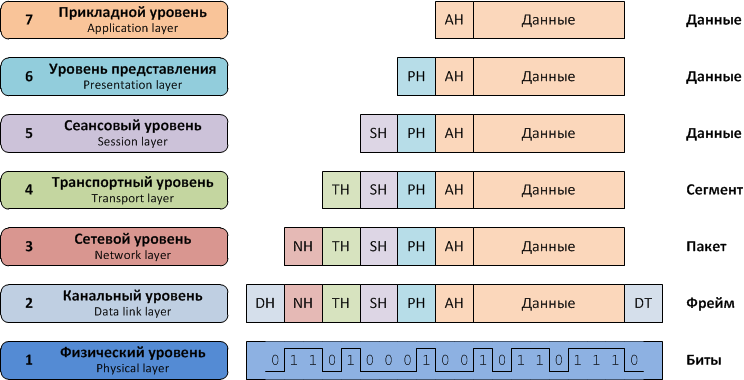
# 

# Лабораторная работа №4. Модель OSI

Мы хотим передать информацию, чтобы это сделать, компьютер превращает информацию в последовательность бит (последовательность 0 и 1). Чтобы это сделать, мы воспользуемся моделью OSI (правила преобразования данных для передачи по сети).

**Сетевая модель OSI** (open systems interconnection basic reference model — Базовая Эталонная Модель Взаимодействия Открытых Систем)



**Физический (L1 уровень)** - решает вопрос, как передать отдельные биты, определяет характеристики сигнала. Какого вида и какой формы должен быть сигнал (пример. Если это Ethernet кабель - какая модуляция будет, какое напряжение, ток. У wi-fi какие волны будут, частота, модуляция).

**Устройства на этом уровне:** Сетевая карта, модем, wi-fi антенна.

**Протоколы:** -

**Канальный уровень** **(L2)**  - как передать осмысленные последовательности из бит. Для передачи в одной канальной среде, появляются **MAC адреса**.

**Устройства на этом уровне:** Сетевой коммутатор (свитч), сетевая карта (работает с MAC).

**Протоколы:** Ethernet (осуществляет доставку данных в одной канальной среде)

**Сетевой уровень (L3)** - передача данных между канальными средами. Появляется понятие **IP адреса**.

**Протоколы:** IP

**Транспортный уровень (L4)** - определяет для какого приложения предназначается отдельный кусок информации. Определяет какую последовательность бит какому приложению передать. Появляется понятие **порта**.

**Протоколы:** UDP (без контроля доставки), TCP (с контролем доставки).

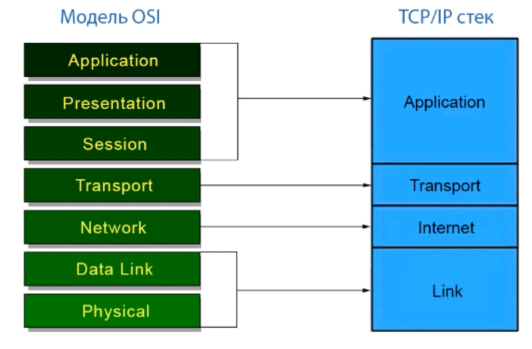
**Сеансовый уровень (L5)** - поддержка связи между приложениями;

