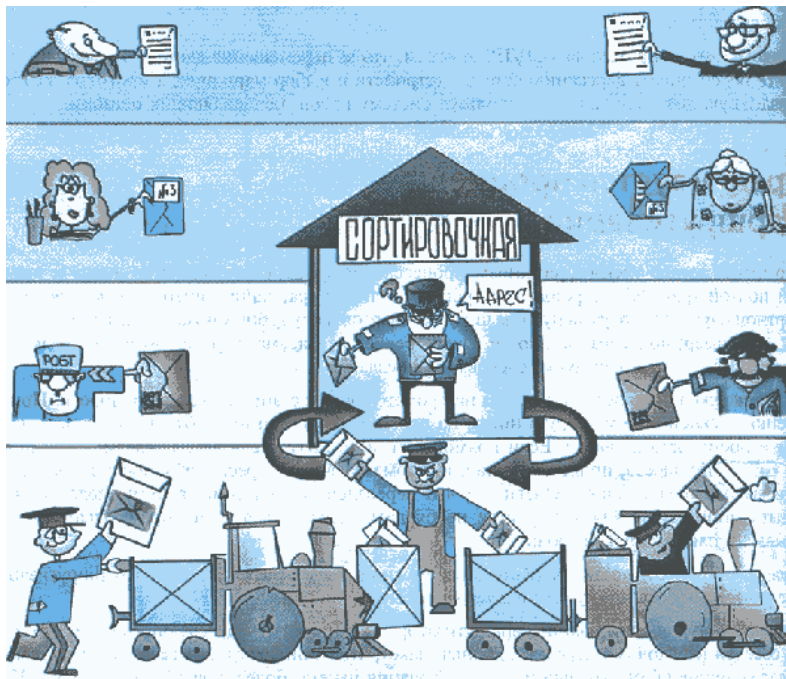


# Основы компьютерных сетей. Тема №3. Протоколы нижних уровней (транспортного, сетевого и канального) / Хабр

 [habr.com/ru/articles/308636](https://habr.com/ru/articles/308636)

Денис

23 сентября 2016 г.



Приветствую всех читателей. Пришло наконец время поговорить о протоколах, находящихся на нижних уровнях. В этой статье будут разобраны протоколы канального, сетевого и транспортного уровней. Присаживайтесь поудобнее и читайте на здоровье.

## Содержание

Как вы помните, я уже говорил о том, что в сетях важно строгое соблюдение всех правил для корректной работы. А именно процесс инкапсуляции и деинкапсуляции. Поэтому, когда в предыдущей статье говорили о протоколах верхних уровней, я вскользь упоминал о некоторых протоколах нижних уровней, так как они постоянно вылезали и напоминали о себе. Объясню почему. Посмотрите сейчас на картинку выше. Тут приведена работа почты. Взгляните на двух лысых дядек вверху, которые написали письмо и светятся от счастья. Но толку не будет от письма, если его не увидит адресат. Для этого они воспользуются почтовой службой. Их письмо примет сотрудница почтового отделения и положит в конверт. Конверт она подпишет, чтобы было понятно от кого оно и кому. Дальше это письмо заберет курьер и отнесет в сортировочный центр. Ниже стоит мужичок в фуражке и фартуке, который жонглирует письмами. Он знает, куда положить письмо, чтобы оно дошло до

адресата. И в самом низу поезд, который является транспортным узлом. Заметьте, что тут важна роль каждого для удачной отправки и доставки письма.

В сетях все тоже самое. Решили вы залезть на сайт и почитать новости. Набираете в строке браузера адрес сайта. Дальше ваш компьютер как то должен эти страницы запросить. И тут уже на помощь придут протоколы пониже, которые являются транспортным узлом. Здесь каждый уровень можно сравнить с вышеописанными личностями на рисунке.

Подведу я всю эту канитель к общему знаменателю и поделюсь примером, который я когда-то для себя вывел. У вас есть оконечное сетевое устройство. Не важно компьютер, ноутбук, планшет смартфон или еще что. Каждое из этих устройств работает по стеку TCP/IP. А значит, оно соблюдает его правила.

- 1) Прикладной уровень. Тут работает само сетевое приложение. То есть веб-браузер, который запускается, например с компьютера.
- 2) Транспортный уровень. У приложения или службы должен быть порт, который он слушает и по которому с ним можно связаться.
- 3) Сетевой уровень. Здесь присутствует IP-адрес. Его еще называют логическим адресом устройства в сети. При помощи него можно связаться с компьютером, на котором запущен этот самый браузер, а значит, и достигаться до самого приложения. Имея данный адрес, он является участником сети и может связываться с другими участниками
- 4) Канальный уровень. Это сама сетевая карточка или антенна. То есть передатчик и приемник. У него есть физический адрес для идентификации этой сетевой карты. Кабели, коннекторы тоже относятся сюда. Это среда, которая свяжет компьютер с другими участниками.

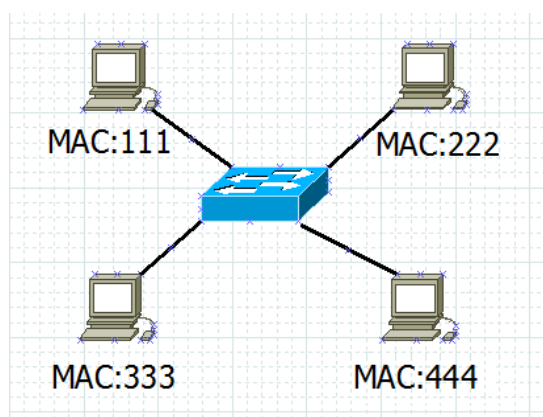
Начнем с самого нижнего уровня. Это канальный и физический уровень, если рассматривать с точки зрения модели OSI и уровень доступа, если смотреть с высоты стека протоколов TCP/IP. Пользуемся мы TCP/IP, поэтому я буду говорить с ее точки зрения. Уровень доступа, как вы поняли, объединяет в себе физический и канальный уровень.

**Физический уровень.** Или как его любят называть «электрический уровень». Задаёт параметры сигнала, а также какой вид и форму имеет сигнал. Если, например, используется Ethernet (который передает данные при помощи провода), то какая модуляция, напряжение, ток. Если это Wi-Fi, то какие использовать радиоволны, частоту, амплитуду. К этому уровню можно отнести сетевые карты, Wi-Fi антенны, коннекторы. На этом уровне вводится такое понятие, как биты. Это единица измерения передаваемой информации.

**Канальный уровень.** Этот уровень используется для того, чтобы передать не просто биты, а осмысленные последовательности из этих бит. Используется для передачи данных в одной канальной среде. Что это значит, я опишу чуть позже. На этом уровне работают MAC-адреса, которые еще называют физические адреса.

Термин «физические адреса» ввели не просто так. Каждая сетевая карта или антенна имеет вшитый адрес, который ей присваивает производитель. В предыдущей статье я упоминал термин «протоколы». Только там это были протоколы верхнего уровня, а если точнее, то прикладного. На канальном уровне работают свои протоколы и количество их не маленькое. Самые популярные — это Ethernet (используется в локальных сетях), PPP и HDLC (они используются в глобальных сетях). Это конечно далеко не все, но Cisco в своей сертификации CCNA рассматривает только их.

Тяжело все это понять в виде сплошного сухого текста, поэтому объясню на картинке.

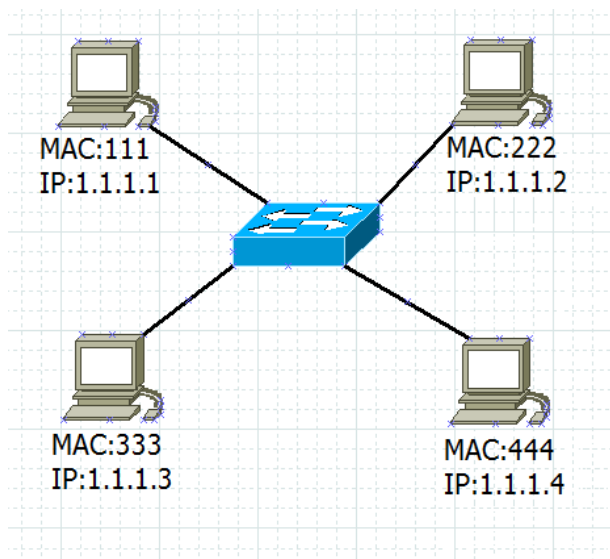


Забудьте сейчас про IP-адреса, модель OSI и стек протоколов TCP/IP. У вас есть 4 компьютера и коммутатор. На коммутатор внимания не обращайтесь, так как это обычная коробка для соединения компьютеров. У каждого компьютера есть свой MAC-адрес, который идентифицирует его в сети. Он должен быть обязательно уникальным. Хотя я и представил их 3-х значными, это далеко не так. Сейчас эта картинка только для логического понимания, а как это работает в реальной жизни, напишу чуть ниже.

Итак. Если один из компьютеров захочет что-то отправить другому компьютеру, то ему потребуется знать только MAC-адрес компьютера, на который он отправляет. Если верхнему левому компьютеру с MAC-адресом 111 захочется что-то отправить нижнему правому компьютеру, то он без проблем отправит это, если будет знать, что у адресата MAC-адрес 444.

Эти 4 компьютера образуют простенькую локальную сеть и одну канальную среду. Отсюда и название уровня. Но для корректной работы узлов в сетях TCP/IP, недостаточно адресации на канальном уровне. Важна еще адресация на сетевом уровне, которая всем известна, как IP-адресация.

Теперь вспоминаем про IP-адреса. И присвоим их нашим компьютерам.



Адреса я присвоил символически, чтобы на базовом уровне понять, как они работают. Вот эти две адресации (канальная и сетевая) работают в тесной связке между собой и по отдельности работать не смогут. Сейчас объясню почему. Мы в повседневной жизни работаем только с IP-адресами или именами, о которых была целая глава в предыдущей статье. С MAC-адресами мы фактически не работаем. С ними работают сами компьютеры. Сейчас смоделирую ситуацию. Я сижу за верхним левым компьютером с IP: 1.1.1.1 и MAC: 111. Захотел я связаться с нижним правым компом и проверить живой он или нет. Я смогу связаться с ним, если буду знать его IP-адрес. MAC-адрес мне его не интересен. Я знаю, что IP-адрес у него 1.1.1.4. И решаю воспользоваться утилитой ping (утилита проверки доступности узла).

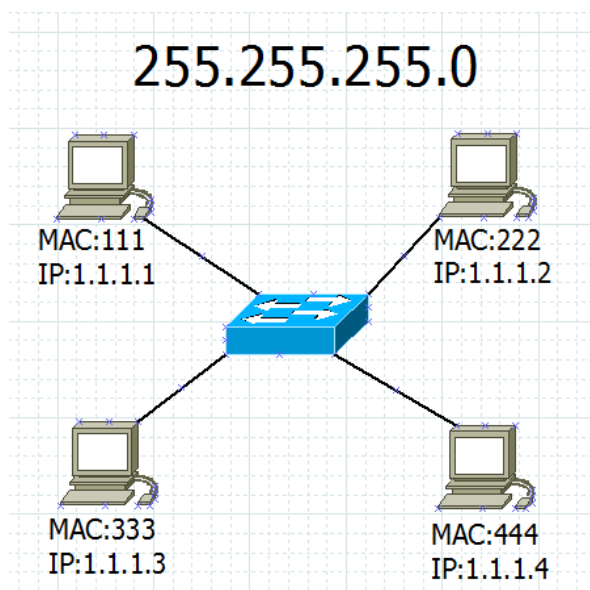
Теперь важная вещь. Компьютер понимает, что он не знает MAC-адрес компьютера, доступность которого надо проверить. Для того, чтобы узнать MAC-адрес по IP-адресу, придумали протокол ARP. Я о нем напишу подробно позже. Сейчас я хочу, чтобы вы поняли зависимости MAC-адреса и IP-адреса. Итак, он на всю сеть начинает кричать: «Кто такой 1.1.1.4». Этот крик услышат все участники сети и, если найдется тот узел, который имеет данный IP-адрес, он отзовется. У меня такой компьютер присутствует и в ответ на этот крик, он ответит: «1.1.1.4 — это я. Мой MAC — 444». Мой компьютер получит это сообщение и сможет продолжить то, что я ему сказал.

