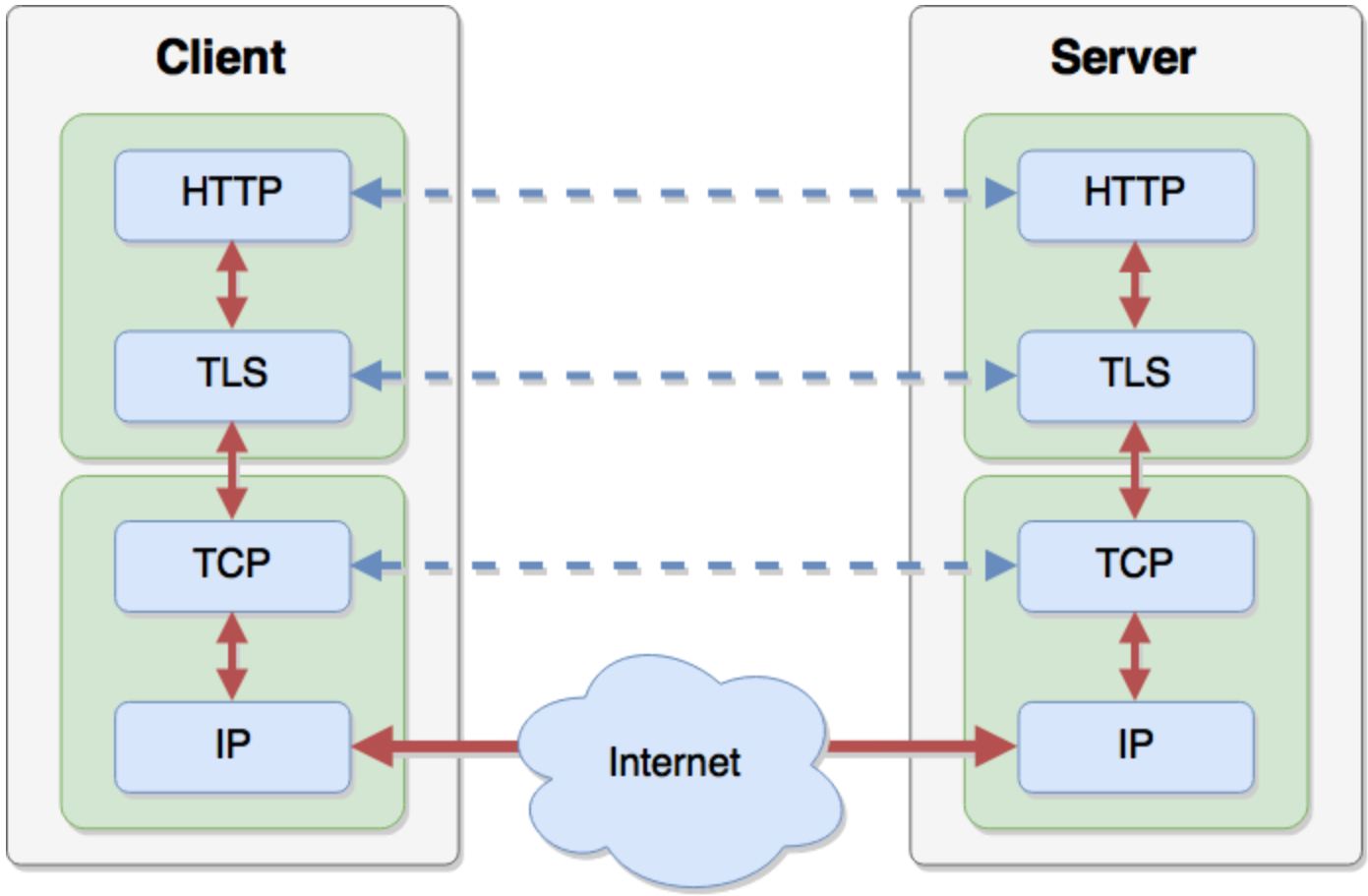
Как происходит HTTP запрос ?

Как происходит HTTP запрос ?

- Браузер анализирует введенный URL и извлекает имя хоста
- Используя систему DNS, браузер преобразует домен в ip адрес
- Устанавливает TCP соединение с web-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 ...

