Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 02.09.2021 10:07:07
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждения высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2017 г.

ПРОТОКОЛ УПРАВЛЕНИЯ ТРАНСПОРТОМ TCP

Методические указания к лабораторной работе
для студентов укрупненной группы специальностей и
направлений подготовки 10.00.00 «Информационная безопасность»

Курск 2017

УДК 621.(076.1)

Составитель: М.О. Таныгин

Рецензент

Кандидат технических наук, доцент кафедры
«Информационная безопасность» И.В. Калуцкий

Протокол управления транспортом TCP [Текст] : методические указания к лабораторной работе/ Юго-Зап. гос. ун-т; сост.: М.О. Таныгин. – Курск, 2017. – 19 с.: ил. 12, табл. 2. – Библиогр.: с. 19.

Содержат сведения по вопросам лабораторной работы по основам мониторинга безопасности инфокоммуникационных систем и сетей. Указывается порядок выполнения лабораторной работы, правила оформления отчета.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальности.

Предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00 «Информационная безопасность».

Текст печатается в авторской редакции

Подписано в печать 24.11.17. Формат 60x84 1/16.
Усл.печ. л. 1,10. Уч.-изд. л. 1,00. Тираж 100 экз. Заказ. Бесплатно. 214/
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Протокол управления транспортом TCP

Протокол управления транспортом (*Transport Control Protocol – TCP*) обеспечивает гарантированную доставку пакетов, соответствующий исходному сообщению порядок следования байт, исключает ошибки передачи, а также реализует управление потоком и производительностью соединения. Надѐжность передачи данных обеспечивается поддержкой TCP следующих механизмов:

последовательной нумерации байтов в передаваемых блоках данных;

подтверждения приема данных с реализацией таймаутов и повторных передач (ретрансмиссий) неподтверждѐнных данных;

предварительной установки соединения между отправителем и получателем;

процедуры нормального и внештатного завершения соединения между отправителем и получателем;

обязательного использования контрольной суммы для защиты TCP-пакета.

Рассмотрим обобщѐнную схему взаимодействия отправителя и получателя с использованием протокола TCP. Сетевое приложение, выполняющееся на компьютере отправителя, передает отправляемые по сети данные программному обеспечению TCP, которое размещает данные в своѐм *выходном буфере* (*send buffer*) (рис. 1). Затем TCP вырезает так называемый *сегмент данных* (*segments*) из буфера, добавляет к нему TCP-заголовок и передает протоколу IP для доставки в виде отдельной дейтаграммы. *Максимальный размер сегмента* (*Maximum Segment Size – MSS*) определяется значением параметра максимальной единицы передачи (*Maximum Transfer Unit – MTU*) технологии канального уровня и суммарным размером заголовков TCP и IP:

MSS = MTU – заголовок TCP – заголовок IP

В случае, если TCP-сегмент передаѐтся в IP-сети, работающей поверх Ethernet (напомним, что Ethernet характеризуется значением MTU = 1500 байт), то при размерах заголовков TCP и IP (без опций), равных по 20 байт, значение MSS будет составлять 1460 байт. Пакетирование данных в сегменты максимального размера обеспечивает максимальную производительность соединения,

поэтому до создания сегмента TCP будет ожидать, пока в выходном буфере не появится соответствующее количество данных.

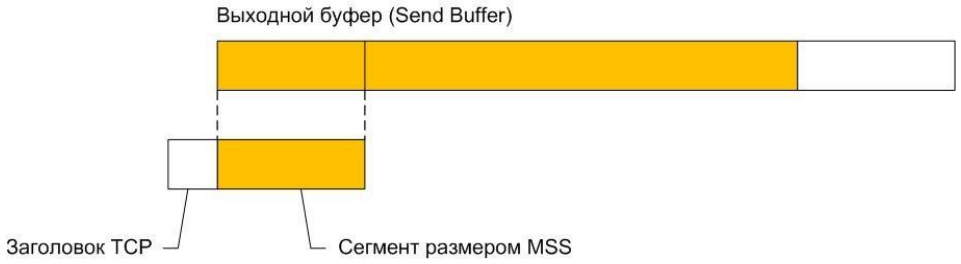

Рис. 1. Формирование сегмента TCP

На практике не всегда размер отправляемого сегмента равен MSS (такие случаи будут рассмотрены далее), однако следует заметить, что при равном MSS размере сегмента при прочих равных условиях производительность соединения будет наибольшей.

Как и для UDP, протокол TCP идентифицирует приложения на стороне отправителя и получателя указанием номеров портов. Номера портов TCP также находятся в диапазоне от 0 до 65535, а порты от 0 до 1023 называются общеизвестными и используются для доступа к стандартным службам, а порты выше 1023 выделяются клиентскому программному обеспечению.

TCP предусматривает присвоение порядкового номера каждому пересылаемому по соединению байту данных. В заголовке сегмента указывается порядковый номер (Sequence Number SEQ#) первого байта поля данных этого сегмента. В подтверждении TCP, высылаемом отправителю, указывается номер подтверждения (Acknowledgement Number ACK#), представляющий собой номер байта следующего за последним байтом в текущем сегменте, полученном получателем. Если подтверждение не приходит за интервал тайм-аута (timeout), данные передаются повторно. Такой механизм называется позитивным подтверждением с ретрансляцией (positive acknowledgment with retransmission) (рис. 2).

