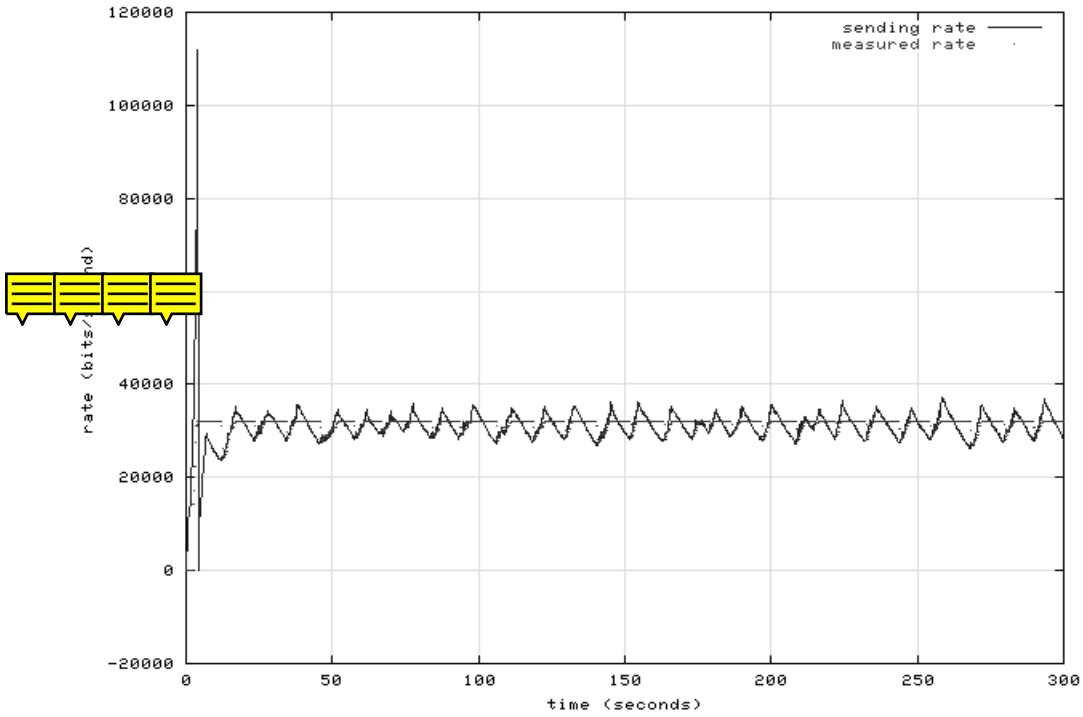

Рис. 31. Зависимость скорости потока от времени при скорости канала 32 Кб/с.

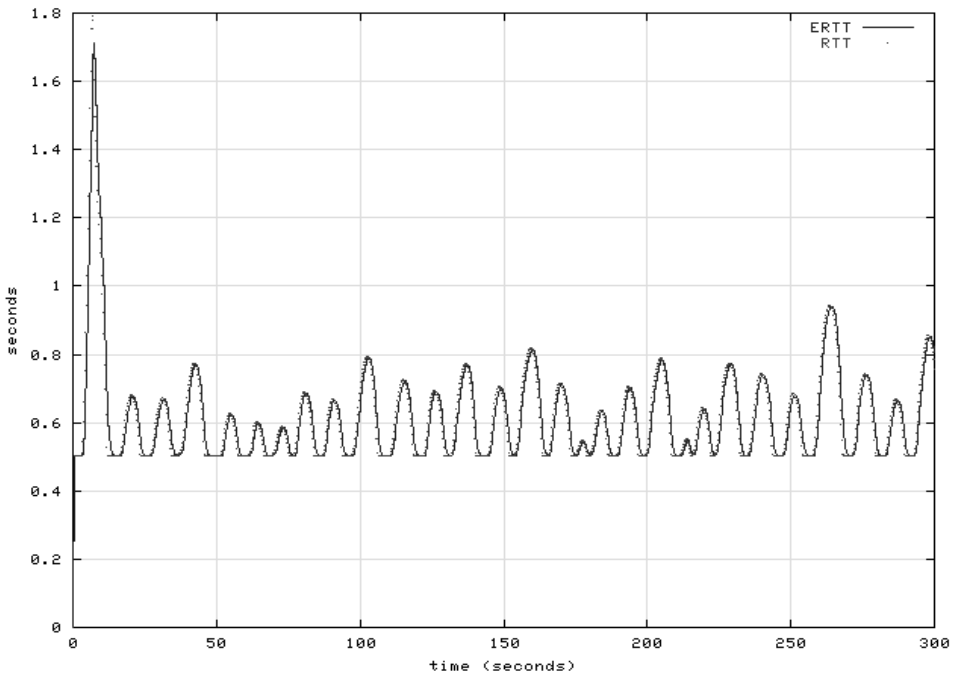

Рис. 32. Зависимость среднего и мгновенного значения RTT от времени при скорости канала 32 Кб/с. Sending rate - устанавливаемая скорость отправки, measured received rate - измеряемая скорость приема потока.

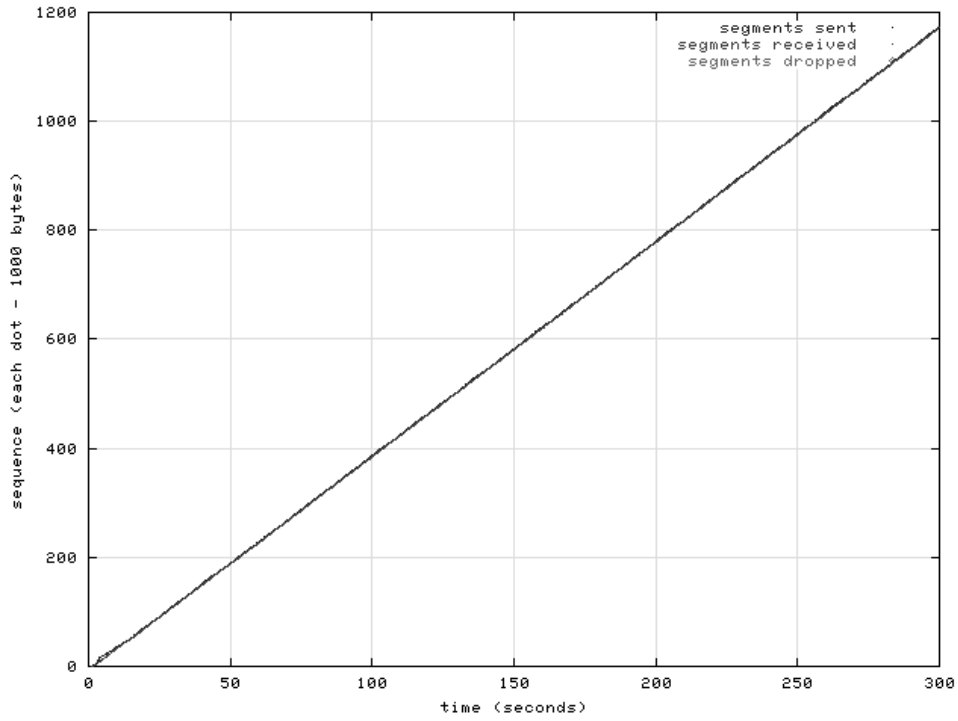

Рис. 33. Зависимость порядка передачи/приема сегментов от времени при скорости канала 32Кб/с. sent - отправленные, received - принятые, dropped - потерянные сегменты.

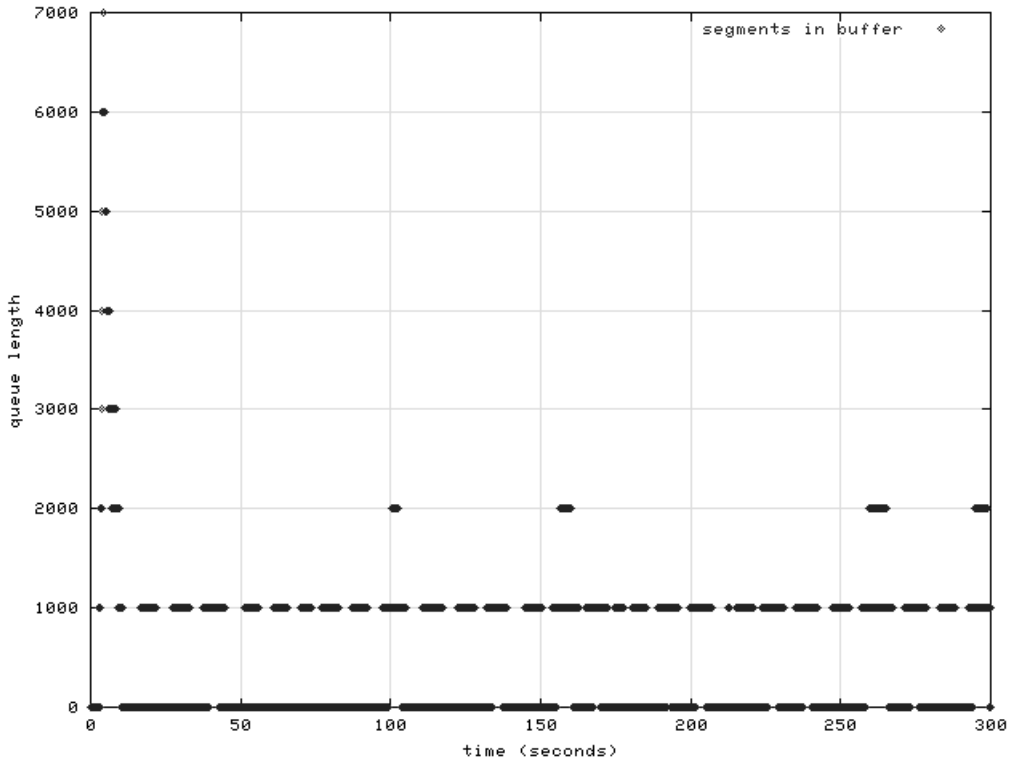

Рис. 34. Зависимость мгновенного значения длины очереди от времени.

4.2.4. Эксперимент 2: 128 Кб/с

Параметры:

Параметр	Значение
ПС каналов 0, 1, 2, 3	10 Мб/с
Задержка каналов 0, 1, 2, 3	0.01 с
ПС каналов 4, 5	128 Кб/с
Задержка каналов 4, 5	0.1 с
Время моделирования	300 с
Макс. размер очереди маршрутизатора	16 Кбайт

Результаты:

Характеристика	Значение
Коэффициент использования ПС в состоянии FT	95.38%
Средняя скорость потока	122285.64 б/с
Число потерянных сегментов	0
Число переданных сегментов	4558
$\overline{RTT}$	0.335 ± 0.048 с.
Минимальное RTT	0.307 с.
$\overline{Q}$	452.667 ± 872.59 байт

Функционирование протокола ARTCP в данных условиях проиллюстрировано на рис. 35, 36, 37, 38.

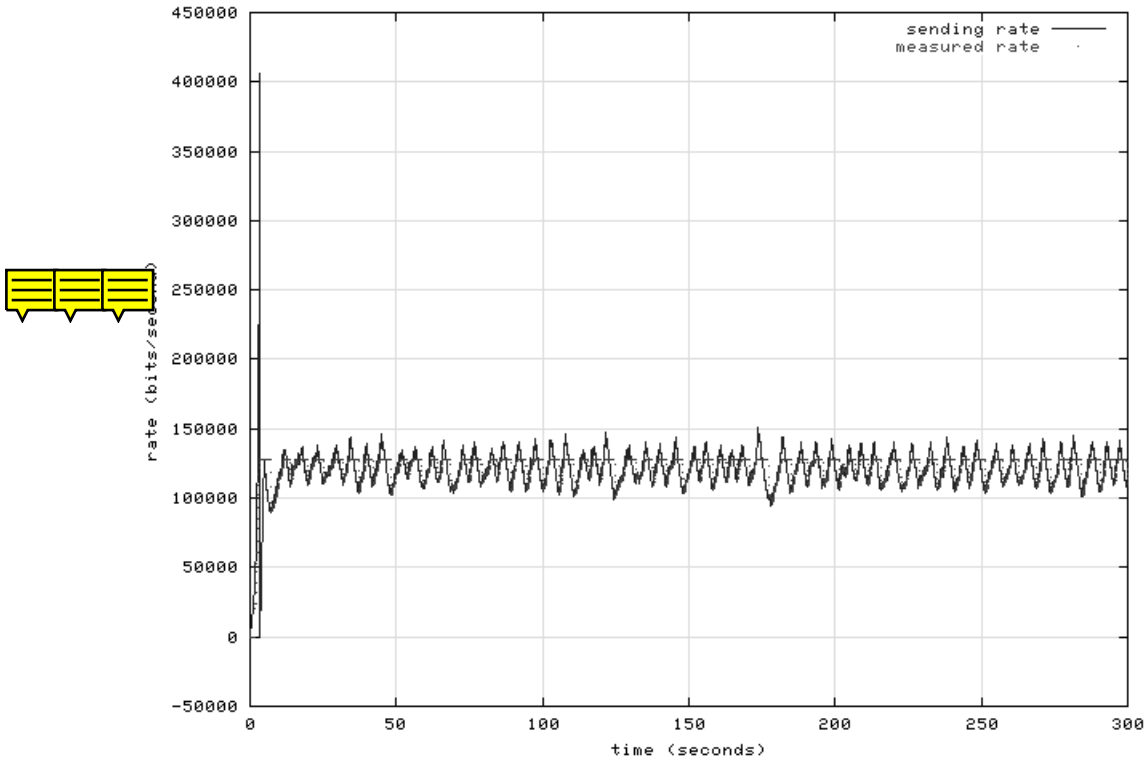

Рис. 35. Зависимость скорости потока от времени при скорости канала 128Кб/с.

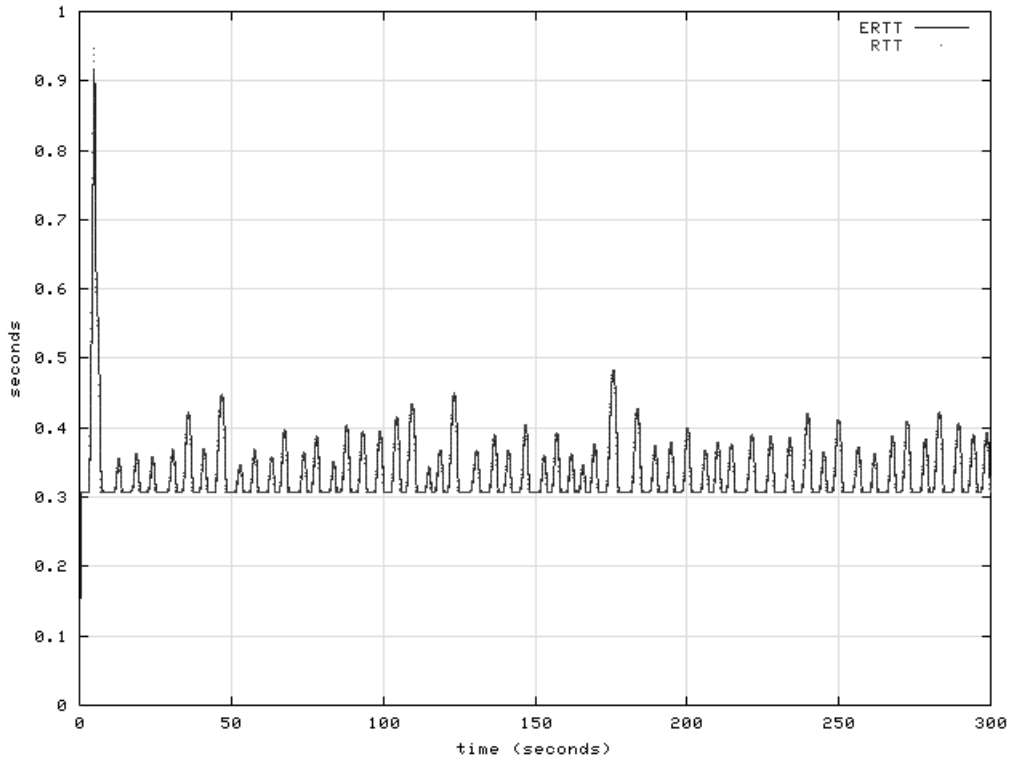

Рис. 36. Зависимость среднего и мгновенного значения RTT от времени при скорости канала 128Кб/с.

