Поговорим о Сетях

Борисенко Глеб

26.09.2024

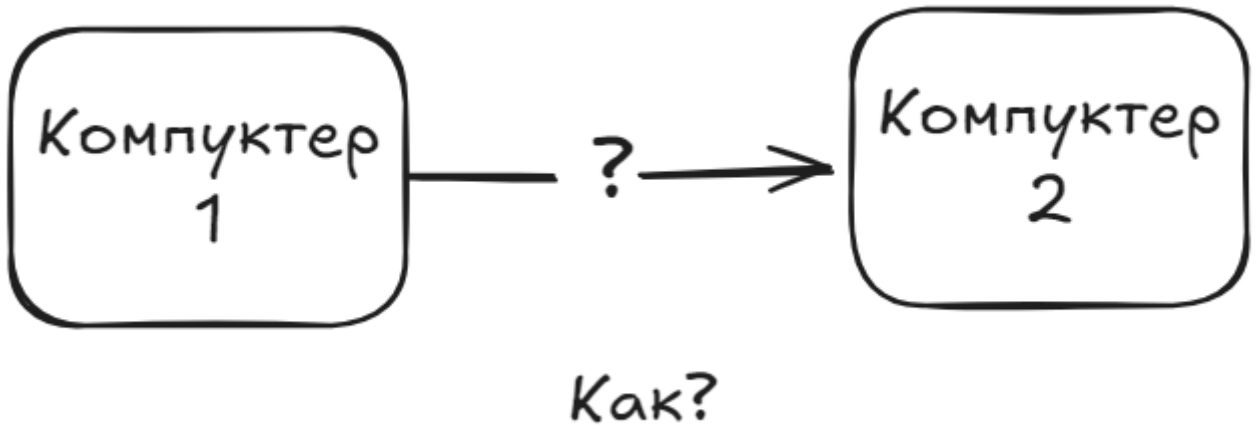
(на самом деле это не говно)

О чем болтали в прошлый раз

- Как писать хороший код ~~чтобы с него не плакали и на вас не кричали~~
- Всякие принципы, паттерны
- Что может помочь (poetry, black, ruff, vscode, etc.)

Начнем с базы

Есть два компуктера. Как передать данные?