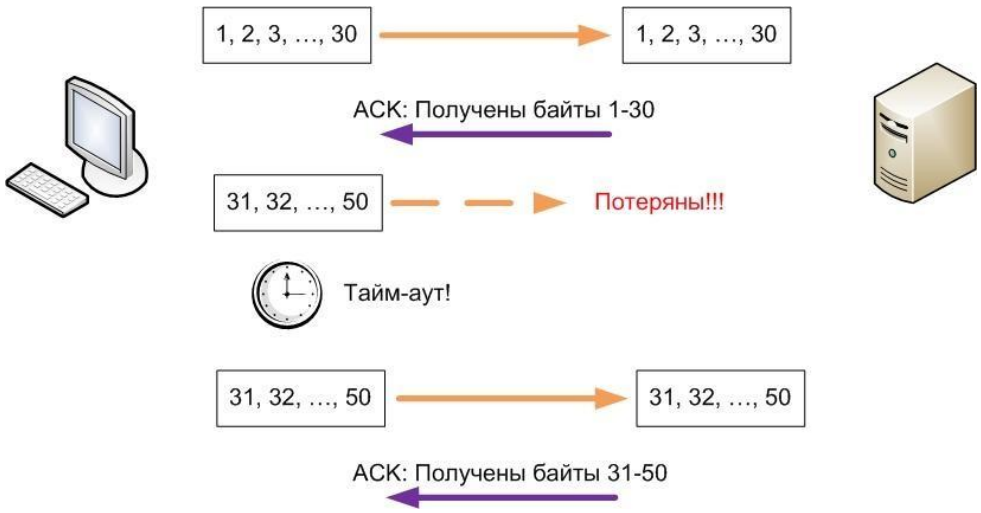

Рис. 2. Позитивное подтверждение с ретрансляцией в TCP

Перед высылкой первого сегмента получателю отправитель инициирует *процедуру установления соединения*, которую часто называют *тройным рукопожатием* (*three-way handshaking*), поскольку для установки соединения партнеры обмениваются тремя сообщениями (рис. 3).

Инициатором соединения является клиент. Во время установления соединения партнеры обмениваются значениями следующих параметров:

начальным порядковым номером (*Initial Sequence Number – ISN*) первого байта отправляемого сегмента (его значение выбирается случайным образом с использованием системного таймера), в примере на рис. 3 ISN клиента равен 1000, а ISN сервера – 8000;

размером буферного пространства для приема данных (так называемого *окна* (*Window*) – будет рассмотрено далее), на рис. 3 показано, что размер окна сервера (64 кбайта) превышает размер окна клиента (8 кбайт), что часто встречается на практике;

значением максимального размера сегмента (MSS).

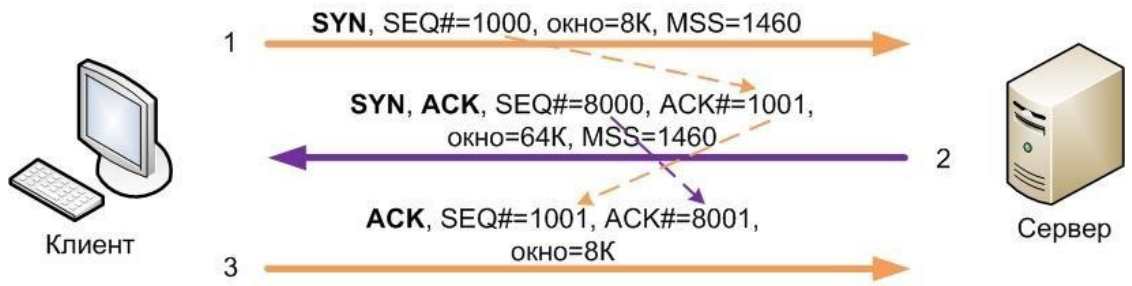

Рис. 3. Установление соединения TCP

TCP обеспечивает полнодуплексный режим работы, одновременно обслуживая два потока данных в разных направлениях. В ответ на инициирующий соединение пакет клиента сервер открывает встречное соединение, пересылая вместе с подтверждением принятия клиентского пакета свои значения указанных выше параметров. В третьем пакете высылается подтверждение клиентом факта установки инициированного сервером соединения. После завершения установления соединения происходит одновременная передача данных в обоих направлениях с присвоением передаваемым байтам последовательных, начиная с ISN + 1, номеров. Например, пакеты данных с сервера передаются одновременно с подтверждениями получения ранее принятых данных в пакетах с клиента.

При пересылке данных TCP подтверждения включаются в пересылаемые сегменты и содержат номер следующего байта, который ожидает получатель в поле данных сегмента.

На рис. 4 первый посланный клиентом сегмент содержит байты с номерами от 1001 до 2000, в его поле ACK указывается значение номера байта 3001, ожидаемого от сервера. Сервер отвечает клиенту сегментом с номерами байтов от 3001 до 4000, в его поле ACK указано значение 2001, означающее, что предыдущая посылка клиента успешно получена. Далее клиент посылает несколько сегментов, не дожидаясь подтверждений от сервера. Сервер использует единственный ACK для подтверждения принятия этих сегментов, экономя полосу пропускания соединения. На рис. 4 также показана пересылка данных при

потере сегмента.

