Лабораторная работа № 3

Цель: *провести анализ производительности протоколов TCP и UDP для заданной конфигурации сети, и сделать заключение о том, какой протокол предпочтительнее использовать.*

Теоретические сведения

1. ТРАНСПОРТНЫЙ УРОВЕНЬ: СРАВНЕНИЕ ПРОТОКОЛОВ TCP И UDP

На этом уровне функционируют два протокола: протокол управления передачей (Transmission Control Protocol) и протокол дейтаграмм пользователя (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования логических соединений.

TCP работает непосредственно над протоколом IP и использует для транспортировки своих блоков данных потенциально ненадежный протокол IP. Надежность передачи данных протоколом TCP достигается за счет того, что он основан на установлении логических соединений между взаимодействующими процессами. Протокол IP используется протоколом TCP в качестве транспортного средства. Перед отправкой своих блоков данных протокол TCP помещает их в оболочку IP-пакета. Протокол IP осуществляет фрагментацию и сборку блоков данных TCP.

На рисунке 1.1 показано, как процессы, выполняющиеся на двух конечных узлах, устанавливают с помощью протокола TCP надежную связь через составную сеть.

Протокол UDP позволяет передавать пакеты дейтаграммным способом, как и протокол уровня межсетевого взаимодействия IP. UDP выполняет только функции связующего звена между сетевым протоколом и пользовательскими процессами, а также многочисленными службами прикладного уровня.

1.1. Понятие портов

Протоколы TCP и UDP взаимодействуют через межуровневые интерфейсы с нижележащим протоколом IP и с вышележащими протоколами прикладного уровня или приложениями. Таким образом, после того, как пакет средствами протокола IP доставлен в компьютерполучатель, данные необходимо направить конкретному процессу-

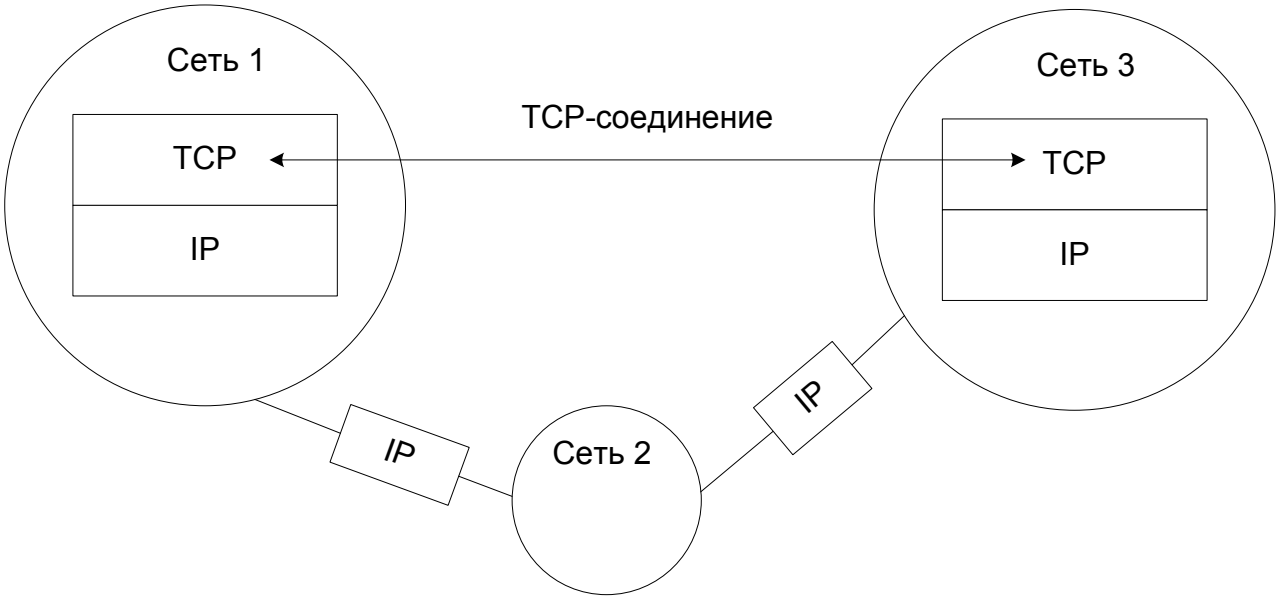

Рис. 1*.1* логическое TCP–соединение

получателю. На одном компьютере могут выполняться несколько процессов, более того, прикладной процесс может иметь несколько точек входа, выступающих в качестве адреса назначения для пакетов данных.