Дальше нужно научиться отличать одну подсеть от другой. И как компьютер понимает, в одной подсети находится он с другим узлом или в разных. Для этого на помощь приходит маска подсети. Масок бывает много и поначалу она кажется страшной, но уверяю вас, что это только сначала так кажется. Ей посвящена будет целая статья и там вы познаете все ее секреты. На данном же этапе, я покажу, как это работает.

Если вы когда-нибудь залезали в настройки сетевых адаптеров или прописывали статический адрес, который вам сообщал провайдер, то видели поле «маска подсети». Она записывается в том же формате, что и IP-адрес, основной шлюз и DNS. Это четыре октета разделенных между собой точками. Если вы этого никогда не видели, то можете открыть командную строку и набрать в ней `ipconfig`. Вы увидите, что-то похожее.

```
IPv4-адрес . . . . . : 192.168.0.101
Маска подсети . . . . . : 255.255.255.0
Основной шлюз . . . . . : 192.168.0.1
```

Это скриншот из командной строки моего ноутбука. Я сижу за домашней точкой доступа, у которой маска 255.255.255.0. Это, наверное, самая простая маска для объяснения и скорее всего у вас она точно такая же. В чем суть. Первые 3 октета (они фиксированы) показывают адрес сети, а 4 октет (он динамический) показывает адрес хоста. Иными словами, данная маска показывает, что нужно проверять первые 3 октета полностью, а четвертый может быть свободным от 0 — 255. Вообще это грубая формулировка. Потому, что с такой маской свободны будут от 1 до 254, где 0 уйдет под адрес сети, а 255 под широковещательный адрес. Но в любом случае это предел одной канальной среды. То есть, когда узлу надо отправить другому узлу сообщение, он берет его адрес и накладывает на него маску и если адрес сети (фиксированная часть) сходится с его адресом, то значит они в одной канальной среде. Объясняю на примере той же картинки.



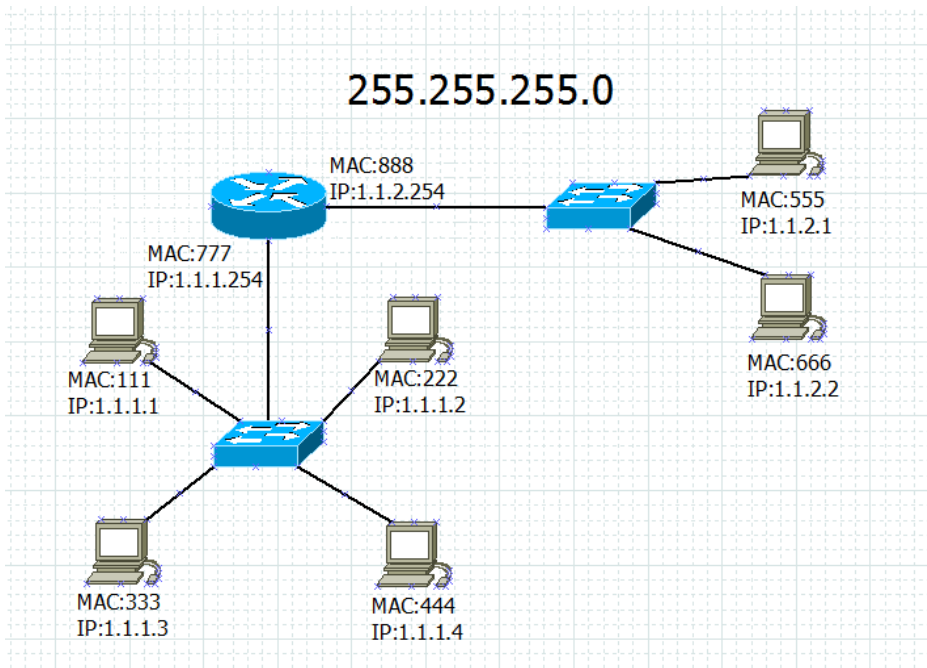
Сижу я за верхним левым компом и хочу отправить нижнему правому. Знаю я и IP-адрес его и MAC-адрес. Мне надо понять, в одной канальной среде мы или нет. Его адрес 1.1.1.4 и маска 255.255.255.0. Маска мне говорит, что 3 октета фиксированы и не должны меняться, а четвертый может быть любым в пределах от 1 до 254. Я накладываю маску на его адрес и на свой адрес и смотрю совпадения и различия.

1.1.1.1

1.1.1.4

Красным подсвечено та область, которая отвечает за сеть. Как видим, у 2-х хостов она одинакова. Значит, они находятся в одной подсети.

Модернизирую сеть и покажу вам ее немного иначе.



Добавилась круглое устройство. Оно называется роутер или маршрутизатор. Слово всем знакомое. Основная его роль — это соединение сетей и выбор лучшего маршрута, о чем будет в дальнейшем рассказано более подробно. И добавился, справа, один коммутатор, с которым соединены 2 компьютера. Маска для всех устройств не изменилась (255.255.255.0).

Посмотрите внимательно на адреса всех устройств. Можно заметить, что у новых узлов и старых отличается 3-ий октет. Давайте разберемся с этим. Я также сижу за компом с MAC:111 и IP:1.1.1.1. Хочу отправить информацию одному из новых узлов. Давайте пусть это будет верхний правый компьютер с MAC:555 и IP:1.1.2.1. Накладываю маску и смотрю.

1.1.1.1

1.1.2.1

И тут уже другая картина. 3-ие октеты различаются, а значит, узлы находятся в разных сетях (правильнее подсетях). Для разрешения таких ситуации, в настройках

каждой операционной системы есть основной шлюз. Его еще называют «шлюз последней надежды». Используется он, как раз, в том случае, когда нужно отправить информацию узлу, находящемуся в другой канальной среде. Для моего компа адрес шлюза — 1.1.1.254. А для компьютера, которому я отправляю данные 1.1.2.254. Логика работы здесь простая. Если узлу, который находился в одной канальной среде, информация доходила напрямую, то для узла находящегося в другой канальной среде, путь будет через маршрутизатор.

Мой комп знает, что адрес шлюза 1.1.1.254. Он крикнет на всю сеть: «1.1.1.254 отзовись». Это сообщение получают все участники канальной среды, но ответит только тот, кто сидит за этим адресом. То есть маршрутизатор. Он отправит ответ, и только после этого мой комп отошлет данные до адреса 1.1.2.254. Причем обратите внимание. На канальном уровне данные будут отправлены на MAC:777, а на сетевом, на IP:1.1.2.1. Это значит, что MAC-адрес передается только в своей канальной среде, а сетевой адрес не меняется на всем своем пути. Когда маршрутизатор получит инфу, он поймет, что на канальном уровне она предназначалась ему, но когда увидит IP-адрес, то поймет, что он промежуточное звено и передать надо в другую канальную среду. Его второй порт смотрит в нужную подсеть. Значит, ему все пришло верно. Но он не знает MAC-адрес адресата. Он начинает так же кричать на всю сеть: «Кто такой 1.1.2.1?». И комп с MAC-адресом 555 отвечает ему. Думаю, что логика работы понятна.

На протяжении двух предыдущих статей и текущей, много раз упоминался термин **«MAC-адрес»**. Давайте разберем, что это такое.

Как я уже говорил, это уникальный идентификатор сетевого устройства. Он уникален и не должен нигде повторяться. Состоит он из 48 бит, из которых первые 24 бита — это уникальный идентификатор организации, который присваивается комитетом IEEE (Институт инженеров электротехники и электроники). А вторые 24 бита назначаются производителем оборудования. Выглядит это так.



Записывают его по-разному. Например:



- 1) 00-50-56-C0-00-08
- 2) 00:50:56:C0:00:08
- 3) 0050.56C0.0008

Как видите, один и тот же адрес можно записать разными способами. Еще обычно его не разделяют, а записывают слитно. Главное знать, что MAC-адрес всегда состоит из 48 бит и состоит из 12 букв и/или цифр. Посмотреть его можно разными способами. Например, в ОС Windows, открыв командную строку, ввести `ipconfig /all`. Многие производители еще записывают его на коробке или на обратной стороне корпуса устройства.

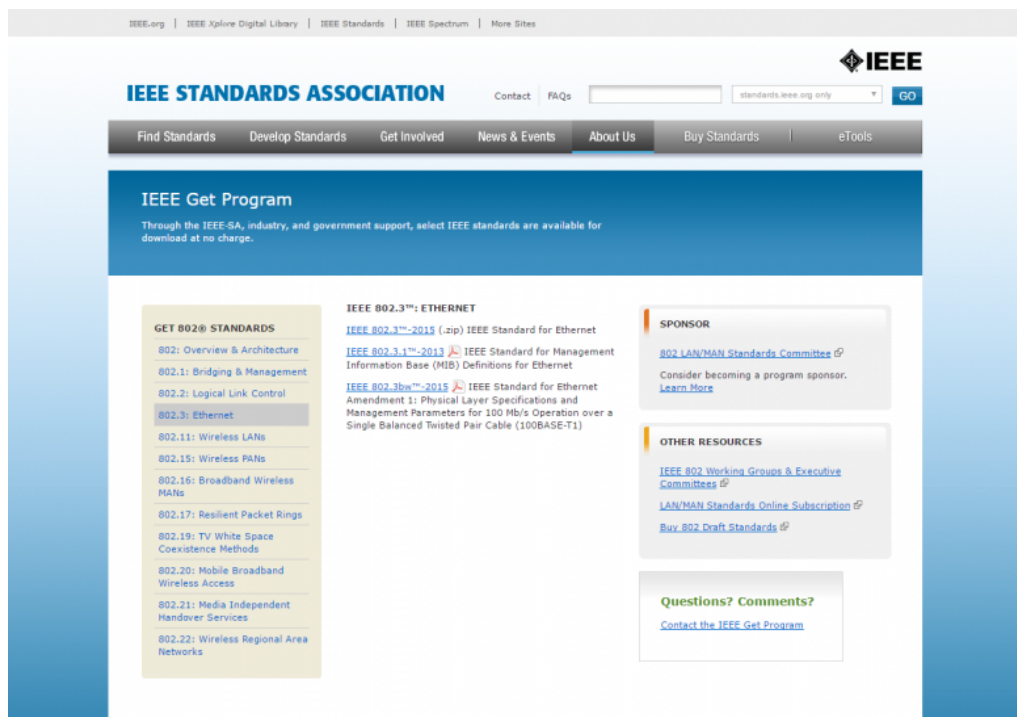


Так что можете посмотреть на свою Wi-Fi точку доступа и увидеть похожую запись. В самом начале я показал MAC-адреса 3-х значными цифрами, что не правда. В том контексте я их употреблял только для простоты объяснения, чтобы не путать вас длинными непонятными записями. Чуть ниже, когда речь дойдет до практики, вы увидите их такими, какие они на самом деле.

Раз мы разобрали адрес на канальном уровне, пришло время разобрать протокол, работающий на данном уровне. Самый популярный на сегодняшний момент протокол, используемый в локальных сетях — это **Ethernet**. IEEE описала его стандартом 802.3. Так что, все версии, которые начинаются с 802.3, относятся именно к нему. Например, 802.3z — это GigabitEthernet через волоконно-оптический кабель; 1 Гбит/с, а 802.3af — это электропитание через Ethernet (PoE — Power over Ethernet).

Кстати, я не сказал об организации **IEEE (англ. Institute of Electrical and Electronics Engineers)**. Эта организация разрабатывает стандарты ко всему, что относится к радиоэлектронике и электротехнике. На их сайте можно найти много документации по существующим технологиям. Вот, что они выдают по запросу «Ethernet»





Давайте разберем, из чего он состоит. Так как сам протокол старый (придуман в 1973 году), то он много раз модернизировался и менял свой формат. В Интернете можно найти все его варианты, но я приведу тот, который приводила Cisco, когда я учился.