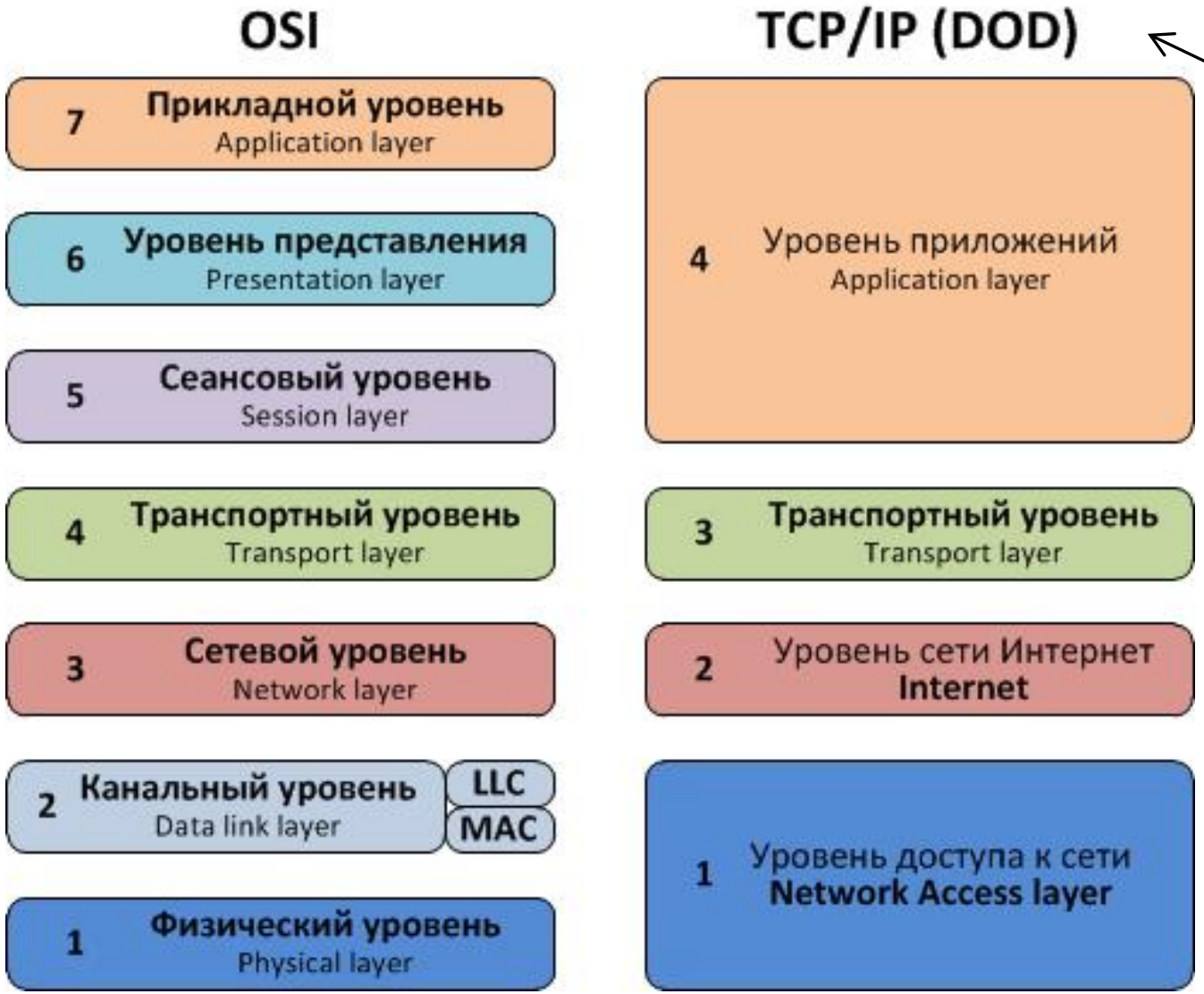
Любые данные – это последовательность единичек и нолей. Это мы все знаем. Но как их передавать по проводам (или воздуху)?

«gleb» -----> 0110 0111 0110 1100 0110 0101 0110 0010

Для этого есть модель OSI

Данные	Прикладной доступ к сетевым службам
Данные	Представления представление и кодирование данных
Данные	Сеансовый Управление сеансом связи
Блоки	Транспортный безопасное и надёжное соединие точка-точка
Пакеты	Сетевой Определение пути и IP (логическая адресация)
Кадры	Канальный MAC и LLC (Физическая адресация)
Биты	Физический кабель, сигналы, бинарная передача данных

