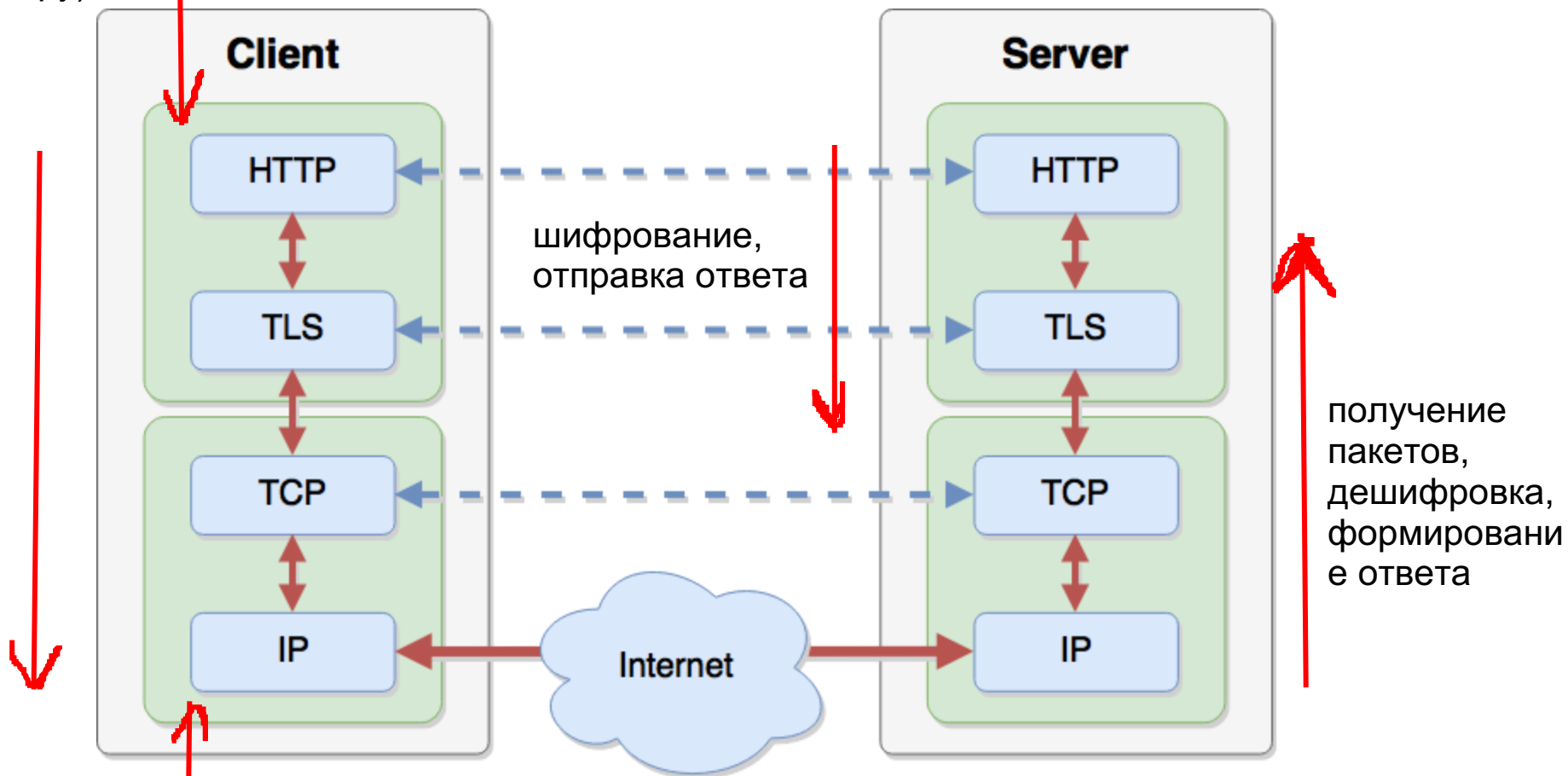


Как происходит
HTTP запрос?

Как происходит HTTP запрос ?