Ethernet-кадр					
8 байт	6 байт	6 байт	2 байта	46-1500 байт	4 байта
Преамбула	MAC-адрес получателя	MAC-адрес источника	Тип(длина)	SNAP/LLC и данные	FCS(Frame Check Sequence)-контроль суммы

- 1) **Преамбула.** Поле, используемое для указания начала кадра. То есть, чтобы приемник смог понять, где начало нового кадра. Раньше, когда использовалась топология с общей шиной и были коллизии, преамбула помогала предотвращать коллизии.
- 2) **MAC-адрес получателя.** Поле, куда записывается адрес получателя.
- 3) **MAC-адрес отправителя.** Соответственно сюда записывается адрес отправителя.
- 4) **Тип (длина).** В этом поле указывается вышестоящий протокол. Для IPv4 — это 0x0800, для ARP — 0x0806, а для IPv6 — 0x86DD. В некоторых случаях сюда может записываться длина поля данных кадра (следующее поле в заголовке).
- 5) **Поле SNAP/LLC + данные.** В этом поле содержатся данные, полученные с высших уровней (или полезные данные).

6) **FCS (от англ. Frame Check Sequence — контрольная сумма кадра)**. Поле в котором подсчитана чек-сумма. По ней получатель понимает, побился кадр или нет.

По ходу написания данной статьи и последующих, будут затронуты и другие протоколы канального уровня. Пока что вышеописанного хватает для понимания его работы.

Переходим к сетевому уровню, и тут нас встречает нашумевший протокол IP. Раз мы говорим о сетевом уровне, то значит протокол, работающий на этом уровне, должен каким-то образом уметь передавать данные из одной канальной среды в другую. Но для начала посмотрим, что это за протокол и из чего он состоит.

**IP (от англ. Internet Protocol)**. Протокол семейства TCP/IP, который был разработан в 80-х годах. Как я говорил ранее, используется для объединения отдельных компьютерных сетей между собой. Также важной его особенностью является адресация, которую называют

**IP-адрес**. На текущий момент существуют 2 версии протокола: IPv4 и IPv6. Пару слов о них:

**1) IPv4.** Использует 32-битные адреса, которые записываются в формате четырёх десятичных чисел (от 0 до 255), разделённых точками. Например, адрес 192.168.0.4. Каждое число разделённое точками называют октетом. Это самая популярная версия на сегодняшний день.

**2) IPv6.** Использует 128-битные адреса, которые записываются в формате восьми четырехзначных шестнадцатеричных чисел (от 0 до F). Например, адрес 2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d. Каждое число разделённое точками называют хекстетом. На заре всеобщей компьютеризации появилась проблема. Стали заканчиваться IP-адреса и нужен был новый протокол, который смог бы обеспечить больше адресов. Так и появился в 1996 году протокол IPv6. Но благодаря технологии NAT, которая будет рассмотрена позже, была частично решена проблема нехватки адресов, и, в связи с этим, внедрение IPv6 затянулось по сегодняшний день.

Думаю понятно, что обе версии предназначены для одних и тех же целей. В этой статье будет разобран протокол IPv4. Про IPv6 будет написана отдельная статья.

Итак, протокол IP работает с блоком информации, который принято называть IP-пакет. Рассмотрим его структуру.

Версия	IHL	Тип обслуживания	Длина пакета	
Идентификатор			Флаги	Смещение фрагмента
Время жизни (TTL)	Протокол		Контрольная сумма заголовка	
IP-адрес отправителя				
IP-адрес получателя				
Опции			Смещение	
Данные				

**1) Версия.** Протокол IPv4 или IPv6.

**2) IHL (от англ. Internet Header Length — размер заголовка).** Так как многие из показанных на картинке полей не фиксированы, то это поле считает размер заголовка.

**3) Тип обслуживания.** Обслуживает размер очередей QoS (Quality of Service — качество обслуживания). Делает он это при помощи байта, который указывает на определенный набор критериев (требование ко времени задержки, пропускной способности, надежности и т.д.)

**4) Длина пакета.** Размер пакета. Если **IHL** отвечает только за размер полей в заголовке (заголовком являются все поля на картинке, кроме поля данных), то длина пакета отвечает за весь пакет в целом, включая пользовательские данные.

**5) Время жизни (TTL- Time To Live).** Поле, используемое для предотвращения циклического пути пакета. При прохождении через маршрутизатор, значение уменьшается на единицу, и когда достигает нуля, пакет отбрасывается.

**6) Протокол.** Для какого вышестоящего протокола предназначается данный пакет (TCP, UDP).

**7) Контрольная сумма заголовка.** Здесь считается целостность полей заголовка. Не данных! Данные проверяются соответствующим полем на канальном уровне.

**8) Опции.** Это поле используется для расширения стандартного заголовка IP. Используется в привычных сетях редко. Сюда записываются данные для какого-нибудь специфического оборудования, которое читает это поле. Например, система управления дверными замками (где идет общение с контроллером), технология умного дома, интернет-вещи и так далее. Привычные сетевые устройства, как роутеры и коммутаторы, будут игнорировать это поле.

**9) Смещение.** Указывает, какому месту принадлежит фрагмент в оригинальном IP. Это значение всегда кратно восьми байтам.

**10) Данные.** Здесь как раз содержатся данные, полученные с вышестоящих уровней. Чуть выше я показал, что в Ethernet-кадре тоже есть поле данных. И в его поле данных будет включен данный IP-пакет. Важно помнить, что максимальный размер Ethernet-кадра равен 1500 байт, а вот размер IP пакета может быть 20 Кбайт. Соответственно весь пакет не влезет в поле данных Ethernet-кадра. Поэтому пакет делят и отправляют частями. И вот для этого используются 3 поля ниже.

**11) Идентификатор.** Это 4-х байтовое число, которое показывает, что все части разделенного пакета одно единое целое.

**12) Флаги.** Указывает, что это не единый, а фрагментированный пакет.

**13) Смещение фрагмента.** Сдвиг относительно первого фрагмента. То есть это нумерация, которая поможет собрать IP-пакет воедино.

**14) IP-адрес отправителя и IP-адрес получателя.** Соответственно эти 2 поля указывают от кого и для кого пакет.

Вот так выглядит IP-пакет. Конечно, для новичков значения многих полей покажутся не совсем понятными, но в дальнейшем это уложится в голову. Например: поле «Время жизни (TTL)». Его работа станет ясна, когда поймете, как работает маршрутизация. Могу дать совет, который я сам применяю. Если видите непонятный термин, выпишите его отдельно и, при наличии свободного времени, попробуйте разобраться. Если никак не лезет в голову, то отложите и вернитесь к его изучению чуть позже. Главное не забрасывать и в конечном итоге все таки добить.

Остался последний уровень из стека TCP/IP. Это **транспортный уровень**. Пару слов о нем. Он предназначен для доставки данных определенному приложению, которое он определяет по номеру порта. В зависимости от протокола, он выполняет разные задачи. Например, фрагментация файлов, контроль доставки, мультиплексирование потоками данных и управление ими. 2 самых известных протокола транспортного уровня — это UDP и TCP. Поговорим о каждом из них подробнее, и начну с UDP, в силу его простоты. Ну и по традиции показываю, из чего он состоит.

Порт источника (16 бит)	Порт назначения (16 бит)
Длина UDP (16 бит)	Контрольная сумма UDP
Данные	

**1) Порт источника.** Порт, используемый клиентом или сервером для идентификации службы. На этот порт, при необходимости, будет посылаться ответ.

**2) Порт назначения.** Здесь указывается порт, который будет являться адресатом. Например, если клиент запрашивает страницу сайта, то порт назначения, по умолчанию, будет 80-ый (протокол HTTP).

**3) Длина UDP.** Длина заголовка UDP. Размер варьируется от 8 до 65535 байт.

**4) Контрольная сумма UDP.** Проверка целостности. Если нарушена, то просто отбрасывает без запроса о повторной отправки.

**5) Данные.** Здесь упакованы данные с верхнего уровня. Например, когда веб-сервер отвечает на запрос клиента и отправляет веб-страницу, то она будет лежать в этом поле.

Как видите, у него не так много полей. Его задачи — это нумерация портов и проверять побился кадр или нет. Протокол простой и не требовательный к ресурсам. Однако он не может обеспечивать контроль доставки и повторно запрашивать побитые куски информации. Из известных сервисов, которые работают с этим протоколом — это DHCP, TFTP. Они рассматривались в предыдущей статье, когда разбирались протоколы верхнего уровня.

Переходим к более сложному протоколу. Встречаем протокол TCP. Смотрим, из чего состоит, и пробегаем по каждому полю.

Порт источника		Порт назначения	
Порядковый номер (Sequence Number)			
Номер подтверждения (Acknowledgment Number)			
Длина заголовка	Зарезервирован	Флаги	Размер окна
Контрольная сумма TCP		Указатель важности	
Опции			
Данные			

**1) Порт источника и порт назначения.** Выполняют те же роли, что и в UDP, а именно нумерация портов.

**2) Порядковый номер.** Номер, который используется для того, чтобы на другой стороне было понятно какой этот сегмент по счету.

**3) Номер подтверждения.** Это поле используется, когда ожидается доставка или подтверждается доставка. Для этого используется параметр ACK.

**4) Длина заголовка.** Используется для того, чтобы понять какой размер у TCP-заголовка (это все поля представленные на картинке выше, кроме поля данных), а какой у данных.

**5) Зарезервированный флаг.** Значение этого поля должно устанавливаться в ноль. Оно зарезервировано под специальные нужды. Например, чтобы сообщить о перегрузках в сети.

**6) Флаги.** В это поле устанавливаются специальные биты для установления или разрыва сессии.

**7) Размер окна.** Поле, указывающее, на сколько сегментов требовать подтверждения. Наверное, каждый из вас наблюдал такую картину. Вы скачиваете какой-то файл и видите скорость и время скачивания. И тут сначала он показывает, что осталось 30 минут, а через 2-3 секунды уже 20 минут. Еще спустя секунд 5, показывает 10 минут и так далее. Это и есть размер окна. Сначала размер окна устанавливается таким образом, чтобы получать больше подтверждений о каждом отправленном сегменте. Далее все идет хорошо и сеть не сбоит. Размер окна меняется и передается больше сегментов и, соответственно, требуя меньше отчетов о доставке. Таким образом, скачивание выполняется быстрее. Как только сеть даст краткий сбой, и какой-то сегмент придет побитым, то размер опять изменится и потребуются больше отчетов о доставке. В этом суть данного поля.

**8) Контрольная сумма TCP.** Проверка целостности TCP-сегмента.

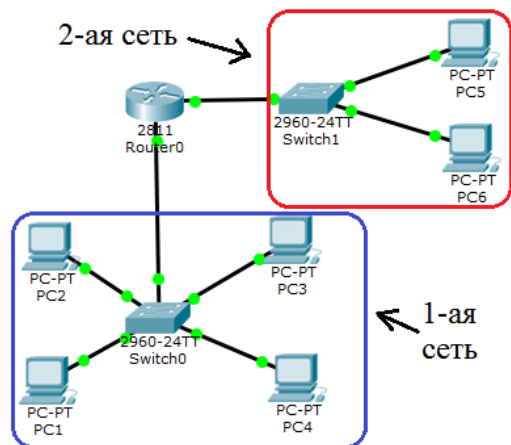
**9) Указатель важности.** Это смещение последнего октета важных данных относительно SEQ для пакетов с установленным флагом URG. В жизни применяется, когда необходимо осуществить контроль потока или состояния протокола верхнего уровня со стороны передающего агента (например, если принимающий агент может косвенно сигнализировать передающему, что не справляется с потоком данных).

**10) Опции.** Используется для каких-нибудь расширенных или дополнительных параметров. Например, для параметра timestamp, который является своеобразной меткой, показывающей время произошедшего события.

