А как гуглить?

Что происходит, когда мы отправляем поисковый запрос в браузере

Представьте: вы налили себе кофе, сели за компьютер и решили, наконец, поменять свою жизнь. Вводите в поисковую Google-строку браузера Chrome фразу: «Зарплаты Python-разработчиков». Ввели, и ждете загрузки страницы.

Что происходит в это время?

1

При нажатии кнопки Enter браузер выполняет преобразование вашего запроса примерно в такой url-адрес:
https://www.google.com/search?q=%D0%B7%D0%B0%D1%80%D0%BF%D0%BB%D0%B0%D1%82%D1%8B+python-%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%BE%D0%B2&oq=%D0%B7%D0%B0%D1%80%D0%BF%D0%BB%D0%B0%D1%82%D1%8B+python-%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%BE%D0%B2&aqs=chrome.0.69i59j0i22i30.936j0j7&sourceid=chrome&ie=UTF-8

В этом адресе можно выделить:

- **протокол взаимодействия:** https
- **домен:** www.google.com
- **путь запроса:** /search
- набор параметров, представленных как ключ=значение, и разделенных амперсандом (&).

Значение ключа **q** — наш поисковый запрос, в котором браузер заранее приводит все кириллические символы из запроса к unicode-стандарту. Т.е. преобразует в кодировку UTF-8, чтобы отвязаться от кодировки пользователя, которая, кстати, видна в параметре **ie**.

Также браузер добавляет некоторую техническую информацию:

- **sourceid** — браузер, из которого был отправлен запрос
- **aqs** — хитрый недокументированный параметр, который расшифровывается как Assisted Query Stats.

2

Каждый домен связан с одним или несколькими IP-адресами, которые представляют собой уникальный номер сетевого компонента в Сети.

IP-адрес можно утрированно представить как адрес дома, сайт — как дом, а домен — как название дома в «народе».

Пример:
Домен — «Дом-сороконожка»,
Адрес — Россия, Москва, Беговая, 34.

Нельзя отправить письмо в «Дом-сороконожку», но можно на Беговую, 34. Поэтому браузеру необходимо узнать, какой IP-адрес скрывается за доменным именем.

Прежде всего, браузер смотрит на настройки пользователя, например, на файл **/etc/hosts** для Linux, где пользователь может руками задать некоторый IP-адрес для домена. Иногда это необходимо для разработки: например, в закрытых сетях, когда нет другой возможности получить IP-адрес сайта.

Далее браузер проверяет: посещался ли этот сайт ранее, и есть ли запись о его IP-адресе в DNS-кэше. У каждого доменного имени есть параметр TTL — *Time To Live*, который обозначает срок хранения кэшированного домена. Если TTL превышен, то браузер удаляет этот домен из кэша и ищет новую информацию у DNS-провайдера.

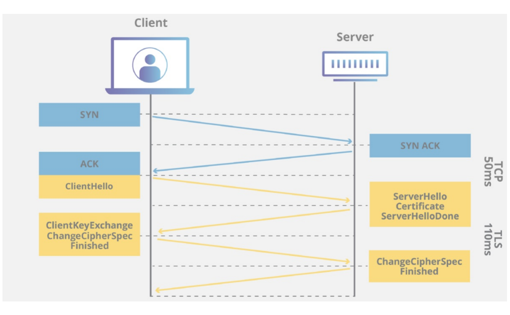
Если адреса в DNS-кэше нет, то браузер обращается к заданным в настройках DNS-провайдерам. DNS расшифровывается как *Domain Name System* и представляет собой ряд серверов, в которых есть таблица сопоставления домена и IP-адреса. Если на выбранном сервере нет информации о домене, то браузер запрашивает информацию у следующего DNS-сервера. И так далее, пока не найдется.

3

Теперь разберемся, зачем нужна буква **s** на конце привычного http-протокола. Она показывает необходимость в настройке протокола шифрования между клиентом и сервисом. Для этого используется протокол *TLS* или устаревший *SSL*, то есть https = HTTP + SSL(TSL).

TLS и SSL предполагают под собой асимметричное шифрование, главная цель которого — не передавать ключ к расшифровке по сети, поэтому для его настройки используется «TLS-рукопожатие».

Подробнее почитать о TLS можно тут.



4

На картинке с TLS-рукопожатием видно три голубых блока, рядом с которыми написано TCP. Это *Transmission Control Protocol,* который используется для установки канала для передачи сообщений. Для его настройки применяется *«трехстороннее»* рукопожатие: на картинке это рукопожатие как раз показано тремя голубыми блоками. TCP-канал гарантирует доставку сообщений до адресата ровно в том порядке, в котором они были отправлены.

Например, вы отправили сообщение другу, а какой-нибудь маршрутизатор отключился и потерял его. Тогда подтверждение получения сообщения от друга не придет, и вы отправите сообщение повторно с тем же номером.

В разговоре про DNS мы упомянули протокол *UDP* — это неряшливый, но быстрый брат TCP. Он не устанавливает соединение, а сразу идет к цели, не тратя времени на всякие формальности. Если вы проводите банковскую транзакцию, то лучше положиться на более медленный, но надежный TCP. А если, например, стримите видео, то потеря одного кадра не так важна, как общая скорость стрима — поэтому лучше выбрать UDP.

Итак, при установке TLS уже было выполнено трехстороннее TCP-рукопожатие, и теперь у нас есть надежный канал связи.