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называются *портами*. Таким образом, адресом назначения, который используется протоколом TCP или UDP, является идентификатор (номер) порта прикладной службы. Номер порта в совокупности с номером сети и номером конечного узла однозначно определяет прикладной процесс в сети. Этот набор идентифицирующих параметров называется *сокет* (socket). Такое определение сокета можно применять не только в терминах TCP/IP, но и, к примеру, в терминах IPX.

Назначение номеров портов прикладным процессам осуществляется либо централизованно, если эти процессы представляют собой популярные общедоступные службы (FTP – 21, telnet – 23), либо локально для тех служб, которые еще не стали столь распространенными, чтобы закреплять за ними стандартные номера. Централизованное присвоение службам номеров портов выполняется организацией *Internet Assigned Numbers Authority (IANA)*. Эти номера затем закрепляются и опубликовываются в стандартах Internet.

Протокол TCP (также как и UDP) ведет для каждого порта две очереди: очередь пакетов, поступающих в данный порт из

сети, и очередь пакетов, отправляемых данным приложением через порт в сеть. Процедура обслуживания протоколом запросов, поступающих от нескольких различных прикладных служб называется *мультиплексированием*. Обратная процедура распределения протоколом поступающих от сетевого уровня пакетов между набором высокоуровневых служб, идентифицированных номерами портов, называется *демультиплексированием*.

1.2. Протокол транспортного уровня TCP

Сегменты и потоки.

Единицей данных для протокола TCP является сегмент. Информация, поступающая по протоколу TCP в рамках соединения от протоколов более высокого уровня, рассматривается протоколом TCP как неструктурированный поток байтов.

Поступающие данные буферизуются. Для передачи на сетевой уровень из буфера вырезается некоторая часть данных, – это и есть *сегмент*. Отличительной особенностью TCP является то, что он подтверждает получение не пакетов, а байтов потока.

Сегменты, посылаемые через соединение могут иметь разный размер. Оба участника соединения должны договариваться о максимальном размере посылаемого сегмента. Размер выбирается таким образом, чтобы при упаковке в IP-пакет сегмент помещался в него целиком без фрагментации, т.е. размер максимального сегмента не должен превосходить максимального размера поля данных IP-пакета.

Соединения.

Для организации надежной передачи данных предусматривается установление логического соединения между двумя прикладными процессами. Т.к. соединение устанавливается через ненадежную среду IP, то во избежание ошибочной инициализации соединений используется специальная многошаговая процедура подтверждения связи.

Соединение в протоколе TCP идентифицируется парой полных адресов обоих взаимодействующих процессов – *сокетов*. Каждый из взаимодействующих процессов может участвовать в нескольких соединениях.

Формально соединение – это набор параметров, характеризующий процедуру обмена данными между двумя процессами. К таким параметрам относятся:

- Согласованные размеры сегментов.
- Объемы данных, которые разрешено передавать без подтверждения.

- Начальные и текущие номера передаваемых байтов.

В рамках соединения осуществляется обязательное подтверждение правильности приема для всех переданных сообщений и, при необходимости, выполняется повторная передача.

Установка связи по протоколу.

Этапы, из которых состоит процесс установки связи по протоколу таковы:

- Узел-отправитель запрашивает соединение, посылая сегмент с установленным флагом синхронизации (SYN).
- Узел-адресат подтверждает получение запроса, отправляя обратно сегмент с:
  - установленным флагом синхронизации;
  - порядковым номером начального байта сегмента, который он может послать;
  - подтверждением, включающим порядковый номер следующего сегмента, который он ожидает получить.
- Запрашивающий узел посылает обратно сегмент с подтверждением номера последовательности и номером своего подтверждения (ACK).

Этап соединения проиллюстрирован на рис. 1.2

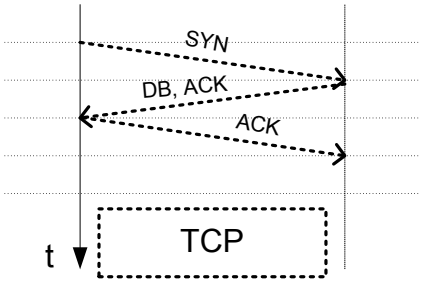

Рис. 1.2 этап соединения по протоколу TCP, SYN – пакет синхронизации, DB – блок данных, ACK – подтверждение

Из рисунка 1.2 видно, что соединение выполняется в три этапа, гарантирующих безошибочное установление связи.