**11) Данные.** Практически тоже самое, что и в протоколе UDP. Здесь инкапсулированы данные с вышестоящего уровня.

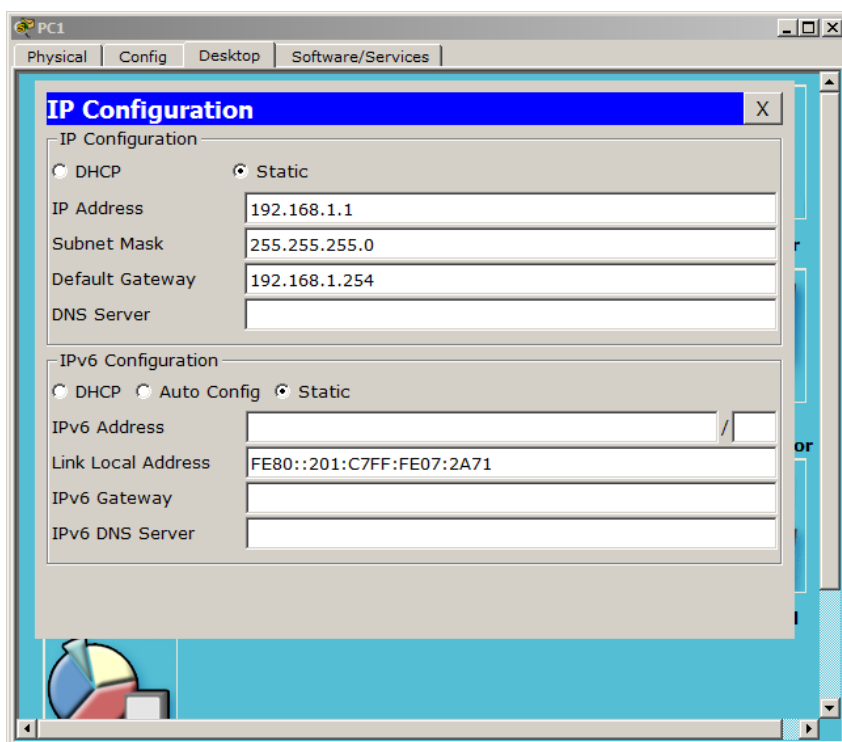
Увидели мы строение протокола TCP и вместе с этим закончили разговор о транспортном уровне. Получилась такая краткая теория по протоколам, работающих на нижних уровнях. Старался объяснить как можно проще. Сейчас еще все это дело опробуем на практике и добьем пару вопросов.

Я открываю CPT и соберу схему, аналогичную одному из рисунков выше.



Здесь наблюдаем первую сеть, состоящую из 4-х компьютеров и коммутатора, который объединяет эти компьютеры. И 2-ую сеть, состоящую из двух компьютеров и коммутатора. Объединяет эти 2 сети маршрутизатор. Перейдем к настройке устройств и после смоделируем ту ситуацию, которую мы рассматривали в самом начале на картинке.

Открываю компьютер PC1 и пропишу сетевые настройки.



Не стал я мудрить с адресом и воспользовался самым простым, который постоянно на глазах:

1) IP-адрес — 192.168.1.1

2) Маска подсети — 255.255.255.0. Эту маску мы рассматривали выше. Напомню, что адрес сети у других хостов в той же локальной сети, должен быть 192.168.1, а адрес хоста может быть от 1 до 254.



3) Основной шлюз — 192.168.1.254. Это адрес маршрутизатора, на который будут отправляться данные для хостов другой локальной сети.

Чтобы не было много однотипных картинок, я не буду приводить скриншоты остальных 3-х компьютеров, а только приведу их настройки.

#### **PC2:**

- 1) IP-адрес — 192.168.1.2
- 2) Маска подсети — 255.255.255.0.
- 3) Основной шлюз — 192.168.1.254.

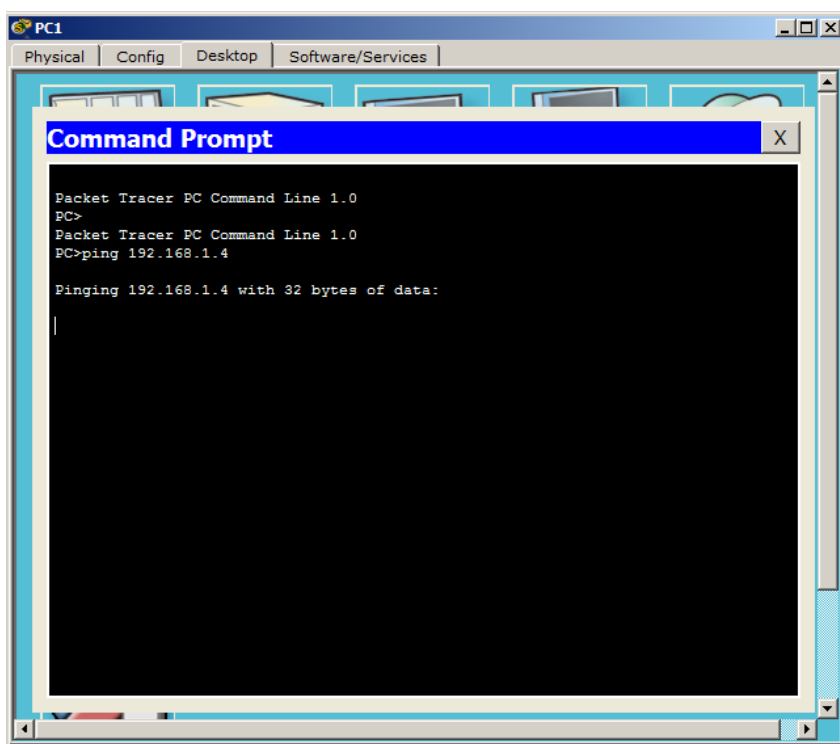
#### **PC3:**

- 1) IP-адрес — 192.168.1.3
- 2) Маска подсети — 255.255.255.0.
- 3) Основной шлюз — 192.168.1.254.

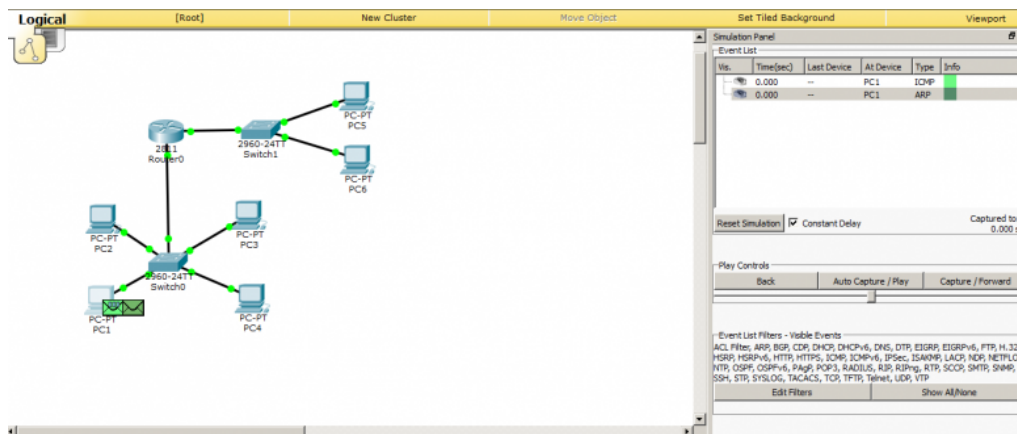
#### **PC4:**

- 1) IP-адрес — 192.168.1.4
- 2) Маска подсети — 255.255.255.0.
- 3) Основной шлюз — 192.168.1.254.

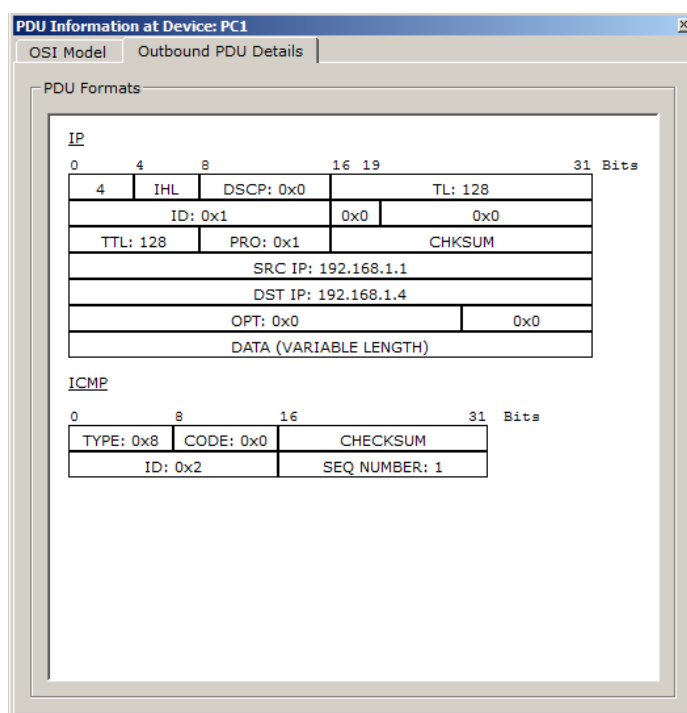
На данной настройке пока остановимся и посмотрим, как работает наша локальная сеть. Перевожу CPT в режим симуляции. Допустим, я сижу за компьютером PC1, и требуется проверить доступность PC4 командой ping. Открываю командную строку на PC1.



Как только нажимаю ENTER, на схеме появляются 2 конверта.



Один из них ICMP, с которым работает сама команда ping. Сразу открываю его и смотрю.



Вижу данные IP и ICMP. Тут нет ничего интересного, за исключением нескольких полей. А именно, цифра 4 в верхнем левом углу данных IP, которая говорит о том, что используется протокол IPv4. И 2 поля с IP-адресом источника и назначения (SRC:192.168.1.1 и DST:192.168.1.4).

Но тут ping сталкивается с проблемой. Он не знает MAC-адрес получателя. То есть, адрес канального уровня. Для этого он использует протокол ARP, который сможет опросить участников сети и узнать MAC-адрес. Мы про него вскользь говорили в предыдущей статье. Давайте поговорим о нем подробнее. Не буду изменять традиции. Картинку в студию!

Тип протокола канального уровня		Тип протокола сетевого уровня
Длина физического адреса в байтах	Длина логич. адреса в байтах	Код операции
Физический адрес отправителя		
Логический адрес отправителя		
Физический адрес получателя		
Логический адрес получателя		

**1) Тип протокола канального уровня (Hardware type).** Думаю понятно из названия, что тут указывается тип канального уровня. Мы пока рассматривали только Ethernet. Его обозначение в этом поле — 0x0001.

**2) Тип протокола сетевого уровня (Protocol type).** Тут, аналогично, указывается тип сетевого уровня. Код IPv4 — 0x0800.

**3) Длина физического адреса в байтах (Hardware length).** Если это MAC-адрес, то размер будет 6 байт (или 48 бит).

**4) Длина логического адреса в байтах (Protocol length).** Если это IPv4-адрес, то размер будет 4 байта (или 32 бита).

**5) Код операции (Operation).** Код операции отправителя. Если это запрос, то код 0001. В случае ответа — 0002.

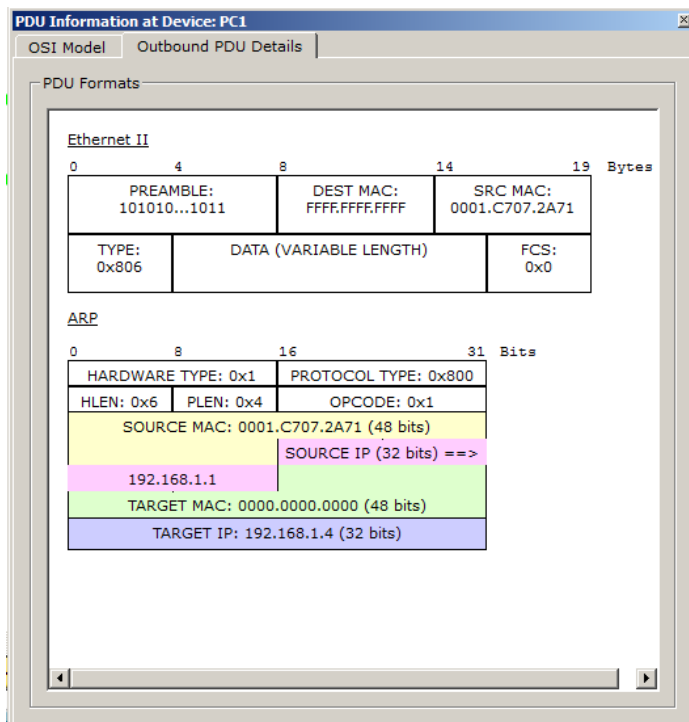
**6) Физический адрес отправителя (Sender hardware address).** MAC-адрес отправителя.