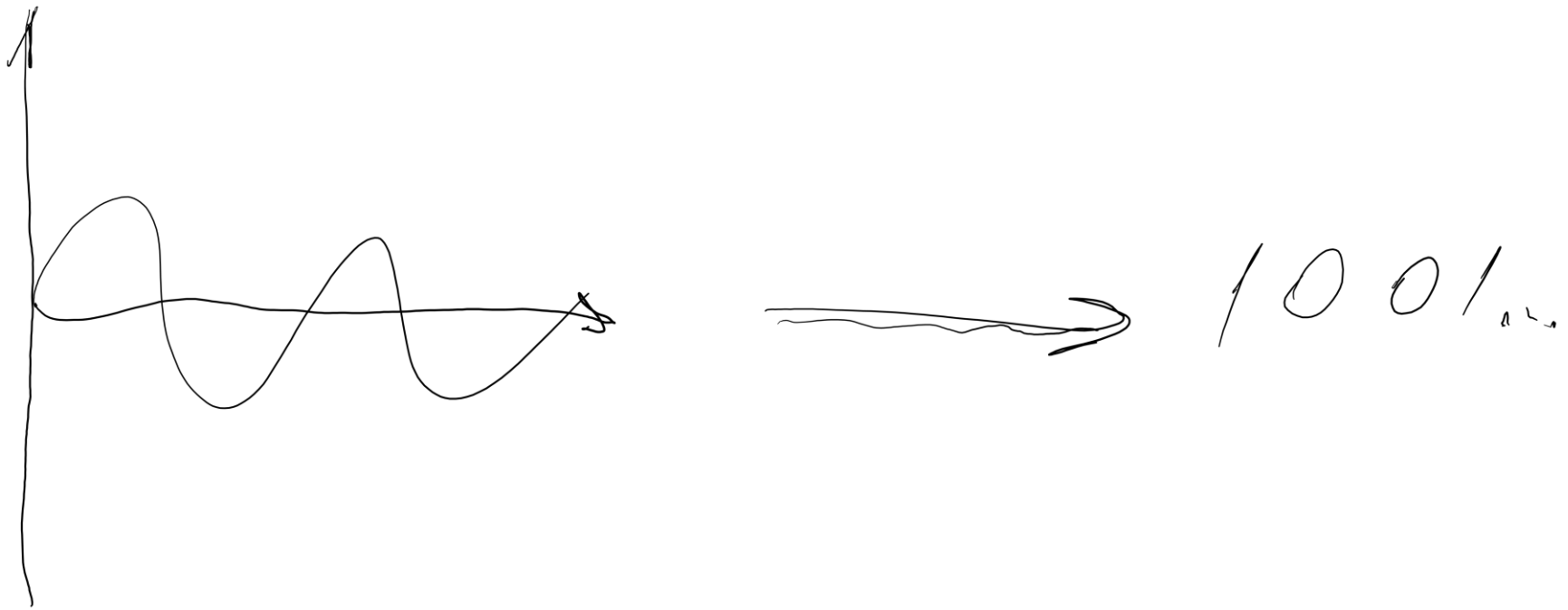
Модель интернета (TCP/IP)