Реализация скользящего окна в TCP.

Во время установленного соединения правильность передачи каждого сегмента должна подтверждаться квитанцией получателя. Квитирование – это один из традиционных методов обеспечения надежной связи. В протоколе TCP используется частный случай квитирования – алгоритм скользящего окна.

Идея состоит в том, что окно определено на множестве нумерованных байтов неструктурированного потока данных,

поступающих с верхнего уровня и буферизуемых протоколом TCP. В качестве квитанции получатель сегмента отсылает ответное сообщение (сегмент), в которое помещается число на единицу большее, чем максимальный номер байта в полученном сегменте. Это число называют номером очереди.

Особенности реализации скользящего окна в протоколе TCP изображены на рис.1.3. Если размер окна равен W, а последняя

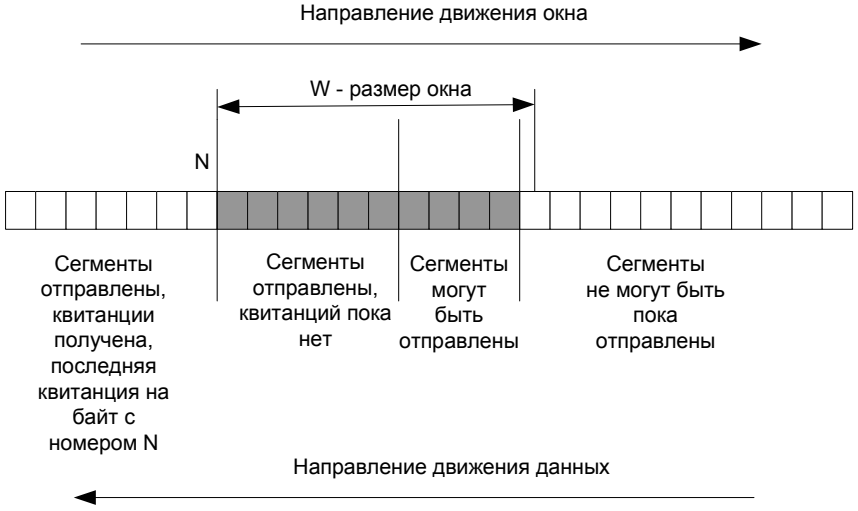

Рис. 1.3 скользящее окно в протоколе TCP

по времени квитанция содержала значение N, то отправитель может посылать новые сегменты до тех пор, пока в очередной сегмент не попадет байт с номером $N + W$. Этот сегмент выходит за рамки окна, и передача в таком случае должна быть приостановлена до прихода следующей квитанции.

Выбор тайм-аута очередной квитанции – важная задача, результат решения которой сильно влияет на производительность протокола TCP. Время не должно быть слишком коротким – для исключения повторных передач, снижающих полезную пропускную способность системы. И время не должно быть слишком большим во избежание длинных простоев, связанных с ожиданием несуществующей или потерявшейся квитанции.

Алгоритм определения тайм-аута. Для TCP он состоит в том, что при каждой передаче сегмента засекается время до прихода квитанции о его приеме (время оборота). Получаемые значения времен оборота усредняются весовыми коэффициентами, возрастающими от предыдущего значения к последующему, то есть усиливается влияние последних замеров. В качестве тайм-аута выбирается среднее время оборота, умноженное на специальный коэффициент. На практике значение этого коэффициента должно превышать 2.

В сетях с большим разбросом времени оборота при выборе тайм-аута учитывается и дисперсия этой величины.

Поскольку каждый байт пронумерован, то каждый из них может быть опознан. Механизм опознавания является накопительным, поэтому опознавание номера X означает, что все байты с предыдущими номерами уже получены.

Этот механизм позволяет регистрировать появление дубликатов в условиях повторной передачи.

Нумерация байтов в пределах сегмента осуществляется так, чтобы первый байт данных сразу вслед за заголовком имел наименьший номер, а следующие за ним байты имели номера по возрастающей.

Размер окна, посылаемый получателем данных с каждым сегментом, определяет диапазон номеров очереди, которое отправитель окна (он же получатель данных) готов принять в настоящее время. Такой механизм связан с наличием в данный момент места в буфере данных.