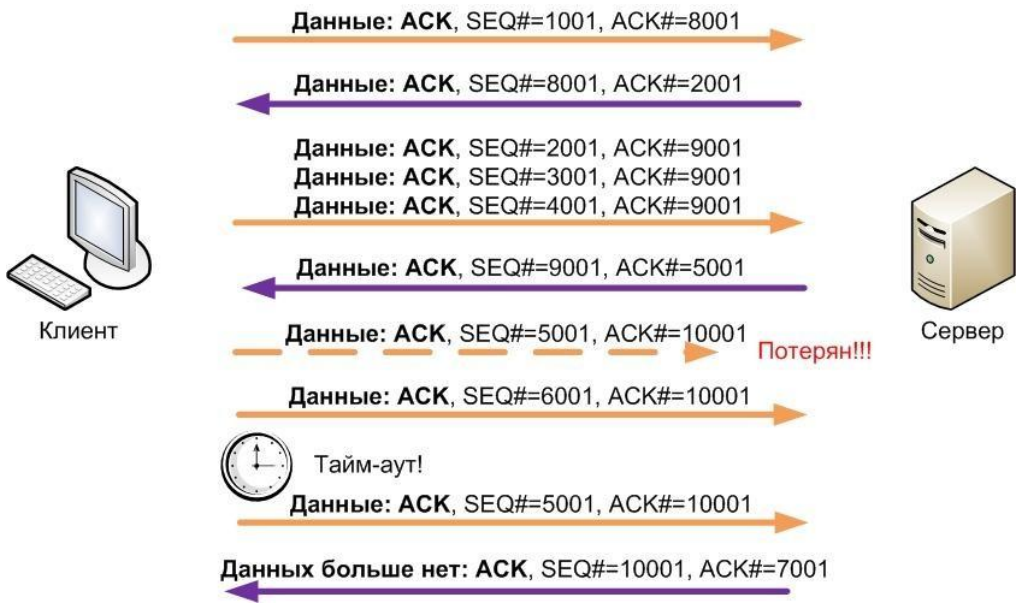

Рис. 4. Пересылка данных TCP

Нормальное завершение TCP соединения может быть инициировано любой стороной (клиентом или сервером). На рис. 5 иллюстрируется завершения сеанса при его инициализации сервером (например, после ввода команды logout пользователем в сеансе telnet). В этом случае выполняются следующие действия:

приложение на сервере указывает TCP на закрытие соединения;

TCP сервера посылает *заключительный сегмент* (*Final Segment □ FIN*), информируя своего партнера о том, что данных для отправки больше нет;

TCP клиента посылает ACK;

клиентское приложение сообщает своему TCP о закрытии соединения;

TCP клиента посылает сообщение FIN;

TCP сервера получает FIN от клиента и отвечает на него сообщением ACK;

TCP сервера указывает своему приложению на закрытие соединения.

Рис. 5. Завершение соединения TCP

Обе стороны могут одновременно начать закрытие. В этом случае обычное закрытие соединения завершается после отправки каждым из партнеров сообщения ACK. Каждая из сторон может запросить *внезапное завершение соединения* (*abrupt close*). Это допустимо, когда приложение желает завершить соединение или когда TCP обнаруживает серьезную коммуникационную проблему, которую не может разрешить собственными средствами. Внезапное завершение запрашивается посылкой партнеру одного или нескольких сообщений *Сброс* (*Reset – RST*), что указывается соответствующим флагом в заголовке TCP (флаги будут описаны далее).

Протокол TCP реализует механизмы управления потоком поступающих хосту-приемнику данных с целью предотвращение переполнения пакетами входного буфера. Во время установки соединения каждый из партнеров выделяет пространство памяти для входного буфера соединения и уведомляет об этом противоположную сторону. *Приемное окно* (*Receive Window*) – это пространство во входном буфере, ещѐ не занятое данными. Освобождение буфера от принятых данных выполняет приложение получателя. Этот процесс зависит от производительности и загруженности процессора хоста получателя (причем не только обработкой данных TCP соединения). Каждый посланный приемником ACK содержит сведения о текущем состоянии приемного окна в поле *Окно* (*Window*), в зависимости от которого регулируется поток данных от источника. Обычно подтверждения ACK высылаются не на каждый пересланный сегмент, а на непрерывный блок из нескольких сегментов, собранный в

приемном окне. Это позволяет не отбрасывать пришедшие не по порядку сегменты, а упорядочивать их в соответствии с последовательными номерами и размерами сегментов.

В некоторых случаях требуется пересылка сегментов, размер которых меньше MSS (например, при работе в командном режиме, когда каждый пакет содержит только байты одной команды). Для этого используется так называемое *выталкивание* (*Push*) *данных из выходного буфера*, при этом отправителем в заголовке TCP выставляется флаг Push.

Также TCP может пересылать в сегменте *срочные данные* (*Urgent Data – URG*) вместе с другими данными. В этом случае сегмент маркируется флагом URG, а двухбайтовое поле Указатель срочности содержит смещение в байтах, которое должно быть добавлено к значению SEQ# заголовка TCP для получения SEQ# последнего байта срочных данных в данном сегменте. Принимающее приложение должно быть в состоянии определить, когда появится указатель срочности. Приложение находится в *режиме срочности* (*Urgent Mode*) все время, пока читает данные с текущей позиции до указателя срочности. После того как указатель срочности принят, приложение возвращается в нормальный режим. Примером использования режима срочности является ситуация, когда пользователь прерывает загрузку данных, при этом в поток передаваемых данных вставляется команда, на которую указывает указатель срочности и которую должно распознать и обработать приложение.

Описанные механизмы поддерживаются значениями полей заголовка TCP, формат которого приведен на рис. 6.

Поля *Порт источника* (отправителя) и *Порт назначения* (получателя) содержат 16 битовые значения портов, указывающих на области памяти с приложениями отправителя и получателя соответственно.

Поле *Порядковый номер* содержит 32-битовые значение номера первого байта данного сегмента, начиная от ISN. Поле *Номер подтверждения* содержит значение последовательного номера, ожидаемого от противоположной стороны следующим.