Преимущественно будем смотреть сюда

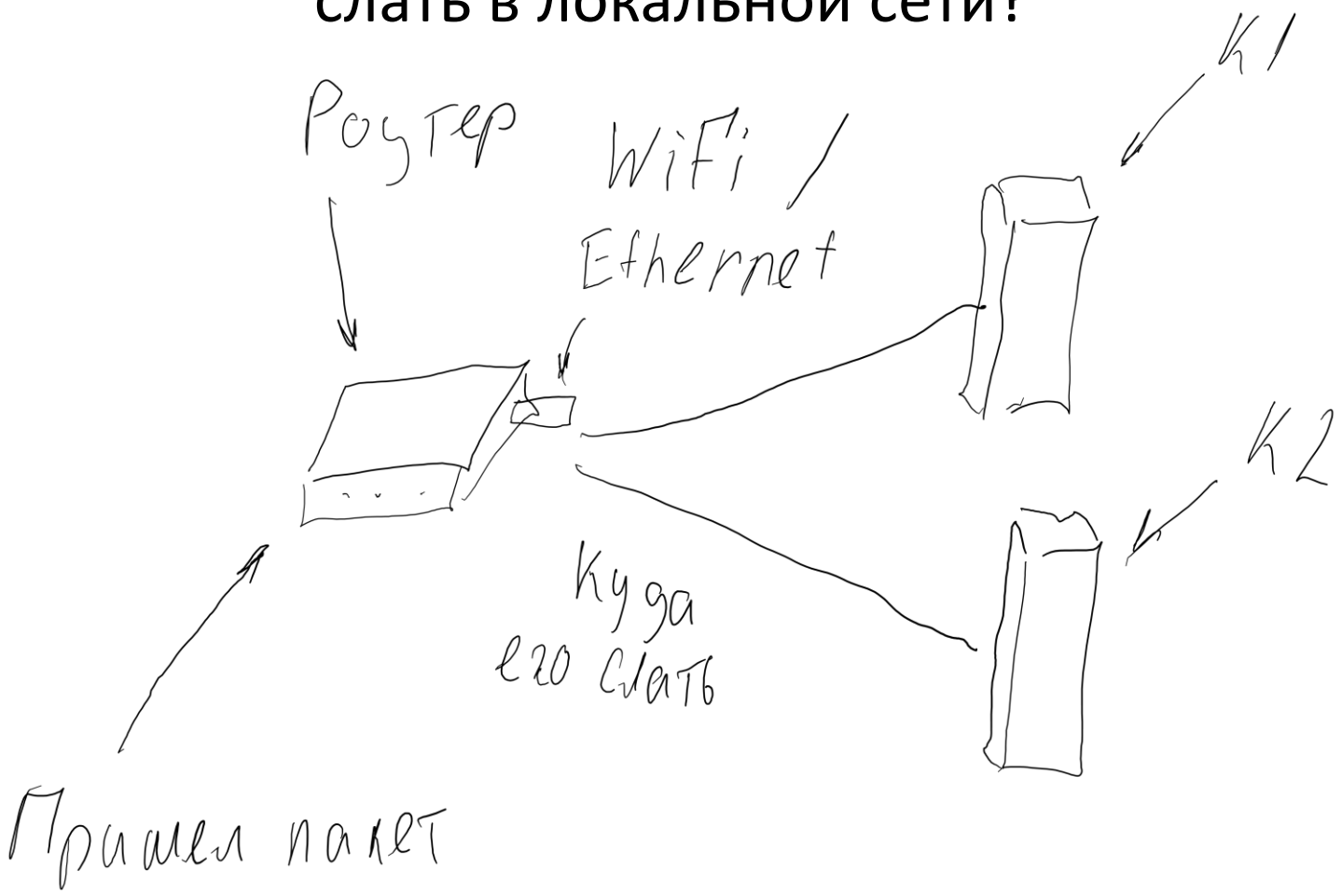
Физический

Сигналы в единички и нолики



Канальный

Вводим понятие пакета/фрейма – единицы