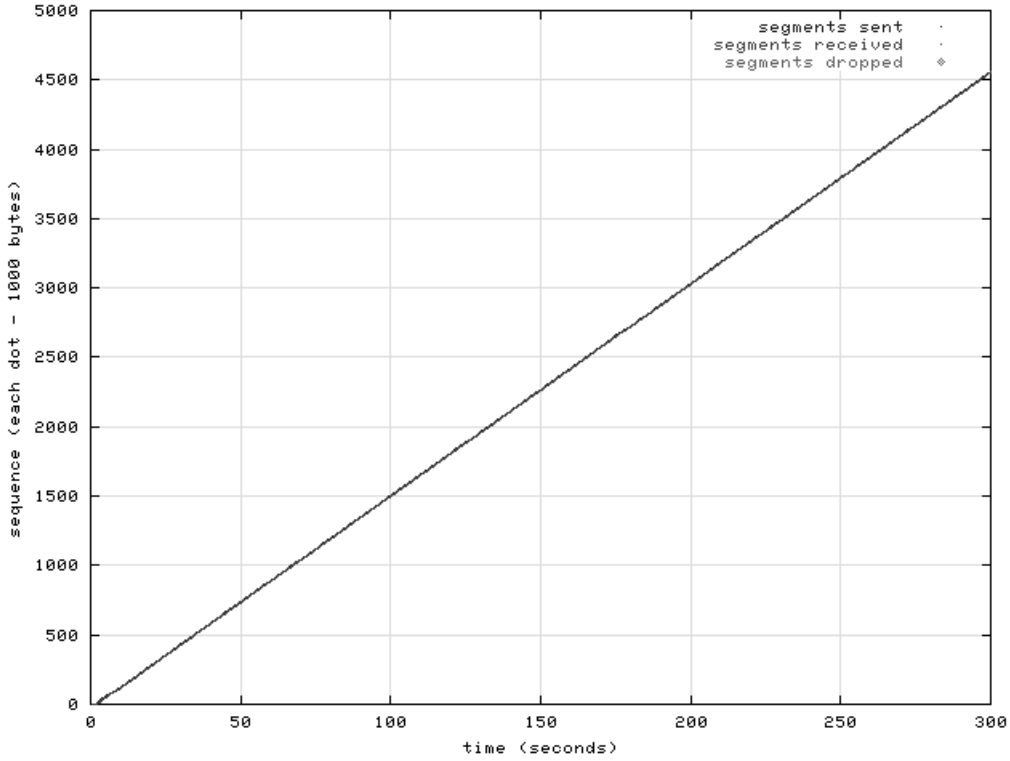

Рис. 37. Зависимость порядка передачи/приема сегментов от времени при скорости канала 128 Кб/с.

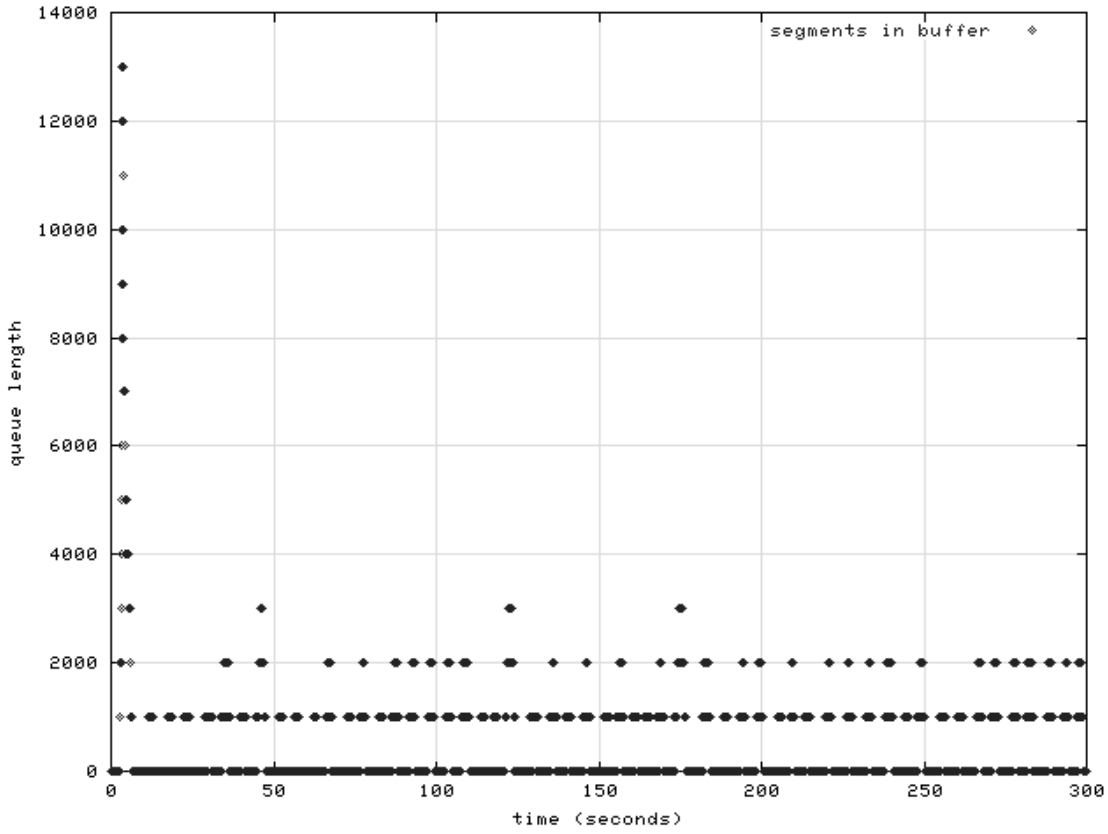

Рис. 38. Зависимость мгновенного значения длины очереди от времени.

4.2.5. Выводы

Эксперименты в сценарии 1 показали, как изолированный протокол ARTCP адаптируется к доступной ПС канала. При этом протокол ARTCP совершает переходы в необходимые режимы в полном соответствии с описанным ранее алгоритмом управления потоком. В двух проведенных экспериментах потерь пакетов не происходит. Для дальнейшего изучения протокола необходимо определить характер зависимости его показателей от основных параметров моделируемой сети.

4.3. Сценарий 2: определение важнейших параметров сети

4.3.1. Задача

Перед тем, как переходить к дальнейшим экспериментам, определим характер зависимости основных характеристик протокола от параметров сети.

Основными характеристиками протокола, которые определяются посредством статистической обработки данных модельного эксперимента, являются коэффициенты U, F и

средняя длина очереди Q. Нужно понять, зависят ли эти характеристики и если зависят, то как, от следующих параметров сети: ПС, число потоков, RTT, BER.

Пропускная способность сети является важнейшей ее характеристикой. Поскольку протокол ARTCP должен адаптироваться к доступной ПС сети, то именно значение ПС канала может оказывать наибольшее влияние на функционирование протокола.

Коэффициент равноправия разделения ПС не должен зависеть от ПС канала, поскольку вероятность увеличения скорости потока ARTCP, в соответствии с его алгоритмом, определяется лишь его скоростью потока и разностью между минимальным и измеряемым RTT. Именно поэтому коэффициент F не должен зависеть от RTT.

Таким образом, необходимо провести эксперименты для получения зависимостей характеристик ARTCP от ПС канала для разного числа соединений. По результатам экспериментов вычислим средние значения U, F, Q. Далее проверим, существует ли для этих коэффициентов зависимость от RTT. Влияние вероятности BER на характеристики ARTCP будем изучать при сравнении ARTCP с TCP.

4.3.2. Топология

Для проведения измерений при разных значениях числа потоков были произведены эксперименты на 10-ти вариантах сетевой топологии, содержащих от 2 до 20 узлов, между которыми соответственно существовало 1-10 одновременных потоков. Общая схема эксперимента приведена на рис. 39. Значение ПС каналов между маршрутизаторами R1 и R2 изменяется в пределах 64 Кб/с и 2048 Кб/с.

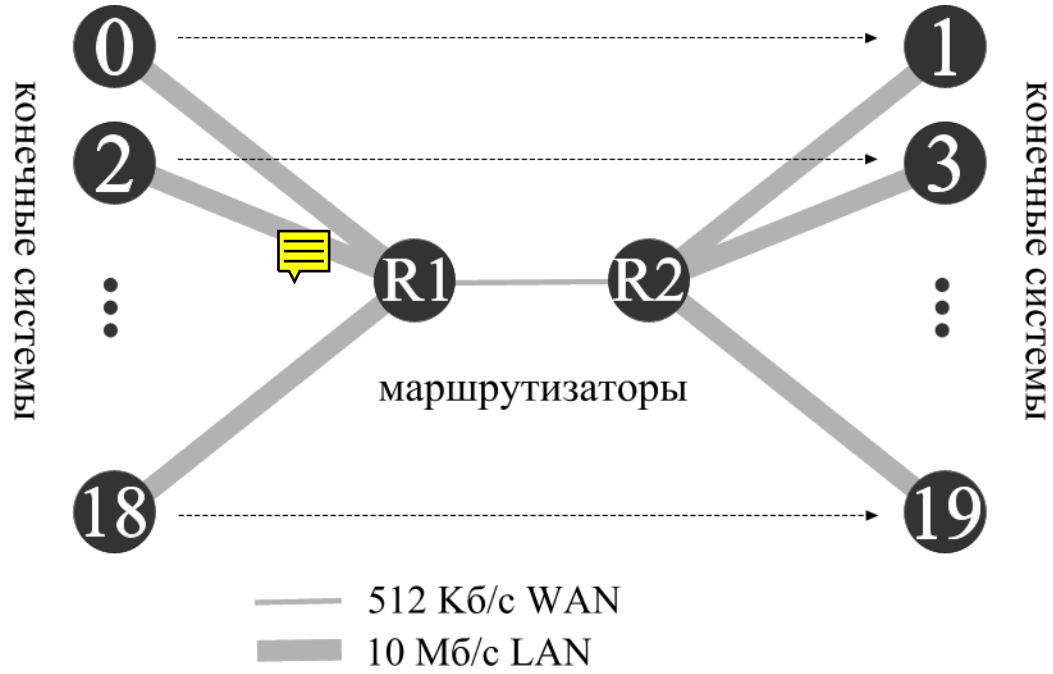

Рис. 39. Топологическая схема 10, с 20-ю парами источник-получатель.

4.3.3. Эксперимент 1: влияние ПС и числа потоков

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	От 64 Кб/с до 2.048 Мб/с, шаг 32 Кб/с
Задержка каналов WAN	0.1 с
Длительность эксперимента	500 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков ARTCP	От 1 до 9
Число экспериментов	50 по каждому значению ПС

Проведем по 50 экспериментов с одним потоком ARTCP для каждого значения ПС сети из диапазона 64-2048 Кб/с при шаге 32 Кб/с. Длительность каждого эксперимента в полученной серии из 3050 равна 300 с. По данным каждого эксперимента определяем значения U, F, Q. Проведем такие же серии измерений для сети с 2, 3, …, 9 ARTCP потоками. На рис. 40, 41 приведены графики зависимости коэффициента U от ПС канала для 1, 2, … 9 потоков.

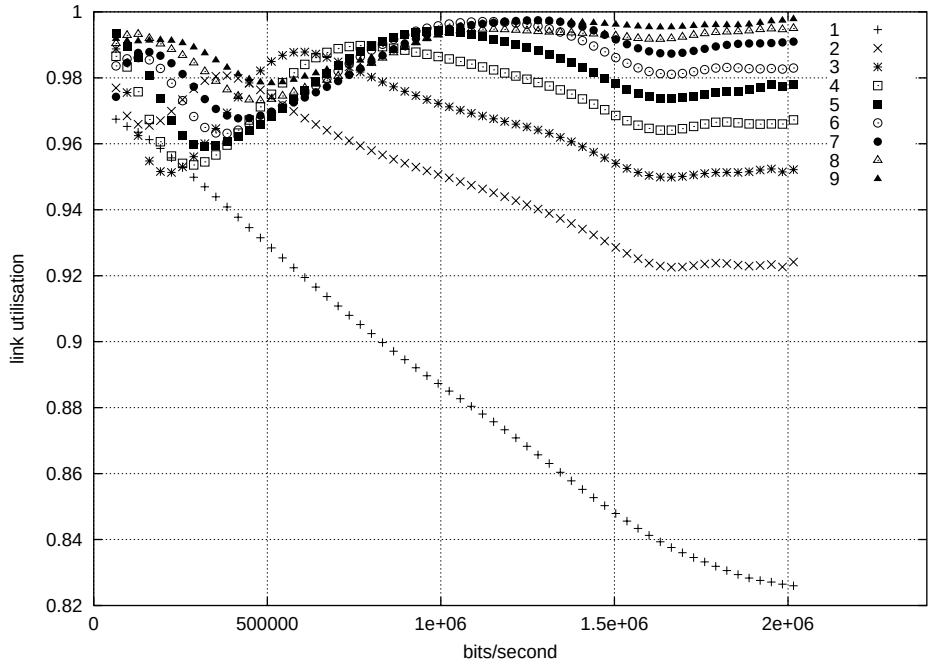

Рис. 40. Зависимость коэффициента использования ПС от ПС канала для 1-9 потоков ARTCP.

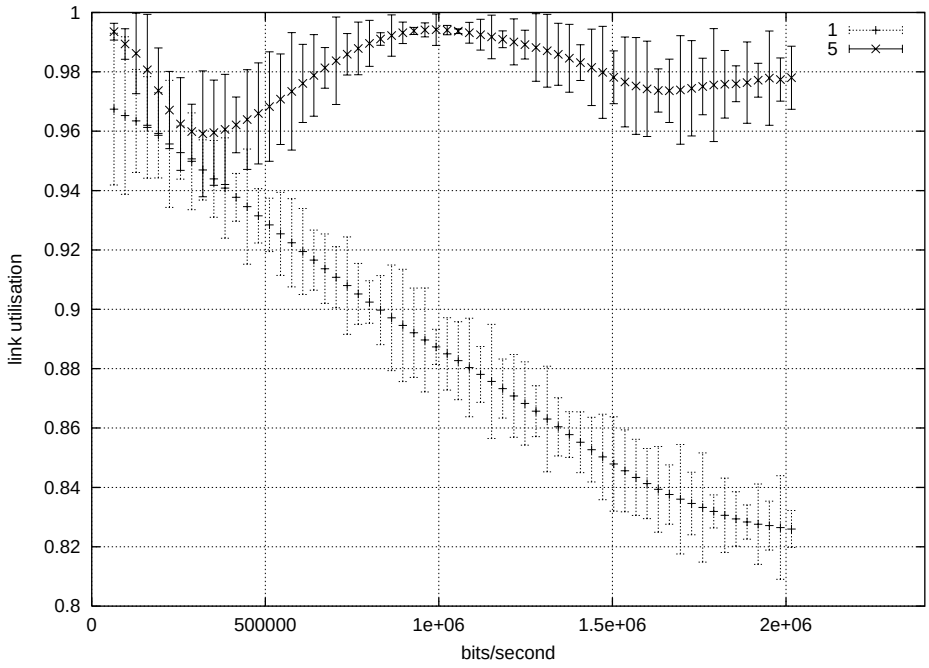

Рис. 41. Зависимость коэффициента использования ПС от ПС канала для 1 и 5 потоков. Отрезками обозначено 2σ.

