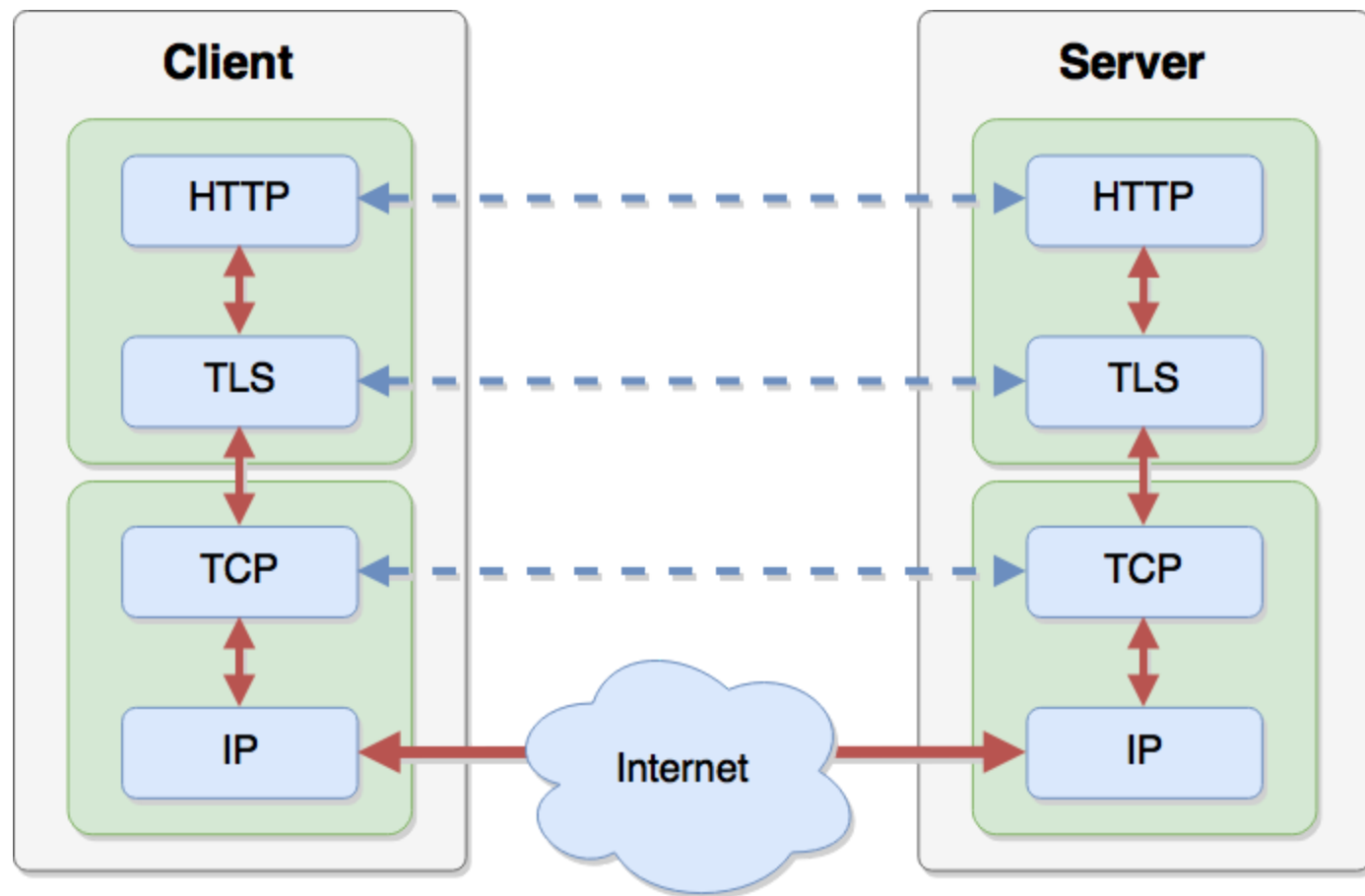


Как происходит  
HTTP запрос ?