данных здесь. Куда его слать в локальной сети?



Протоколы канального уровня

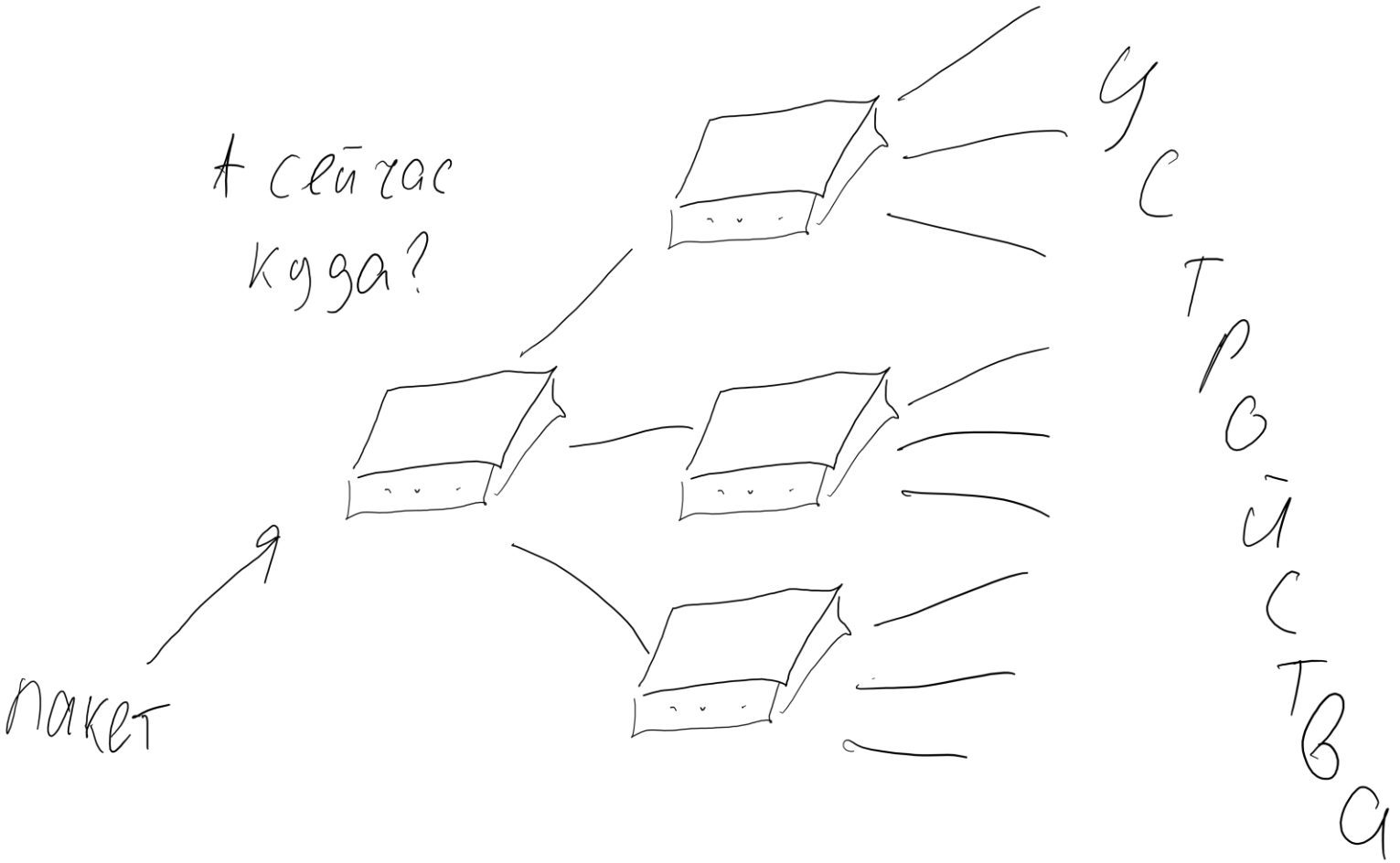
- Ethernet
- Протоколы WiFi – 80x.xx
- Есть еще