<table>
<tr><td colspan="16">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td><td colspan="16">16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td></tr>
<tr><td colspan="16">Порт источника</td><td colspan="16">Порт назначения</td></tr>
<tr><td colspan="32">Порядковый номер</td></tr>
<tr><td colspan="32">Номер подтверждения</td></tr>
<tr><td colspan="4">Длина заголовка (смещение данных)</td><td colspan="6">Зарезервировано</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Окно</td></tr>
<tr><td colspan="16">Контрольная сумма</td><td colspan="16">Указатель срочности</td></tr>
<tr><td colspan="32">Опции</td></tr>
<tr><td colspan="32">Данные</td></tr>
</table>

Рис. 6. Заголовок TCP

Четырèхбитовое поле *Длина заголовка* содержит размер заголовка в четырèхбайтовых словах (если заголовок не содержит опций, то его размер равен 5 × 4 байта = 20 байт).

Следующие 6 бит за полем длины заголовка зарезервированы и обычно равны 0.

Поле флагов TCP содержит шесть однобитовых флагов, передающих назначение пакета:

URG – устанавливается в единицу для данных, которые должны быть обработаны получателем срочно, в этом случае в поле *Указатель срочности* указывается последний байт срочных данных;

ACK – устанавливается в единицу для сегментов, выполняющих подтверждение принятых сегментов (на практике для всех пакетов TCP-сеанса, кроме первого);

PSH – устанавливается в единицу приложением, отправляющим данные, чтобы указать TCP, что нужно отправлять данные из выходного буфера, не дожидаясь его заполнения до максимального размера сегмента;

RST – устанавливается в единицу для завершения сеанса в связи с ошибкой или внештатной ситуацией;

SYN – устанавливается в единицу при установлении соединения;

FIN – устанавливается в единицу при нормальном завершении соединения.

Поле *Окно* предназначено для указания текущего размера окна входного буфера получателя.

Контрольная сумма вычисляется как 16-битовое дополнение до единицы суммы дополнений до единицы всех 16-битовых слов псевдозаголовка, аналогичного псевдозаголовку UDP (значение поля Протокол в случае TCP = 6) и заголовка TCP.

Опции TCP могут занимать целое число байт, в качестве примера опций можно привести опцию, передающую значение MSS при установке соединения. Заголовок TCP должен заканчиваться на 32-битной границе, для этого может использоваться заполнение нулями.

Задание для самостоятельной работы

Наиболее простой способ исследовать сообщения ICMP – использование программы `ping`. Откройте окно командной строки (терминал) и подготовьте в нѐм команду `ping`, в качестве параметра которой укажите IP-адрес соседнего компьютера сети. Запустите анализатор протоколов *Wireshark* и настройте в нѐм фильтр на захват только ICMP-пакетов (для этого необходимо в окне, открытом командой Capture-Options в поле Capture Filter прописать `icmp`). Запустите анализатор на захват пакетов, перейдите в окно командной строки (терминала) и нажмите клавишу Enter, инициируя отправку `ping`-пакетов. Приведите в отчѐт структуру ICMP-заголовка и значения его полей для эхо-запроса и эхо-ответа. Для расшифровки полей сообщений используйте описание стандарта на ICMP (RFC-792).

Повторите эксперимент по захвату пакетов, но уже используйте программу `tracert` (`traceroute`), в качестве параметра которой укажите имя любого интернет-сервера (если администратор сети запретил пересылку ICMP-пакетов в

Интернете, попробуйте в качестве целевого сервера использовать имя или IP-адрес компьютера из соседней с Вами сети/подсети). Фильтр в Wireshark должен оставаться таким же, поскольку сообщения `tracert` (`traceroute`) – это также ICMP-сообщения. Захватите последовательность ICMP-пакетов и приведите в отчѐт значение поля TTL заголовка IP, структуру ICMP-заголовка и значения его полей для первых двух запросов, а также аналогичные параметры для ответов на эти запросы. Для расшифровки полей сообщений используйте описание стандарта на ICMP (RFC-792) .

Для захвата пакетов с сообщениями ICMP о недостижимости адресата постройте в программе Cisco Packet Tracer сеть, приведенную на рис. 7 . Задайте конфигурацию хостам и маршрутизаторам, например такую, которая указана в табл. 1.

Таблица 1 Адресная информация сети, использующейся для исследования ICMP

Название	IP-адрес	Маска	Шлюз	Примечание
PC0	192.168.1.1	255.255.255.0	192.168.1.254	Отправитель пакетов
PC1	192.168.2.1	255.255.255.0	192.168.2.254	Получатель пакетов
Router0-Fa0/0	192.168.1.254	255.255.255.0		Шлюз

Если не сконфигурировать правило маршрутизации для Router0, из которого он понимал бы, что сеть `192.168.2.0/24` находится за Router1 и па- кеты необходимо принимать именно ему, то при получении пакетов, направляемых в эту сеть, маршрутизатор не знает, куда их передавать. В этом случае он должен выслать отправителю ICMP-сообщение о недостижимости адресата.

Создайте расширенный протокольный блок данных (Complex PDU), выполнив щелчок на соответствующей кнопке и заполнив информацию, помеченную на рис. 7 (здесь имитируется отправка сообщения протокола прикладного уровня *telnet*).