**Уровень представления (L6)** - преобразует данные из одной формы в другую (кодирование и декодирование);

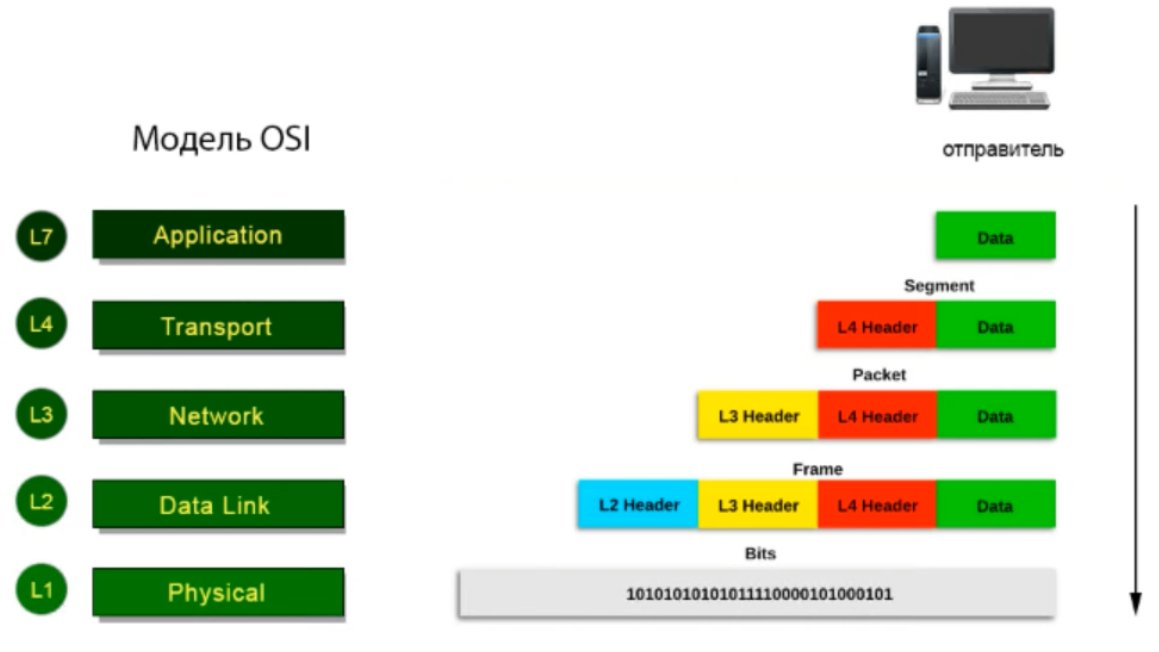
**Прикладной уровень (L7)** - для взаимодействия приложений со всеми другими уровнями.

*Благодаря многоуровневой системе, при написании приложения, программисту не надо думать как его программа будет общаться с 1 по 4 уровень.*

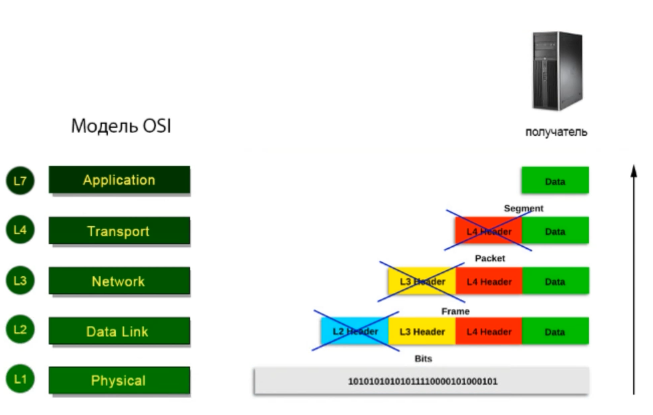
Модель **TCP/IP** - всё тоже самое только менее детально, более просто представление процессов, не отличается от OSI.



**Процесс передачи.** Каждый уровень инкапсулирует, заворачивает информацию прошлого уровня в свою оболочку (заголовки). Уровни 6-5 нас не интересуют.



Процесс получения, мы декапсулируем, чтобы получить данные.



**Структура заголовка протокола Ethernet**

Из чего состоит **L2 Header**

**(определяет как данные будут передаваться внутри канальной среды):**



* **MAC получателя** стоит в начале, для того чтобы умные Свитчи с технологией [Cut-Through](https://en.wikipedia.org/wiki/Cut-through_switching), которая принимает решение до получения всего кадра, на какой порт предложить кадр.
* **Тип (длина)** - указывается длина данных, которые содержатся в след. поле. Без неё было бы не понятно, когда заканчиваются данные и начинается чек сумма.
* **SNAP/LLC** - указывает для какого вышестоящего протокола предназначаются данные (обычно для IP, раньше был [IPX](https://ru.wikipedia.org/wiki/IPX));
* **+ данные** - ваша информация, которую вы передаете, до 1500 байт - это размер [MTU](https://ru.wikipedia.org/wiki/Maximum_transmission_unit) (т.к. чек сумма работает неэффективно после 1500 байт).