По графику можно сделать вывод, что в случае одного потока ARTCP существует зависимость между U и ПС канала. Коэффициент использования ПС канала для одного ARTCP потока снижается с ростом ПС. Однако даже при ПС канала равной 2 Мб/с эффективность использования ПС для ARTCP не меньше 0.825. Для двух потоков ARTCP снижение коэффициента U с ростом ПС происходит медленнее и прекращается на значении $U \approx 0.925$. Для большего числа ARTCP потоков коэффициент U не падает с ростом ПС и приближается к единице для всего диапазона ПС с ростом числа потоков. Для числа потоков более 5 среднее значение коэффициента U не опускается ниже 0.95.

Падение эффективности использования сети одним ARTCP потоком при росте ПС можно объяснить консервативностью алгоритма управления потоком ARTCP, стремящегося избежать любого накопления данных в буфере маршрутизатора. В режиме FT вероятность снижения скорости при росте RTT выше, чем вероятность увеличения скорости. Кроме того, вероятность увеличения скорости тем ниже, чем выше развитая соединением скорость (см. коэффициент speedup). Если же потоков ARTCP несколько, то снижение скорости одного из них компенсируется временным повышением скорости другого, поэтому с ростом числа потоков общее поведение системы становится более стабильным и ресурсы сети используются более полно.

Коэффициент равноправия разделения ресурсов для любого числа ARTCP потоков не зависит от ПС сети и близок к единице (рис. 42). Средняя длина очереди Q зависит лишь от числа потоков и не зависит от ПС сети (рис. 47).

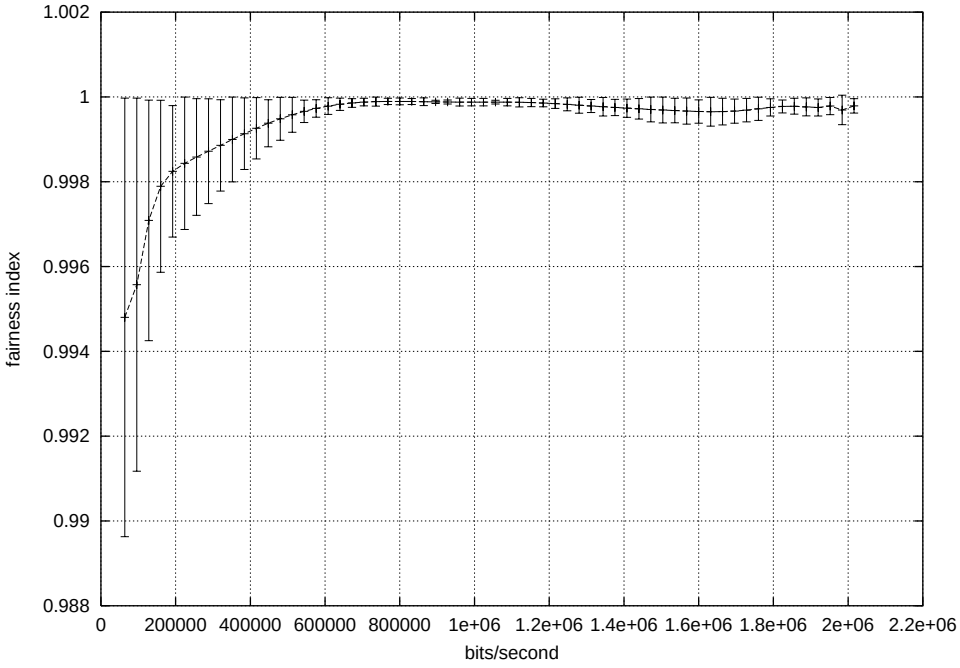

Рис. 42. Зависимость коэффициента равноправия разделения ПС от ПС канала. Для числа потоков от 1 до 9.

Итак, коэффициент использования ПС для нескольких одновременных ARTCP потоков близок к единице при всех значениях ПС сети от 64 Кб/с до 2.048 Мб/с.

4.3.4. Эксперимент 2: влияние RTT

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	54, 128, 256, 512 Кб/с
Задержка каналов WAN	От 0.01 до 0.33 с при шаге 0.02 с
Длительность эксперимента	300 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков	От 1 до 9
Число экспериментов	По одному для каждого значения задержки

Проверим теперь наличие зависимости характеристик протокола от значения RTT. Предполагаем, что ни один из параметров U, F и Q не зависит от RTT, поскольку алгоритм ARTCP предполагает зависимость лишь от разности текущего и минимального значения

RTT, а не от его абсолютного значения. Для проверки этой гипотезы проведем эксперимент длительностью 300 с для каждого из значений задержки передачи каналов WAN от 0.01 до 0.33 с шагом 0.02 для каждого из значений ПС: 64, 128, 256 и 512 Кб/с и каждого из 9 значений числа потоков. 36 средних значений U, полученных из 36 серий по 16 измерений не отличаются от соответствующих данному числу потоков и значению ПС результатов предыдущего эксперимента. Кроме того, в каждой из 36 серий не наблюдается зависимости U от RTT. Аналогичные измерения были проведены и для значений Q и F, которые также не обнаружили зависимости от RTT.

4.3.5. Выводы

Таким образом, коэффициент использования ПС сети (U) зависит от ПС сети лишь для одного ARTCP потока, при наличии более 5 ARTCP потоков можно пренебречь зависимостью U от ПС. В этом случае при росте ПС значение U стабилизируется на величине тем более близкой к единице, чем больше число потоков. Кроме того, U не зависит от RTT соединения.

Коэффициент равноправия разделения ПС не зависит от RTT или числа потоков. Зависимость его от ПС сети очень слаба в изученных пределах (64-2048 Кб/с) и ей можно пренебречь.

Средняя длина очереди Q зависит лишь от числа потоков. С ростом количества одновременных соединений Q медленно растет.

Поскольку для числа потоков, превосходящего 5, коэффициенты U и F практически не зависят от ПС канала, то дальнейшее исследование ARTCP будем приводить при одном или нескольких фиксированных значениях ПС.

4.4. Сценарий 3: ARTCP и TCP в условиях ошибок передачи

4.4.1. Задача

Превосходство ARTCP над TCP должно наиболее ярко проявляться при работе по каналам, с ненулевой вероятностью битовых ошибок, поскольку в отличие от TCP, алгоритм протокола ARTCP нечувствителен к потерям сегментов.

Задачей экспериментов в этом сценарии является сравнение коэффициента использования ПС протоколами ARTCP и TCP при разных значениях BER. Поскольку для более 5 потоков ARTCP область изменения коэффициента U мала, то будем проводить исследования при нескольких фиксированных значениях ПС канала.

4.4.2. Топология

Для экспериментов по данному сценарию используется топологическая схема с 10 парами источник-получатель (рис. 39). Протокол TCP моделируется на такой же топологии в ПО NS. Время задержки передачи на канале с наименьшей ПС составляет 0.1 с в каждом направлении. Значения ПС канала фиксированы и составляют 256, 512 и 1024 Кб/с.

4.4.3. Эксперимент

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	256, 512, 1024 Кб/с
Задержка каналов WAN	От 0.1 с
Длительность эксперимента	500 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков	10
Число экспериментов	По 50 для каждого значения BER
BER	$[0,\ 6\times10^{-5}]$

Определим значения суммарной скорости 10 ARTCP и 10 TCP потоков, разделяющих общий канал (отдельно, для каждого протокола) с ПС 256 Кб/с и значениями BER из промежутка $[0,\ 6\times10^{-5}]$. По каждому значению BER проводим по 50 экспериментов длительностью 500 с. В приведенных ниже результатах используются суммарная достигнутая всеми соединениями скорость потока. На графике зависимости скорости потоков от времени (рис. 43) ясно видно, что, начиная со значения 1×10^{-5}, скорость TCP резко снижается, а скорость ARTCP потока остается близкой к максимальной скорости.

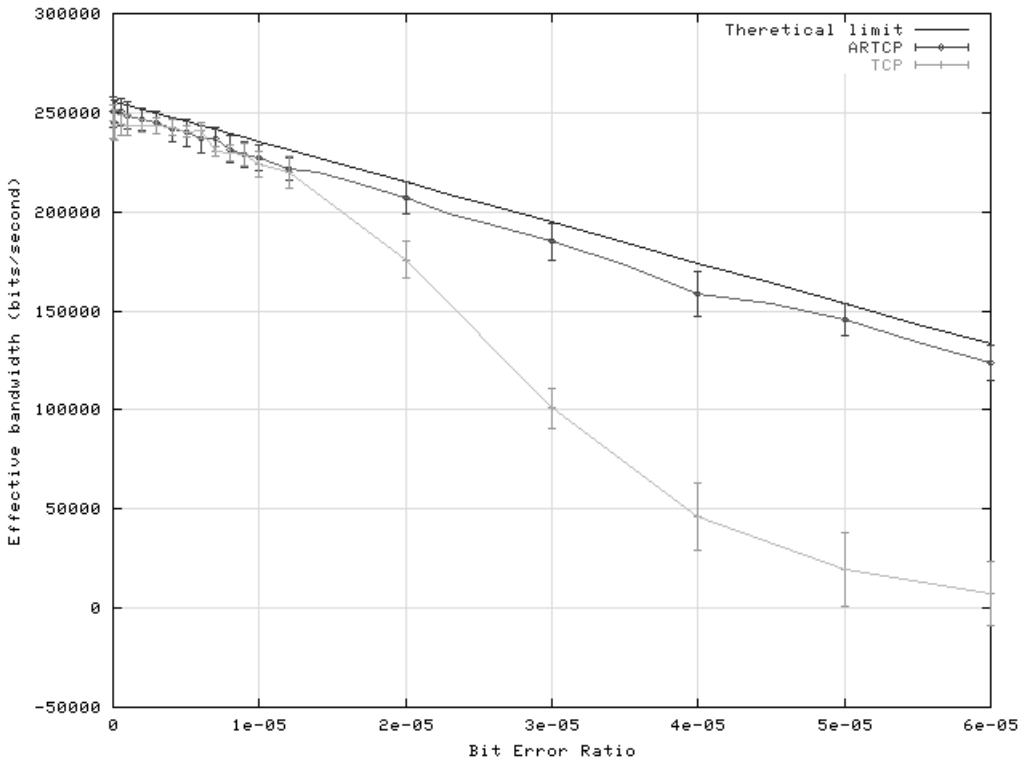

Рис. 43. Зависимость коэффициента использования ПС от вероятности битовых ошибок канала. ПС канала равна 256 Кб/с.

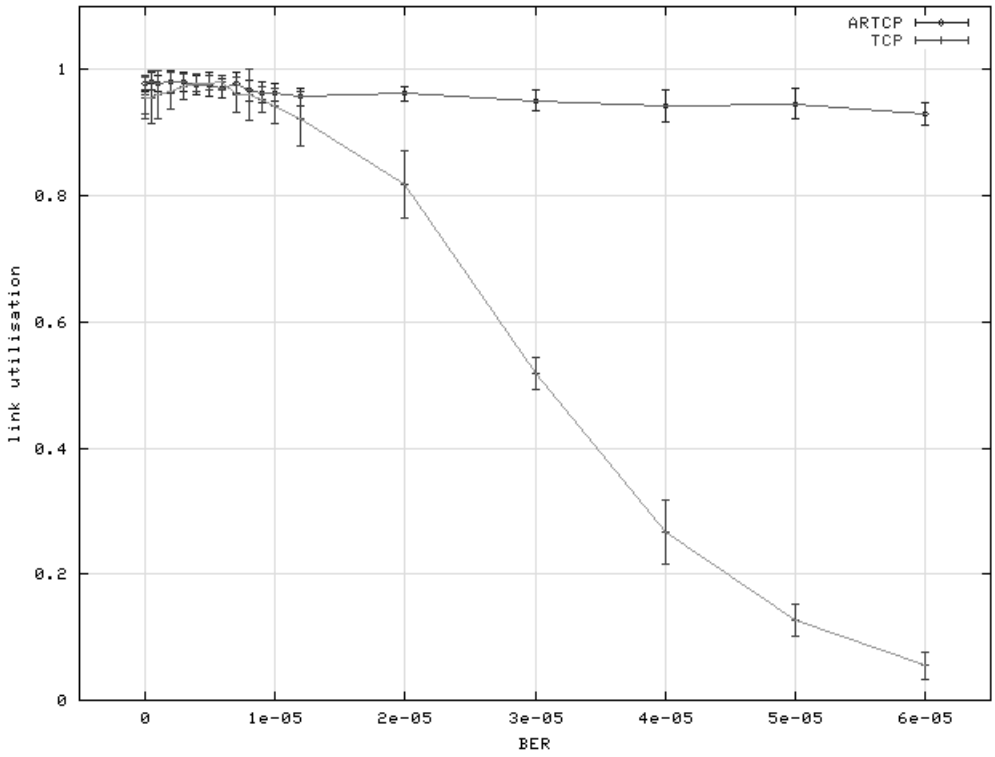

Рис. 44. Зависимость коэффициента использования ПС от вероятности битовых ошибок канала.

Максимальная ПС в данном случае вычисляется как $ПС \times (1 - BER)^S$. Для обобщения результатов построим график зависимости U от BER. Такой график представлен на рис. 44. Проведем аналогичную серию экспериментов для ПС канала, равной 512 и 1024 Кб/с. Как и следовало ожидать, экспериментальные значения зависимости U от ПС для других значений ПС канала практически неотличимы от уже полученной зависимости при ПС=256 Кб/с. Это происходит потому, что, как показано ранее, коэффициент U для большого числа ARTCP потоков почти не зависит от ПС канала.

4.4.4. Выводы

Как видно на рис. 44, эффективность использования ПС канала протоколом ARTCP не зависит от вероятности битовых ошибок на канале. На канале с вероятностью битовых ошибок превышающей 1×10^{-5} протокол ARTCP существенно превосходит TCP по эффективности использования ПС.

4.5. Сценарий 4: ARTCP и TCP - коэффициент использования

4.5.1. Задача

Создавая искусственную перегрузку в сети, TCP приводит к потерям сегментов, ретрансляция которых снижает эффективность TCP по сравнению с ARTCP. Вследствие этого, коэффициент использования ПС канала для TCP должен снижаться с увеличением числа потоков. Необходимо провести эксперимент для получения зависимости коэффициента U для протокола TCP в зависимости от ПС для разного числа потоков.

4.5.2. Топология

Для моделирования 1-9 потоков ARTCP будем проводить эксперимент на топологических схемах использованных в эксперименте 1 сценария 2.

4.5.3. Эксперимент

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	От 64 Кб/с до 2.048 Мб/с, шаг 32 Кб/с
Задержка каналов WAN	0.1 с
Длительность эксперимента	500 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков TCP	От 1 до 9
Число экспериментов	50 по каждому

	значению ПС

Проведем серию измерений, аналогично, сценарию 2, теперь для протокола TCP. Значения коэффициента U для TCP проявляют выраженную тенденцию к снижению при росте числа потоков (рис. 45).