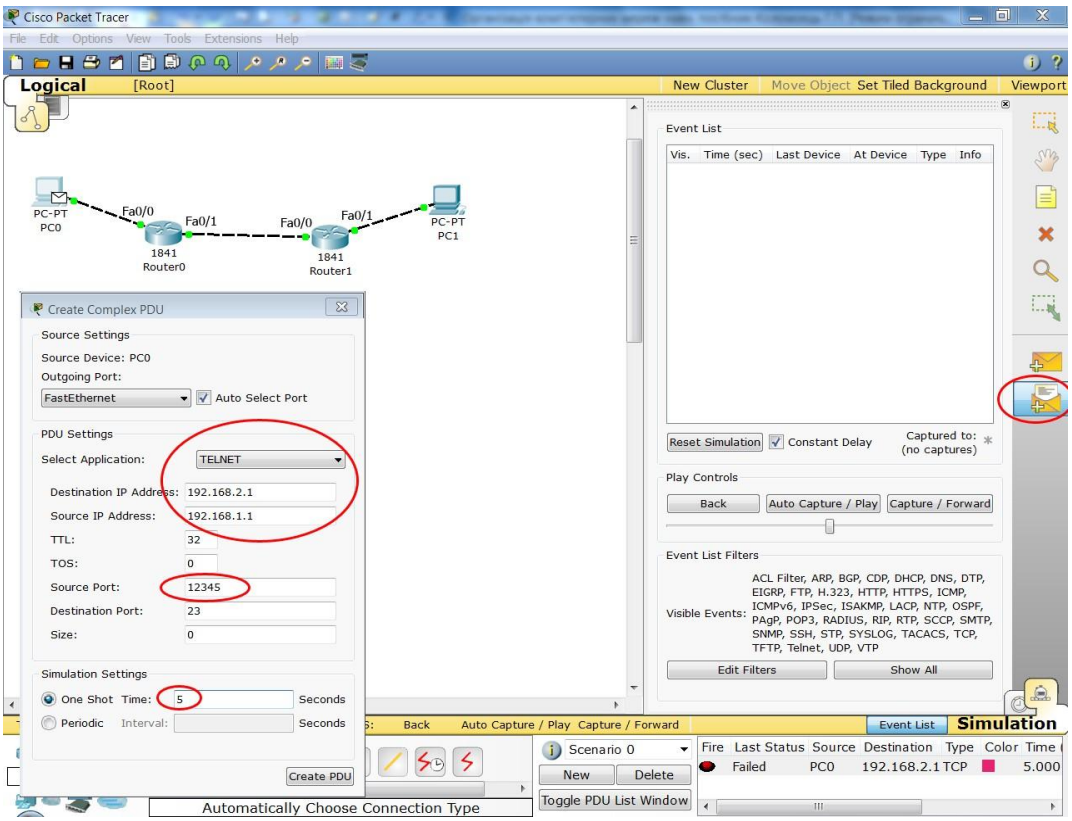

Рис. 7. Исследование ситуации, в которой адресат недостижим

Запустите симуляцию, выполнив щелчок по кнопке Auto Captire/Play, Вы увидите, что после получения пакета с сообщением *telnet* Router0 высылает хосту ICMP-сообщение. Просмотрите его в виртуальном анализаторе протоколов, дважды выполнив щелчок по сообщению в списке Event List. При- ведите в отчѐт структуру и значения полей ICMP-заголовка.

Протокол транспортного уровня UDP используется, в частности, *системой доменных имен* (*Domain Name System – DNS*), позволяющей определять значение IP-адреса по доменному имени типа zhu.edu.ua. Сообщения DNS от хоста-клиента до DNS-сервера локальной сети и обратно обычно пересылаются в поле данных UDP-пакетов, поскольку обычно локальные сети считаются надѐжными. В операционных системах Windows и Linux с установленным по умолчанию сетевым стеком TCP/IP имеется

программа nslookup, позволяющая выполнять интерактивные запросы к DNS.

Откройте окно командной строки в Windows (либо окно терминала в Linux) и введите команду nslookup <DNS-имя сервера>, пока не нажимайте клавишу Enter (в качестве имени сервера может быть выбрано любое имя известного Вам, например, WWW-сервера). Запустите анализатор протоколов *Wireshark*, выполните команду Capture-Options и в открывшемся окне в поле Capture Filter: введите udp (то есть будут захватываться только кадры, содержащие UDP-пакеты). Нажмите на кнопку Start, перейдите в окно командной строки (терминала) и нажмите клавишу Enter. Выполните анализ захваченных кадров. Приведите в отчет дамп заголовка UDP-пакета и его расшифровку.

Ещѐ одним протоколом, использующим в качестве транспортного UDP, является *протокол динамического конфигурирования хоста* (*Dynamic Host Configuration Protocol – DHCP*), позволяющий, в частности, автома- тически задавать узлам сети адресную информацию сетевого уровня (IP- адрес, маску подсети и т.д.). Мы сможем исследовать протокольные блоки данных, передаваемые этим протоколом путѐм моделирования в програм- ме *Cisco Packet Tracer*. Для этого запустите программу и создайте в ней простейшую сеть, состоящую из сервера, соединѐнного с коммутатором и компьютера, пока не подсоединѐнного к коммутатору (рис. 8).

Выполните настройку адресной информации сервера, компьютер по умолчанию использует протокол DHCP (рис. 8). Выполните щелчок на команде Services в окне настроек сервера и выберите команду DHCP. В открывшемся окне Вы можете задать размер пула адресов с помощью параметров Start IP Address и SubnetMask (в примере на рис. 9 адрес пула начинается с 192.168.1.10 и заканчивается 192.168.1.254, что составляет 246 адресов). Здесь также можно указать IP-адреса шлюза данной сети и локального DNS-сервера, в этом случае эти параметры также автоматически назначаются хостам сети.