- Браузер анализирует введенный URL и извлекает имя хоста
- Используя систему DNS, браузер преобразует домен в ip адрес IPv4 - число(32 бита)
- Устанавливает TCP соединение с web-сервером TCP устанавливается после успешного получения пакетов сервером
- Если протокол https, устанавливает TLS соединение поверх TCP ()
- Формирует HTTP запрос, отправляет его, HTTP ответ
- Браузер закрывает соединение (для HTTP/1.0)
- Далее процесс парсинга и отображения документа ...

уровень клиента(варьируется от браузера к браузеру)



уровень ядра ОС,
реализовано на низком уровне
(на уровне сетевых вызовов к ядру)

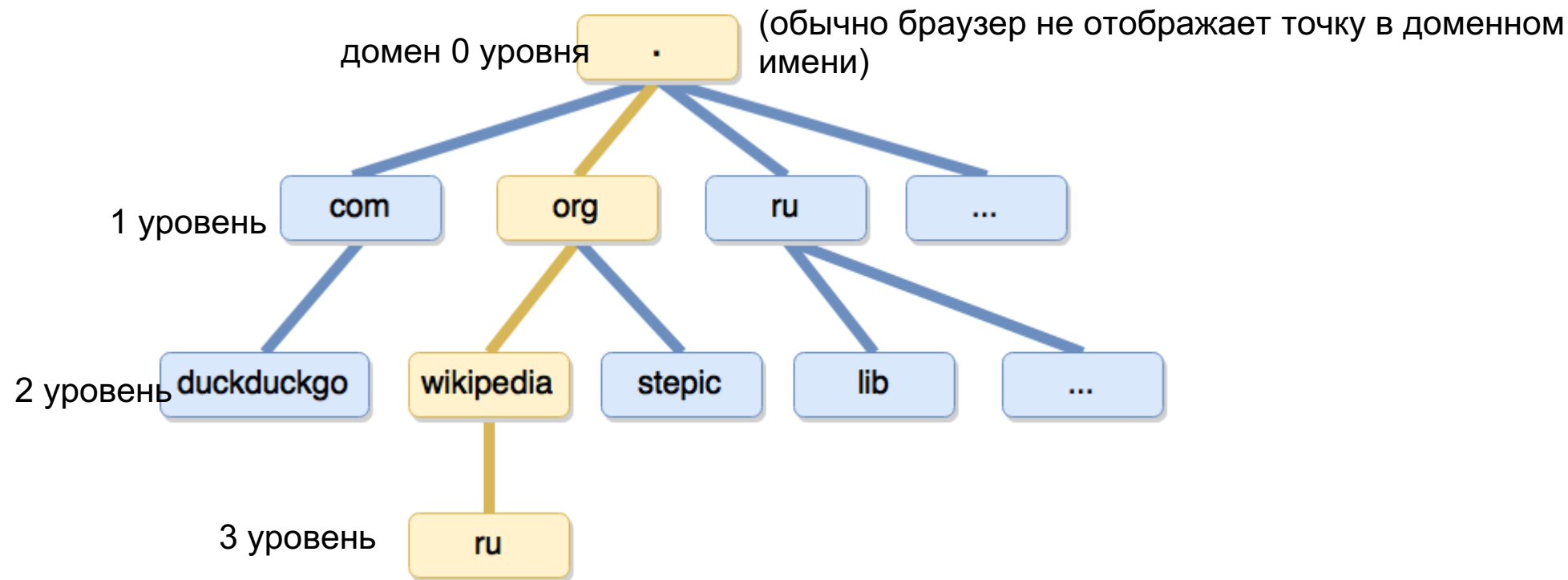
отправка IP
пакета

DNS

Domain Name System

DNS - это распределенная база данных, хранящая информацию о доменах, в первую очередь отображение доменных имен на IP адреса машин, обслуживающих эти домены