Но его можно увеличить в настройках сетевой карты, повысив скорость работы сети но понизив эффективность работы проверки целостности. Это можно делать, если вы уверены что у вас сеть работает стабильно. Эти увеличенные кадры называются [jumbo frame](https://ru.wikipedia.org/wiki/Jumbo-%D0%BA%D0%B0%D0%B4%D1%80) (jumbo-кадр). Но и принимающая сторона должна уметь работать с jumbo-кадрами. Кроме того это повышает нагрузку на роутер.

* **FCS** - чек сумма, для проверки не побился ли кадр при передаче (плохо обжат провод, неверно работает оборудование, были эл. маг наводки, в итоге 0 превратился в 1, а 1 в 0 и всё, сбой). Как это работает: при отправке данных считается чек сумма (математ операция), при получении выполняется та же операция, если сходится то всё хорошо. Если кадр побился, то он будет отправлен по новой.
* **Преамбула** - использовалась в топологии с общей шиной, когда одновременно было невозможно передавать кадры одновременно, она предотвращала коллизии, сообщая все участникам, кто сейчас будет передавать данные.

**Структура IP заголовка (L3 Header)**

для передачи между канальными средами

****

*Появляется IP адресация!*

**Поля:**

1. **Версия** - какая версия IP протокола v4 или v6
2. **IHL** - размер заголовка (многие поля плавающие в L3)
3. **Тип обслуживания** - для механизма очередей, в какой очерёдности посылать пакет (при большой нагрузке можем сказать насколько важный пакет для передачи)
4. **Длина пакета** - какого размера весь пакет, включая пользовательские данные.
5. **Время жизни (TTL)** - для того чтобы предотвратить петли маршрутизации. Мы задаём определённое количество жизней (255 обычно) проходя через каждый роутер отнимается “одна жизнь”. В итоге через 255 роутеров, пакет удалится. Это необходимо, чтобы не было блуждающих в сети пакетов.
6. **Протокол** - для какого вышестоящего протокола пакет (для UDP или TCP);
7. **Контрольная сумма заголовка** - считается чек сумма от полей (**не проверяется целостность данных!!**), только целостность заголовка.
8. **Опции** - редко используется, дописывается для специфических устройств (например - система контроля дверных замков);
9. **Смещение** - добивает 00 к полю Опции.
10. **Данные** - наши данные.

**Идентификатор, Флаги и Смещение фрагмента** - используются когда мы не можем передать данные одним пакетом, и разбиваем его на несколько пакетов.

**Например, разбили наши данные на 4 части,** то:

* **Идентификатор** - пишется единый для всех 4х ID.
* **Флаги** - указано что пакет фрагментированный а не единый. В последнем фрагменте также стоит флаг что он последний.
* **Смещение фрагмента** - нумерация пакета, какой пакет по счёту.

**Структура UDP заголовка (L4 Header)**



Осуществляет разделение трафика приложений.

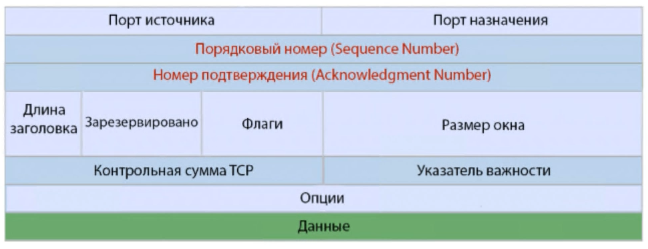
Мы хотим поиграть в игру [Delta Force](https://ru.wikipedia.org/wiki/Delta_Force_(%D0%B8%D0%B3%D1%80%D0%B0)) по сети через UDP, для этого:

1. Отправляем запрос, генерируя случайный **порт источника** (например 2000),
2. **Порт назначения указываем** = 35.
3. Сервер игры принимает на 35 порту данные,
4. Меняет местами порт источника и порт назначения и отвечает на 2000 порт.