Чем больше окно, тем большую порцию неподтвержденных данных можно послать в сеть. Но если пришло большее количество данных, чем может быть принято программой TCP, данные будут отброшены. Это приведет к излишним пересылкам информации и ненужному увеличению нагрузки на сеть и программу TCP. С другой стороны, указание малого размера окна может ограничить передачу данных скоростью, которая определяется временем путешествия по сети каждого сегмента в отдельности. Чтобы избежать применения малых окон, получателю данных предлагается откладывать изменение окна до тех пор, пока свободное место не составит 20-40 % от максимально возможного объема памяти для этого соединения. Отправитель также не должен спешить с посылкой данных, до того времени, пока окно не станет достаточно большим.

Разработчики протокола TCP предложили схему, согласно которой при установлении соединения заявляется большое окно, но впоследствии его размер существенно уменьшается.

Если сеть не справляется с нагрузкой, то возникают очереди в промежуточных узлах-маршрутизаторах и в конечных узлах-компьютерах.

При переполнении приемного буфера конечного узла "перегруженный" узел отправляет TCP-квитанцию, помещая в нее новый, уменьшенный размер окна. Если узел совсем отказывается от приема, то в квитанции устанавливается нулевой размер окна. Однако, даже после этого приложение может послать сообщение на отказавшийся от приема порт, сопроводив его (сообщение) пометкой

"срочно". В такой ситуации порт обязан принять сегмент, даже если для этого придется вытеснить их буфера уже находящиеся там данные.

После приема квитанции с нулевым размером окна протоколотправитель время от времени делает контрольные попытки продолжить обмен данными. Если приемник уже готов принимать информацию, то в ответ он посылает квитанцию с указанием ненулевого размера окна.

Другим проявлением перегрузки сети является переполнение буферов в маршрутизаторах. В таких ситуациях они могут централизованно изменить размер окна, посылая управляющие сообщения некоторым конечным узлам, что позволяет им дифференцированно управлять интенсивностью потока данных в разных частях сети.

1.3. Протокол транспортного уровня UDP

Общее описание.

Протокол User Datagram Protocol (UDP) обеспечивает неориентированную на соединение службу доставки дейтаграмм по принципу "максимального усилия". Это означает, что получение всей дейтаграммы или правильной последовательности не гарантируется. Протокол UDP используется приложениями, не требующими подтверждения. Обычно такие приложения передают данные небольшого объема за один раз. К примеру, это: сервис имен Net-BIOS, сервис SNMP, сервис дейтаграмм NetBIOS.

Порты протокола UDP.

Для использования протокола UDP приложение должно знать IP-адрес и номер порта получателя. Порт действует как мультиплексная очередь сообщений, то есть он может получать несколько сообщение одновременно. Важно отметить, что порты UDP отличаются от портов TCP несмотря на использование одних и тех же значений номеров.

Описание работы UDP.

Порт UDP легче всего представить в виде очереди. В большинстве реализаций, когда прикладная программа "договаривается" с операционной системой об использовании данного порта, операционная система создает внутреннюю очередь, которая хранит приходящие сообщения. Часто приложение может указать или изменить размеры очереди. Когда узел получает дейтаграмму по UDP-протоколу, он проверяет, нет ли порта назначения с таким номером среди используемых портов. Если таких портов нет, UDP посылает ICMP-сообщение об ошибке "порт недоступен" и уничтожает дейтаграмму. Если есть, UDP добавляет новую дейтаграмму в очередь порта, где

прикладная программа может ее получить. Если очередь порта уже переполнена, то тогда UDP уничтожает новую дейтаграмму.

1.4. Сравнение производительности TCP и UDP

Последовательность действий (запросов и ответов) для протоколов TCP и UDP представлена на рис. 1.4. Как видно из рисунка, передача

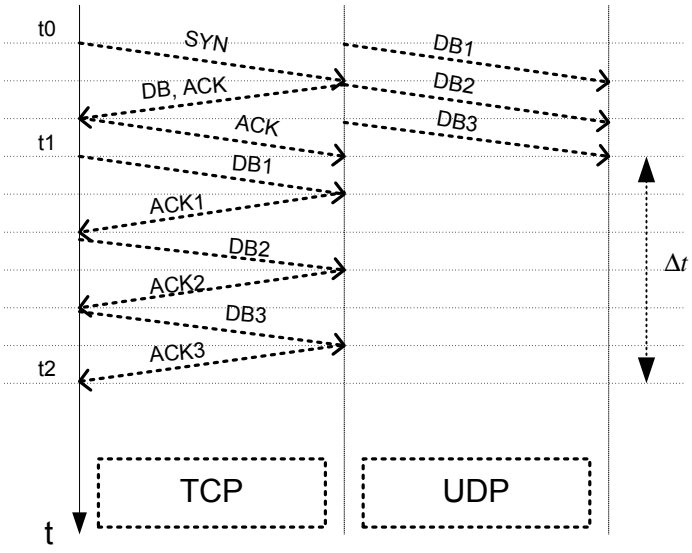

Рис. 1.4 последовательность запросов и ответов в TCP и UDP, SYN – пакет синхронизации, DB – блок данных, ACK – подтверждение