**7) Логический адрес отправителя (Sender protocol address).** IP-адрес отправителя.

**8) Физический адрес получателя (Target hardware address).** MAC-адрес получателя. Если это запрос, то, как правило, адрес неизвестен и это поле остается пустым.

**9) Логический адрес получателя (Target protocol address).** IP-адрес получателя.

Теперь, когда мы знаем, из чего он состоит, можно посмотреть на его работу в СРТ. Кликаю по второму конверту и наблюдаю следующую картину.



И вот протокол ARP во всей красе. На 2-ом уровне работает протокол Ethernet. Остановимся и посмотрим на его поля.

**1) Преамбула** — здесь битовая последовательность, которая говорит о начале кадра.

**2) Далее идет MAC-адрес источника и получателя.** В адресе источника записан MAC-адрес компьютера, который является инициатором, а в адресе получателя записан широковещательный адрес FF-FF-FF-FF-FF-FF (то есть для всех узлов в канальной среде).

**3) Тип** — здесь указан вышестоящий протокол. Код 0x806 означает, что выше стоит ARP. Я, если честно, не могу точно сказать, на каком уровне он работает. В разных источниках указано по-разному. Кто то говорит, что на 2-ом уровне OSI, а кто-то говорит, что на 3-ем. Я считаю, что он между ними работает. Так как тут есть адреса, присущие каждому из уровней.

Про данные и чек-сумму много говорить не буду. Данные здесь никак не указаны, а чек-сумма нулевая.

Поднимаемся чуть повыше и здесь протокол **ARP**.

**1) Hardware Type** — код канального уровня. СРТ убрала лишние нули и вставила 0x1 (тоже, что и 0x0001). Это Ethernet.

**2) Protocol Type** — код сетевого уровня. 0x800 — это IPv4.

**3) HLEN** — длина физического адреса. 0x6 означает 6 байт. Все верно (MAC-адрес занимает 6 байт).

**4) PLEN** — длина сетевого адреса. 0x4 означает 4 байта (IP-адрес занимает 4 байта).

5) **OPCODE** — код операции. 0x1 означает, что это запрос.

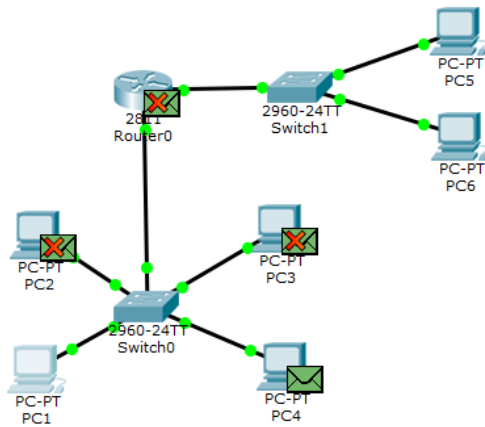
6) **Source Mac** — здесь MAC-адрес отправителя. Можно сравнить его с адресом в поле протокола Ethernet и убедиться в правильности.

7) **Source IP** — IP-адрес отправителя.

8) **Target MAC** — так как это запрос и канальный адрес не известен, то он пустой. CPT показала его нулями, что равносильно.

9) **Target IP** — IP-адрес получателя. Как раз тот адрес, который пингуем.

Посмотрим, что будет происходить дальше в сети.



Протокол ARP опрашивает все хосты в локальной сети и только один отвечает на этот запрос. Это PC4. Посмотрим, чем он ответит.

The screenshot shows the PDU Information at Device: Switch0 window. The packet is an ARP response. The Ethernet II header shows the source MAC as 000B.BE40.B9AA (PC4) and the destination MAC as 0001.C707.2A71 (PC1). The ARP section shows the hardware type as 0x1, protocol type as 0x800, and the opcode as 0x2. The source MAC is 000B.BE40.B9AA (48 bits) and the source IP is 192.168.1.4 (32 bits). The target MAC is 0001.C707.2A71 (48 bits) and the target IP is 192.168.1.1 (32 bits).

Ethernet II			
0	4	8	14
PREAMBLE: 101010...1011		DEST MAC: 0001.C707.2A71	
TYPE: 0x806		DATA (VARIABLE LENGTH)	
FCS: 0x0			

ARP			
0	8	16	31
HARDWARE TYPE: 0x1		PROTOCOL TYPE: 0x800	
HLEN: 0x6		PLEN: 0x4	
SOURCE MAC: 000B.BE40.B9AA (48 bits)		OPCODE: 0x2	
192.168.1.4		SOURCE IP (32 bits) ==>	
TARGET MAC: 0001.C707.2A71 (48 bits)			
192.168.1.1		TARGET IP: 192.168.1.1 (32 bits)	

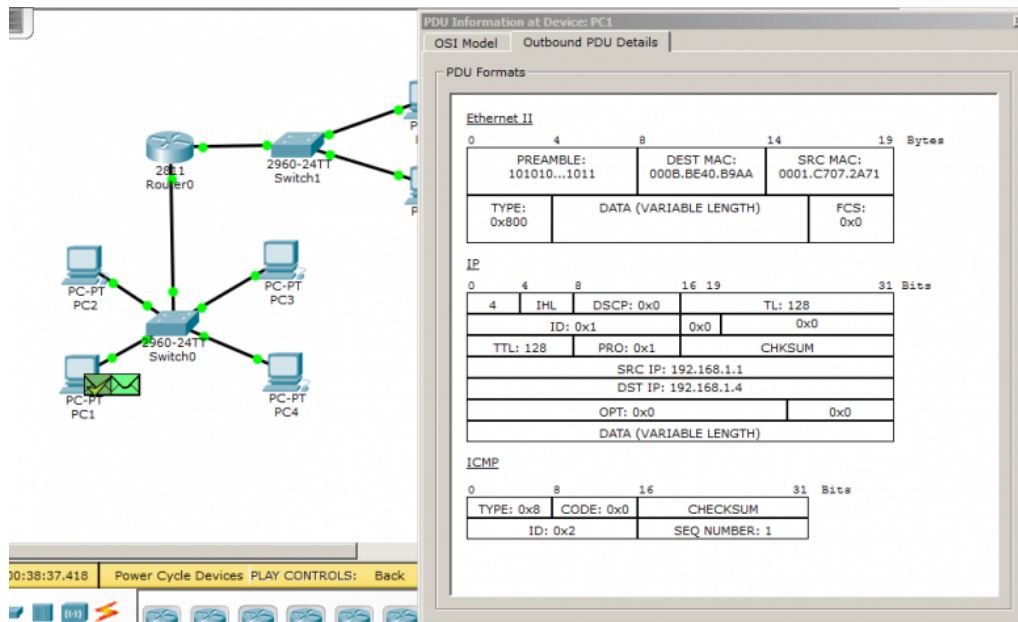
Вот он выплевывает что-то на коммутатор. Открываю его и вижу некоторые изменения, а именно:

1) В поле источника протокола Ethernet теперь записан MAC-адрес PC4, а в поле назначения MAC-адрес инициатора, то есть PC1.

2) В поле OPCODE теперь значение 0x2, то есть ответ.

3) Поменялись поля логических и физических адресов в протоколе ARP. Source MAC и Destination MAC аналогичны тем, что в протоколе Ethernet. В поле Source IP — адрес 192.168.1.4 (PC4), а в поле Destination IP — адрес 192.168.1.1 (PC1).

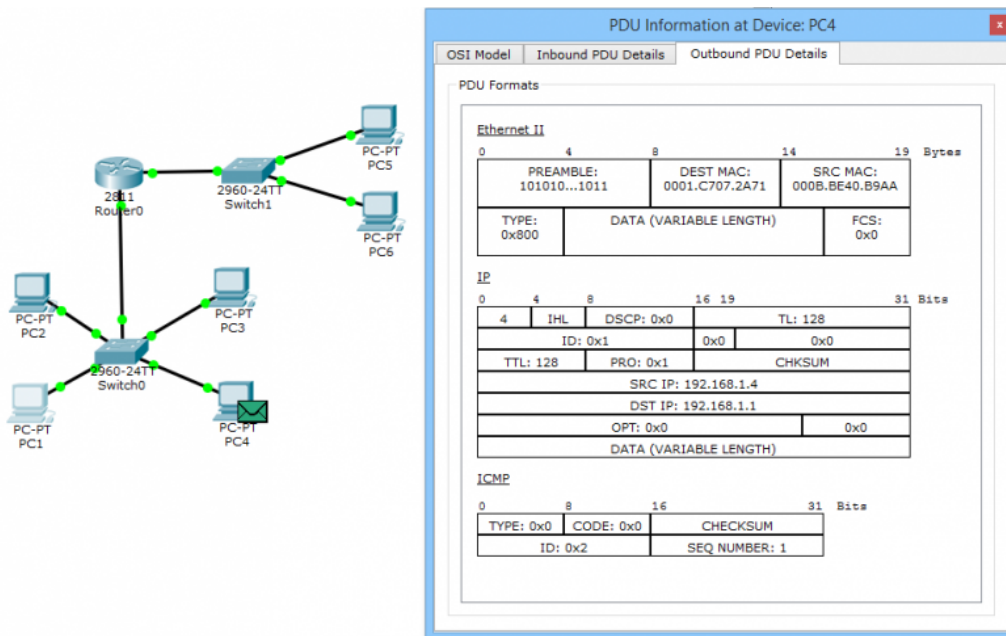
Как только эта информация достигнет PC1, он сразу формирует ICMP-сообщение, то есть ping.



Открываю его и смотрю. Это блок данных, состоящий из работы 3-х протоколов: Ethernet, IP и Ping.

- 1) В Ethernet протоколе ничего нового, а именно MAC-адрес отправителя — PC1, MAC-адрес получателя — PC4, а в поле Type — 0x800 (протокол IPv4)
- 2) В IP протоколе, в поле Версия — 4, что означает протокол IPv4. IP-адрес отправителя — PC1, а IP-адрес получателя — PC4.
- 3) В ICMP протоколе, в поле Type — код 0x8 (эхо-запрос).

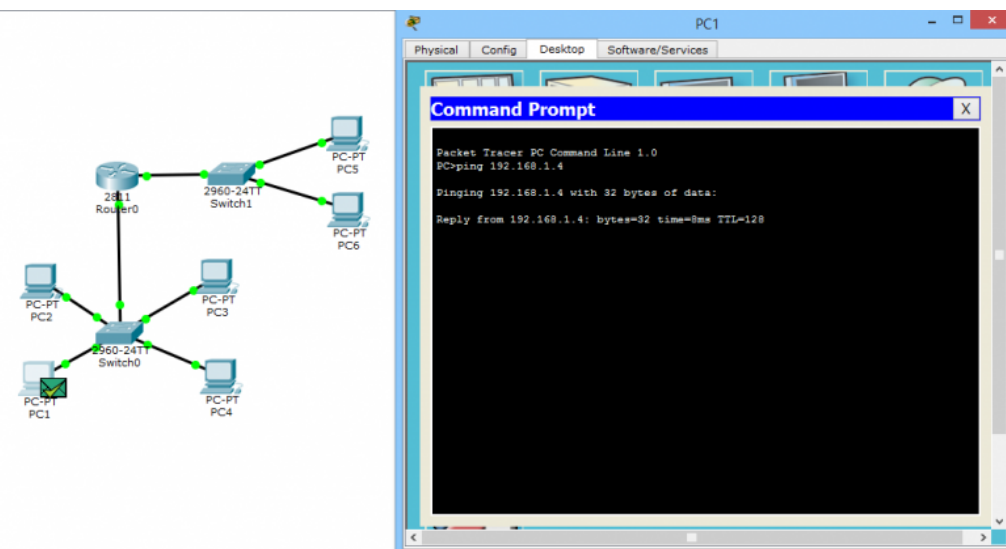
Посылает он эхо-запрос, а я смотрю, чем ответит PC4.