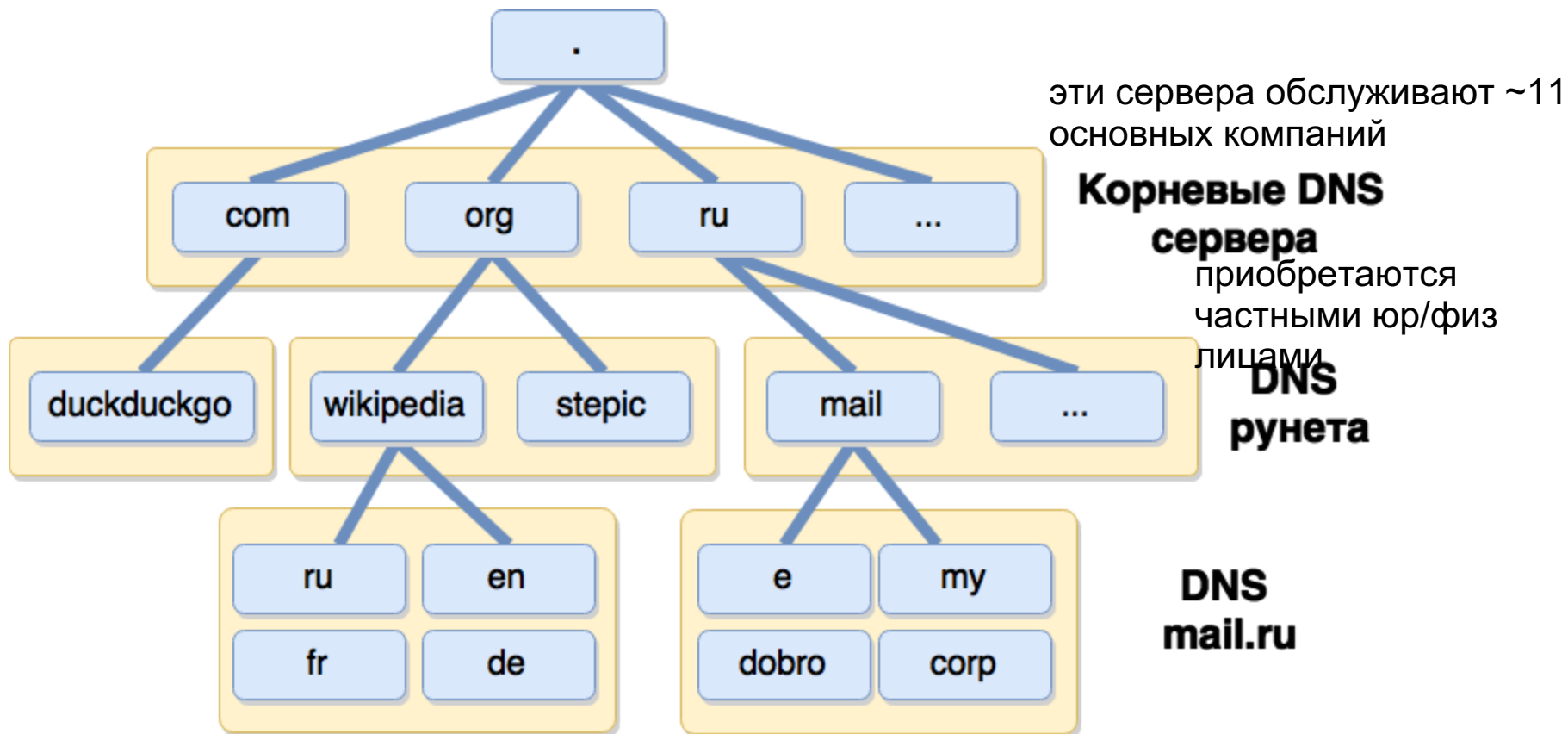
Пространство доменных имен

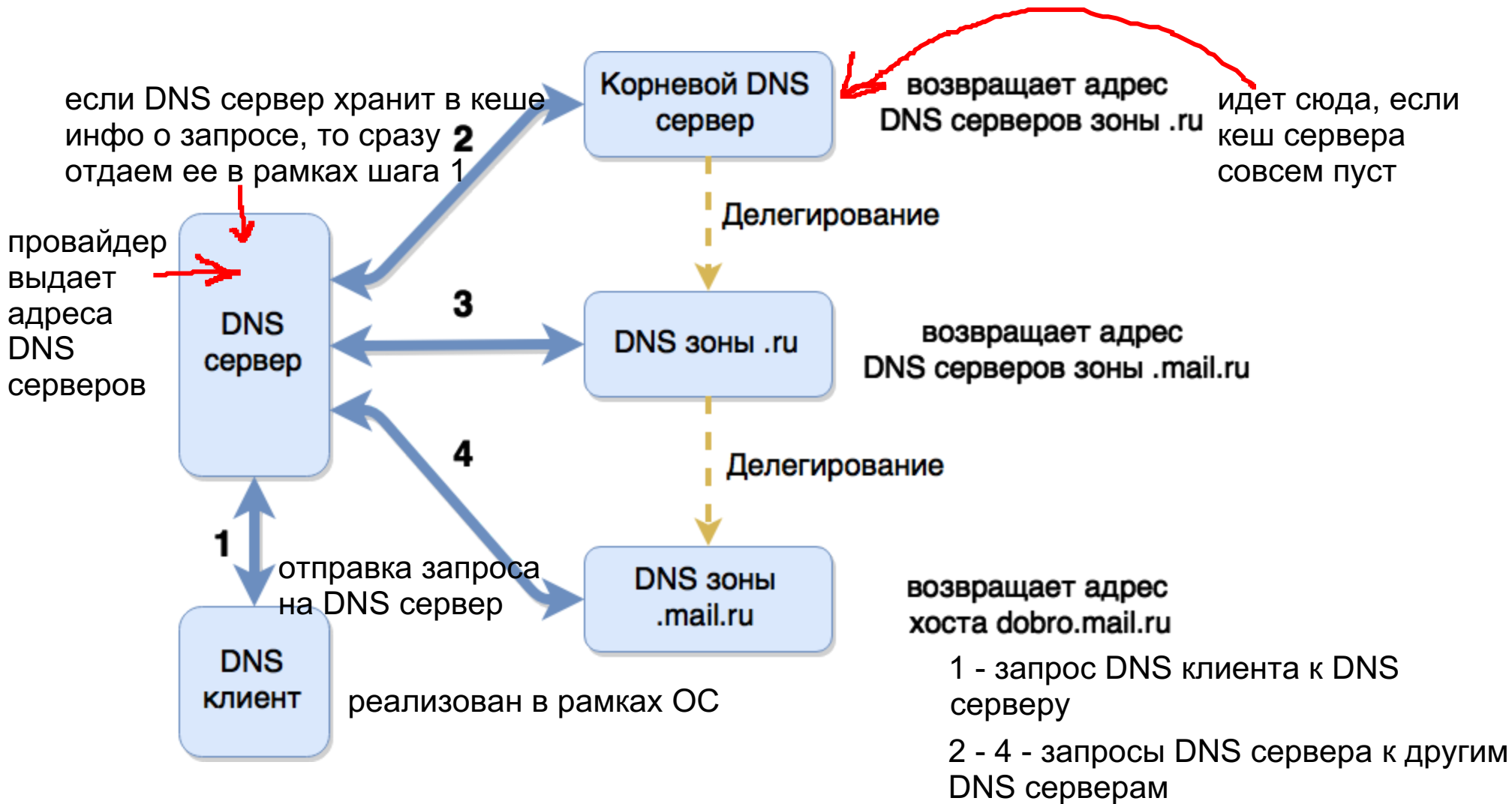


Домены и зоны

База DNS разделена на **зоны**. Каждая зона находится под единым административным контролем. Проще говоря обслуживается одной организацией.

Хранение информации о доменах более высокого уровня может быть **делегировано** другим зонам.





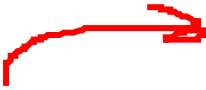
Что содержит зона DNS сервера ?

- A - IPv4 адрес(а) для данного домена
- AAAA - IPv6 адрес(а)
- NS - адрес(а) DNS серверов обслуживающих данную зону
- MX - адрес(а) почтовых серверов для данного домена

TCP

Зачем нужен TCP ?