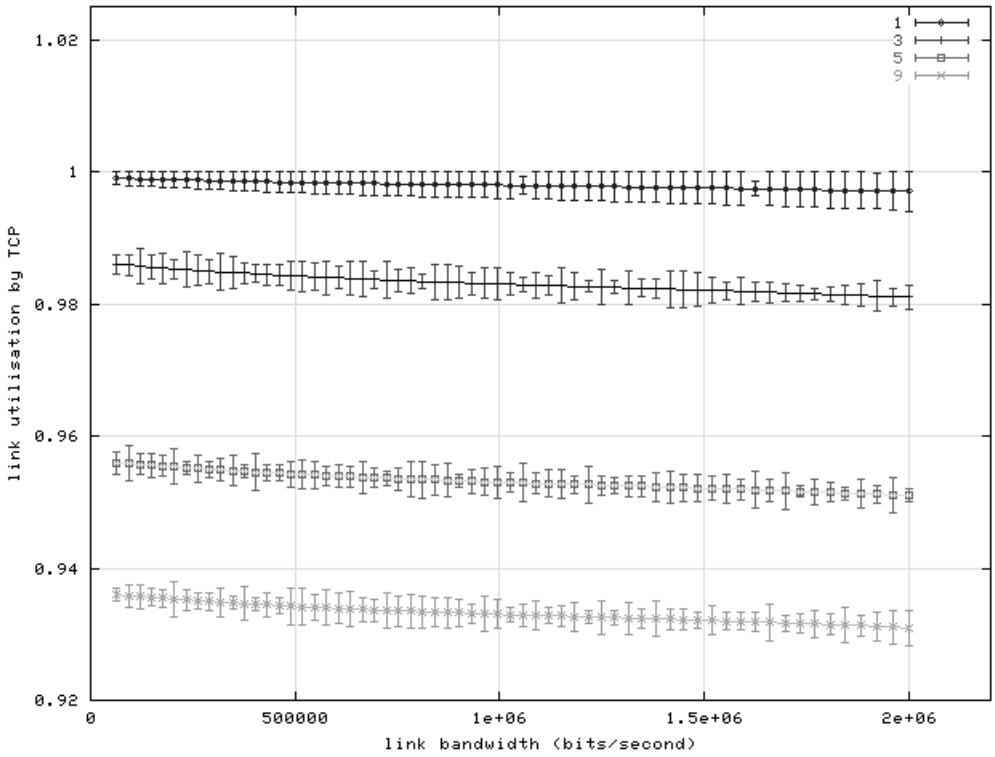

Рис. 45. Зависимость коэффициента использования ПС от ПС сети для 1, 3, 5, 9 одновременных потоков TCP

Для изолированного TCP соединения значение коэффициента U всегда близко к единице. Это является следствием возникновения так называемой синхронизации по подтверждениям, которая в случае наличия одного соединения поддерживает его скорость равной скорости канала без возникновения потерь сегментов. Однако при увеличении числа потоков возникают потери сегментов (уже в случае двух потоков), которые приводят к снижению эффективности протокола. Так для 10 потоков коэффициент U составляет примерно 0.935.

Сравнение рис. 45 и рис. 40 показывает явное превосходство ARTCP перед TCP при росте числа потоков.

4.5.4. Выводы

Таким образом, даже в случае традиционных проводных сетей эффективность протокола ARTCP выше по сравнению с TCP уже при числе потоков равном 5 и более.

Для небольшого числа активных соединений эффективность использования ПС канала для TCP несколько выше. Это происходит за счет того, что TCP полностью заполняет очередь на выходном интерфейсе маршрутизатора так, что обслуживающее устройство никогда не простаивает. При таких же условиях ARTCP стремясь обеспечить минимальное заполнение очереди не всегда успевает передать очередной пакет. С ростом количества соединений растет и средняя длина очереди в маршрутизаторе для ARTCP, а коэффициент использования ПС канала приближается к единице. В случае протокола TCP рост числа соединений приводит к увеличению вероятности потери сегмента за счет переполнения очереди (при использовании дисциплины управления очередью DropTail[20]) или увеличения вероятности отбрасывания данных (при использовании дисциплины RED) [79]. Ретрансляции отброшенных сегментов приводят к уменьшению эффективности использования ресурсов сети. В экспериментальной топологии данный эффект минимален, поскольку потерянные пакеты проходят лишь высокоскоростной канал, соединяющий отправителя с первым маршрутизатором. Однако, как продемонстрировано в работе [78], для более сложной сети, в состав которой входят несколько перегруженных каналов эффект ретрансляции пакетов существенно снижает эффективность использования ресурсов.

4.6. Сценарий 5: ARTCP и TCP - коэффициент равноправия

4.6.1. Задача

Рассмотрим поведение коэффициента равноправия разделения ПС для протоколов TCP и ARTCP в зависимости от числа соединений. Поведение коэффициента F для ARTCP было исследовано в экспериментах по сценарию 2. Было показано, что коэффициент F в случае ARTCP фактически не зависит от числа потоков и ПС канала. Необходимо определить поведение коэффициента F для протокола TCP. Как было показано в [80], коэффициент равноправия разделения ПС для TCP не зависит от скорости канала. Поставим эксперимент для определения значения F при разном числе потоков TCP и фиксированном значении ПС.

4.6.2. Топология

Для проведения эксперимента используется топологическая схема идентичная схеме эксперимента 2, однако, значение ПС канала фиксировано и равно 256 Кб/с.

---

[20] отбрасывание последнего полученного пакета в том случае, если количество свободного пространства в очереди не позволяет разместить в ней пакет данного размера.

4.6.3. Эксперимент

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	256 Кб/с
Задержка каналов WAN	0.1 с
Длительность эксперимента	100 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков TCP	От 1 до 9
Число экспериментов	100 по каждому значению числа потоков

Для каждого из значений числа потоков от 1 до 9 проводим по 100 модельных экспериментов длительностью 100 с. По каждой из 9 серий определяем среднее значение коэффициента F (рис. 46) для каждого значения числа одновременных потоков.

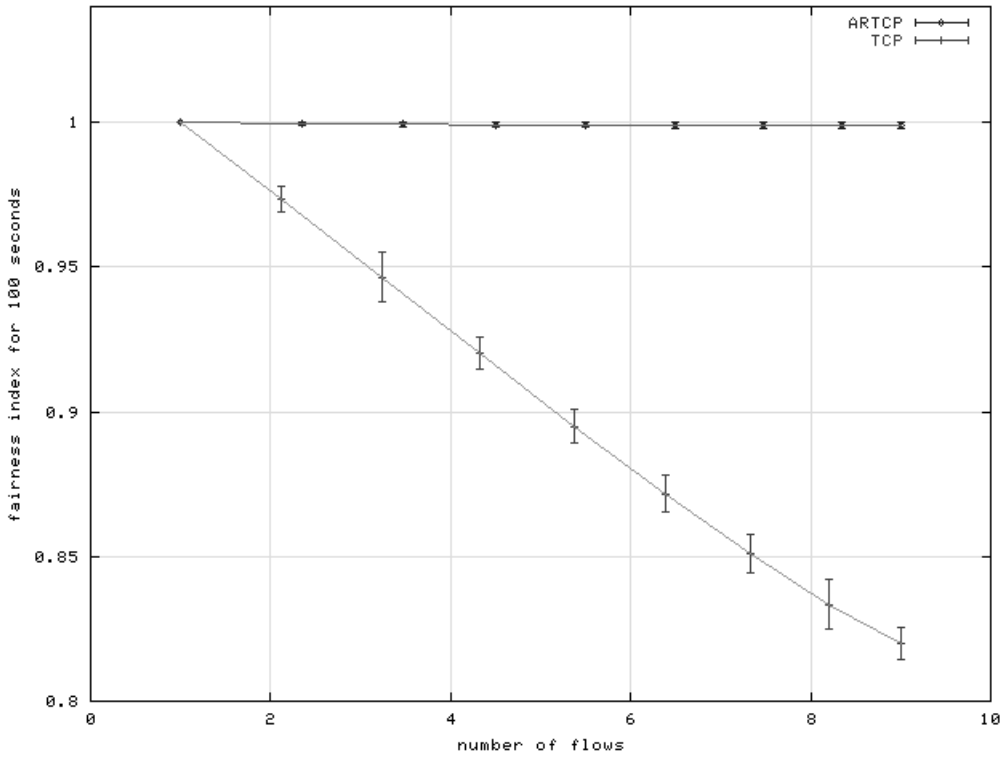

Рис. 46. зависимость коэффициента равноправия разделения ПС от числа потоков. ARTCP и TCP. Время измерения 100 с.

Главным отличием здесь является то, что для ARTCP коэффициент равноправия разделения ПС растет при росте числа соединений, в то время как для TCP увеличение количества соединений приводит к снижению равноправия разделения ПС.

4.6.4. Выводы

Соединения протокола ARTCP более равноправны между собой, чем TCP, причем с ростом числа соединений для протокола TCP значение F снижается, а для ARTCP постоянно и близко к 1.

4.7. Сценарий 6: ARTCP и TCP средняя длина очереди

4.7.1. Задача

За счет использования более консервативного механизма определения максимальной доступной ПС, протокол ARTCP во всех случаях должен обеспечивать существенно меньшую, чем TCP среднюю длину очереди в маршрутизаторах. Проверим эти утверждения на экспериментальных данных. Эксперимент будем проводить с фиксированным значением ПС при различном числе потоков. Зависимость средней длины очереди от числа ARTCP потоков была получена в сценарии 2, поэтому здесь необходимо установить эту зависимость для TCP.

4.7.2. Топология

Для проведения эксперимента используется топологическая схема идентичная схеме эксперимента 2, однако, значение ПС канала фиксировано и равно 256 Кб/с.

4.7.3. Эксперимент

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	256 Кб/с
Задержка каналов WAN	0.1 с
Длительность эксперимента	500 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков TCP	От 1 до 9
Число экспериментов	100 по каждому значению числа потоков

Для каждого числа потоков от 1 до 9 проводим 100 экспериментов длительностью 500 с. По данным 9 серий определим средние значения длины очереди (рис. 47).

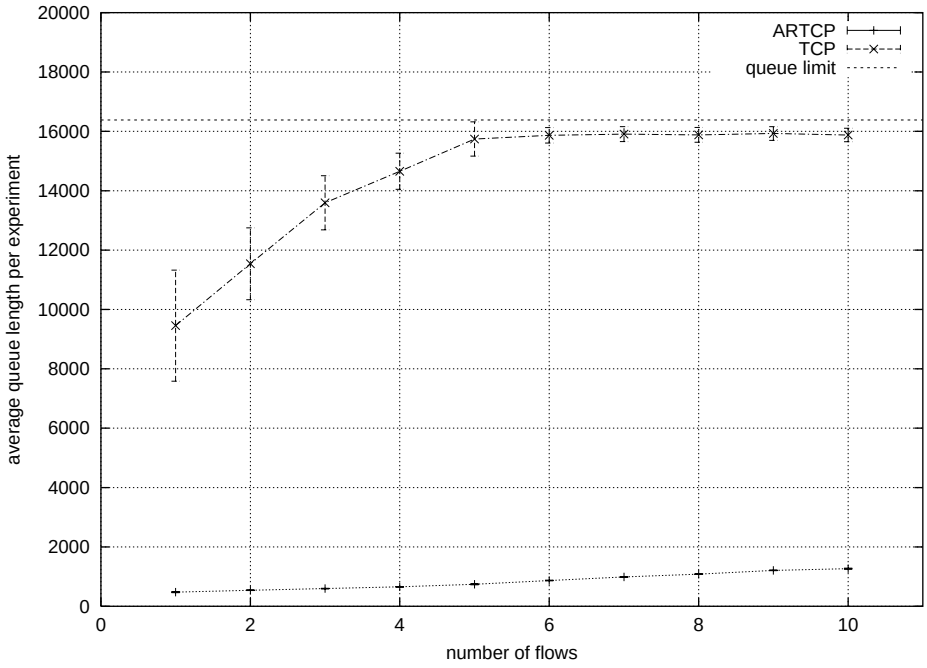

Рис. 47. Зависимость средней длины очереди от количества соединений.

Протокол TCP определяет доступную пропускную способность сети, полностью насыщая ее трафиком, и вызывая переполнение очередей в сетевых устройствах. Вследствие этого средняя длина очереди при моделировании TCP потоков всегда является максимальной (начиная с 5-ти активных соединений). Следствием этого являются постоянно происходящие потери пакетов, что проиллюстрировано на рис. 49 в отличие от ARTCP, заполнение очередей для которого минимально и отсутствуют связанные с переполнением буфера потери пакетов (рис. 48).

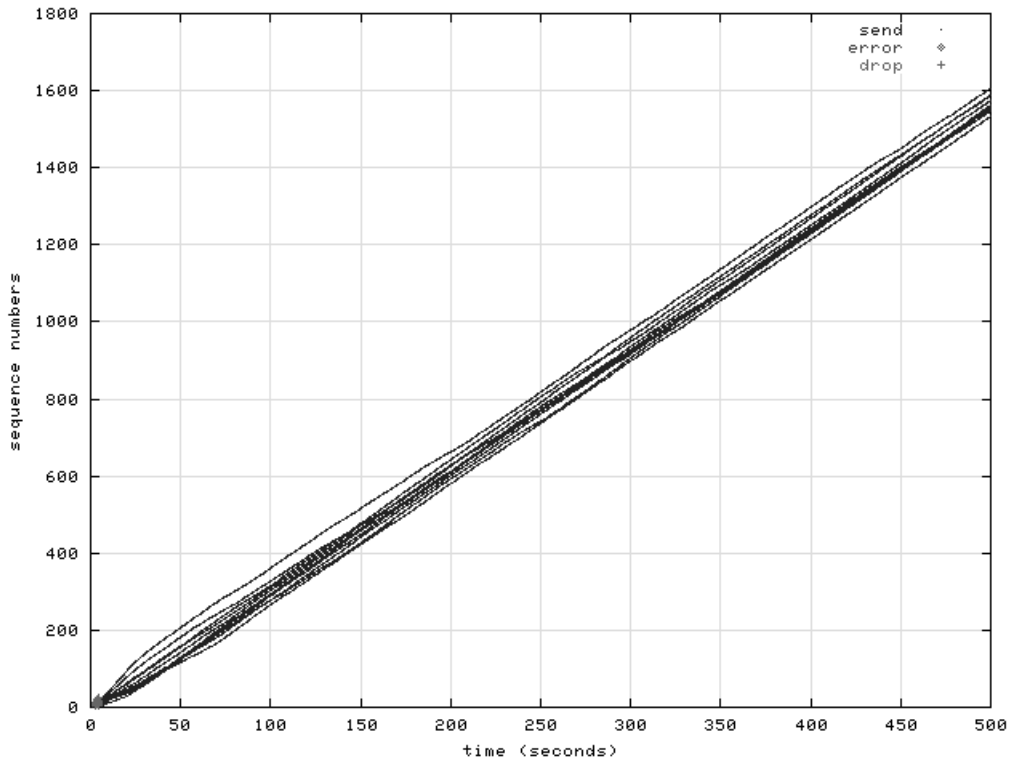

Рис. 48. Последовательность передачи для 10-ти ARTCP потоков при пропускной способности канала 256 Кб/с.

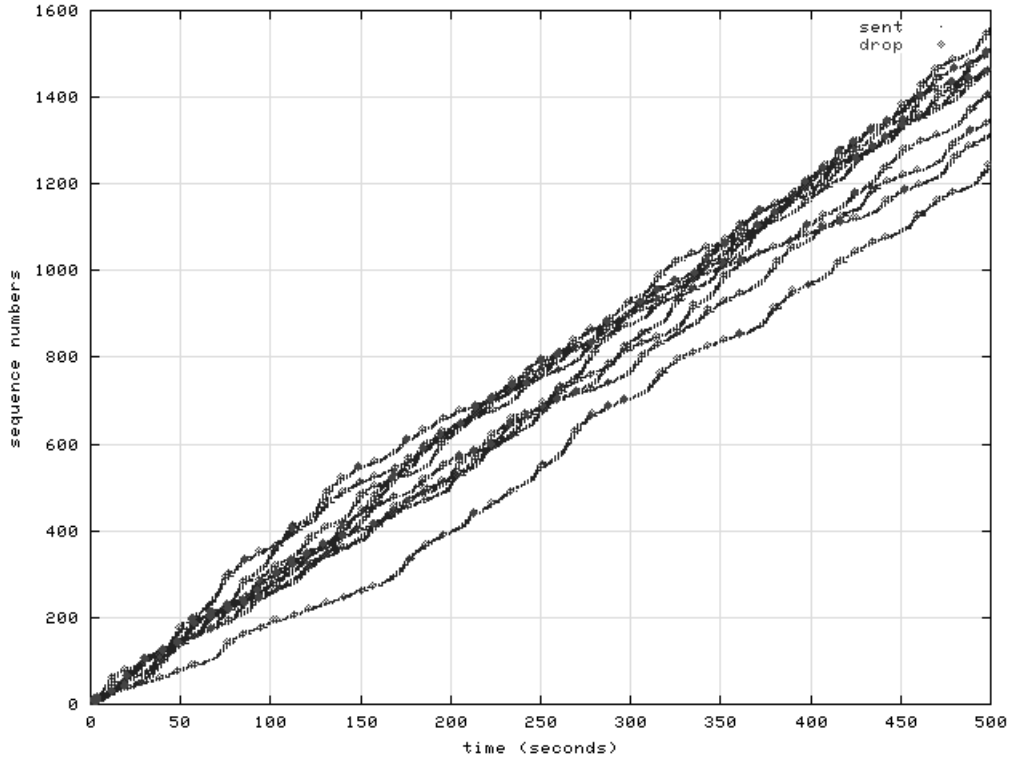

Рис. 49. Последовательность передачи для 10-ти TCP потоков при пропускной способности канала 256 Кб/с).