началась в момент времени t_0 по обоим протоколам. К моменту времени t_1 протокол UDP завершил передачу, в то время, как по протоколу TCP передача закончится только к моменту времени t_2. Легко получить разницу во времени $\Delta t = t_2 - t_1$. Таким образом, протокол UDP работает быстрее, чем TCP. Данные по протоколу UDP отсылаются получателю друг за другом и не требуется получение подтверждения об успешной доставке данных получателю. UDP не тратит время на установление связи в несколько этапов. Стоит отметить, что TCP путем механизма подтверждений гарантирует успешную доставку данных получателю, в то время, как данные посланные через UDP могут и не дойти до адресата.

Порядок выполнения работы

1. В качестве схемы сети взять результат выполнения соответствующего варианта лабораторной работы №1. Установить коэффициенты прохождения пакетов согласно вашему варианту.
2. Протестировать отправку по UDP и по TCP 20 сообщений с K1 на K3.
3. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения скорости доставки информации.
4. Протестировать отправку по UDP и по TCP 20 сообщений с K2 на K1.
5. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения надежности доставки информации.
6. Подсчитать процент потерь пакетов. С учетом того, что должно теряться не более 7% пакетов. Объяснить, как привести сеть к требуемому лимиту по потерям.
7. Проанализировать время соединения, сделать вывод о том, какой протокол быстрее справился с поставленной задачей (необходимо учитывать требуемую надежность).
8. Определить состояние, при котором сеть начинает удовлетворять требованиям по потери пакетов. То есть подобрать такие значения коэффициентов пропускания, при которых будет теряться не более 7% пакетов. Разрешается использовать диапазон значений длины 10, то есть можно найти интервал значений коэффициентов пропускания длины 10, где на нижней границе сеть не удовлетворяет критерию потерь пакетов, а на верхней заданный критерий удовлетворяется.

Отчет должен содержать: анализ производительности протоколов TCP и UDP для заданной конфигурации сети при коэффициенте пропускания равном 100, расчет процента потерь пакетов и анализ производительности сети для обоих протоколов в условиях недоброкачественных линий передач для обоих протоколов, оценку удовлетворения сетью критерия по потере пакетов, анализ времени соединения. В отчете также необходимо привести вывод о том, какой протокол предпочтительнее использовать в данной конфигурации сети.

Варианты заданий

Вариант 1. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 1. Установите коэффициент прохождения пакетов между узлами Connector и Boss_R в 82.

Обозначения в задании: K1 – Boss, K2 – Hacker, K3 – OFFICE2 pc1.

Вариант 2. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 2. Установите коэффициент прохождения пакетов между узлами H_OFFICE1 и OFF_R в 71.
Обозначения в задании: K1 – BIG BOSS, K2 – M_CH_S, K3 – OFFICE1 pc4.
Вариант 3. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 3. Установите коэффициент прохождения пакетов между узлами Connector и Hacker в 75.
Обозначения в задании: K1 – Boss, K2 – Hacker, K3 – OFFICE2 pc1.
Вариант 4. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 4. Установите коэффициент прохождения пакетов между узлами BOSS HUB и R2 в 85.
Обозначения в задании: K1 – BIG BOSS, K2 – M_CH_S, K3 – OFFICE1 pc4.
Вариант 5. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 5. Установите коэффициент прохождения пакетов между узлами HBosses и center в 80.
Обозначения в задании: K1 – MegaBoss, K2 – Manager2, K3 – FileServer.
Вариант 6. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 6. Установите коэффициент прохождения пакетов между узлами HManagers и center в 78.
Обозначения в задании: K1 – Manager3, K2 – PrintServer, K3 – MicroBoss.
Вариант 7. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 7. Установите коэффициент прохождения пакетов между узлами Hub2 и R1 в 85.
Обозначения в задании: K1 – Station1, K2 – Remote1, K3 – Station2.
Вариант 8. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 8. Установите коэффициент прохождения пакетов между узлами Hub2 и R1 в 55.
Обозначения в задании: K1 – Station1, K2 – Remote1, K3 – Station2.
Вариант 9. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 9. Установите коэффициент прохождения пакетов между узлами Hub1 и R1 в 75.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 10. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 10. Установите коэффициент прохождения пакетов между узлами Hub1 и R1 в 65.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.
Вариант 11. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 11. Установите коэффициент прохождения пакетов между узлами R–C–M и HManagers в 75.
Обозначения в задании: K1 – Chief, K2 – Manager1, K3 – Service.
Вариант 12. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 12. Установите коэффициент прохождения пакетов между узлами R–M–S и HService в 80.
Обозначения в задании: K1 – Manager3, K2 – Service, K3 – Chief.
Вариант 13. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 13. Установите коэффициент прохождения пакетов между узлами H1 и R131 в 77. А также между узлами H2 и R120 в 90.
Обозначения в задании: K1 – Remote1, K2 – Remote2, K3 – Remote3.
Вариант 14. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 14. Установите коэффициент прохождения пакетов между узлами H3 и Remote3 в 80. А также между узлами H2 и R120 в 85.
Обозначения в задании: K1 – Remote2, K2 – Remote3, K3 – Remote1.