Перекошил у меня СРТ, и пришлось перезапустить его. Только теперь ICMP конверт не светло-зеленового цвета, а смесь зеленого и синего. Но это без разницы. Это одни и те же данные.

Ну что же, смотрю, чем ответил PC4. Поля источника и назначения в протоколах Ethernet и IP поменялись местами. А в поле Type протокола ICMP изменилось значения с 0x8 на 0x0 (означает эхо-ответ).

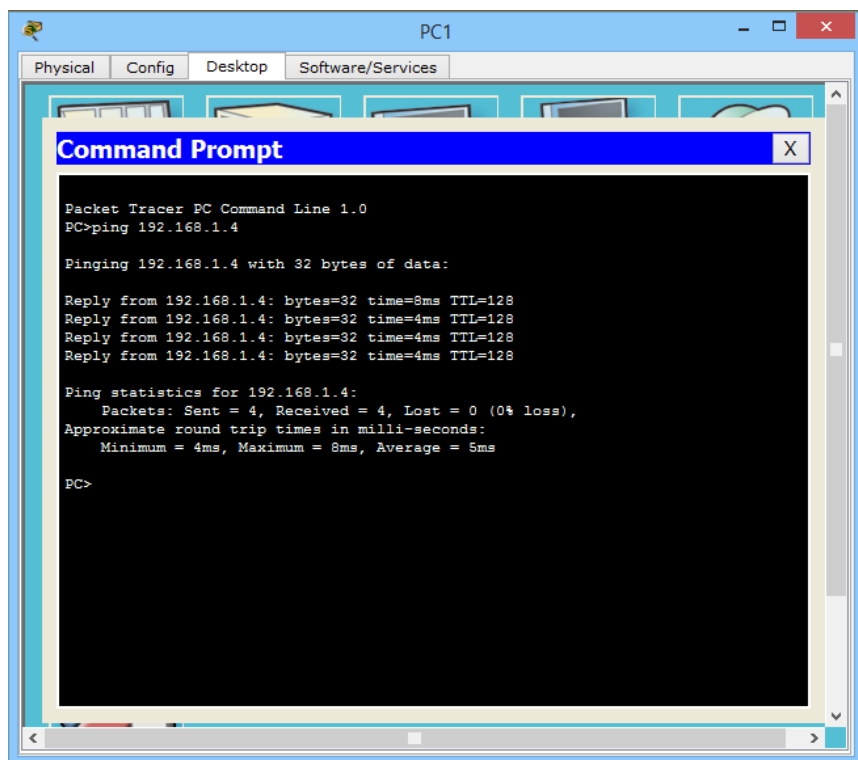
Судя по логике, как только этот ответ достигнет PC1, в консоли PC1 должна появиться запись. Давайте проверим.



И действительно. Появилась запись о доступности PC4, размер данных (32 байта), задержка по времени (8 мс) и TTL или время жизни (128). TTL показывает, сколько маршрутизаторов преодолел пакет. У меня пакет гулял в пределах локальной сети, поэтому данное поле не изменилось.



По умолчанию пинг отправляет 4 запроса. Следовательно, PC1 сформирует еще 3 аналогичных ICMP. Показывать путь каждого пакета я не буду, а приведу финальный вывод консоли на PC1.



```
PC1
Physical Config Desktop Software/Services
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.4

Pinging 192.168.1.4 with 32 bytes of data:

Reply from 192.168.1.4: bytes=32 time=8ms TTL=128
Reply from 192.168.1.4: bytes=32 time=4ms TTL=128
Reply from 192.168.1.4: bytes=32 time=4ms TTL=128
Reply from 192.168.1.4: bytes=32 time=4ms TTL=128

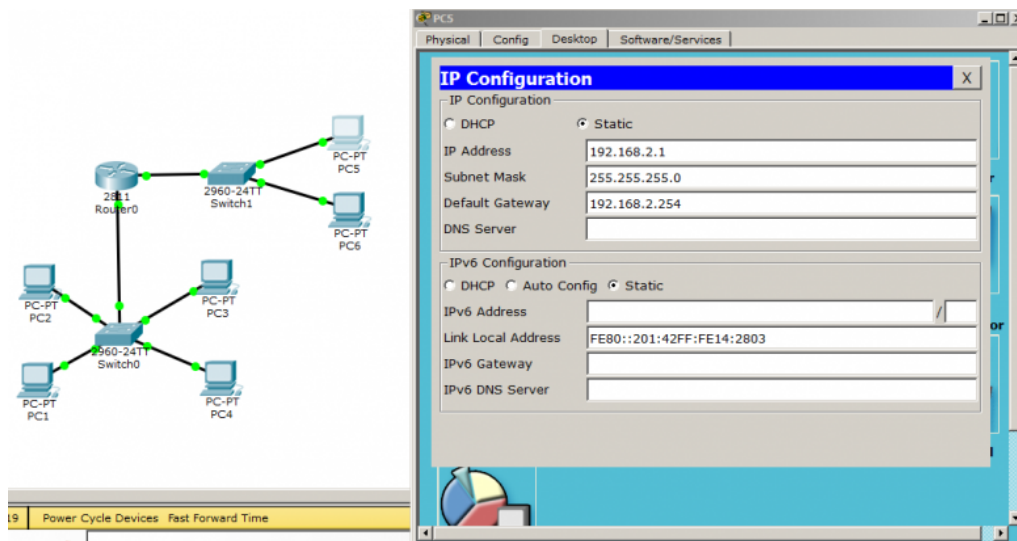
Ping statistics for 192.168.1.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 4ms, Maximum = 8ms, Average = 5ms

PC>
```

И как видите, действительно 4 ответа. Заметьте, что первый пришел с задержкой в 8 мс, а 3 последних в 4 мс. Это связано с работой протокола ARP, так как сначала PC1 не знал MAC-адрес PC4 и ждал, когда ему сообщат. Хотя в СРТ встречается ситуация, что в режиме реального времени, первый пакет вообще теряется. Особенно часто это проявляется, когда проверяется доступность хоста, находящегося в другой канальной среде.

Увидели мы, как работает передача данных в одной канальной среде. Теперь посмотрим, что произойдет, если хосты окажутся в разных канальных средах или подсетях. Напомню, что сеть настроена не до конца. А именно, нужно настроить маршрутизатор и вторую подсеть. Чем сейчас и займемся.

Открываю я компьютер с именем PC5 и пропишу сетевые настройки.



Заметьте, что сетевая адресация в первой канальной среде, была 192.168.1.X, а во 2-ой 192.168.2.X. При маске 255.255.255.0, это означает, что первые 3 октета фиксированы, а 4-ый октет в пределах от 1 до 254. И так как у нас 3-ие октеты различаются, то это разные канальные среды.

Привожу настройки PC6:

- 1) IP-адрес — 192.168.2.2
- 2) Маска подсети — 255.255.255.0
- 3) Основной шлюз — 192.168.2.254

Хосты во 2-ой канальной среде настроены и прекрасно работают. Для того, чтобы они смогли общаться с хостами из 1-ой канальной, нужно настроить маршрутизатор, который соединяет эти среды. Маршрутизатор настраивается через CLI (то есть в консольном виде) и проще будет приводить сюда не скриншоты, а команды.

- 1) Router>enable — *переход в привилегированный режим*
- 2) Router#configure terminal — *переход в режим глобальной конфигурации*
- 3) Router(config)#interface fastEthernet 0/0 — *переходим к настройке порта 0/0, который смотрит на первую канальную среду*
- 4) Router(config-if)#ip address 192.168.1.254 255.255.255.0 — *вешаем на этот порт IP-адрес. Так как этот порт будет являться основным шлюзом для 1-ой канальной среды, то указываем ему тот IP, который прописали хостам*
- 5) Router(config-if)#no shutdown — *включаем этот интерфейс. По умолчанию все порты на цисковских маршрутизаторах выключены*
- 6) Router(config-if)#exit — *выходим из режима настройки fastEthernet 0/0*
- 7) Router(config)#interface fastEthernet 0/1 — *переходим к настройке порта 0/1, который смотрит на вторую канальную среду*
- 8) Router(config-if)#ip address 192.168.2.254 255.255.255.0 — *вешаем сюда адрес, который будет являться основным шлюзом для хостов во 2-ой канальной среде*
- 9) Router(config-if)#no shutdown — *аналогично включаем его*
- 10) Router(config-if)#end — *пишем команду, которая отбросит в*

*привилегированный режим*

11) Router#copy running-config startup-config — *сохраняем настройки в памяти маршрутизатора*

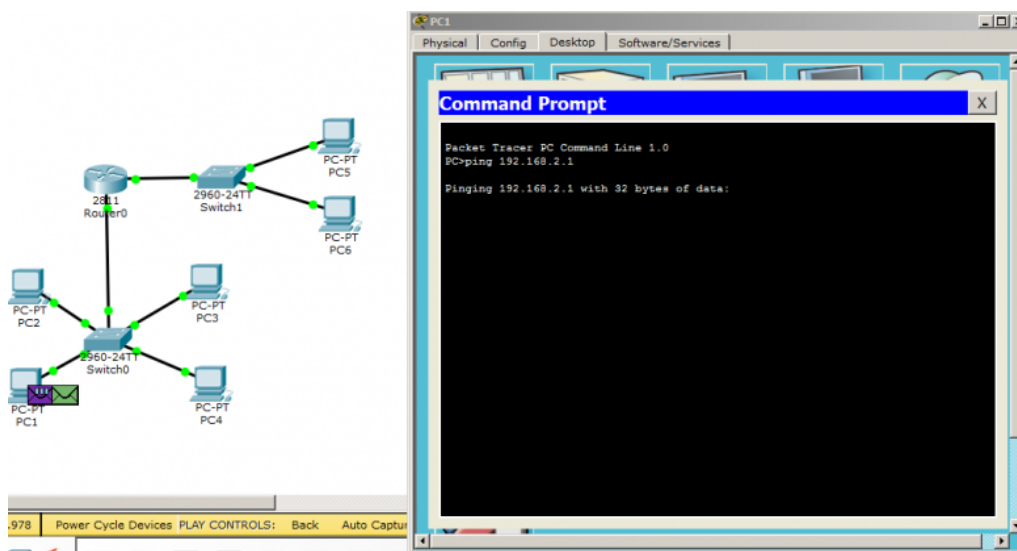
На этом этапе настройка маршрутизатора окончена. Немного забегу вперед и покажу полезную команду «show ip route». Она показывает все известные маршрутизатору сети и маршрут до них.

```
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.1.0/24 is directly connected, FastEthernet0/0
C    192.168.2.0/24 is directly connected, FastEthernet0/1
```

Исходя из этой таблицы, можно удостовериться, что он знает и про 1-ую канальную среду, и про 2-ую. Отлично. Осталось дело за малым — это проверить доступность PC5 из PC1. Пробую. Переключаю CPT в режим симуляции. Открываю командную строку и пингу 192.168.2.1.



Как только нажимаю ENTER, сразу появляется 2 конверта: ICMP и ARP. Остановимся и посмотрим на них подробнее. Сейчас может показаться, что передача между разными канальными средами ничем не отличается от передачи в одной канальной среде, но это не так. И сейчас вы это увидите.