Процедуру автоматического назначения адресной информации, а также протокольные передаваемые при этом блоки данных можно увидеть в режиме Simulation. Перейдите в этот режим

и настройте фильтрацию только пакетов с DHCP-сообщениями в окне, открывающемся щелчком по кнопке Edit Filters (рис. 10).

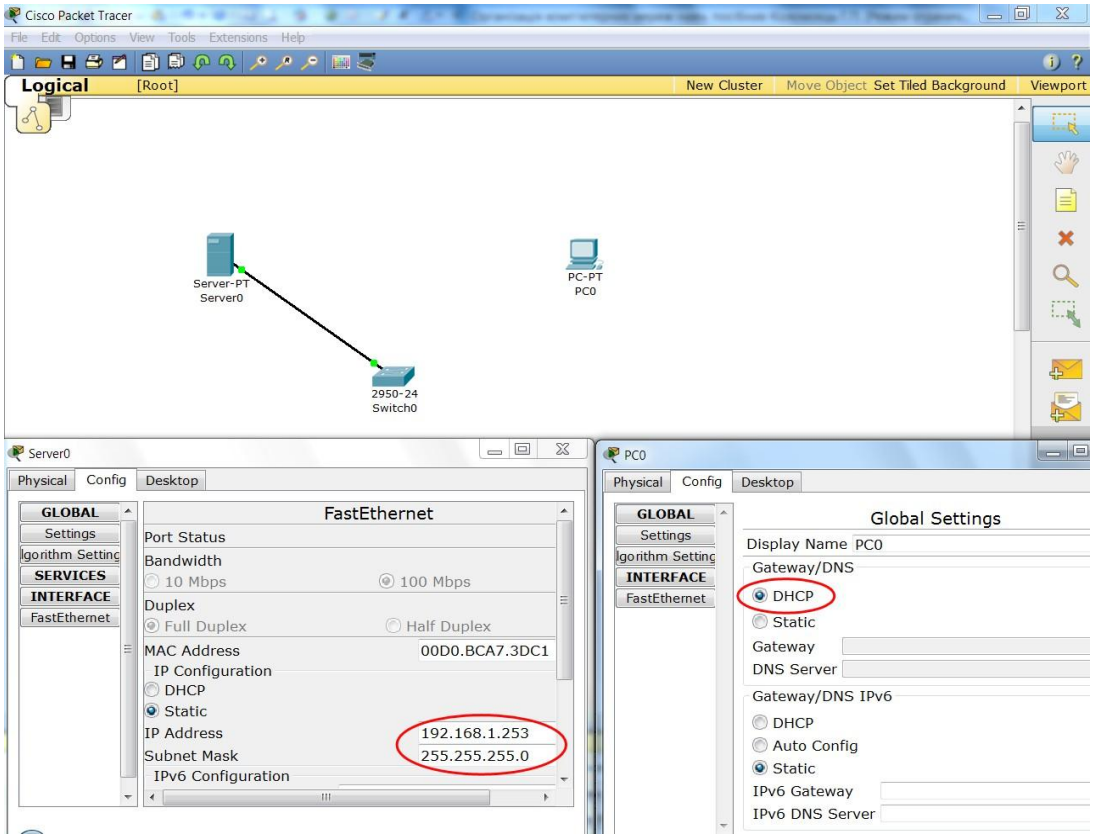

Рис. 8. Сеть с DHCP сервером

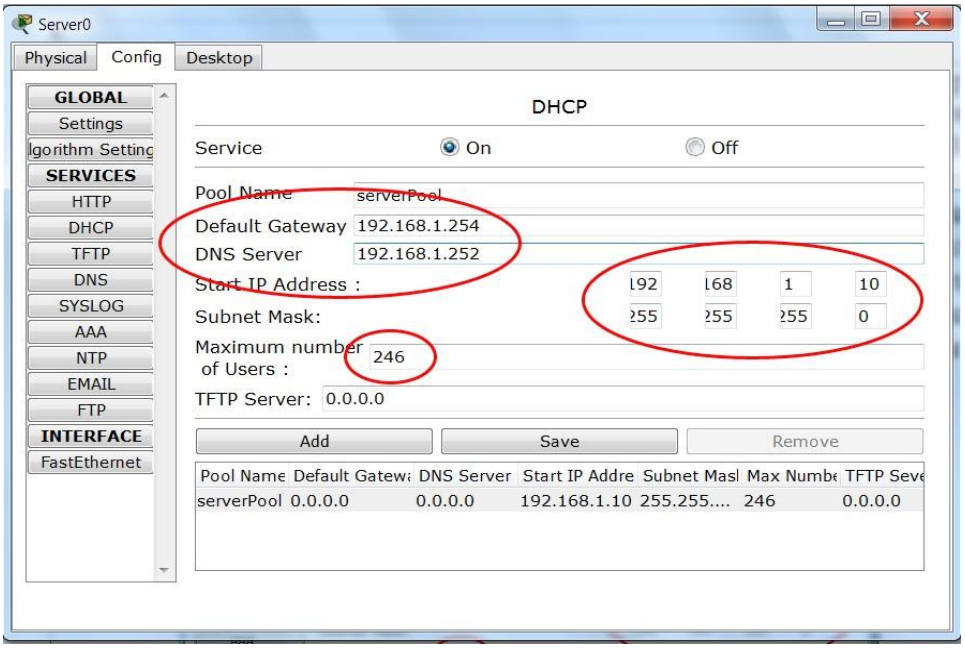

Рис. 9. Настройки DHCP сервера

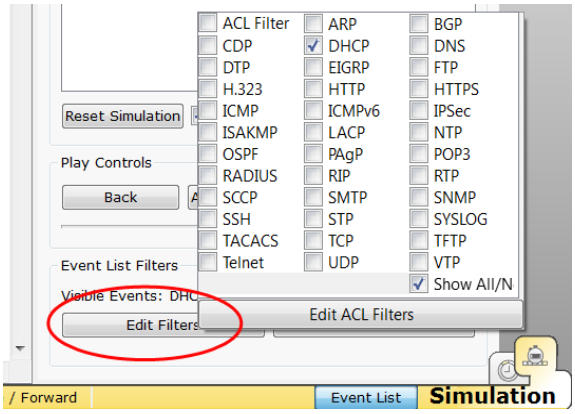

Рис. 10. Настройка фильтрации DHCP-сообщений

Далее подсоедините компьютер к коммутатору и запустите симуляцию, нажав кнопку Auto Capture/Play. После этого по сети будут переданы несколько пакетов с сообщениями DHCP между компьютером и сервером. Выполните щелчок по одному из них для просмотра структуры захваченного кадра (рис. 11).

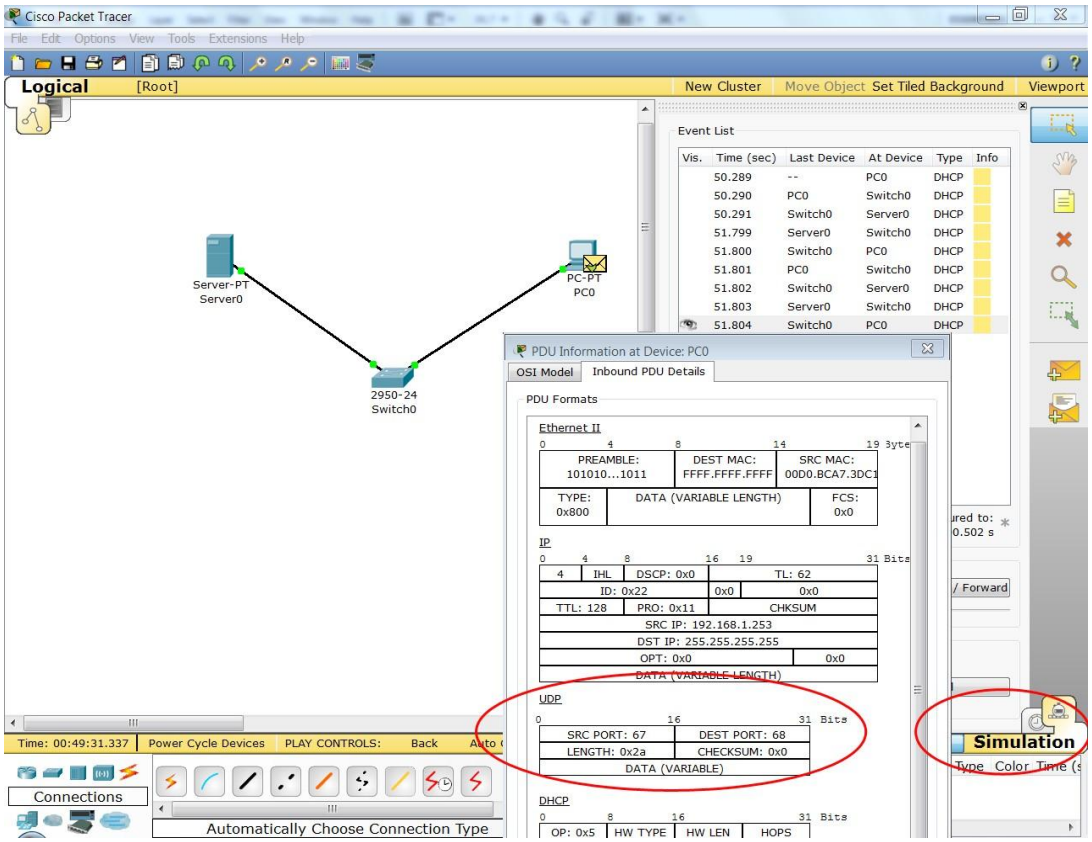

Рис. 11. Захват кадров с DHCP-сообщениями

Откройте окно с конфигурацией компьютера и убедитесь, что

он автоматически получил адрес из сконфигурированного Вами пула.

Повторите описанный эксперимент, установив в качестве начального адреса пула адрес `192.168.1.X`, где `X` = Вашему порядковому номеру в журнале академгруппы. Приведите в отчѐт заголовок UDP вместе с расшифровкой его полей.

Протокол транспортного уровня TCP используется большинством популярных протоколов прикладного уровня, в частности, при пересылке HTTP-пакетов между клиентами (браузерами) и HTTP-серверами (Web- серверами). Запустите программу-браузер и задайте в ней адрес любого сервера в Интернете, но пока не инициируйте соединение с ним (рекомендуется выбрать сервер с небольшим количеством информации на его главной странице, например, http://ya.ru).

Запустите программу анализатор протоколов *Wireshark* и настройте в ней фильтр для захвата пакетов TCP. Включите захват и инициируйте соединение с выбранным сервером в браузере. По окончании загрузки страницы остановите захват. Вы должны были захватить пакеты HTTP-сеанса, начиная с пакетов, устанавливающих соединение, и заканчивая пакетами, осуществляющими нормальное завершение соединения. Поскольку могли захватиться пакеты, направленные другим серверам, можно их отфильтровать, подав команду, оставляющую только пакеты, отправленные выбранному Вами серверу или полученные от него. Для этого необходимо из окна захваченных пакетов выяснить IP-адрес сервера (для http://ya.ru он 77.88.21.3), ввести в поле Filter команду ip.dst == 77.88.21.3 or ip.src == 77.88.21.3 и нажать кнопку Apply (рис. 12).