Пример выполнения лабораторной работы

Файл со схемой сети: lab4_sample.jfst. Компьютер PC1 имеет IP-адрес 192.168.0.1. Компьютер PC2 имеет IP-адрес 192.168.0.2.
Задание:

1. Установить коэффициент пропускания всех линий равный 100.
2. Протестировать отправку по UDP и TCP 20 сообщений с PC2 на PC1.
3. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения скорости доставки информации.
4. Установить коэффициент пропускания 65.
5. Протестировать отправку по UDP и TCP 20 сообщений с PC2 на PC1.
6. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения надежности доставки информации.

7. Подсчитать процент потерь пакетов. С учетом того, что должно теряться не более 7% пакетов. Объяснить, как привести сеть к требуемой надежности.
8. Проанализировать время соединения, сделать вывод о том, какой протокол быстрее справился с поставленной задачей (необходимо учитывать требуемую надежность).
9. Определить состояние, при котором сеть начинает удовлетворять требованиям по потери пакетов. То есть подобрать такие значения коэффициентов пропускания, при которых будет теряться не более 7% пакетов. Разрешается использовать диапазон значений длины 10, то есть можно найти интервал значений коэффициентов пропускания длины 10, где на нижней границе сеть не удовлетворяет критерию потерь пакетов, а на верхней заданный критерий удовлетворяется.

Порядок выполнения работы будет следующим:

1. Убедимся, что коэффициент пропускания на всех линиях (в том числе между PC1 и PC2) равен 100.
2. Выберем PC1 и запустим на нем UDP-приложение (UPD-сервер), выбрав в качестве прослушиваемого порт 7.
   Программа выдаст следующее сообщение:
   `pc1Applicationisnowlisteningonport7.`
3. Выберем PC2 и пошлем через UDP-приложение 20 сообщений со строкой "rsh"на PC1. Программа выдаст похожее на следующее сообщение (для первого из двадцати сообщений):

```
pppccc222Startttsendingechomessage'rsh'to192.168.0.1:7
     CCCrrreeeaaaeeedddUDPPPpacketfor192.168.0.1:7.
pppccc111tARResponsepacketto192.168.0.2
     SSSeeennnddiiinnnggggpppaaaccckkkeeetttfffrrrooommmPPPrrroootttooo
pppccc2 1
111ProtocolStackreceivedpacketfromlocaaalInteeerface.
pppcccConfirmedPackeeetisfffforthisNetworkLyerDviceee.
1UDPpacketrecivedrom192.168.0.2:3000messag:
     "rsh".UDPPort7hasstatus"busy"fromnow.
pppccc111AAAppppppllliiicccaaatttiiiooonnnReeecievingechomessage'rsh'fr
                Sndingechomessage'rsh'toclient.
pppccc111CreatedUDPpacketforrr192.168.0.2:3000.
     SendingpacketfromPotocccolStack(tooo192.168.0.2)...
pppccc222ProtocolStackreceivedpaketfromlcaaalInteeerface
     ConfirmedPackeeetisfffforthisNetworkLyeeerDvice.
pc2UDPpacketrecivedrom192.168.0.1:7mssage:"rsh".
```

```
pc2 Recievingechomessage'rsh'fromserver.
pc1 Serverclosingconnection. Nowlisteningon7.
pc1 Applicationisnowlisteningonport7.
```