Обеспечивают:

- Получение доступа к среде передачи
- Выделение границ кадра
- Аппаратная адресация (MAC-адрес)
- Обеспечение достоверности принимаемых данных (контрольные суммы)

Сетевой

Подсетей много. Опять вопрос: куда слать?



Что он делает

- Отвечает за трансляцию логических адресов и имён в физические
- Определение кратчайших маршрутов
- Маршрутизацию (IP-адрес, маска подсети на роутере)
- Отслеживание неполадок и заторов в сети.

Здесь работают IPv4, IPv6, IPSec, ICMP, etc.

IPv4

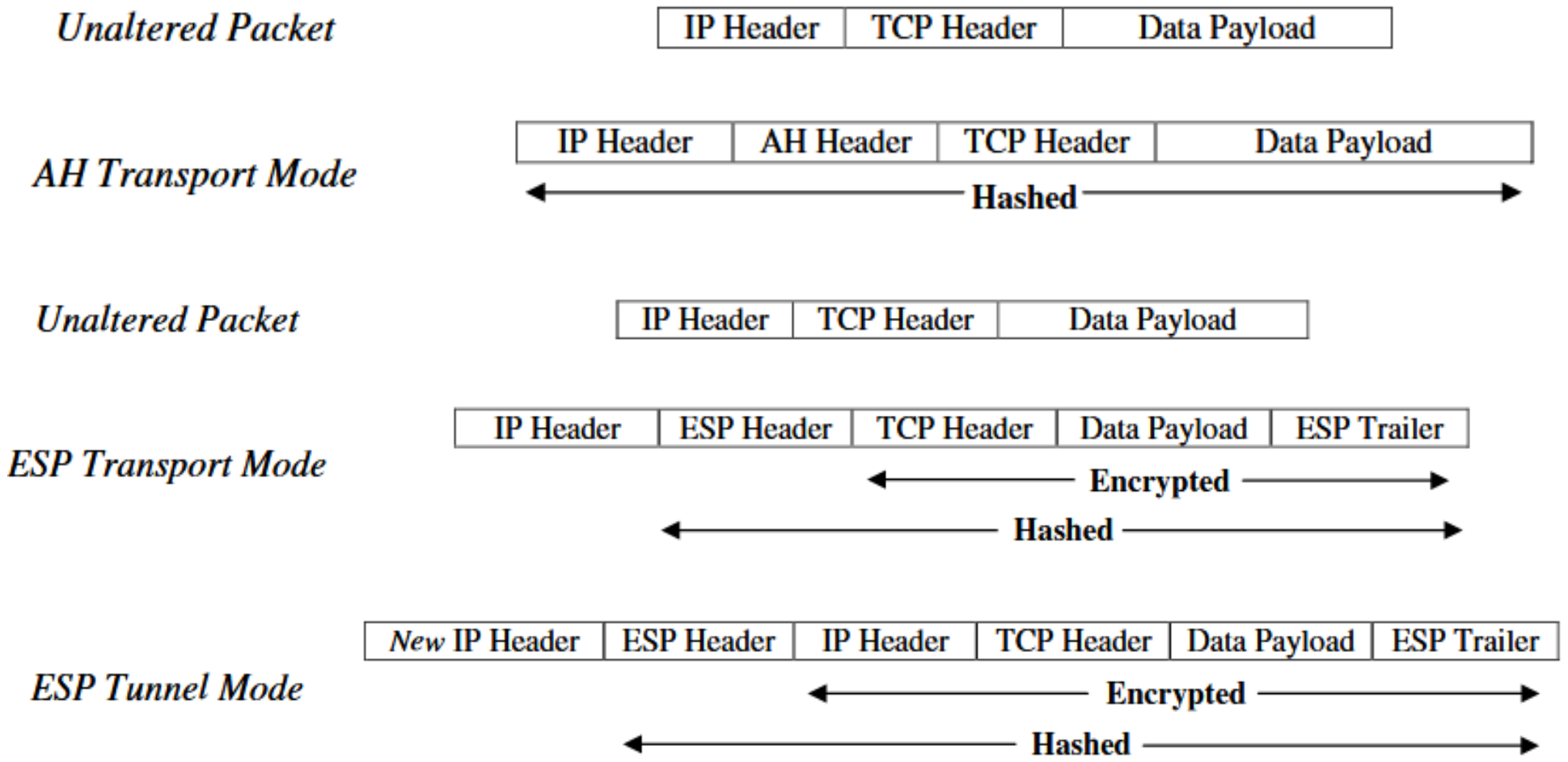
- 32-битные (четырёхбайтные) адреса, ограничивающие адресное пространство 4 294 967 296 (2^{32}) возможными уникальными адресами;
- CIDR – ~~вкусный алкогольный напиток~~ бесклассовая адресация по маскам
- У пакета есть время жизни – time-to-live (TTL)

IPv6

- Круто
- Модно
- Молодежно
- Адресов сильно больше (здесь адрес 16 байт вместо 4-х)
- Возможна поддержка огромных пакетов до 4 гб
- Появилось многоадресное вещание

IPSec

Authentication Header (AH) и Encapsulating Security Payload (ESP)



Транспортный

Что обеспечивает

- Доставка данных до конкретного «слушателя»
- Фрагментация длинных и объединение коротких сообщений

- Вводится понятие порта – целого неотрицательного числа, являющегося адресом конкретного «слушателя» на устройстве
- Порт – 2 байта в пакете, число от 1 до 65 535 (у 0 особое значение)

User Datagram Protocol (UDP)

- Тупо адреса портов, длина нагрузки и контрольная сумма
- Всё
- Важно: нет механизма отслеживания и лечения потери пакетов

- Используется обычно для передачи потоковых данных – видео, аудио, игры там, где потеря пакетов не так страшна, а задержка нужна минимальная

Transmission Control Protocol (TCP)