**Контрольная сумма** - если не сходится, пакет отбрасывается (удаляется). Такое допустимо при отправке аудио или видео сигнала. Но недопустимо при передаче например электронной почты.

*UDP просто отправляет данные, без установки сессии, ему не важно данные дойдут или нет.*

**Структура TCP заголовка**



**Функции:**

1. Выполняет функцию **контроля доставки**. Если контрольная сумма не сходится, то данные запрашиваются еще раз.
2. **Механизм сборки пакета в правильной последовательности** (при разбивке данных на разные пакеты). Наши пакеты могут прийти не в правильной последовательности, TCP всё равно соберёт всё верно.
3. **Установление сессии** путем трёхэтапного рукопожатия.
4. Пори источника - случайный.
5. Порт назначения - обычно конкретный, исходя из [сервисов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BF%D0%BE%D1%80%D1%82%D0%BE%D0%B2_TCP_%D0%B8_UDP).
6. Порядковый номер - рассмотрим позже
7. Номер подтверждения - рассмотрим позже
8. **Длина заголовка** - определяет сколько бит заголовок, сколько бит данные.
9. Флаги - для установления сессии
10. Размер окна - число, говорящее на какое число пакетов отправлять подтверждение о получении пакетов. Размер этого числа зависит от качества работы сети, постепенно увеличивается, если потери встречаются очень редко.
11. Контрольная сумма - математ. операция от всех полей и данных для контроля целостности.
12. Указатель важности - уровень важности данного пакета.
13. Опции - у TCP есть масса дополнений и расширений, взаимодействие с которыми происходит через это поле.

**Процесс установления TCP сессии:**

1. Генерируется случайный порт источника;
2. Порт назначения всегда определен, например 80 порт, если мы стучимся к веб-серверу;
3. В поле порядковый номер генерируется случайное число;
4. Устанавливается флаг SYN (флаг желания установки сессии);

**Данные отправляются, доходят до сервера;**

1. Сервер меняет местами порт источника и порт назначения;
2. Порядковый номер - генерируется любое число;
3. Номер подтверждения, прибавляет 1 к порядковому номеру, который пришел;
4. К флагу SYN добавляется флаг ACK

**Отправляет данные клиенту;**

1. Клиент меняет местами порт источника и порт назначения;
2. Номер подтверждения, прибавляет 1 к порядковому номеру, который пришел;
3. Оставляет только флаг ACK

**Данные отправляются, доходят до сервера;**

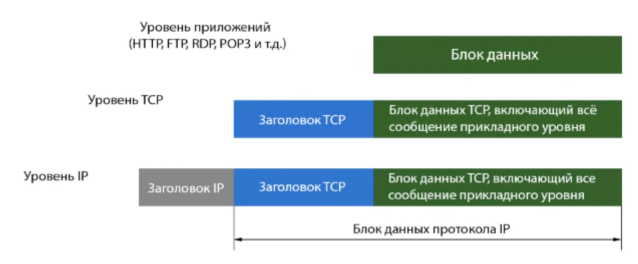
Установка сессии завершена! У нас произошло трехэтапное рукопожатие (3-way handshake).

Когда сессия установлена, поля Порядковый номер и Номер подтверждения используются по другому. В Порядковом номере указывается номер пакета, в номере подтверждения указываются номера пакетов которые уже были получены стороной отправления.