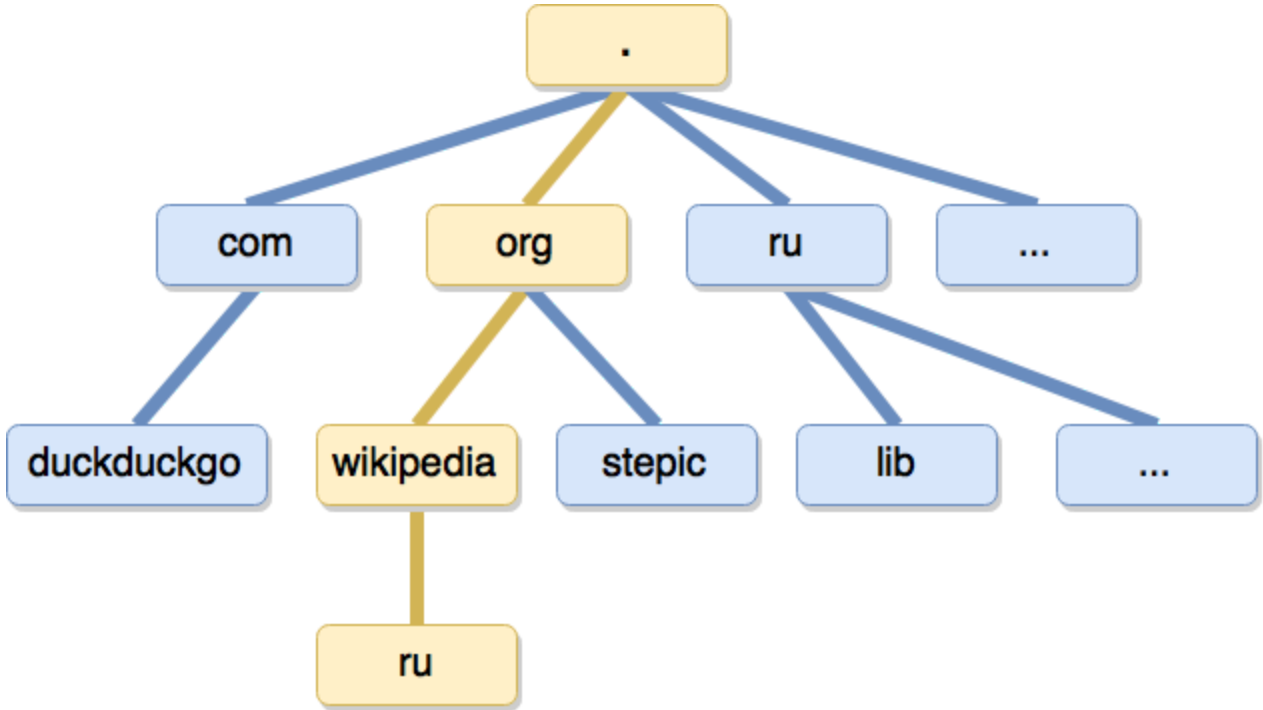

Client
Server
HTTP
HTTP
TLS
TLS
TCP
TCP
IP
Internet
IP

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

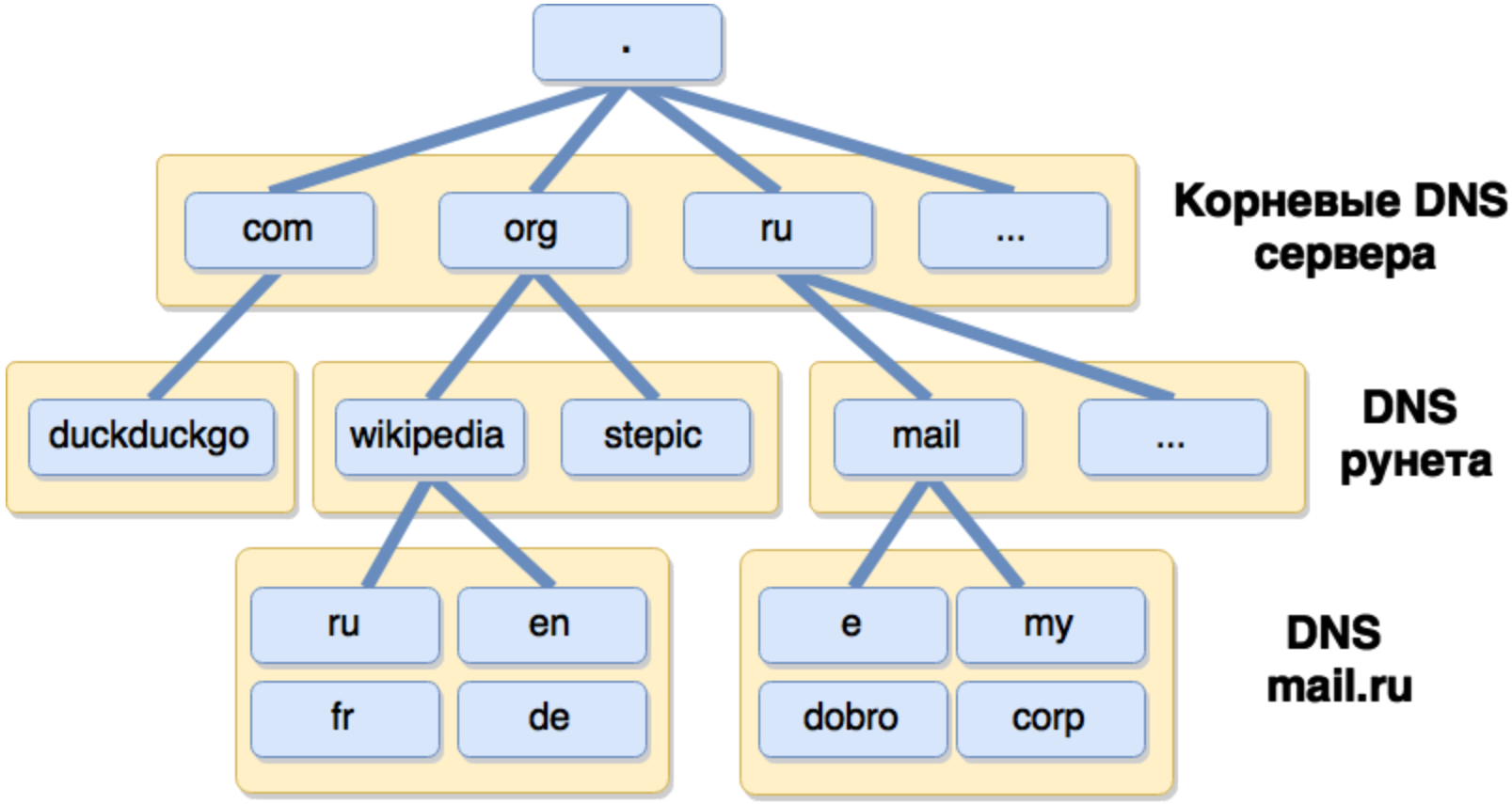
Пространство доменных имен

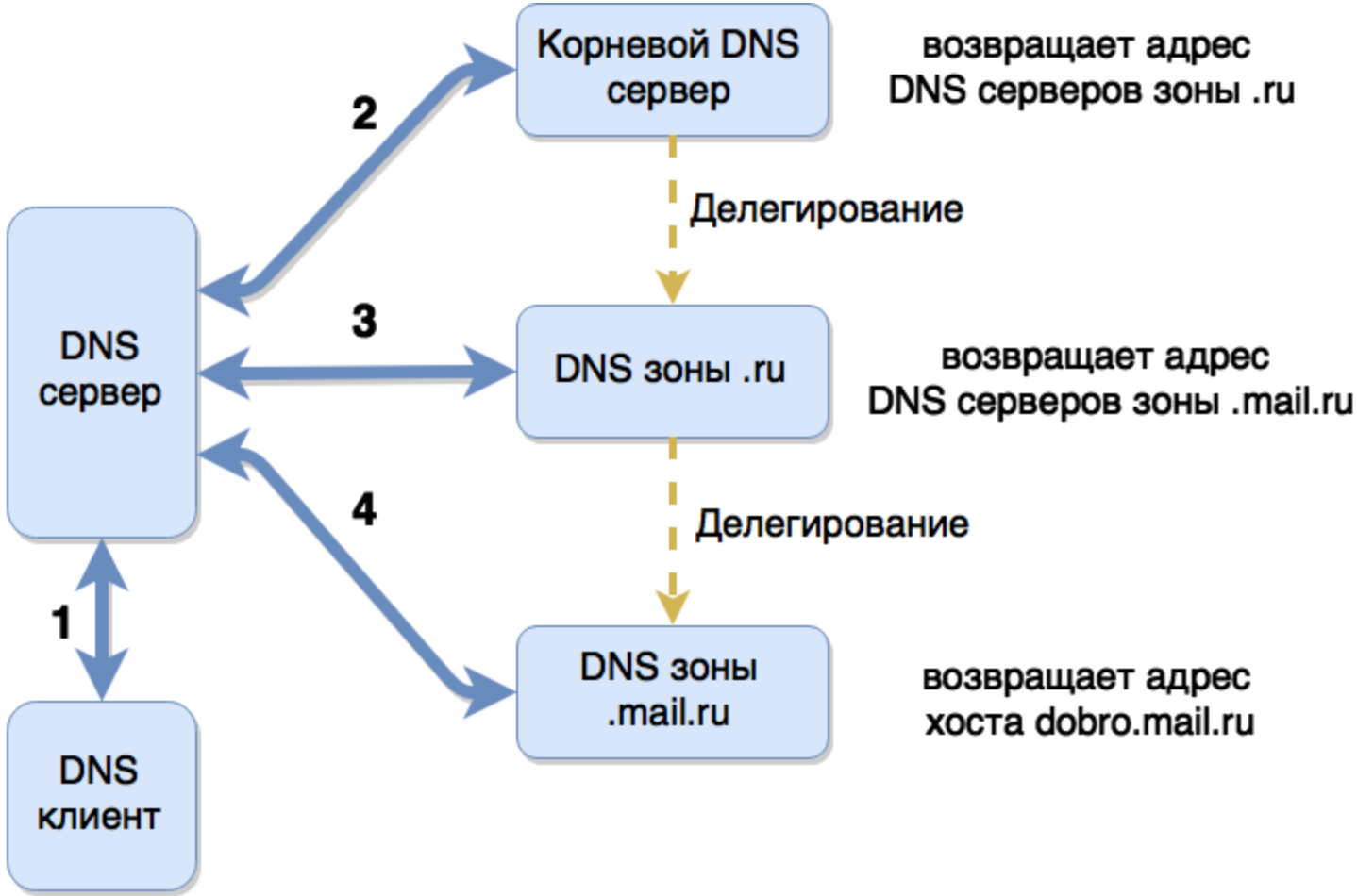


Домены и зоны

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.


.
com
org
ru
...
duckduckgo
wikipedia
stepic
mail
...
ru
en
fr
de
e
my
dobro
corp
Корневые DNS сервера
DNS рунета
DNS mail.ru


Корневой DNS сервер
возвращает адрес DNS серверов зоны .ru
2
Делегирование
DNS сервер