4.7.4. Выводы

В отличие от протокола TCP, который максимально заполняет очередь маршрутизатора, стремясь определить ПС сети, ARTCP не допускает переполнения очередей, поддерживая минимальной среднюю длину очереди. Благодаря этому при работе ARTCP потери сегментов не происходят, а значение времени RTT близко к минимуму. Сокращение средней длины очереди также является важным преимуществом протокола ARTCP.

4.8. Сценарий 7: 1 ARTCP и 1 CBR

4.8.1. Задача

Рассмотрим взаимодействие одного ARTCP потока с одним CBR потоком. Протокол ARTCP должен эффективно использовать оставшуюся от потока CBR пропускную способность сети. Работа соединения происходит в двух фазах: фаза, когда существует только один ARTCP поток на канале и фаза, когда включен CBR. Причем, после включения CBR потока, ARTCP, занимавший ранее всю ПС, должен снизить скорость своего потока. В этом сценарии CBR поток включается позже, чем ARTCP, поскольку нас интересует именно адаптация ARTCP к ПС канала, после ее скачкообразного понижения на величину скорости CBR (R_{CBR}). Проверим, существует ли зависимость между коэффициентом U и значением (ПС- R_{CBR}), при фиксированном значении ПС канала 256 Кб/с.

4.8.2. Топология

Для проведения эксперимента в данном сценарии используется топологическая схема с двумя источниками и двумя получателями, приведенная на рис. 50.

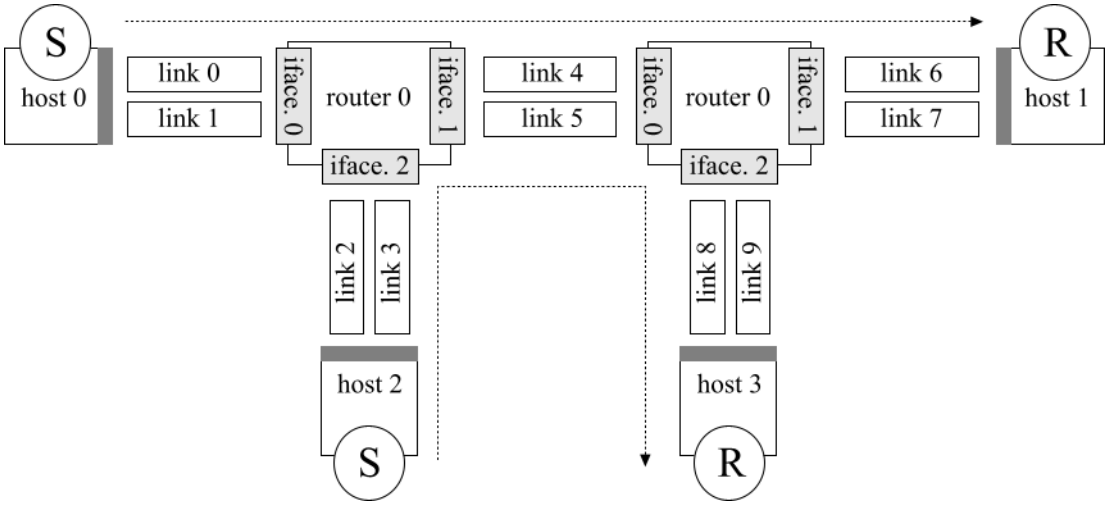

Рис. 50. Элементы топологии 2.

4.8.3. Эксперимент

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	256 Кб/с
Задержка каналов WAN	0.1 с
Длительность эксперимента	500 с
Макс. размер очереди маршрутизатора	32 Кбайт
Число потоков ARTCP	1
Число потоков CBR	1
Скорость потока CBR	От 48 Кб/с до 208 Кб/с с шагом 16 Кб/с
Число экспериментов	100 по каждому значению R_{CBR}
Момент запуска потока CBR	Выбирается из интервала 90-110 с по случайному закону с равномерным распределением
Момент остановки CBR	Выбирается из интервала 390-410 с по случайному закону с равномерным распределением

Проводим по 100 измерений U для каждого значения CBR от 48 до 208 Кб/с с шагом 16 Кб/с. Полученные средние значения U по 100 измерений для каждого из значений разности ПС-R_{CBR} фактически не зависят от (ПС-R_{CBR}). Полученное значение U составляет 0.9726 ± 0.003 (рис. 51).

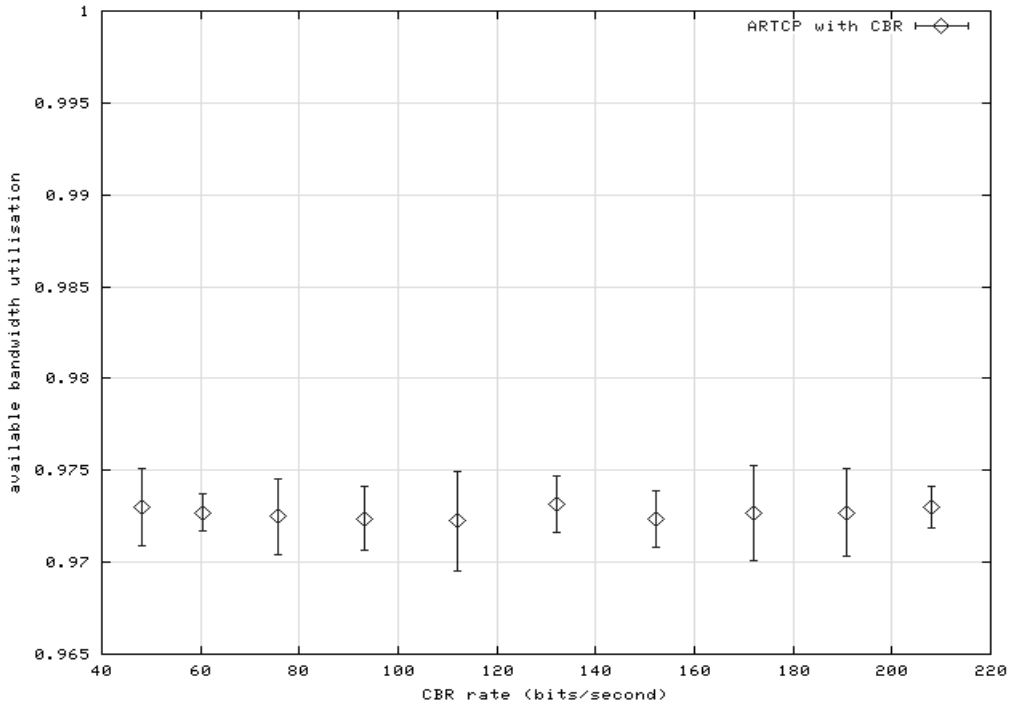

Рис. 51. Значения коэффициента использования ПС протоколом ARTCP при различных значениях скорости протокола CBR. ПС канала составляет 256 Кб/с.

Поведение соединений в типичном эксперименте этого сценария проиллюстрировано на рис. 52, 53, 54, 55.

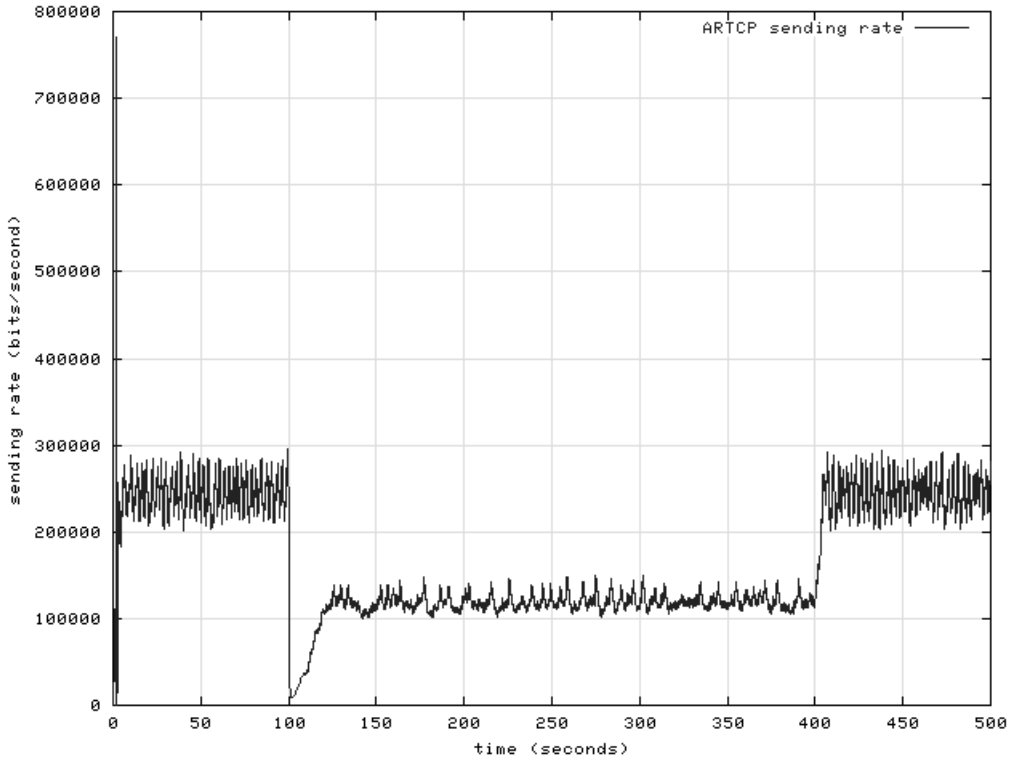

Рис. 52. Зависимость скорости потока от времени.

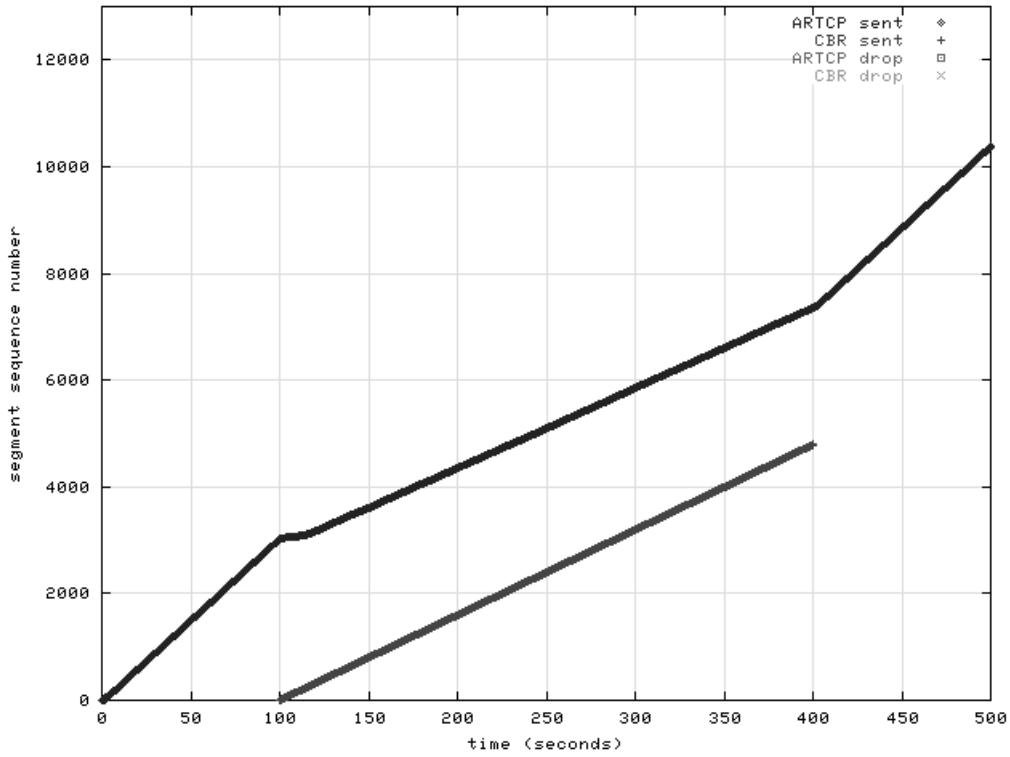

Рис. 53. Зависимость последовательности передачи данных от времени.

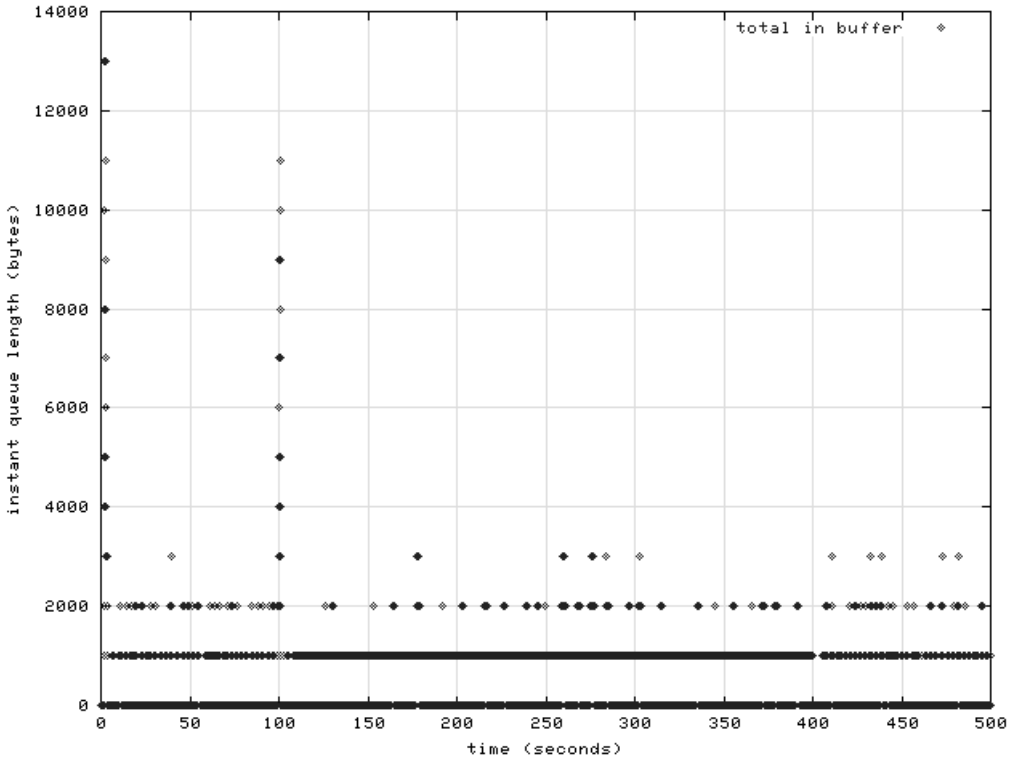

Рис. 54. Зависимость мгновенной длины очереди от времени.