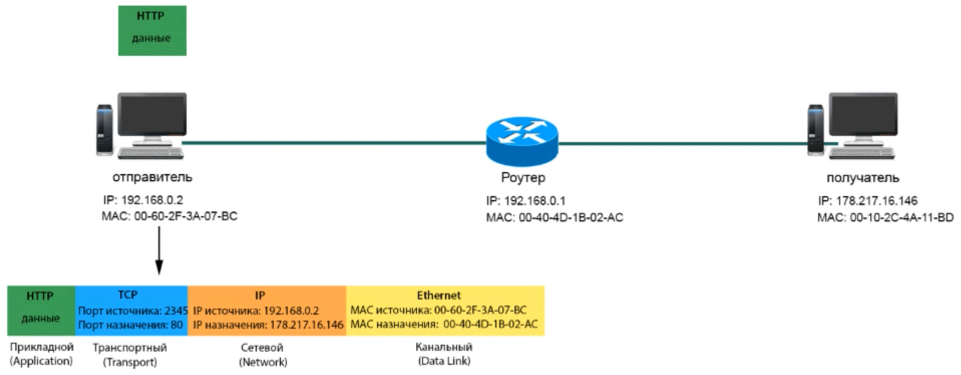
Подробно рассмотрим поле **данные**, для каждого из уровней.

1. На канальном уровне данными является: Заголовок IP + другие данные;
2. Данными для заголовка IP будет: Заголовок TCP с данными;
3. Данными для заголовка TCP будет: например протокол http с пользовательскими данными (например, с html кодом);

Всё это дело называется ”Инкапсуляция данных”



## Рассмотрим процесс передачи данных по сети через роутер



У нас есть отправитель, который хочет отправить какие то HTTP данные. начинает работать модель OSI которая

1. берет эти данные,
2. добавляет TCP заголовок,
3. добавляет IP заголовок,
4. добавляет Ethernet заголовок.

* **Транспортный уровень:** придумывает себе порт, порт получения известен.
* **На сетевом уровне:** задается IP получателя и отправителя (для передачи данных в разных канальных средах);
* **На канальном уровне:** задаются MAC получателя в нашей кнанальной среде (т.е. роутера) и MAC отправителя.

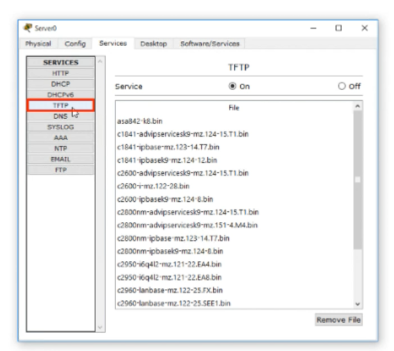
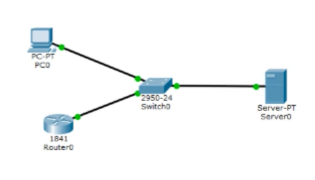
Роутер: отрезает Ethernet заголовок с MAC (он больше не нужен) и добавляет свой Ethernet заголовок, вбивая туда как MAC отправителя свой, и MAC получателя.

## Рассмотрим процесс передачи данных по сети через коммутатор

## 

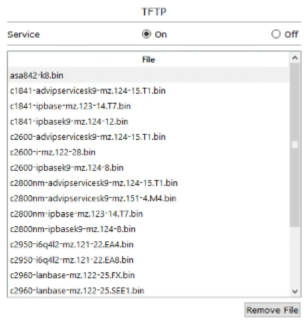
Всё тоже самое, только канальный уровень не отрезается и MAC сразу задан у целевой машины.

Задание. Разверните в CPT сеть для демонстрации работы протокола UDP включите сервис TFTP (урезанная версия протокола FTP без поддержки аутентификации и работающая на UDP) на сервере. Настройте роутер (воспользуйтесь инструкциями с прошлой лабораторной работы).



После того как схема готова,

1. Наберите команду # copy tftp flash: (запрос на копирование в флешь память роутера).
2. Далее указываем IP адрес сервера.
3. Далее указываем название файла который хотим скачать (список файлов доступен на сервере). Включаем режим симуляции.



1. Далее у нас спрашивают имя назначения. Здесь мы можем задать новое имя скачиваемому файлу. Можем ввести любое своё значение.
2. После чего наблюдайте за обменом пакетов, сначала пойдёт ARP потом STP и только потом TFTP. Посмотрите что содержится внутри TFTP
3. 