# 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 ...

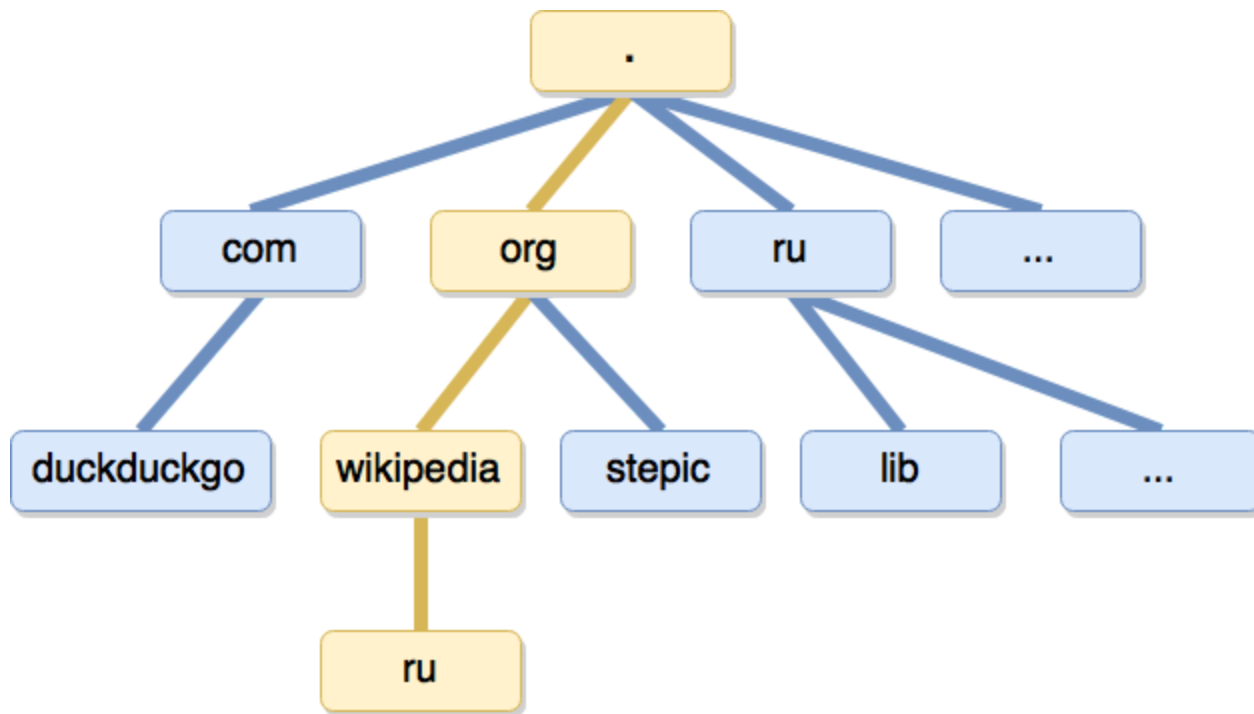


DNS

# 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

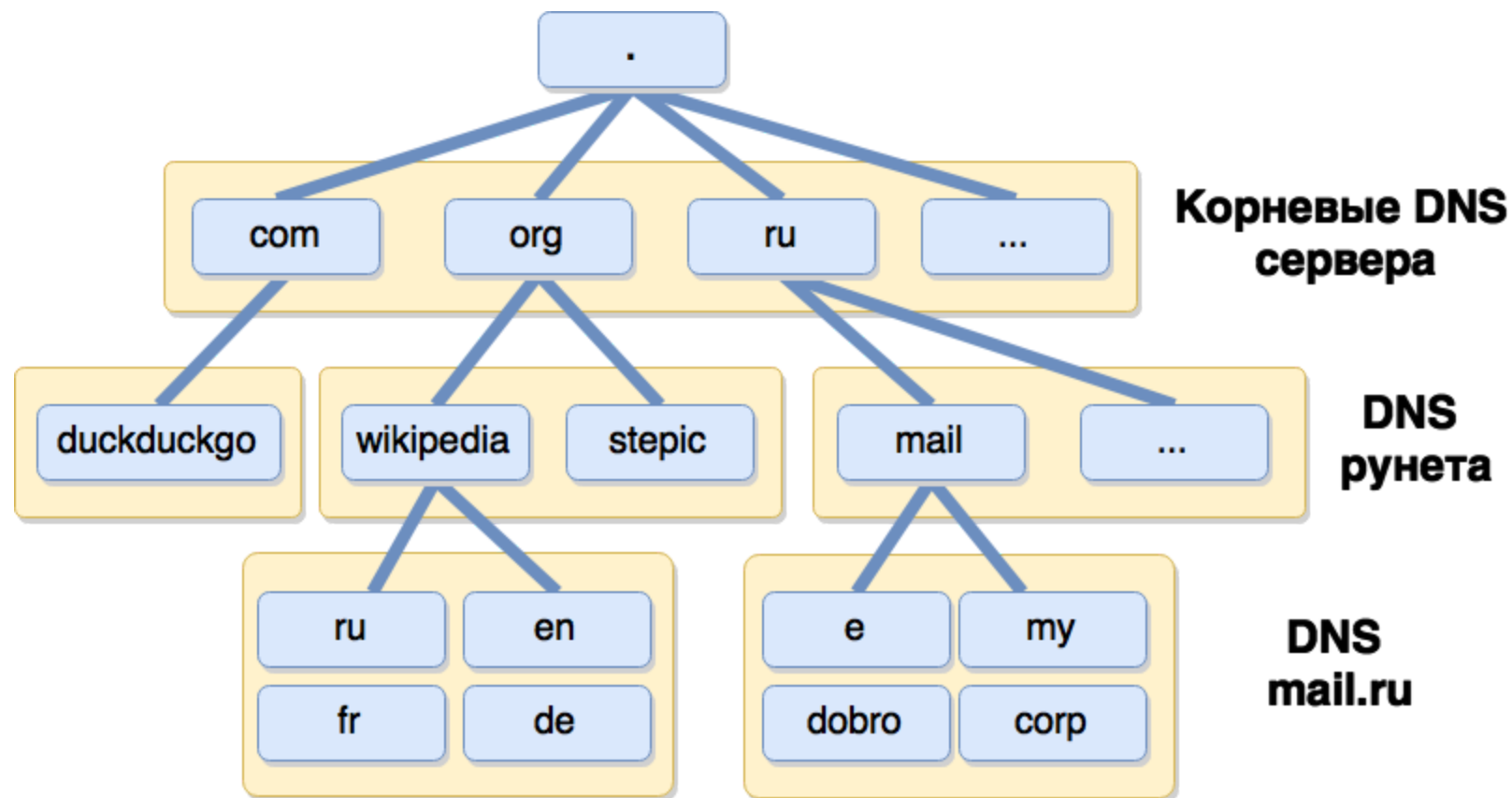