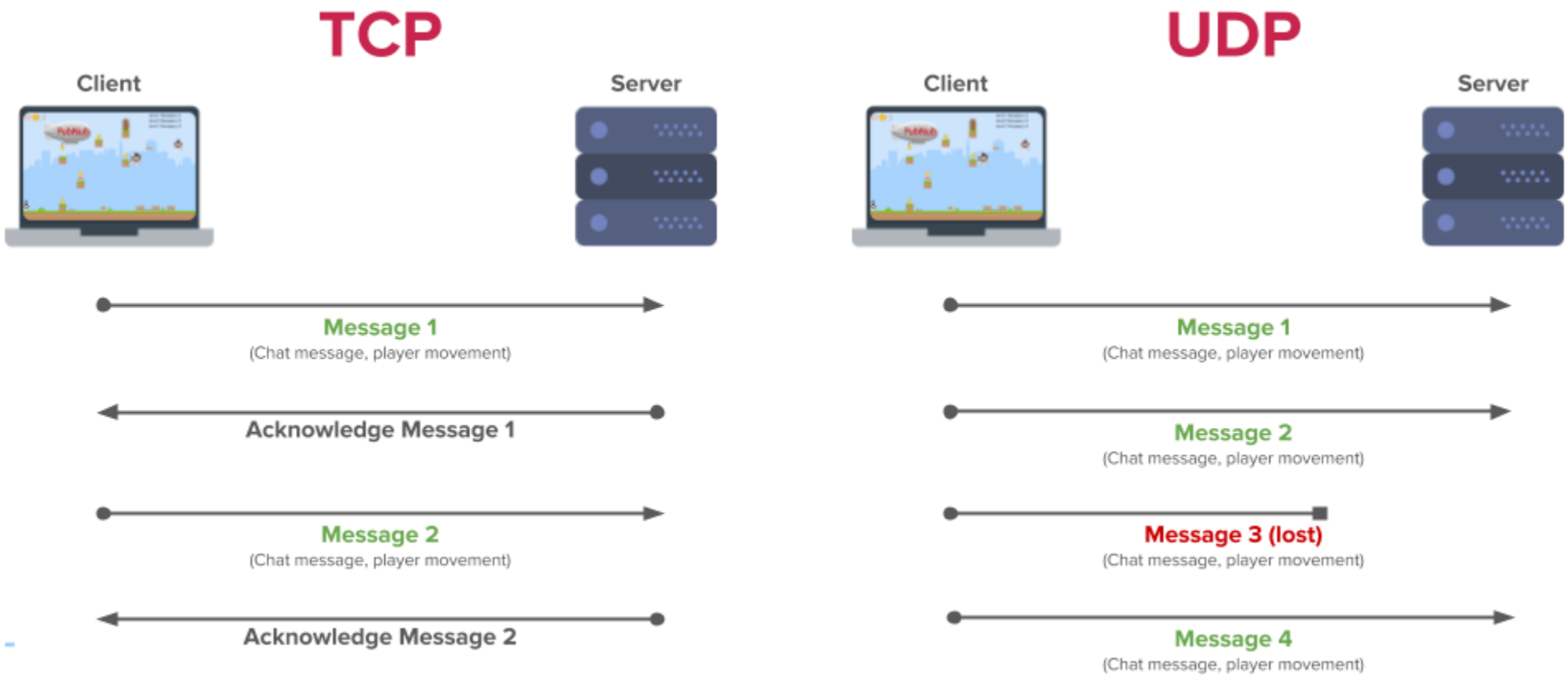
То же самое, что в UDP, плюс:

- Порядковый номер переданного байта в текущем соединении
- Номер подтверждения полученного байта (до этого байта все байты были получены)
- Флаги всякие служебные
- Размер окна, после которого ждать подтверждение получения
- Опциональная шелуха

Transmission Control Protocol (TCP)

- Взаимное рукопожатие при соединении
- Взаимное рукопожатие при разъединении
- Взаимное подтверждение получения при передаче данных

TCP vs UDP


TCP
Client
Server
Message 1
(Chat message, player movement)
Acknowledge Message 1
Message 2
(Chat message, player movement)
Acknowledge Message 2
UDP
Client
Server
Message 1
(Chat message, player movement)
Message 2
(Chat message, player movement)
Message 3 (lost)
(Chat message, player movement)
Message 4
(Chat message, player movement)

Сеансовый

- Поддержание сеанса связи

- gRPC – посмотрим на след паре

Уровень представления

- Кодирование/декодирование данных
- Запросы приложений, полученные с уровня приложений, он преобразует в формат для передачи по сети, а полученные из сети данные преобразует в формат, понятный приложениям

- ASCII, JPEG, etc.
- Часто это уровень реализован в самом приложении

Прикладной

- позволяет приложениям использовать сетевые службы:
  - удалённый доступ к файлам и базам данных,
  - пересылка электронной почты;
- отвечает за передачу служебной информации;
- предоставляет приложениям информацию об ошибках;
- формирует запросы к уровню представления.