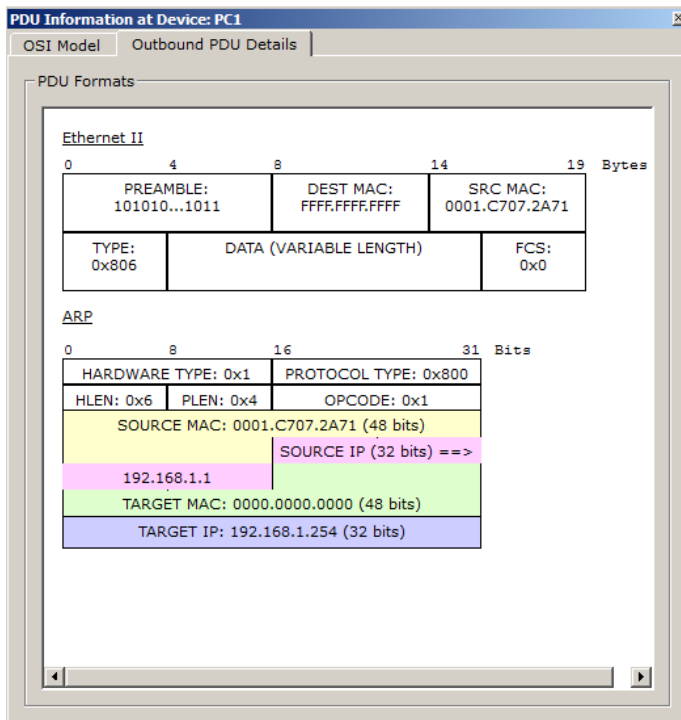
Сначала посмотрим на ICMP.

PDU Information at Device: PC1				
OSI Model		Outbound PDU Details		
PDU Formats				
<b>IP</b>				
0	4	8	16 19	31 Bits
4	4	8	16	31
IHL		DSCP: 0x0		TL: 128
ID: 0x1		0x0		0x0
TTL: 128		PRO: 0x1		CHKSUM
SRC IP: 192.168.1.1				
DST IP: 192.168.2.1				
OPT: 0x0		0x0		
DATA (VARIABLE LENGTH)				
<b>ICMP</b>				
0	8	16	31	Bits
0	8	16	31	
TYPE: 0x8		CODE: 0x0		CHECKSUM
ID: 0x2		SEQ NUMBER: 1		

Здесь пока в принципе ничего интересного. В поле источника — IP-адрес PC1, а в поле назначения — IP-адрес PC5.

Что же будет происходить дальше. PC1 видит, что проверяется доступность хоста, находящегося в другой канальной среде (путем наложения маски на свой IP-адрес и IP-адрес получателя). И кроме IP-адреса он не знает о получателе ничего.

Соответственно в таком виде отправлять пакет ICMP нельзя. Зато он знает, что у него есть основной шлюз, который, скорее всего, знает что-то про канальную среду, в которой находится PC5. Но возникает еще одна сложность. Он знает IP-адрес шлюза (который я ему прописал в сетевых настройках), но не знает его MAC-адреса. Тут ему приходит на помощь протокол ARP, который опросит всех участников канальной среды и найдет его MAC-адрес. Посмотрим, как заполнены поля.

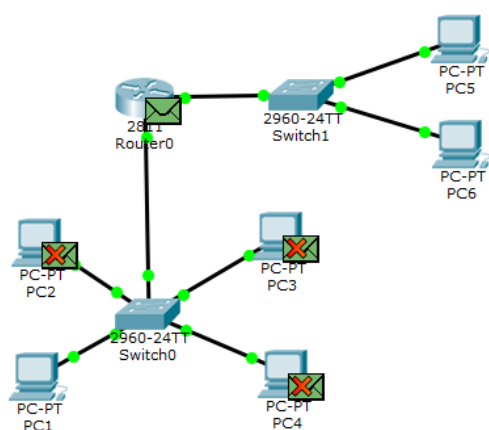


На канальном уровне (протокол Ethernet): Поле источника — MAC-адрес PC1, а в поле назначения — широковещательный адрес (то есть всем участникам).

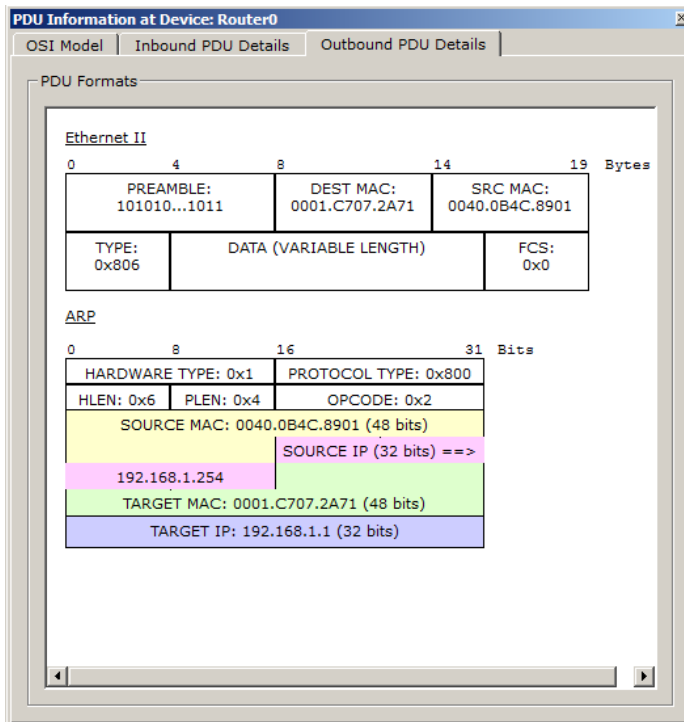
И чуть повыше (протокол ARP):

- 1) SOURCE MAC — тот же PC1, а DESTINATION MAC пустой (его должен заполнить тот, для кого этот запрос предназначен).
- 2) SOURCE IP — адрес PC1, а вот DESTINATION IP — адрес основного шлюза.

Смотрим, что будет происходить дальше.



3 компьютера отбросили пакет, и только маршрутизатор понял, что это для него. Смотрим, чем ответит.



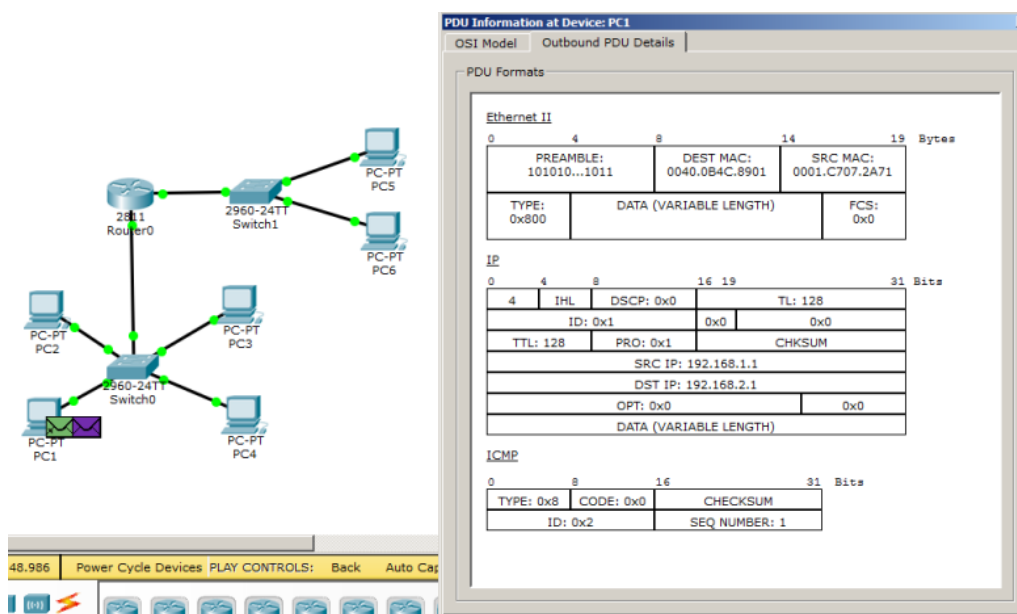
Ethernet:

- 1) Source MAC — сюда он вставляет свой MAC-адрес (а именно MAC-адрес fastEthernet0/0).
- 2) Destination MAC — сюда записывает MAC-адрес PC1 (то есть того, кто запросил).

ARP:

- 1) Source MAC и Destination MAC аналогично записям в протоколе Ethernet.
- 2) Source IP — свой IP-адрес.
- 3) Target IP — IP-адрес PC1.

Идем дальше.



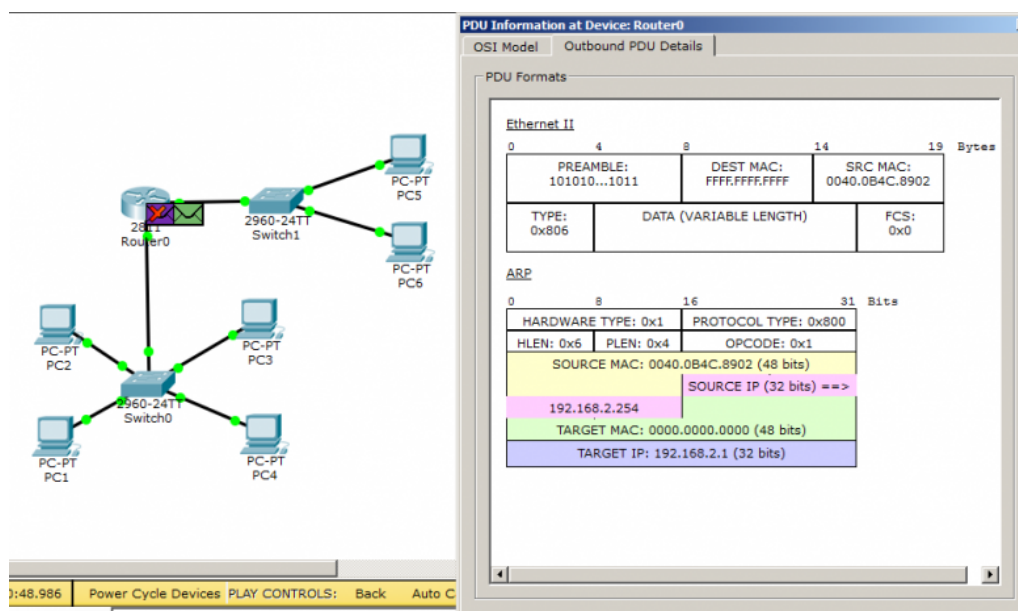
Как только ARP доходит от маршрутизатора к PC1, то сразу PC1 отправляет ICMP сообщение на маршрутизатор (или основной шлюз). И вот здесь прошу обратить

особое внимание. А именно, на поля источника и назначения (и в протоколе Ethernet, и в протоколе IP).

- 1) SRC MAC: здесь указан MAC-адрес PC1.
- 2) DEST MAC: MAC-адрес маршрутизатора.
- 3) SRC IP: IP-адрес PC1.
- 4) DST IP: IP-адрес PC5.

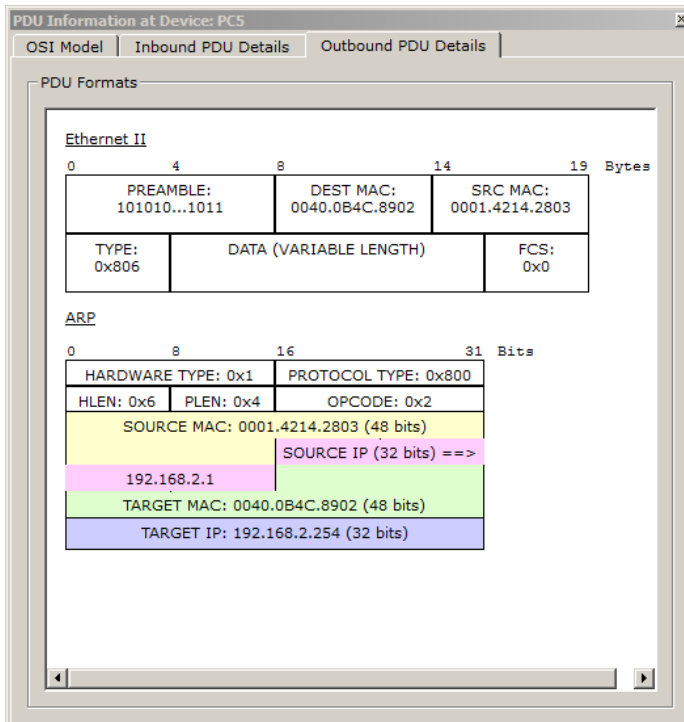
Что это значит. Адреса на сетевом уровне (то есть IP-адреса) не меняются, чтобы знать от кого и кому предназначается информация. А адреса на канальном уровне (MAC-адреса) могут спокойно меняться, переходя из одной канальной среды в другую. Это очень важно понять и запомнить!