5

Мы готовы к формированию HTTP-запроса. HTTP расшифровывается как *Hyper Text Transfer Protocol,* то есть протокол для передачи гипертекста. В самой часто используемой на сегодняшний день версии этого протокола HTTP/1.1 информация представлена в текстовом виде, то есть ее можно прочитать не только роботам, но и обычным человекам. Наш запрос к Google преобразовался в такой вид (для наглядности параметры запроса обрезаны):

```
GET /search?sourceid=chrome&ie=UTF-8 HTTP/1.1
Host: www.google.com
User-Agent: Chrome/90.0.4430.85
Accept-Language: en-GB,en;q=0.9,ru-RU;q=0.8,ru;q=0.7,en-US;q=0.6,la;q=0.5
…
```

- **GET** — это метод HTTP-запроса, о чем пойдет речь в следующих карточках. После указан наш путь с передаваемыми параметрами, а затем версия протокола: HTTP/1.1.
- **Host** — домен. Это то то, куда мы должны отправить наш запрос.

В следующих строках идет ряд *заголовков.* Эта специальная информация, которая передается в виде *ключ/значение.*

- **User-Agent** — заголовок, с помощью которого мы говорим сайту, с какого браузера и устройства отправили запрос. Чтобы, например, открыть мобильную версию сайта, если зашли с телефона.
- **Accept-Language** — заголовок, с помощью которого говорим, что хотим только русско- и англоязычные сайты в поисковой выдаче. О заголовках мы еще поговорим дальше.

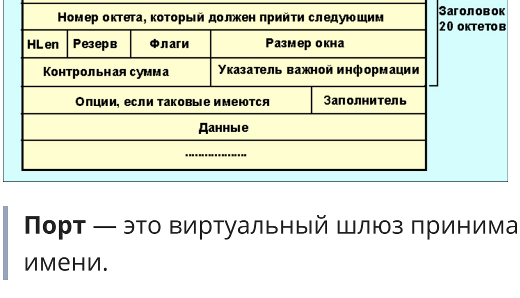
6

Помните, на этапе установки защищенного соединения был настроен TCP-канал с помощью трехстороннего рукопожатия?

Чтобы воспользоваться этим каналом для отправки HTTP-запроса, необходимо преобразовать запрос в *TCP-пакет.* Для этого HTTP-данные преобразуются в бинарный вид и кладутся в поля для данных TCP-пакета.

В TCP-пакете уже указаны:

- его порядковый номер относительно созданного TCP-канала
- контрольная сумма
- порт, на который должен прийти запрос



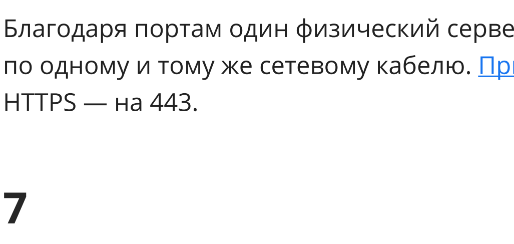
> **Порт** — это виртуальный шлюз принимающей стороны, который обычно пишется в конце доменного имени.

Продолжая аналогию с IP-адресом как адресом дома, порт можно сравнить с номером квартиры, в которой могут обработать отправляемый запрос. Каждый порт можно назначить какому-нибудь сервису. Например, порт 80 отдать под ответы на HTTP-запросы, а порт 587 — под отправку электронной почты.

Благодаря портам один физический сервер может обрабатывать множество протоколов, приходящих по одному и тому же сетевому кабелю. Принято, что протокол HTTP работает на порту 80, а, например, HTTPS — на 443.

7

Далее необходимо упаковать TCP-пакет в IP-пакет, добавив данные о своем IP-адресе и об IP-адресе получателя. Данные TCP-пакета также кладутся в место для данных в IP-пакете.



8

После этого IP-пакет инкапсулируется в *Ethernet-кадр,* в котором указан физический MAC-адрес устройства, которому адресована эта информация.

9

После этого Ethernet-пакет в виде световых сигналов или радиоволн побитово передается по сети и доходит до получателя с помощью сложной системы хабов, коммутаторов и маршрутизаторов.

> Процесс превращения HTTP-запроса в набор физических сигналов называется *инкапсуляцией.*

10

Получив запрос, сервер Google выполняет обратное преобразование набора световых пучков или электрических сигналов в HTTP-запрос, повторяя всю последовательность шагов с 5 по 8, но в обратном направлении, выполняя *деинкапсуляцию.* По полученному тексту запроса веб-сервис выполняет необходимую работу и формирует данные для ответа, в нашем случае — список результатов поиска по запросу «зарплаты Python-разработчиков».

11

Создается HTTP-ответ, который выглядит примерно таким образом:

```
HTTP/1.1 200 OK
Server: gws
Date: Sun, 10 Oct 2021 23:26:07 GMT
Accept-Ranges: bytes
Content-Type: text/html; charset=UTF-8
Content-Length: 12
<html>
…
```

На первой строке указана версия протокола (HTTP/1.1), статус ответа (200) и текстовое описание статуса (OK). О статусах ответа мы поговорим позже.

Далее идет ряд заголовков, несущих информацию, например, о типе данных в ответе (Content-Type) и об их длине (Content-Length).

12

Происходит точно такая же инкапсуляция HTTP-ответа, как в случае HTTP-запроса, но уже на сервере Google, и ответ отправляется обратно к клиенту.

13

Браузер пользователя обрабатывает HTTP-ответ и, используя свой графический движок, отрисовывает html-разметку. Мы видим страницу с поисковой выдачей.

Такой путь проходит HTTP-запрос. Не пугайтесь его сложности: обычно для работы backend-разработчика достаточно понимания протокола HTTP — это покрывает практически все задачи, которые могут возникнуть в процессе разработки.

Далее мы более детально рассмотрим самые важные в разработке вещи, касающиеся сетевого взаимодействия в целом, и HTTP-протокола в частности.