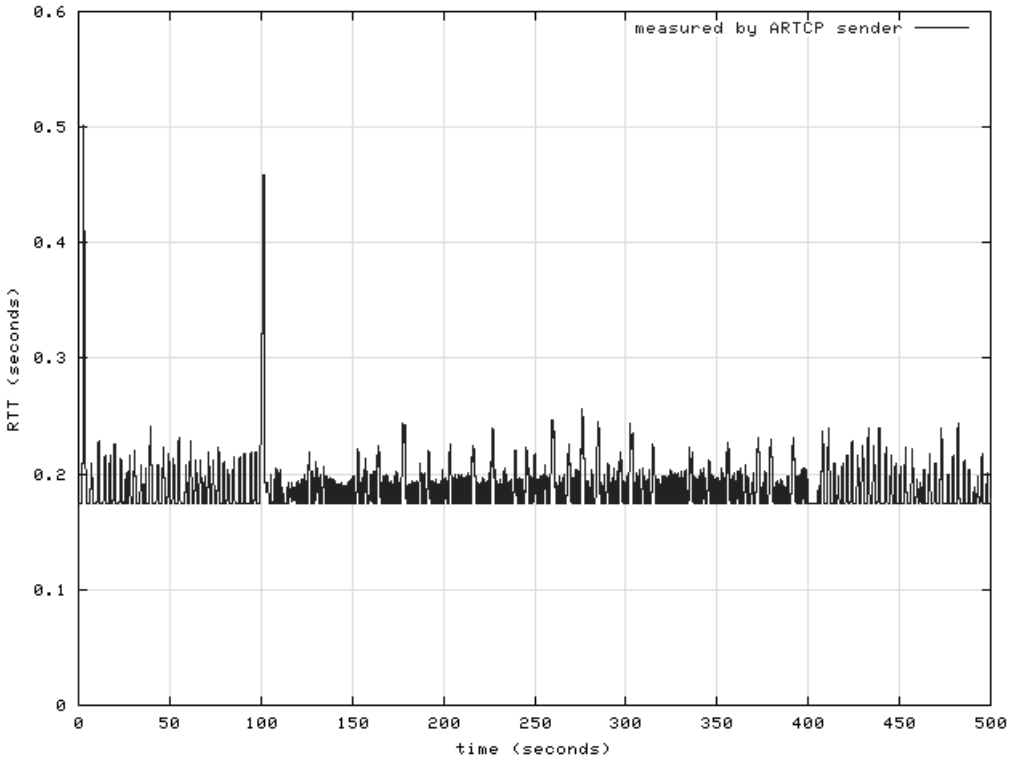

Рис. 55. Зависимость измеряемого RTT от времени.

4.8.4. Выводы

По результатам эксперимента в этом сценарии можно сделать вывод о том, что протокол ARTCP качественно адаптируется к ПС канала не занятой потоком CBR. В данных условиях не происходит потерь ни ARTCP сегментов, ни сегментов CBR. Для фиксированных значений ПС канала скорость ARTCP в присутствии CBR практически не зависит от ПС- R_{CBR} .

4.9. Сценарий 8: 2 ARTCP и 1 CBR

4.9.1. Задача

Далее детально изучим взаимодействие двух ARTCP потоков разделяющих общий канал в присутствии CBR потока и без него. В работе системы выделяются периоды, когда в ней присутствует только один ARTCP поток, два ARTCP потока и два ARTCP потока совместно с CBR. После включения второго ARTCP потока, уже существующий должен уменьшить свою скорость до значения равного половине ПС канала. После включения CBR потока, оба ARTCP потока должны уменьшить свою скорость так, чтобы на долю каждого приходилась половина от разности ПС- R_{CBR} . Задачей моделирования по сценарию 4 является проверка качества адаптации ARTCP к доступной доле пропускной способности канала. Необходимо рассмотреть взаимодействие двух ARTCP потоков до и после включения CBR.

4.9.2. Топология

Для исследования аспектов взаимодействия потоков протокола ARTCP в условиях наличия потока протокола CBR используется топология, изображенная на рис. 56. Стрелкой указано направление передачи данных. Узлы, отмеченные знаком **S** являются источниками, узлы **R** - приемниками потоков.

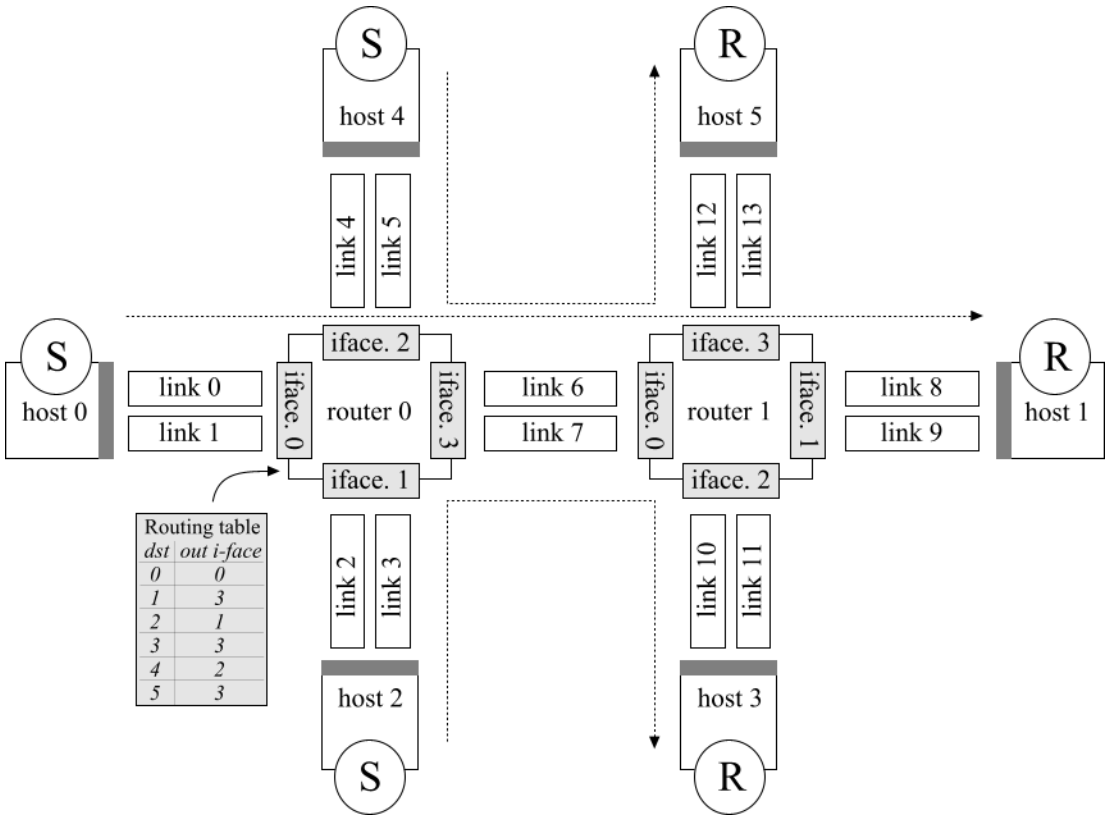

Рис. 56. Элементы топологии 3.

4.9.3. Эксперимент

Параметры:

Параметр	Значение
ПС каналов LAN	10 Мб/с
Задержка каналов LAN	0.01 с
ПС каналов WAN	256 Кб/с
Задержка каналов WAN	0.1 с
Длительность эксперимента	500 с
Макс. размер очереди маршрутизатора	32 Кбайт
Скорость потока CBR	От 48 Кб/с до 208 Кб/с с шагом 16 Кб/с
Число экспериментов	100
Число потоков ARTCP	2
Число потоков CBR	1
Скорость потока CBR	Выбирается из интервала 50-200 Кб/с по случайному закону с равномерным распределением
Число экспериментов	100 по каждому значению R_{CBR}
Момент запуска потока CBR	Выбирается из интервала 90-110 с по случайному закону с равномерным распределением
Момент остановки CBR	Выбирается из интервала 390-410 с по случайному закону с равномерным

	распределением
Момент запуска второго ARTCP потока	Выбирается из интервала 190-210 с по случайному закону с равномерным распределением

Поскольку, как выяснилось в предыдущем сценарии, коэффициент U не зависит от ПС- R_{CBR}, то значения скорости CBR выбираем по случайному закону с равномерным распределением из промежутка 50-200 Кб/с. Выбираем 100 значений и для каждого проводим эксперимент длительностью 500 с. По случайному закону с равномерным распределением выбираем и значения моментов запуска второго ARTCP потока и CBR потока из промежутков 90-110 и 190-210 с соответственно в каждом эксперименте.

Полученные результаты таковы: для двух ARTCP потоков в присутствии CBR потока $U = 0.981 \pm 0.012$; для двух ARTCP потоков в отсутствие CBR потока $U = 0.971 \pm 0.023$; число потерянных сегментов во всех случаях равно нулю; для двух ARTCP потоков в присутствии CBR потока $F = 0.989 \pm 0.011$; для двух ARTCP потоков в отсутствие CBR потока $F = 0.97 \pm 0.028$.

График зависимости скорости соединений от времени приведены на рис. 57 и график зависимости порядкового номера передаваемых сегментов от времени - на рис. 58. Для визуализации выбран типичный эксперимент данного сценария.

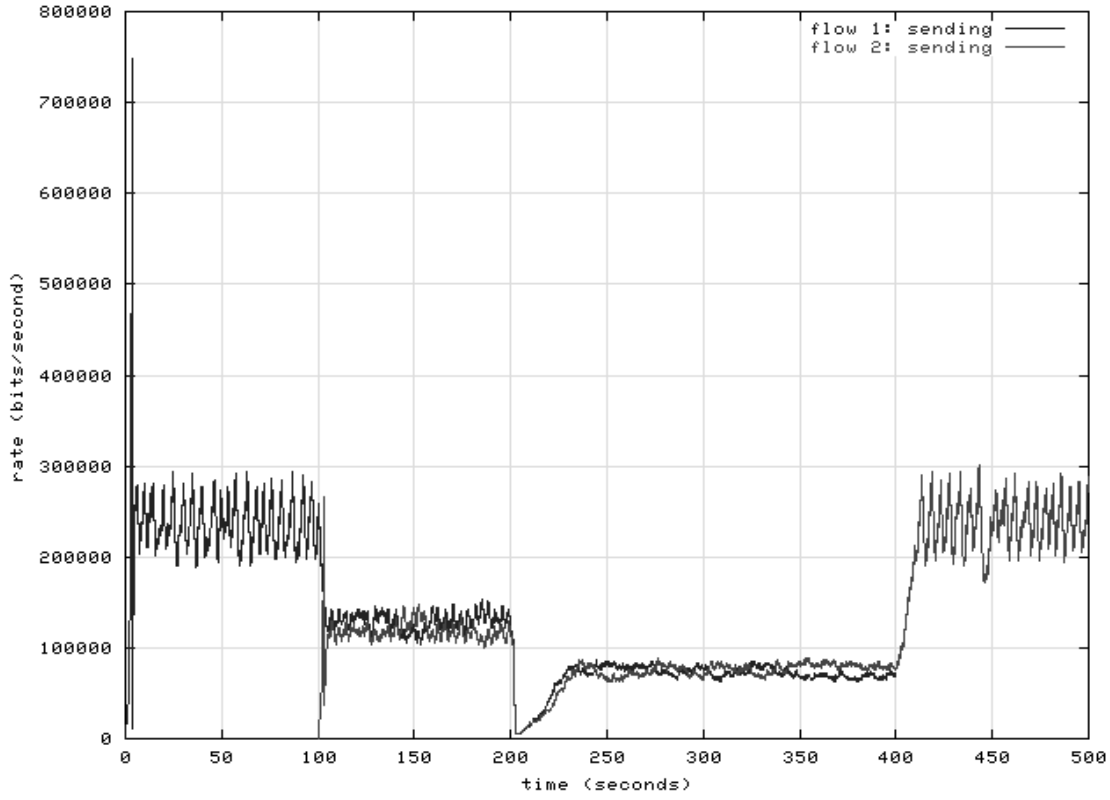

Рис. 57. Зависимость скорости ARTCP потоков 1 и 2 от времени.

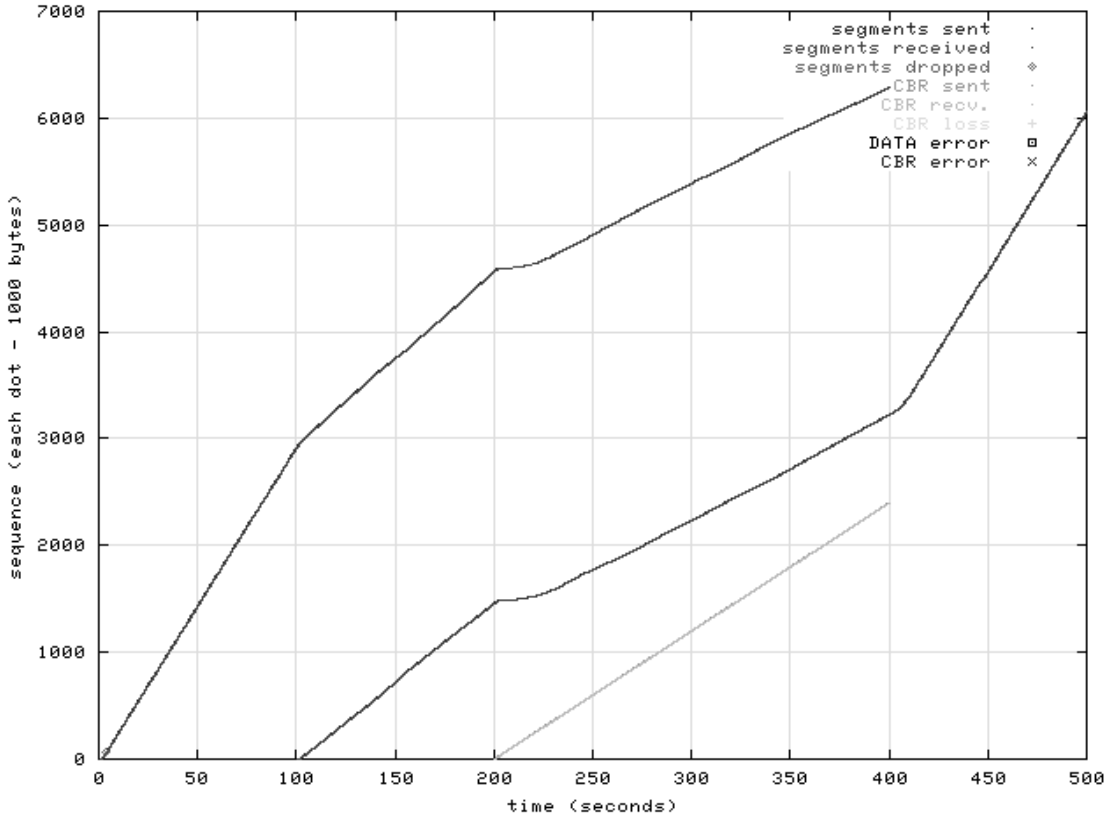

Рис. 58. Зависимость последовательности передачи данных от времени.

4.9.4. Выводы

Таким образом, два ARTCP потока хорошо адаптируются к доступной ПС канала как на фоне CBR потока, так и без него.

4.10. Сценарий 9: свойство самоподобия трафика ARTCP

4.10.1. Задача

Основным методом анализа коммуникационных сетей является теория систем массового обслуживания. Однако большинство результатов этой теории получено в предположении о конечности дисперсий как интервалов между поступлениями сегментов, так и длительностей их обслуживания. Экспериментальное изучение трафика в TCP/IP сетях (В. Леланд и др.) показало, что такое предположение о конечности дисперсии неверно. В классических работах В. Виллингера и М. Таггу показано, что трафик в сетях архитектуры TCP/IP обладают свойством самоподобия.