# Пространство доменных имен



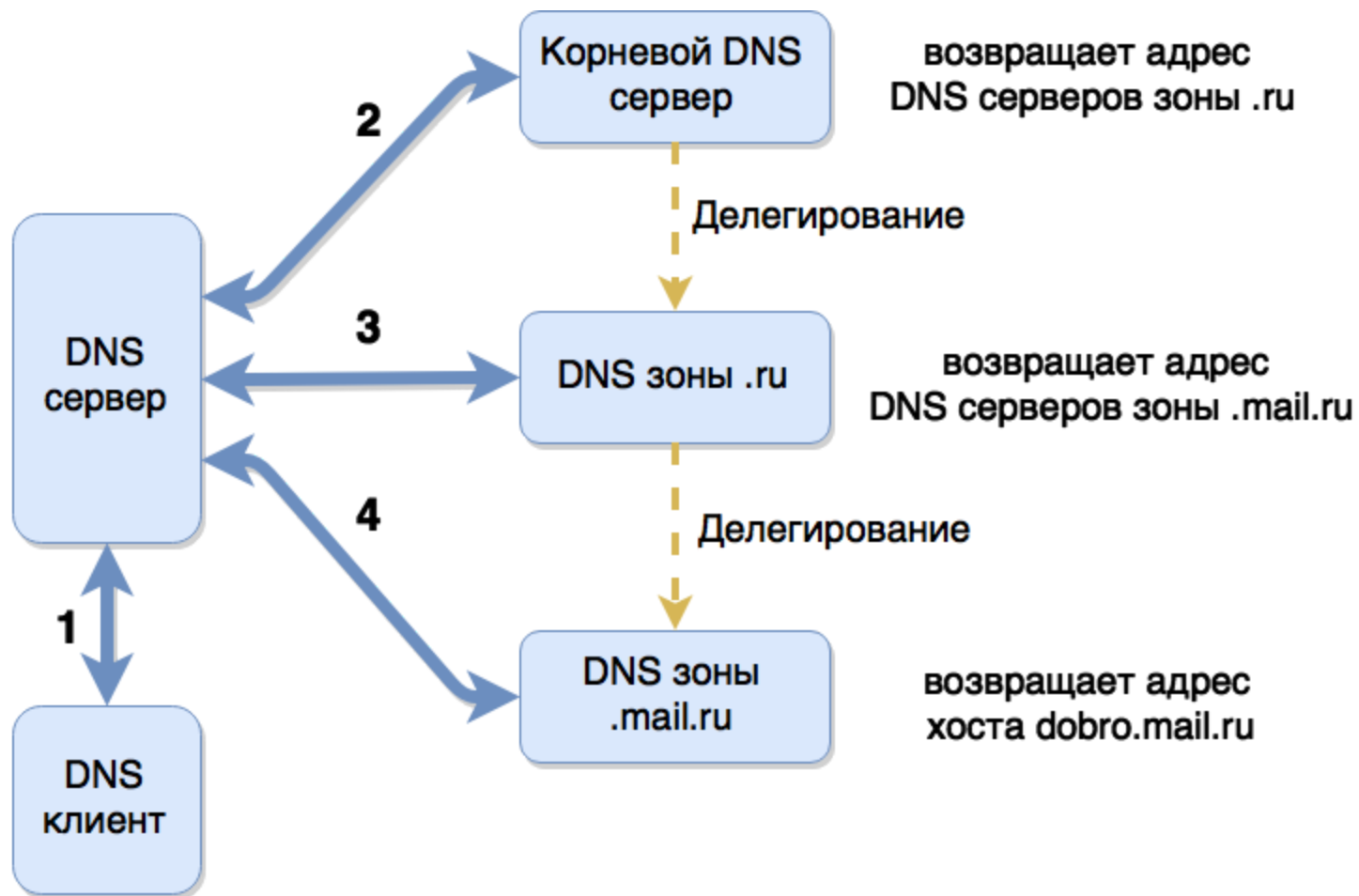
# Домены и зоны

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.

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