TCP - протокол, обеспечивающий надежную последовательную доставку данных. Фактически, TCP предоставляет интерфейс, похожий на файловый ввод/вывод для сетевых соединений. (СОКЕТЫ)

- Надежная доставка  система контрольных сумм(проверка целостности пакетов), система id пакетов(проверка порядка отправки пакетов), система подтверждения получения пакетов с определенным таймаутом(в случае любого провала/несоответствия - переотправка пакетов)
- Полнодуплексная передача параллельная передачи данных от клиента к серверу и обратно (duplex - двухсторонний)
- Контроль потока - защита от переполнения

алгоритм дробления данных по группам пакетов(

ПОРТ - программа на конкретной машине(в сети идентифицируется с помощью IP), устанавливающая(открывающая) соединение по конкретным протоколам(в зависимости от порта) с помощью интерфейса сокетов --> за счет портов определяется, какой программе какой пакет должен быть доставлен

TCP порты

TCP порт - это «адрес» сетевого соединения в пределах одного хоста. TCP порты позволяют поддерживать множество открытых соединений на одной машине.

Номер порта - целое число, не больше 65535. Порты ниже 1024 требуют привилегий суперпользователя для использования.

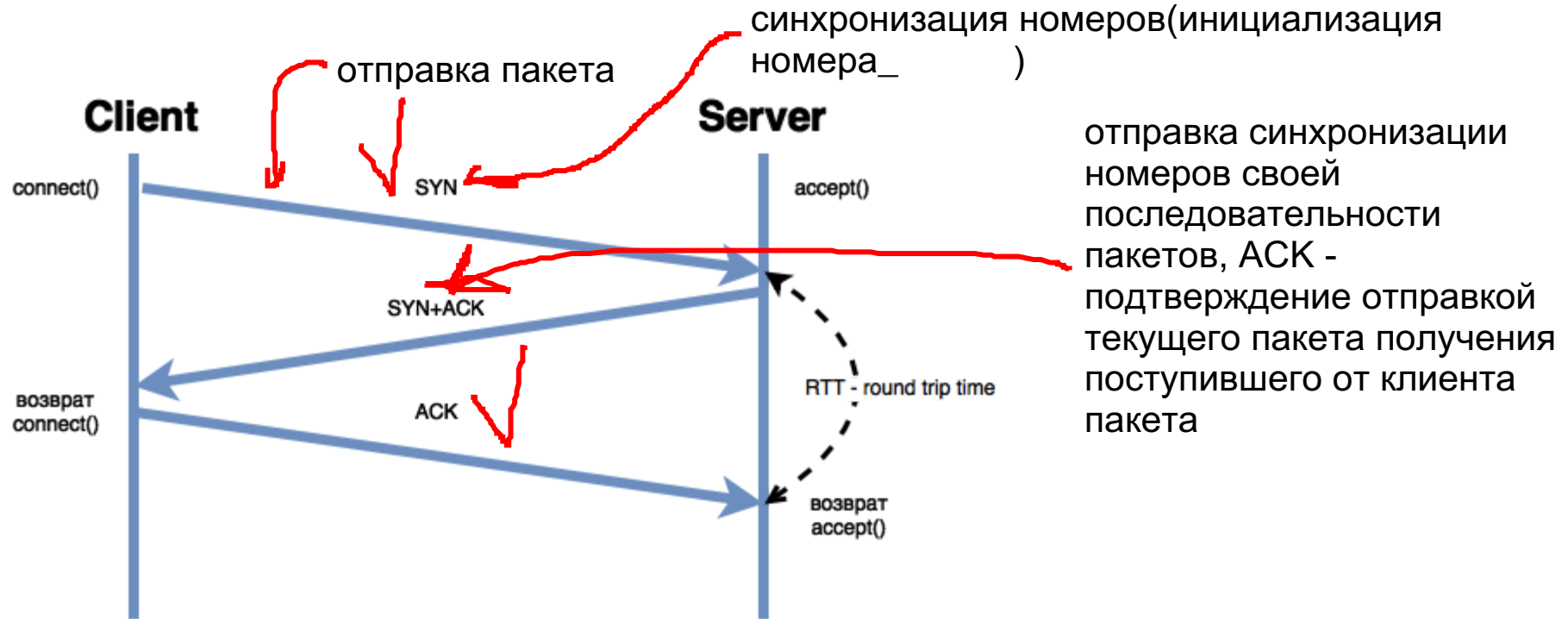
Стандартные TCP порты

- 20,21 - FTP ПРОТОКОЛ ДЛЯ ПЕРЕДАЧИ ФАЙЛОВ
- 22 - SSH УДАЛЕННОЕ УПРАВЛЕНИЕ ОС, ТУННЕЛИРОВАНИЕ TCP-СОЕДИНЕНИЙ
- 25 - SMTP сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.
- 80 - HTTP ПРОТОКОЛ ДЛЯ ПЕРЕДАЧИ HTML-ДОКУМЕНТОВ И ИНЫХ ДАННЫХ В СЕТЯХ TCP/IP
- 443 - HTTPS

расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов TLS или устаревшего в 2015 году SSL

TCP HANDSHAKE - УСТАНОВКА СОЕДИНЕНИЯ, ПОКА БЕЗ ПЕРЕДАЧИ ДАННЫХ

Установка TCP соединения



ВСЕГО ОТПРАВЛЕНО 3
ПАКЕТА

при попытке доступа к серверу со стороны клиента у клиента всегда неявно открывается это
самое соединение на каком-либо порту

Структура заголовка

кол-во соединений между
2-мя машинами в рамках
2-ух конкретных IP - число
портов(2^{16} - src/dest port),
для увеличения числа
возможных соединений
можно повесить несколько
IP адресов

Бит	0 — 3	4 — 9	10 — 15	16 — 31
0	Порт источника, Source Port			Порт назначения, Destination Port