Смотрим, что происходит. Пакет доходит до маршрутизатора и сразу перечеркивается. А все из-за того, что он не знает MAC-адрес PC5. Теперь он формирует ARP-запрос и пытается его узнать. Привожу скриншот этого запроса.

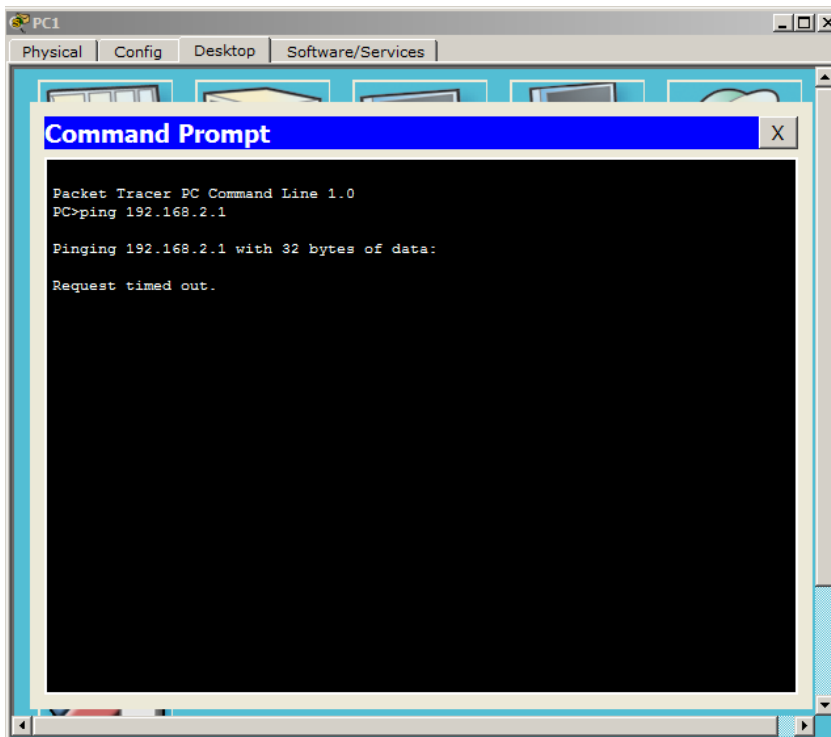


Далее PC5 получит его и сформирует ответ.

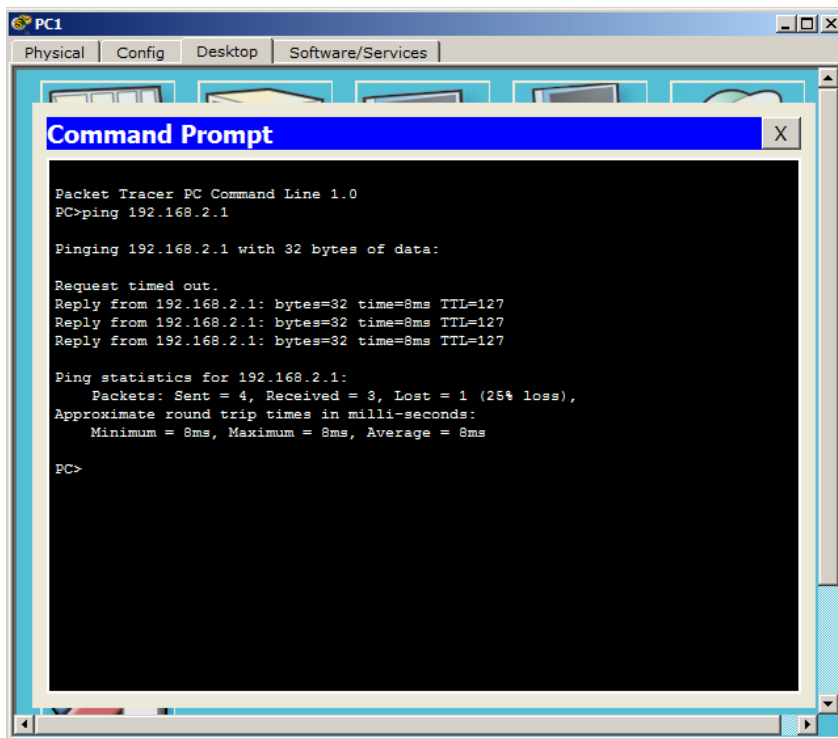




Как только этот ответ дойдет до маршрутизатора, он будет знать канальный адрес PC5. Но вот что произошло. Пока тянулась канитель с ARP у маршрутизатора и PC5, у PC1 истекает время ожидания ответа отправленного ICMP. Показываю картинку.



После истечения времени ожидания, он формирует второе ICMP, ответ которого уже дойдет без проблем, так как MAC-адреса известны. Следом он сформирует 3-ье и 4-ое ICMP. Привожу конечный итог.



И если внимательно присмотреться, то можно заметить, что TTL снизился на единицу и теперь равен 127. Это произошло из-за того, что пакет преодолел один транзитный участок (маршрутизатор).

Вот таким образом работает передача данных из одной канальной среды в другую (или из одной сети в другую). Тут, кстати, не важно, сколько канальных сред вам надо будет преодолеть, чтобы попасть до получателя. Принцип все равно будет такой.

В предыдущей статье, когда рассматривались протоколы верхнего уровня, мы немного касались транспортного уровня. Предлагаю вспомнить этот уровень и крепко закрепить.

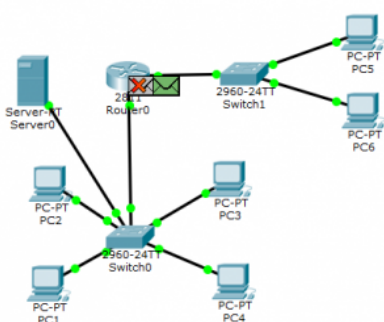
Начну, как всегда, с простого. И это протокол UDP. Как я уже говорил выше, он используется для того, чтобы передать данные определенному протоколу вышестоящего уровня. Делает он это при помощи портов. Один из протоколов, работающих с UDP — это TFTP(Trivial File Transfer Protocol). Протокол этот мы рассматривали в предыдущей статье. Поэтому трудностей возникнуть не должно. Для демонстрации потребуется добавить в сеть сервер с включенной службой TFTP.

Настройки сервера следующие:

- 1) IP-адрес — 192.168.1.5
- 2) Маска подсети — 255.255.255.0
- 3) Основной шлюз — 192.168.1.254

Служба TFTP включена по умолчанию, но лучше проверить. Далее переключаю CPT в режим симуляции и попробую сохранить конфигурацию маршрутизатора на TFTP-сервер:

- 1) Router>enable — *переход в привилегированный режим.*
- 2) Router#copy startup-config tftp: — *пишу команду copy (то есть скопировать), далее startup-config (что именно скопировать) и tftp: (куда скопировать).*
- 3) Address or name of remote host []? 192.168.1.5 — *выходит сообщение с запросом адреса или имени сервера, где я пишу его адрес.*
- 4) Destination filename [Router-config]? — *следом он спрашивает, под каким именем его сохранить на сервере и предлагает стандартное имя. Меня это устраивает, и я жму ENTER.*



Vis.	Time(sec)	Last Device	At Device	Type	Info
<input checked="" type="checkbox"/>	0.000	--	Router0	TFTP	
<input checked="" type="checkbox"/>	0.000	--	Router0	ARP	

Reset Simulation ☒ Constant Delay

Play Controls

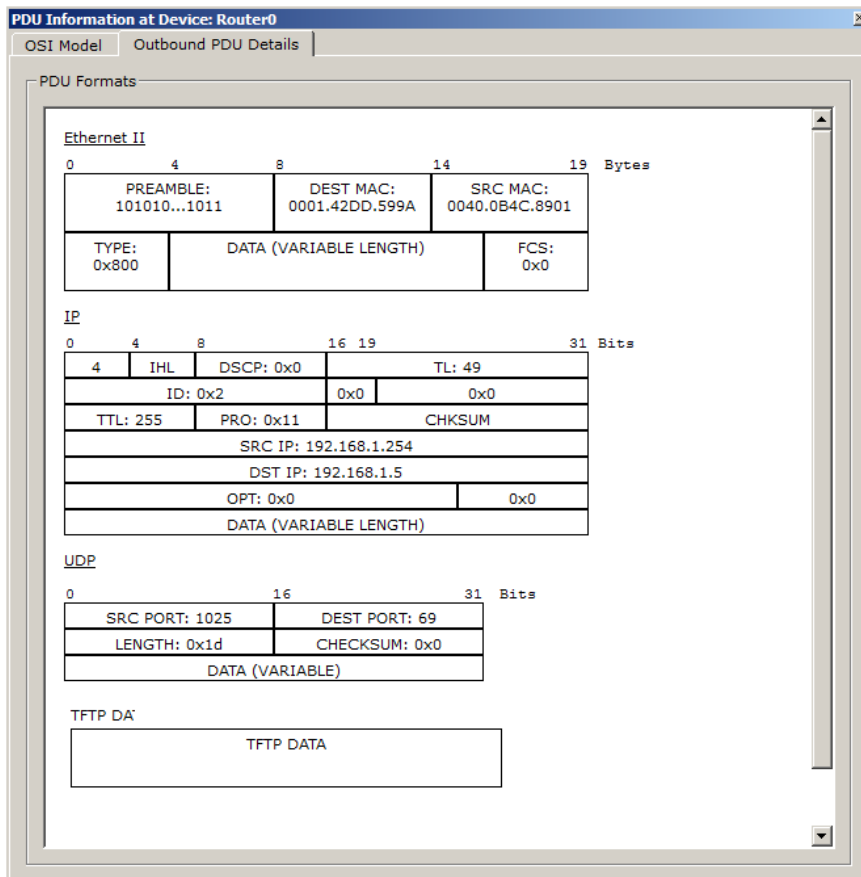
Back

Auto Capture / Play

Event List Filters - Visible Events  
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, HTTPS, ICMP, ICMPv6, IPsec, ISAKMP, LACP, NDP, NETFLOW, NTP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, ...  
[Edit Filters](#)

Сразу маршрутизатор формирует 2 конверта. Один из них — это перечеркнутый TFTP, а второй ARP. Думаю, догадались, что перечеркнут он из-за того, что не знает MAC-адрес сервера.

Пропущу я момент работы ARP, так как мы вдоволь на него насмотрелись.



Давайте подробнее разберем, что маршрутизатор отправляет на сервер.

### Ethernet:

- 1) Source MAC — адрес маршрутизатора.
- 2) Destination MAC — адрес сервера.
- 3) Type — 0x800 (означает, что выше работает протокол IP).

### IP:

- 1) Protocol — 0x11 (означает, что выше работает протокол UDP).
- 2) Source IP — адрес маршрутизатора.
- 3) Destination IP — адрес сервера.

### UDP:

- 1) Source Port — динамически созданный порт (1025).
- 2) Destination Port — порт, который слушает TFTP-сервер (зарезервированный 69 порт).

### TFTP:

Здесь находятся сами данные.

Так и работает протокол UDP. Он не устанавливает сессии, не требует подтверждения доставки, а если что-то потеряется, он не запрашивает повторно. Его работа — это указать номер порта и отправить. Что там будет происходить дальше, его не интересует. Но возникают случаи, когда это не устраивает и все эти параметры критически важны. Тогда на помощь приходит протокол TCP.