На настоящий момент теория массового обслуживания для самоподобных потоков только начинает развиваться, и в ней отсутствуют изученные теоретические модели для систем, моделирующих сетевой трафик. Поэтому модельный эксперимент является основным способом изучения TCP/IP трафика.

Для определения того, обладает ли трафик свойством самоподобия, обычно вычисляется коэффициент Хёрста. Целью данного сценария является выявление свойства самоподобия ARTCP трафика.

4.10.2. Топология

Топологическая схема эксперимента представлена на рис. 39. Согласно схеме через территориальную сеть проходит трафик между двумя ЛВС - по 10 узлов в каждой. Данные снимаются с маршрутизатора R1. ПС каналов WAN составляет 512 Кб/с.

4.10.3. Эксперимент

Для вычисления коэффициента Хёрста ARTCP трафика, был проведен модельный эксперимент, результатом которого явилась серия из 147036 измерений, суммирующих события прихода сегментов с данными на маршрутизатор R1 от 10-и активных источников за периоды 0.1 с. Время моделирования составило 19174 с, а общее число событий поступления сегментов с данными в маршрутизатор R1: 1208031.

График фрагмента (9000-12000 с) исходной серии измерений приведен на рис. 59, а на рис. 60 изображен результат сглаживания фрагмента последовательным применением wavelet symlet8 декомпозиции уровня 10, отбрасывания коэффициентов разложения превышающих 150 и восстановления сигнала. Для wavelet анализ применялась программа Matlab.

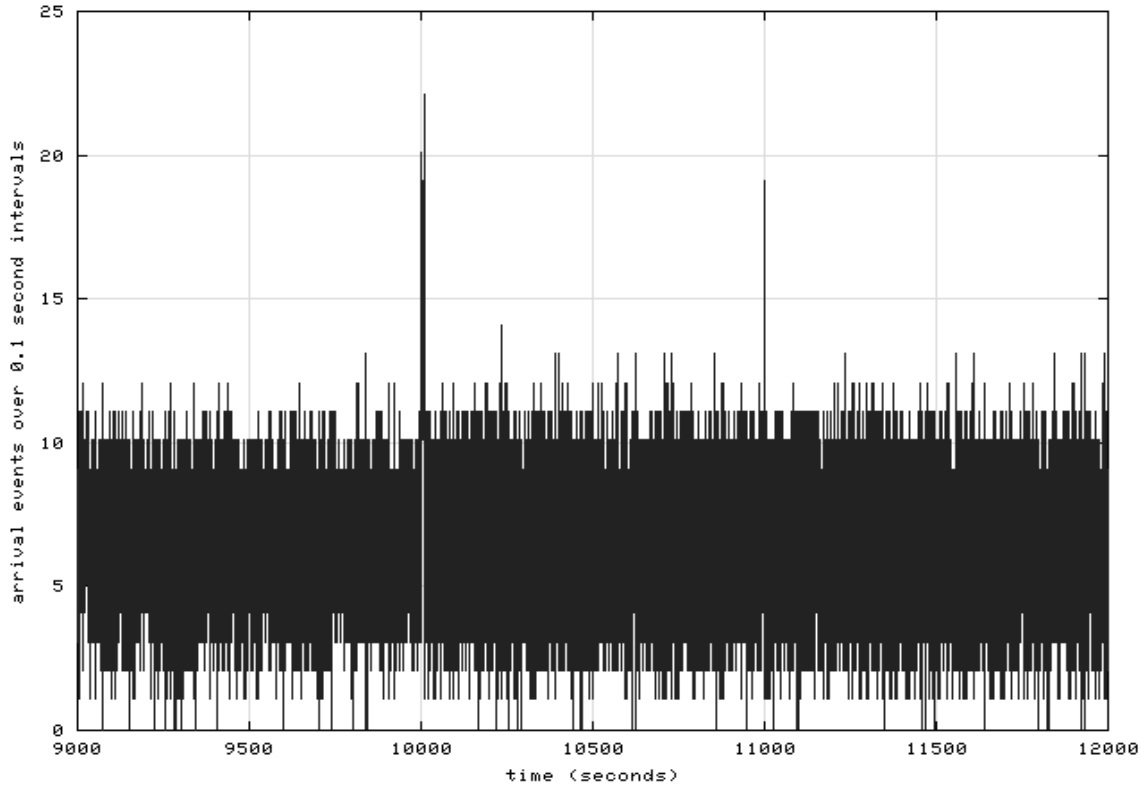

Рис. 59. Фрагмент полученной серии измерений.

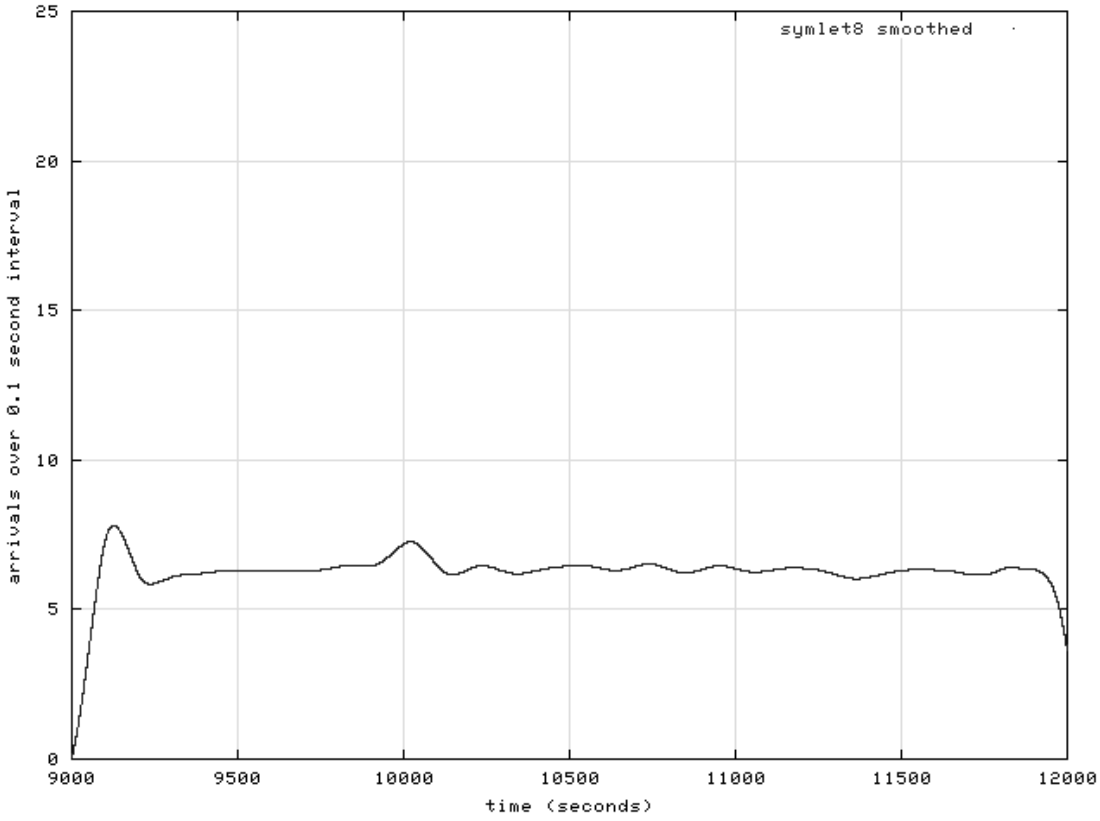

Рис. 60. Фрагмент серии измерений после сглаживания с применением sym8 wavelet. Искажением на краях можно пренебречь.

Полученная исходная серия подверглась статистической обработке с применением методов R/S статистики (рис. 61) и aggregated variance (рис. 62). По результатам применения обоих методов был вычислен коэффициент Хёрста: по методу R/S он равен 0.63, по методу aggregated variance: 0.65.

Для этого мною были разработаны программы на языке C, выполняющие вычисления по методам R/S и AVM достаточно быстро. Линейная аппроксимация по методу наименьших квадратов производилась с помощью программы статистического анализа PSPP[21].

[21] программа статистической обработки распространяется под лицензией GNU, информация о ней доступна по адресу http://www.gnu.org/software/pspp/pspp.html

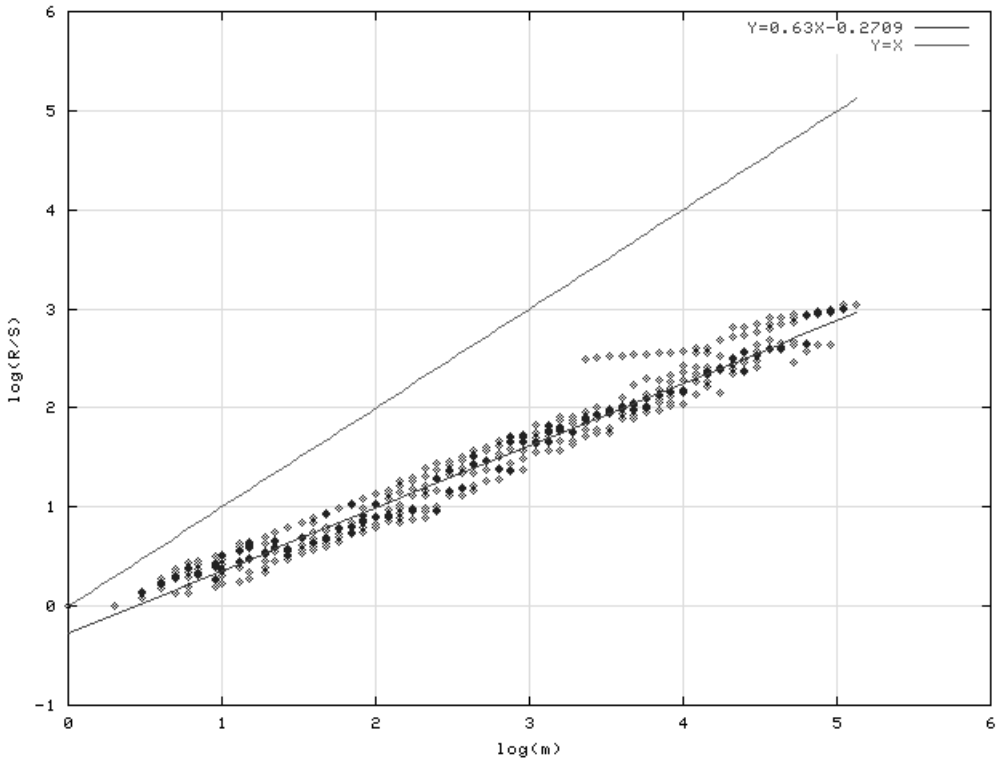

Рис. 61. результат применения метода Rescaled adjusted range (R/S).

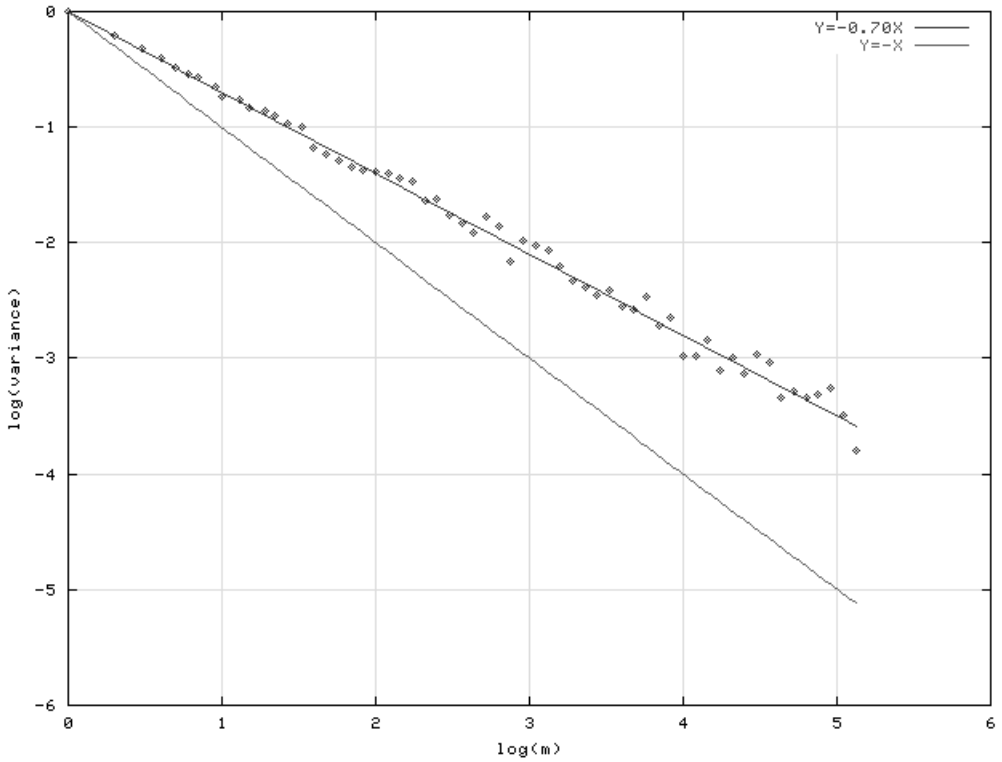

Рис. 62. Результат применения метода aggregated variance.

4.10.4. Выводы

Таким образом, трафик ARTCP, как и другой сетевой трафик по Вилингеру и Таггу [97, 95], обладает свойством самоподобия. Использование метода имитационного моделирования протокола ARTCP является в настоящий момент единственно возможным средством его исследования.

Наличие свойства самоподобия у трафика, полученного на имитационной модели, так же как и у трафика реальных сетей, указывает на то, что разработанная модель хорошо воспроизводит процессы, происходящие в реальных сетях.

Основные выводы

1. В настоящей работе дано описание нового транспортного протокола ARTCP, отличающегося от стандартного протокола TCP в нескольких основных аспектах. ARTCP в качестве сигнала о перегрузке в сети использует не потерю сегмента, а темпоральные характеристики потока. Сегменты ARTCP отправляются в сеть не в виде всплеска, а разделенные заданными временными интервалами. Измерение значения межсегментных интервалов у получателя позволяет оценить значение доступной ПС. ARTCP определяет доступную ПС соединения, не доводя сеть до состояния перегрузки, поэтому средняя длина очередей существенно снижается, и устраняются связанные с этим потери сегментов. Благодаря механизму диспетчеризации сегментов их отправка в сеть происходит без всплесков, более равномерно. Поэтому, во-первых, снижается потребность в буферном пространстве маршрутизаторов, а во-вторых, уменьшается разброс времени задержки сегментов в сети. В работе приведено подробное описание алгоритма протокола ARTCP и создана его модельная реализация в виде класса на языке C++.
2. Для исследования свойств протокола ARTCP создана универсальная имитационная программная модель, позволяющая изучать процессы, происходящие в сети с точки зрения транспортного протокола. Эта модель, построенная с помощью объектно-ориентированных методов на языке C++, дает возможность конструировать топологические схемы большой сложности и задавать любые условия их функционирования. Имитационная модель состоит из набора топологических элементов сети и объектов протоколов. В модели полностью осуществлена реализация протокола ARTCP и сервиса сети с коммутацией пакетов.
3. Результаты модельного эксперимента, проведенного на имитационной модели, показывают существенное превосходство адаптивного алгоритма управления скоростью потока протокола ARTCP по сравнению с TCP. Особенно хорошо ARTCP должен функционировать в беспроводных сетях. Обнаруженное у трафика моделируемой сети, в которой функционирует протокол ARTCP свойство самоподобия, во-первых, свидетельствует о том, что модель хорошо воспроизводит свойства реальных сетей, а во-вторых, служит основанием использования именно метода модельного эксперимента для исследования нового протокола.