32	Порядковый номер, Sequence Number (SN)			
64	Номер подтверждения, Acknowledgment Number (ACK SN)			
96	Длина заголовка	Зарезервировано	Флаги	Размер Окна
128	Контрольная сумма			Указатель важности
160	Опции (необязательное, но используется практически всегда)			
160/192+	Данные			

Флаги заголовка

- **URG** — поле «Указатель важности»
- **ACK** — поле «Номер подтверждения»
- **PSH** — пуш данных в приложение пользователя
- **RST** — оборвать соединения, сбросить буфер (очистка буфера)
- **SYN** — синхронизация номеров последовательности
- **FIN** — завершение соединения

Пример ТСР клиента

нам нужно создать сокет, подключиться к серверу послать ему данные, принять данные и закрыть соединение

```
import socket
```

```
req = b'Hello tcp!'
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect(('127.0.0.1', 1234))
```

адрес и порт, к которому привязан серверный сокет

```
s.send(req)
```

```
rsp = s.recv(1024)
```

```
s.close()
```

на данном этапе происходит TCP HANDSHAKE

блокировка клиентского сокета

FIN

SYN, ACK

Пример ТСП сервера


```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('127.0.0.1', 1234))
s.listen(10)
while True:
    conn, addr = s.accept()
    while True:
        data = conn.recv(1024)
        if not data: break
        conn.send(data)
    conn.close()
```

привязка серверного сокета к конкретному сетевому интерфейсу("" - любой в рамках системы) + порт

подключений queue

новый сокет и адрес клиента. Именно этот сокет и будет использоваться для приема и посылке клиенту данных.

Как правильно читать данные из сокета ?



ПОСТУПАЕТ ИЗ ПРОТОКОЛА БОЛЕЕ ВЫСОКОГО УРОВНЯ:
HTTP

ДЛИНА СООБЩЕНИЯ

```
def myreceive(sock, msglen):  
    msg = b''  
    while len(msg) < msglen:  
        chunk = sock.recv(msglen-len(msg))  
        if chunk == b'':  
            raise RuntimeError('broken')  
        msg = msg + chunk  
    return msg
```

Как правильно записывать данные в сокет ?

```
def mysend(sock, msg):  
    totalsent = 0  
    while totalsent < len(msg):  
        sent = sock.send(msg[totalsent:])  
        if sent == 0:  
            raise RuntimeError('broken')  
        totalsent = totalsent + sent
```