# TCP порты

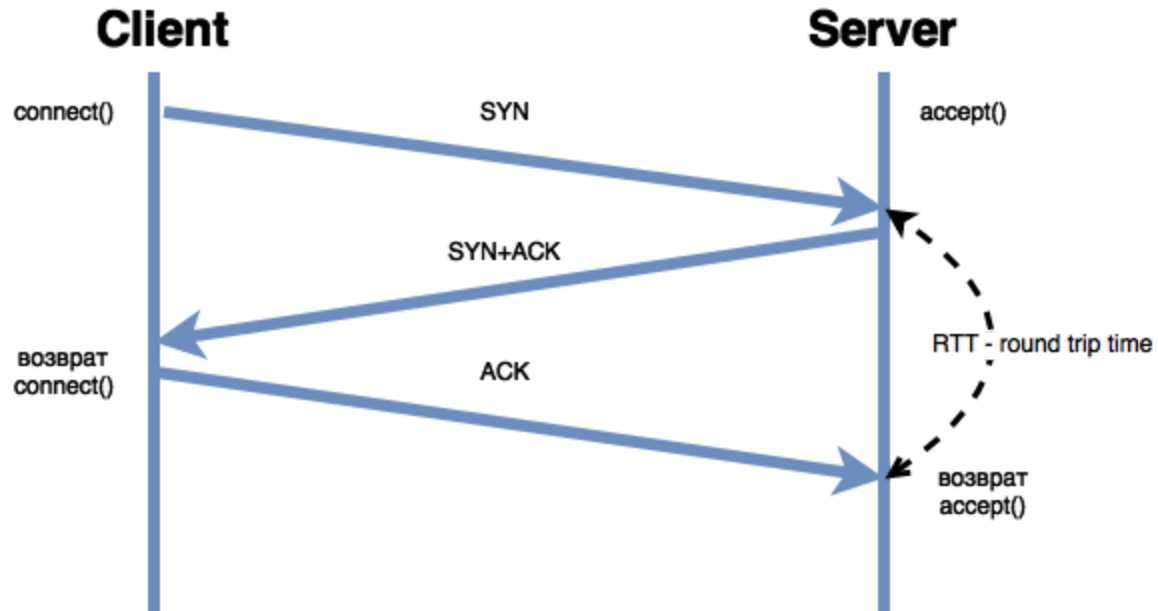
TCP порт - это «адрес» сетевого соединения в пределах одного хоста. TCP порты позволяют поддерживать множество открытых соединений на одной машине.