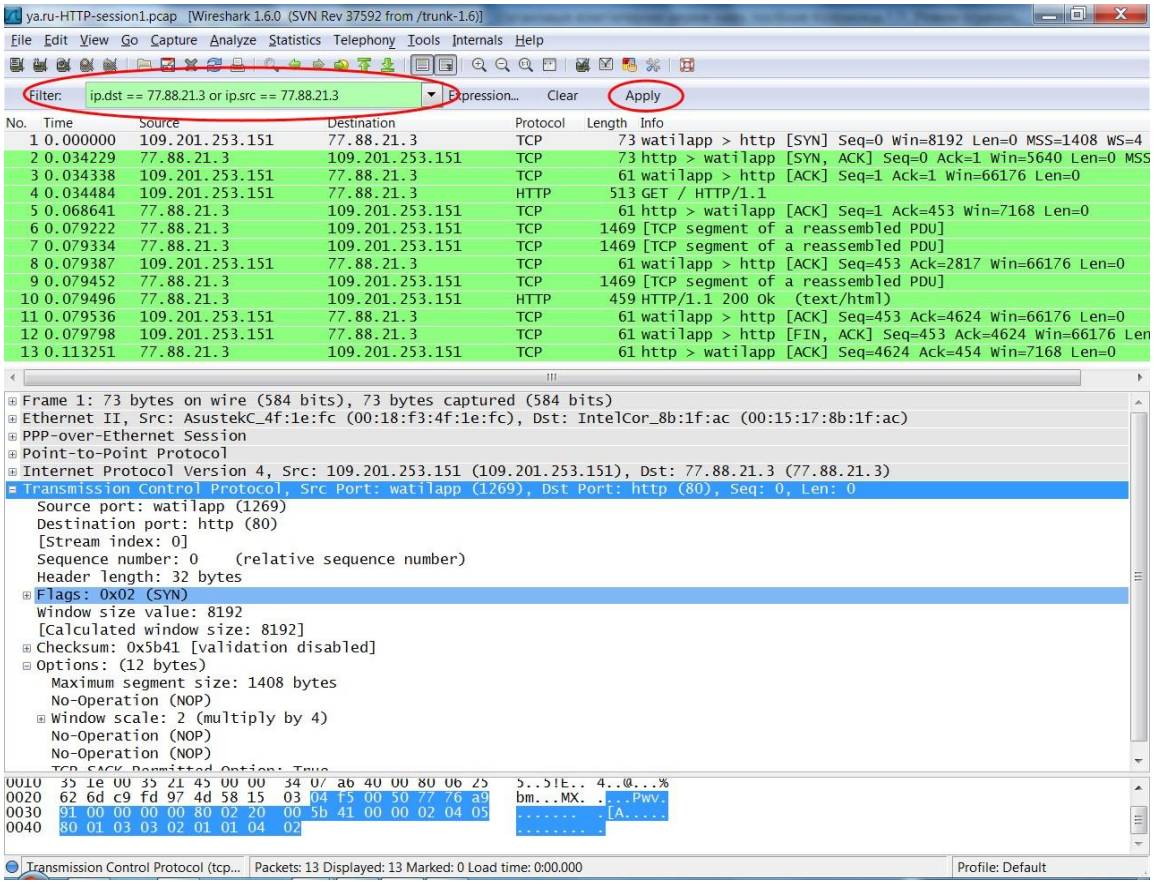

Рис. 12. Захват кадров с DHCP-сообщениями

Выполните анализ захваченных кадров, заполнив приведенную ниже таблицу, подставляя вместо данных примера свои данные (табл. 2).

Таблица 2

Данные TCP-сеанса

Клиент:		IP = 109.201.253.151		Порт = 1269		MSS = 1408	
Сервер:		IP = 77.88.21.3		Порт = 80		MSS = 1410	
№ па-кета	Отправи- тель	Получа-тель	SEQ# hex	ACK# hex	Раз-ность dec	Flags	Длина дан-ных
1	Клиент	Сервер	7776A991	0		SYN	0
2	Сервер	Клиент	2C815468	7776A992		SYN, ACK	0
3	Клиент	Сервер	7776A992	2C815469		ACK	0
4	Клиент	Сервер	7776A992	2C815469	453	PSH, ACK	453
5	Сервер	Клиент	2C815469	7776AB56		ACK	0
6	Сервер	Клиент	2C815469	7776AB56		ACK	1408
…							

В поля SEQ# и ACK# вводите не относительные номера, которые предлагает Wireshark в окне анализа структуры заголовков, а абсолютные значения этих параметров из дампа. Рассчитайте для пакетов с ненулевым размером поля данных разность ACK# следующего пакета с противоположной стороны и SEQ# текущего пакета. Учитывая, что SEQ# является номером первого байта текущего сегмента, добавьте к полученному значению единицу, переведите в десятичную систему и запишите результат в столбец Разность.

Пример в таблице: $7776AB56_{H} - 7776A992_{H} = 1C4_{H} + 1 = 1C5_{H} = 453_{10}$.

Укажите в отчѐте диапазон номеров пакетов, устанавливающих TCP- соединение, передающих данные в этом соединении и завершающих его.