длина сообщения

все отправляется блоками максимум по 1500 байт, при totalsent == 0

попытаемся отправить все, в sent пойдет успешно отправленная часть msg

кол-во отправленных байт

TLS

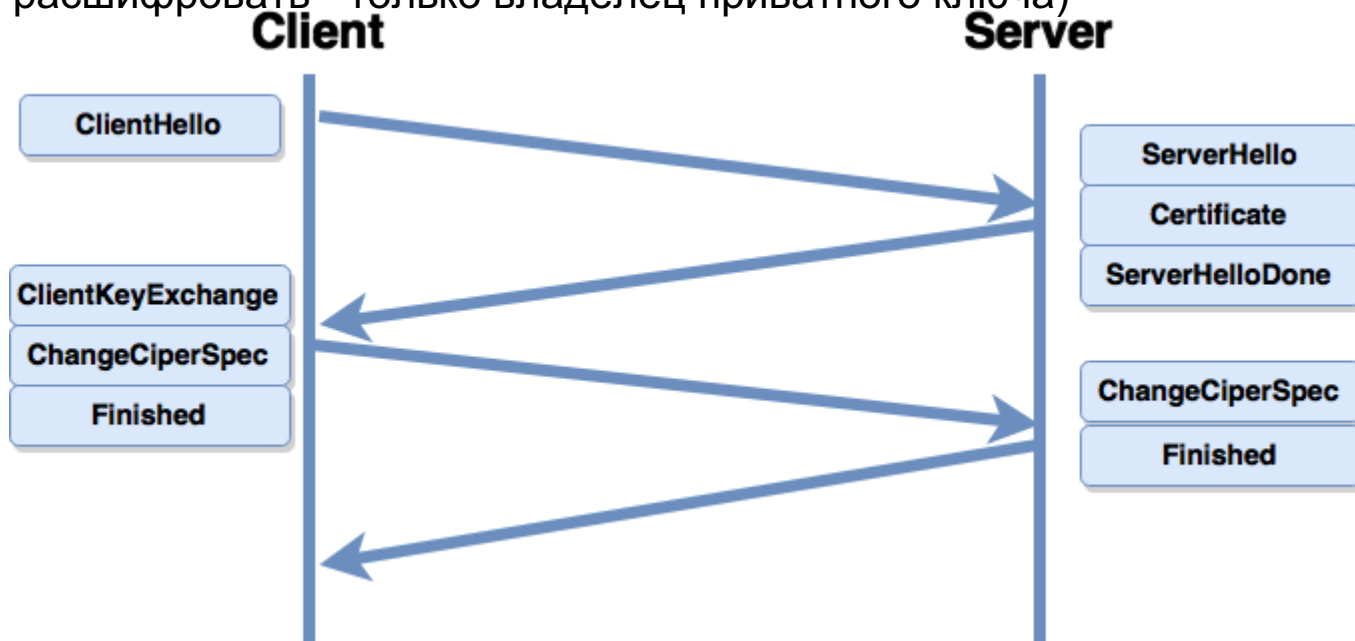
TLS - Transport Layer Security

TLS (а ранее SSL) - криптографический протокол, обеспечивающий безопасную передачу данных между хостами в Internet.

- Аутентификация сервера (и клиента) действительно ли сервер, которому подключаемся, владелец домена
- Шифрование и сжатие передаваемой информации
- Защита от подмены и проверка целостности сообщений

Установление TLS соединения

(домен + TLS сертификат) асимметричное шифрование - на сервере есть приватный и публичный ключи, на клиент отправляется публичный ключ(то есть зашифровать сообщение может кто угодно, расшифровать - только владелец приватного ключа)



- ClientHello - клиент указывает желаемые опции соединения
- ServerHello - сервер подтверждает опции соединения
- Certificate - сервер посылает клиенту свой сертификат
- Клиент проверяет сертификат.

На данном этапе соединение может быть отклонено

- ClientKeyExchange - клиент отправляет серверу ключ симметричного шифрования (или параметры для его генерации)
- Finished - сервер подтверждает завершение рукопожатия

Неутешительный вывод

Установление TCP и TLS соединения требует существенного времени. Минимум 1 RTT для TCP соединения и 1-2 RTT для TLS соединения.

Под RTT понимается Round Trip Time - время, необходимое для передачи IP дейтаграммы к серверу и обратно.