4. Выберем меню статистики узла PC1 и проверим, сколько UDP дейтаграмм он получил и отправил. Будет выведен следующий результат: "Received UDP segments: 20", что означает, что получено 20 UDP дейтаграмм, и "Sent UDP segments: 20", что означает, что отправлено 20 UDP дейтаграмм. При заданных параметрах сети процент потерь равен 0%, что удовлетворяет требованиям.
5. Обнулим статистику узла PC1. Теперь установим коэффициент пропускания линии между двумя узлами в значение, равное 65 и снова пошлем с PC2 на PC1 20 UDP дейтаграмм.
6. Выберем меню статистики узла PC1 и проверим, сколько UDP дейтаграмм он получил и отправил. Будет, с большой вероятностью, выведен следующий результат: "Received UDP segments: 13", что означает, что получено 13 UDP дейтаграмм, и "Sent UDP segments: 13", что означает, что отправлено 13 UDP дейтаграмм.
7. Выберем меню статистики узла PC2 и проверим, сколько UDP дейтаграмм он получил и отправил за все время нашего опыта. Будет, с большой вероятностью, выведен следующий результат: "Received UDP segments: 26", что означает, что получено 26 UDP сегментов, и "Sent UDP segments: 40", что означает, что отправлено 40 UDP сегментов. При заданных параметрах сети процент потерь больше, чем 7%, что не удовлетворяет требованиям. Можно попробовать использовать протокол TCP.
8. Выберем PC1 и запустим на нем TCP-приложение (TCP-сервер), выбрав в качестве прослушиваемого порт 8.
   Программа выдаст следующее сообщение:
   ```
   pc1 Applicationisnowlisteningonport8.
   ```
9. Выберем PC2 и пошлем через TCP-приложение 20 сообщений со строкой "ppp"на PC1. Программа выдаст похожее на следующее сообщение (для первого из двадцати сообщений, дошедших до PC1):
   ```
   pc2 Connectingtohost192.168.0.1:8.
        Pleasewait...
   pppccc2TCPSYN-packetfor192.168.0.1:8.
   111 PrrrotocolStackreceivedpacketttfromlocalInterface.
   ppcccCeatedARPResponsepacketo 192.168.0.2
   1 SendingpacketfromProtocccolStack(tooo192.168.0.2)...
   pc2 ProtocolStackreceivedpaketfroml calInterface
   ```

```
pc2SendingpacketfromProtocolStack(to192.168.0.1).
pc1ProtocolStackreceivedpacketfromlocalInterface.
pc1TCPSYN-packetreceivedfrom192.168.0.2:3000.
     TCPPort8hasstatus"busy"fromnow.
     CreatedTCPSYN-packetfr192.168.0.2:3000.
pppccc111SendingpppaaaccckkkeeetttfromProtooocolStack(to111999222...111
2TCPSYN-withACKreceivedfrom1:8.
     TCPPort3000stillhasstatus"busy".
pppccc222CreatedTCPacknowledgementpacketfooor192.168.0.1:8.
     SendingpacketttfromProtocolStack(ttt192.168.0.1).
pc1TCPpacketwihestablishinnngconnnnecccionACKreceived
     from192.168.0.2:3000.Conectioonfirmed!
     NewTCPconnectionestablished!
pc2Startsendingechomessage'ppp'to
     192.168.0.1:8
pppccc222CreatedTCPdatapacketfor192.168.0.111:8...
     SendingpacketfromProtocccolStack(tooo92168.0.1)...
pppccc111PrrrotocolStackreceivedpaketfromlcalInterface
     CeatedTCPacknowledgementpacket
     for192.168.0.2:3000.
pc1SendingpacketfromProtocolStack
     (to192.168.0.2).
pppccc222ProtocolStttackreceivedpacketfromlocalInterface.
     TCPpackewithestablishingcooonnnnnneeeccctttiiiooonnnACK
     receivedfrom192.168.0.1:8.Cconfirmed!
pc1TCPpacketwithdatareceivedfrom192.168.0.2:3000.
     Passingdatatoapplicationprogram.
pppccc111Reeecievingechomessage'ppppp'fromclient.
     Sndingechomessage'pp'toclient.
```