3
DNS зоны .ru
возвращает адрес DNS серверов зоны .mail.ru
4
Делегирование
1
DNS зоны .mail.ru
возвращает адрес хоста dobro.mail.ru
DNS клиент

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
- TXT - текстовая информация о домене

```
$ dig example.ru
;; QUESTION SECTION:
 example.ru.          IN   A

;; ANSWER SECTION:
example.ru.       60   IN   A    5.61.236.163

;; AUTHORITY SECTION:
example.ru.       300  IN   NS   ns2.example.ru.
example.ru.       300  IN   NS   ns1.example.ru.

;; ADDITIONAL SECTION:
ns1.example.ru.   600  IN   A      217.69.139.112
ns1.example.ru.   600  IN   AAAA   2a00:1148:db00::2
ns2.example.ru.   600  IN   A      94.100.180.138
ns2.example.ru.   600  IN   AAAA   2a00:1148:db00::1
```

TCP

Зачем нужен TCP ?

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

- Надежная доставка
- Полнодуплексная передача
- Контроль потока - защита от переполнения

TCP порты

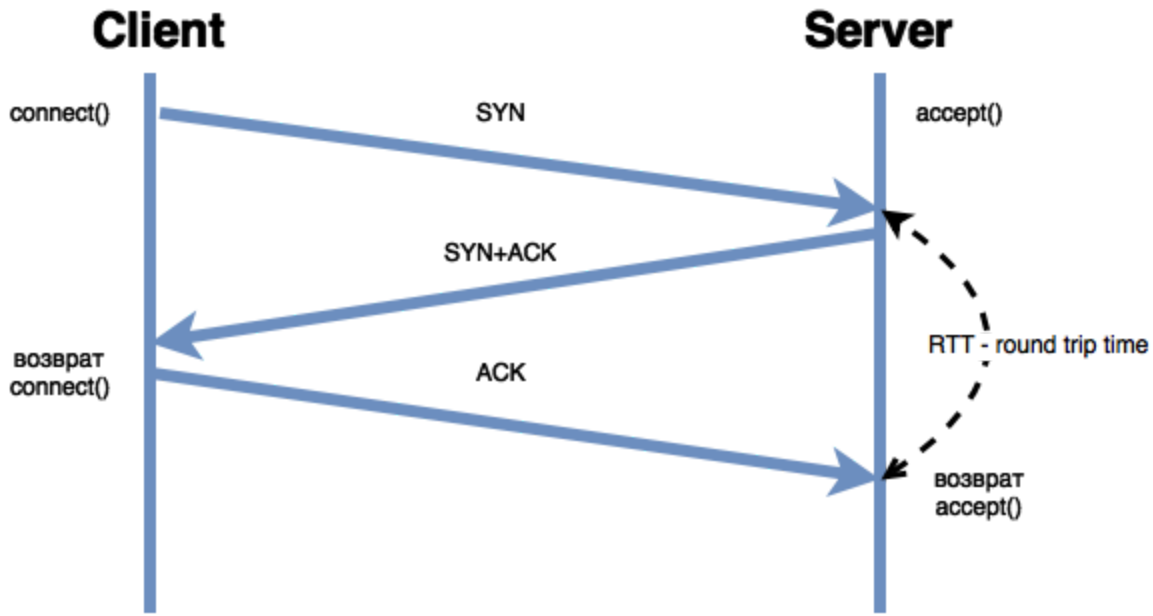
TCP порт - это «адрес» сетевого соединения в пределах одного хоста. TCP порты позволяют поддерживать множество открытых соединений на одной машине.