P.S.: скопировано из вики

ХТТП (HTTP)

- Основа – клиент-сервер
- Стартовая строка – метод (GET, POST, etc.) и версия http (0.9, 1.1, etc.)
- Заголовки – параметры ‘key: value’ вроде языка, типа содержимого, типа сервера и т.п.
- Тело сообщения – чаще всего вы будете иметь дело с JSON

HTTPS

- Тот же HTTP, только использующий шифрование SSL и TLS
- Здесь используются сертификаты для «доверия»
- Сами данные шифруются одним из методов криптографического шифрования данных

DNS

- Получение IP-адреса по домену
- Имеет иерархическую структуру «перевода»
- Что-то вроде распределенной базы

SSH

- Протокол прикладного уровня
- Позволяет удаленно управлять ОС
- Есть ssh-тунелирование. Когда на одном конце шифруем, передаем по ssh, и на другом конце расшифровываем

Сокет

- Сокет — название программного интерфейса для обеспечения обмена данными между процессами.
- По сути, это IP:PORT
- В языках программирования есть верхнеуровневые либы (например, requests), которые реализуют под капотом работу с сокетами
- Так, requests – не дефолтная питоновская либа, а socket - дефолтная

Сессия

- Все взаимодействия между браузерами и серверами осуществляются при помощи протокола HTTP, который не сохраняет своё состояние (stateless).
- Сессии являются механизмом, который используют верхнеуровневые либы для отслеживания "состояния" между сервисом и «пользователем».

Кукис

- Куки — небольшой набор данных, отправляемый веб-сервером и хранимый на компьютере пользователя без изменений и какой-либо обработки.
- Веб-клиент (обычно веб-браузер) всякий раз при обращении к соответствующему сайту пересылает эти данные веб-серверу в составе HTTP-запроса.
- Обычно куки применяют для:
  - аутентификации пользователя;
  - хранения персональных предпочтений и настроек пользователя;
  - отслеживания состояния сеанса доступа пользователя;
  - хранения сведений статистики о пользователе.

Прокся(proxy)

- Прокси – промежуточный сервер, «посредник» во взаимодействии между сервером и клиентом
- Могут использоваться для «скрывания» IP
- Обычно не шифруется никак трафик дополнительно
- Могут использоваться для аутентификации, безопасности, балансировки

Масштабирование

- Что делать, когда запросов много и сервис не справляется? МАСШТАБИРОВАТЬСЯ
- Есть два вида масштабирования
- Вертикальное – накидываем мощностей: берем просто сервак покруче
- Очевидно, что тут есть ограничения, в которые легко упереться
- Горизонтальное – берем еще один или несколько серваков, каждый – наш сервис, вуаля (почти)
- Как перенаправлять запросики?

Балансировка трафика

- Хотим масштабировать сервис, чтобы выдерживал большую нагрузку. Как?
- Поднимаем балансировщик – он будет сам перенаправлять по выбранному алгоритму трафик
- Пример балансировщика - Nginx

Как происходит балансировка?

- Есть разные алгоритмы балансировки
- Например, RoundRobin: просто по циклу отправляем пакеты разным серверам
- Может использоваться «взвешенный» подход
- Также храниться «таблица», чтобы пакеты от одного пользователя шли одному и тому же серверу
- В общем, куча нюансов, разные алгоритмы, разные балансировщики

Общение между сервисами

- Казалось бы, просто шлем пакетики и кайфуем, но нет
- Что делать с очередями?
- У HTTP есть таймауты, и если сервис не успевает обработать – мы считаем, что нас послали
- А если пакет прокакался как-то на сервисе?
- Очередь пакетов HTTP, что по дефолту есть, ограничена

Общение между сервисами

- Отсюда приходим к термину Message Broker, по-русски брокеру сообщений
- Менеджментит общение между сервисами, работает с очередями
- Пример: Kafka или RabbitMQ