10. Выберем меню статистики узла PC1 и проверим, сколько TCP сегментов он получил и отправил. Будет выведен следующий результат: "Received TCP segments: 45", "Sent TCP segments: 43", "Sent TCP ACKs: 23", что означает, что отправлено 23 подтверждения, также будет нулевая статистика по отосланным и принятым дубликатам. Выберем меню статистики узла PC2 и проверим, сколько TCP сегментов он получил и отправил. Будет выведен следующий результат: "Received TCP segments: 43", "Sent TCP segments: 45", "Sent TCP ACKs: 23", что означает, что отправлено 23 подтверждения, также будет нулевая статистика по отосланным и принятым дубликатам. Из этого можно сделать

вывод о том, что для хорошей линии передач излишне проводить загрузку канала подтверждениями о получении сегментов, которые занимают около 50% сегментов, задействованных в обмене информацией, однако, были доставлены все 20 сообщений, что удовлетворяет требованиям по процентам потерь.

11. Обнулим статистику узла PC1 и PC2. Теперь установим коэффициент пропускания 60 и снова пошлем с PC2 на PC1 20 TCP сегментов.
12. Выберем PC2 и пошлем через TCP-приложение 20 сообщений со строкой "ppp"на PC1.
    Программа, в нашем случае, выдаст следующее сообщение:
    ```
    pppccc2Packeeetlostduetooophysiiicallinkproblems!
    1 Servrawaitingcnnectontimeout!
        Nowserverislisteningtoport:8.
    pc2 Connectiontimeout! Closingconnection
        tohost:192.168.0.1:8.
    ```
    Это говорит о том, что на PC2 было закрыто подключение к PC1 и на PC1 было закрыто подключение к PC2, т.к. качество линий *в данном примере* не позволяет обмениваться информацией за установленные программой на соединение промежутки времени. При таких параметрах сеть не удовлетворяет требуемым условиям по потерям: не более 7%.
    Если проверить статистику PC2, то можно увидеть, что было отправлено 14 дубликатов, а получено 27 дубликатов. В то время, как было отправлено 10 сегментов, а получено 11.
13. Обнулим статистику узла PC1 и PC2. Теперь установим коэффициент пропускания 88 и снова пошлем с PC2 на PC1 5 TCP сегментов.
14. Выберем меню статистики узла PC2 и проверим, сколько TCP сегментов он получил и отправил за время нашего опыта. Будет, с большой вероятностью, выведен результат такой, что было отправлено 14 TCP сегментов, 8 дубликатов, 6 подтверждений, также было получено 10 сегментов.
15. В результате анализа полученных результатов можно сделать следующие выводы. В условиях качественного обеспечения передачи UDP протокол показал себя с хорошей стороны, так как все дейтаграммы дошли до адресатов. По времени было затрачено 32ms. Не тратилось время на установление соединения и на подтверждения получения пакетов. При плохом качестве линий не все пакеты дошли до пунктов назначения. Оправданием

использования UDP на плохих линиях может стать только то, что информация за время задержки или потери станет неактуальна, и ее можно не передавать (к примеру, видеоконференция через Интернет).

Результаты проведенной работы по протоколу TCP говорят о неэффективном использовании им (данным протоколом) качественных линий, так как дополнительное время тратится на подтверждение пакетов, а также на установление и разрыв связи. В условиях некачественной физической линии использование TCP явно предпочтительнее, так как "потерявшиеся" сегменты пересылаются и, в конечном счете, доходят до адресата. По времени передача по протоколу TCP заняла 344ms, что в 10.75 раза больше, чем время затраченное при передаче через UDP. Таким образом, применение протокола оправдано в случаях, требующих гарантированного получения адресатом всей посылаемой информации (к примеру, проверка электронной цифровой подписи).

Очевидно, что при использовании UDP сеть начинает удовлетворять семипроцентному критерию по потере пакетов при коэффициенте пропускания между узлами PC1 и PC2 не менее 93%. Если использовать TCP, то критерий по потере пакетов удовлетворяется при коэффициенте пропускания между узлами PC1 и PC2, принадлежащем интервалу от 60 до 65.

Контрольные вопросы

1. Какой из протоколов транспортного уровня обеспечивает надежную доставку данных? За счет какого механизма обеспечивается гарантия доставки?
2. Назовите ситуации, в которых применение протокола UDP является целесообразным.
3. Назовите ситуации, в которых применение протокола TCP является целесообразным.
4. Чем в TCP обеспечивается ускорение работы по передаче данных?
5. Сколькими пакетами обмениваются во время UDP-соединения клиент и сервер?
6. Сколькими пакетами обмениваются во время TCP-соединения клиент и сервер? Какие существуют пакеты TCP?