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, <b>Source Port</b>			Порт назначения, <b>Destination Port</b>
32	Порядковый номер, <b>Sequence Number (SN)</b>			
64	Номер подтверждения, <b>Acknowledgment Number (ACK SN)</b>			
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

```
import socket
req = "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 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()
```

# Как правильно читать данные из сокета ?

```
def myreceive(sock, msglen):  
    msg = ''  
    while len(msg) < msglen:  
        chunk = sock.recv(msglen-len(msg))  
        if chunk == '':  
            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
```

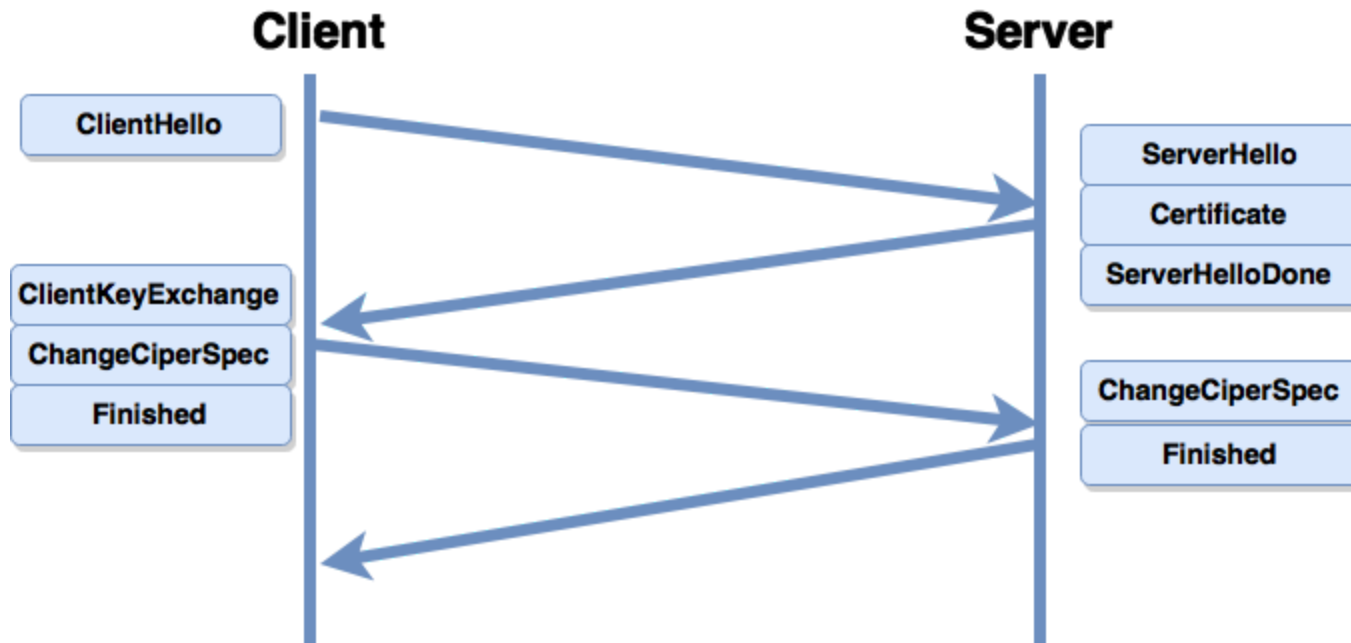
TLS

# 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.