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

Стандартные TCP порты

- 20, 21 - FTP
- 22 - SSH
- 25 - SMTP
- 80 - HTTP
- 443 - HTTPS

Установление TCP соединения



Структура заголовка

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 TCP клиента

```
import socket
req = b'Hello tcp!'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('127.0.0.1', 1234))
s.send(req)
rsp = s.recv(1024)
s.close()
```

Пример TCP 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()
```

Как правильно читать данные из сокета ?

```
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
```

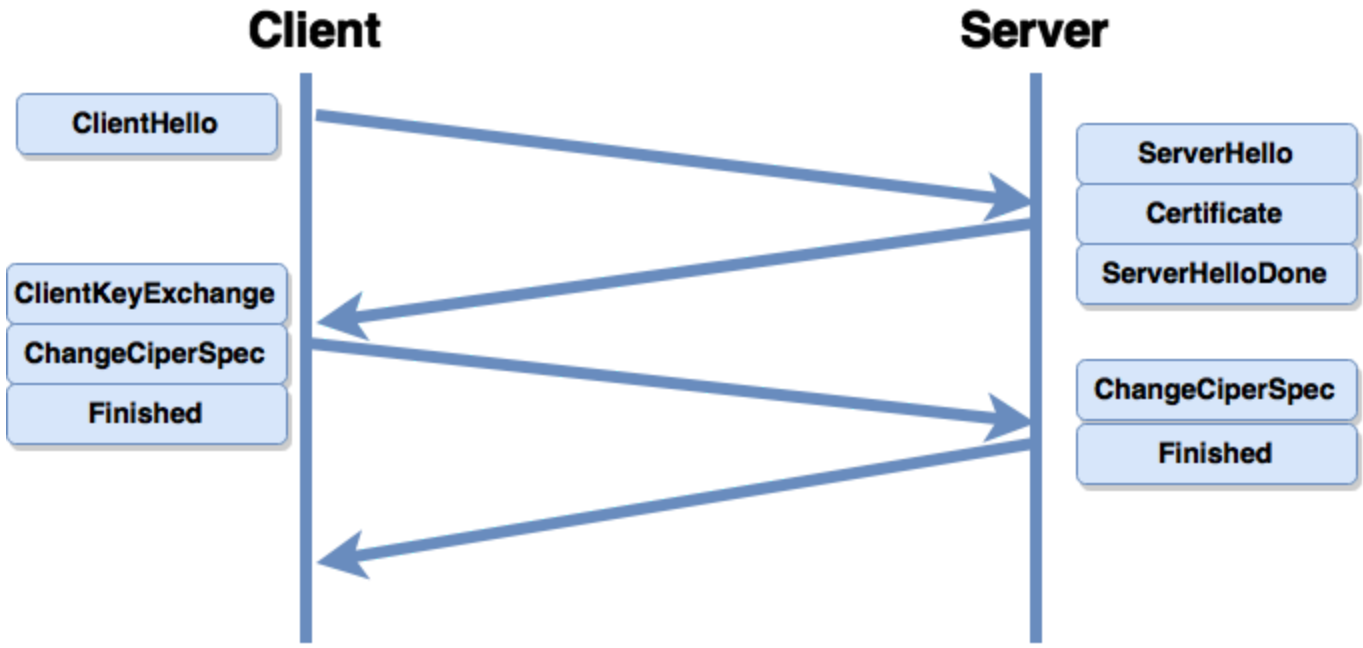
TLS

TLS - Transport Layer Security

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

- Аутентификация сервера (и клиента)
- Шифрование и сжатие передаваемой информации
- Защита от подмены и проверка целостности сообщений

Установление TLS соединения



- 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.