Список литературы

1. Tanenbaum A.S. Computer Networks. Third edition, Prentice-Hall, New Jersey, 1996.
2. Jacobson V. Congestion Avoidance and Control. // ACM SIGCOMM'88. 1988.
3. Holzmann G. Design and Validation of Computer Protocols. Prentice Hall, New Jersey, 1991.
4. Postel J. Transmission Control Protocol. // RFC793 (STD7). 1981.
5. Braden R. T. Requirements for Internet Hosts – Communication Layers. // RFC1122. 1989.
6. Jacobson V., Braden R., Borman D. TCP Extensions for High Performance. // RFC1323. 1992.
7. Karn P., Partridge C. Estimating Round-trip Times in Reliable Transport Protocols. // ACM SIGCOMM'87. 1987.
8. Nagle J. Congestion Control in IP/TCP Networks. // ARPANET Working Group Requests for Comment (RFC-896), DDN Network Information Center, SRI International, Menlo Park, CA. 1984.
9. Бертсекас Д., Галлагер Р. Сети передачи данных. Пер. с англ. М., Мир. 1989.
10. George F., Young G. SNA Flow Control: Architecture and Implementation. // IBM System Journal, vol. 21, no. 2. - 1982. - pp. 179-210.
11. Digital Equipment Corporation. DECnet Digital Network Architecture [phase IV] General Description. // Order AA-N149A-TC, Digital Equipment Corporation. 1982.
12. Postel J. B., Sunshine C. A., and Cohen D. The ARPA Internet Protocol. // Computer Networks, vol. 5, no. 4. - 1981. - pp. 171-261.
13. Yang C., Reddy A. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. // IEEE Network Magazine. Vol. 9, Number 5. - 1995.
14. Brakmo L., O'Malley S., Peterson L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. // ACM SIGCOMM. -1994.- pp. 24-35.
15. Brakmo L., Peterson L. TCP Vegas: End to End Congestion Avoidance on a Global Internet. // IEEE Journal on Selected Areas in Communications, 13(8). -1995.
16. Lefelhocz C., Lyles B., Shenker S., Zhang L. Congestion Control for Best-Effort Service: Why We Need a New Paradigm. // IEEE Network, Vol. 10, N. 1. -1996.
17. Braden. B., Clark D., Crowfort J., Deering S., Estrin D. Recommendations in Queue Management and Congestion Avoidance in the Internet. // RFC 2309 -1998.
18. Floyd S., Jacobson V. Random Early Detection gateways for Congestion Avoidance. // IEEE/ACM Transactions on Networking. Vol.1, N.4. -1993.- p. 397-413.
19. Brakmo L., Peterson L. Performance Problems in BSD4.4 TCP. // ACM Computer Communications Review. 25(5). -1995.- p. 69-86.
20. Caceres R., Danzig P., Jamin S., Mitzel D. Characteristics of Wide-Area TCP/IP Conversations. // ACM SIGCOMM’91. -1991.
21. Fall K., Floyd S. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. // Computer Communications Review. July -1996.
22. Tomey C. Rate-Based Congestion Control Framework for Connectionless Packet-Switched Networks. // Doctor of Philosophy Thesis. University of New South Wales Australian Defence Force Academy. -1997.

23. Benmohamed L., Meerkov S. M. Feedback Control of Congestion in Packet Switched Networks: The Case of a Single Congested Node. // IEEE Transactions on Networking. Vol. 1, No. 6. -1993.- p. 693-707.

24. Charney A. An Algorithm for Rate Allocation in a Packet-Switched Network with Feedback. // M.Sc. thesis. Department of EECS. MIT. -1994.

25. Ramakrishnan K., Jain R. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. // ACM Transactions on Computer Systems. Vol. 8, No. 2. -1990.- p. 158-181.

26. Keshav S. The Packet Pair Flow Control Protocol. // ICSI Tech. Rept. TR-91-028. Computer Science Division, Department of EECS, University of California, Berkeley and International Computer Science Institute. Berkeley, CA. May 1991.

27. Bennett J., Zhang H. Hierarchical Packet Fair Queueing Algorithms. // ACM SIGCOMM'96. Aug 1996.

28. Demers A., Keshav S., Shenker S. Analysis and Simulation of a Fair Queueing Algorithm. // ACM SIGCOMM’89. -1989.- p. 3-12.

29. Floyd S., Jacobson V. Link Sharing and Resource Management Models for Packet Networks. // IEEE/ACM Transactions on Networking. 3(4). -1995.

30. Алексеев И.В. Интегрированные услуги нового поколения Internet. // Сети. № 10. -1999.- с. 102-108.

31. Geria M. Kleinrock L. Congestion Control in Interconnected LANs. // IEEE Network. vol. 2, N. 1. -1988.

32. Floyd S. TCP and Explicit Congestion Notification. // Computer Communications Review. 24(5). -1994.- p. 10-23.

33. Clark D., Lambert M., Zhang L. NETBLT: A High Throughput Transport Protocol. // ACM SIGCOMM '88. -1988.- p. 306-312.

34. Clark D., Lambert M., Zhang L. NETBLT: A Bulk Data Tranfer Protocol. // RFC 969. -1987.

35. Wang Z., Crowcroft J. A New Congestion Control Scheme: Slow Start and Search (Tri-S). // ACM Computer Communications Review. vol. 21. -1991.- p. 32-34.

36. Wang Z., Crowcroft J. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. // ACM computer communications review. vol. 22. -1992.- p. 9-16.

37. Selecting Sequence Numbers. // SIGCOMM/SIGOPS Interprocess Commun. Workshop. ACM. -1975.- p. 11-23.

38. Sunshine C., Dalal Y. Connection Management in Transport Protocols. // Computer Networks. vol. 2. -1978.- p. 454-473.

39. Watson R. Timer-Based Mechanisms in Reliable Transport Protocol Connection Management. // Computer Networks. vol. 5. -1981.- p. 47-56.

40. End-to-End Arguments in System Design. // ACM Trans. on Computer Systems. vol. 2. -1984.- p. 277-288.

41. Deering S., Hinden R. Internet Protocol Version 6 (IPv6) Specification. // RFC 2460. -1998.

42. Bartlett K., Scantlebury R., Wilkinson P. A note on reliable full-duplex transmission over half-duplex lines. // Comm. of the ACM. Vol. 12, No. 5. -1969.- p. 260-265.

43. Cerf V., Kahn R. A protocol for packet network intercommunication. // IEEE Trans. on communications. vol. COM-22, N. 5. -1974.- p. 637-648.

44. Chiu D., Jain R. Networks With Connectionless Network Layer; Part III: Analysis Of The Increase And Decrease Algorithms. // Tech. Rep. DEC-TR-509. Digital Equipment Corporation, Stanford, CA. -1987.

45. Wang Z. Routing And Congestion Control In Datagram Networks. // Doctor Of Philosophy Thesis. University College London. London UK. -1992.

46. Padhye J., Firoiu V., Towsley D., Kurose J. Modelling TCP throughput: a simple model and its empirical validation. // ACM SIGCOMM'98. -1998.

47. Johnson D., Maltz D. Protocols for Adaptive Wireless and Mobile Networking. // IEEE Personal Communications. -February 1996. p. 34-42.

48. Eckhardt D., Steenkiste P. Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network. // Proc. of ACM SIGCOMM '96. -1996.- p. 243-254.

49. Cáceres R., Iftode L. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. // IEEE Journal of Selected Areas in Communication. 13(5). -1995.- p. 850-857.

50. Balakrishnan H., Padmanabhan V.N., Seshan S., Katz R.H. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. // Proc. of ACM SIGCOMM. -1996.- p. 256-269.

51. Balakrishnan H., Padmanabhan V.N., Katz R.H. The Effects of Asymmetry on TCP Performance. // Proc. Of ACM/IEEE International Conf. on Mobile Computing and Networking. -September 1997.

52. Alekseev I.V., Sokolov V.A. Compensation Mechanism for Adaptive Rate TCP. // 1-St International IEEE/Popov Seminar "Internet: Technologies A and Services". P. 68-75, October 1999.

53. Алексеев И.В., Соколов В.А. Протокол TCP с адаптацией скорости. // Моделирование и анализ информационных систем. Т.6, №1. - 1999.- С. 4-12.

54. Алексеев И.В. Математическая модель протокола TCP с адаптацией скорости. // Моделирование и анализ информационных систем. Т.6, №2. - 1999.- С. 51-53.

55. Гольдштейн Б. Протоколы сети доступа. // М., Радио и связь. -1999.

56. Holzmann C. A Theory for Protocol Validation. // IEEE Transactions on Computers. Vol. C-31, N. 8. - 1982. - p. 730-738.

57. Holzmann C. Tracing Protocols. // AT&T Technical Journal. vol. 64. - 1985.- p. 2413-2434.

58. An Improved Protocol Reachability Analysis Technique. // Software, Practice and Experience. vol. 18, N. 2. -1988.- p. 137-161.

59. Bajaj S., Breslau L., Estrin D., Fall K., Floyd S. Improving Simulation for Network Research. // Technical Report 99-702. University of Southern California. -March 1999.

60. Estrin D., Handley M., Heidemann J., McCanne S., Xu Y., Yu H. Network Visualization with the VINT Network Animator Nam. // Technical Report 99-703. University of Southern California. -March 1999.

61. K. Fall Network Emulation in the Vint/NS Simulator. // Proc. of ISCC'99. -1999.

62. Stroustrup B. The C++ Programming Language. Addison-Wesley. Third edition. 1997.

63. Schildt H. C the Complete Reference. // McGraw-Hill, Berkeley CA. Second edition. -1987.

64. McKusick M., Bostic K., Karels M., Osterman J. The Design and Implementation of the 4.4 BSD Operating System. Addison-Wesley, 1996.

65. Floyd S., Jacobson V. Link-sharing and Resource Management Models for Packet Networks. // IEEE/ACM Transactions on Networking. Vol. 3 No. 4. - 1995.- p. 365-386.

66. Wakeman I., Ghosh A., Crowcroft J., Jacobson V., Floyd S. Implementing Real Time Packet Forwarding Policies using Streams. // Usenix 1995 Technical Conference. -January 1995.

67. Keshav S., Sharma R. Issues and Trends in Router Design. // Department of computer science, Cornell University. -1998.

68. Ginsburg D. ATM: Solutions for Enterprise Internetworking. Addison Wesley Longman Limited. UK. -1996.

69. C. Nicoll Overview: Multiservice Networking. // Packet. Cisco Systems Inc. -January 1999.

70. Unix Unleashed. SAMS Publishing. Indiannapolis. 1994.

71. Miller M. Internetworking. M&T Books, New York, 1991.

72. Allman M., Glover D., Sanchez L. Enhancing TCP over Satellite Channels using Standard Mechanisms. // RFC 2488. -1999.

73. Chiu D., Jain R. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. // Computer Networks and ISDN Systems. v. 17. -1989.- p. 1-14.

74. Bach M. J. THE DESIGN OF THE UNIX OPERATING SYSTEM. Prentice Hall, NJ, 1986.

75. Nemeth E., Snyder G., Seebass S., Hein T. UNIX System Administration Handbook. Prentice Hall PTR. NJ. Second edition. 1995.

76. Алексеев И. В. Модель и анализ транспортного протокола ARTCP. // Электронный журнал "Исследовано в России". № 27. - 2000.- С.395-404. http://zhurnal.mipt.rssi.ru/articles/2000/027.pdf

77. Balakrishnan H., Seshan S., Amir E., Katz R. Improving TCP/IP Performance over Wireless Networks. // Proc. Mobicom'95. - June 1995.

78. Floyd S. Connections with Multiple Congested Gateways in Packet Switched Networks, Part 1: One-Way Traffic. // ACM Computer Communications Review. 21 (5). -1991.- p. 30-47.

79. Morris R. TCP Behavior with Many Flows. // IEEE International Conference on Network Protocols. US. - October 1997.

80. Henderson T., Sahouria E., McCanne S., Katz R. On Improving the Fairness of TCP Congestion Avoidance. // Proc. IEEE Globecom '98. -1998.

81. Bonald T. Comparison of TCP Reno and TCP Vegas via Fluid Approximation. // Rapport de recherche N. 3563. INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE. -1998.

82. Altman E., Bolot J., Nain P., Elouadghiri D., Erramdani M., Brown P., Collange D. Performance Modelling of TCP/IP in Wide-Area Network. // Rapport de recherche N. 3142. INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE. - 1997.

83. Lakshman T., Madhow U. The Performance of TCP/IP for Networks with High Bandwidth-delay Product and Random Loss. // IEEE/ACM Transactions on Networking. -June 1997.

84. Comer D. Internetworking with TCP/IP, Vol. II: Design Implementation and Internals. Prentice Hall, NJ. 1994.

85. Comer D. Internetworking with TCP/IP, Vol. III: Client - Server Programming and Applications. Prentice Hall, NJ. 1994.

86. Stevens R. Advanced Programming in the UNIX Environment. Addison-Wesley. 1992.

87. Shepard T. TCP Packet Trace Analysis. // Masters of Science Thesis. Massachusetts institute of Technology. MIT/LCS/TR-494. -February 1991.

88. Frost V., Melamed B. Traffic modelling for telecommunications networks. // IEEE Communications Magazine. 32(2). -1994.- p. 70-80.

89. Leland W., Taqqu M., Willinger W., Wilson D. On the Self-Similar Nature of Ethernet Traffic (Extended Version). // IEEE/ACM Transactions on Networking. 2(1). -1994.- p. 1-15.

90. Gusella R. A Measurement Study of Diskless Workstations Traffic on an Ethernet. // IEEE Trans. on Communications. 38(9). - 1990.- p. 1557-1568.

91. Paxson V., Floyd S. Wide-Area Traffic: The Failure of Poisson Modelling. // IEEE/ACM Transactions on Networking. 3(3). - 1995.- p. 226-244.

92. Floyd S., Jacobson V. The Synchronization of Periodic Routing Messages. // IEEE/ACM Transactions on Networking. 2(2). - 1994.- p 122-136.

93. Willinger W., Taqqu M., Erramili A. A Bibliographical Guide to Self-Similar Traffic and Performance Modeling for Modern High-Speed Networks. // Stochastic Networks: Theory and Applications, Clarendon Press (Oxford University Press). Oxford. - 1996.- p. 339-366.

94. Taqqu M., Teverovsky V., Willinger W. Estimators for long-range dependence: an empirical study. // Fractals. vol. 3, n. 4. - 1995.- p. 785-788.

95. Taqqu M., Teverovsky V., Willinger W. Is network traffic self-similar or multifractal? // Fractals. n. 5 - 1997.- p. 63-73.

96. Riedi R., Vehel J. TCP traffic is multifractal: a numerical study. // INRIA research report 3129. -March 1997.

97. Riedi R., Willinger W. Towards an Improved Understanding of Network Traffic Dynamics. // preprint chapter from the book "Self-similar Network Traffic and Performance Evaluation". - 1999.

98. Taqqu M., Willinger W., Sherman R. Proof of Fundamental Result in Self-Similar Traffic Modelling. // Computer Communications Review. n. 27. - 1997.- p. 5-23.

99. Crovella M., Taqqu M., Bestavros A. Heavy-Tailed Probability Distributions in the World Wide Web. // preprint chapter from the book "A Practical Guide to Heavy Tails: Statistical Techniques and Applications". Alder R., Feldman R., Taqqu M. Birkhauser. Boston, US. -1998.

100. Willinger W., Taqqu M., Sherman R., Wilson D. Self-similarity through high variability: Statistical analysis of Ethernet LAN traffic at source level. // IEEE/ACM Transactions of Networking. n. 5. -1997.- p. 71-86.

101. Хакен Г. Синергетика. М., Мир. С. 379. 1980.