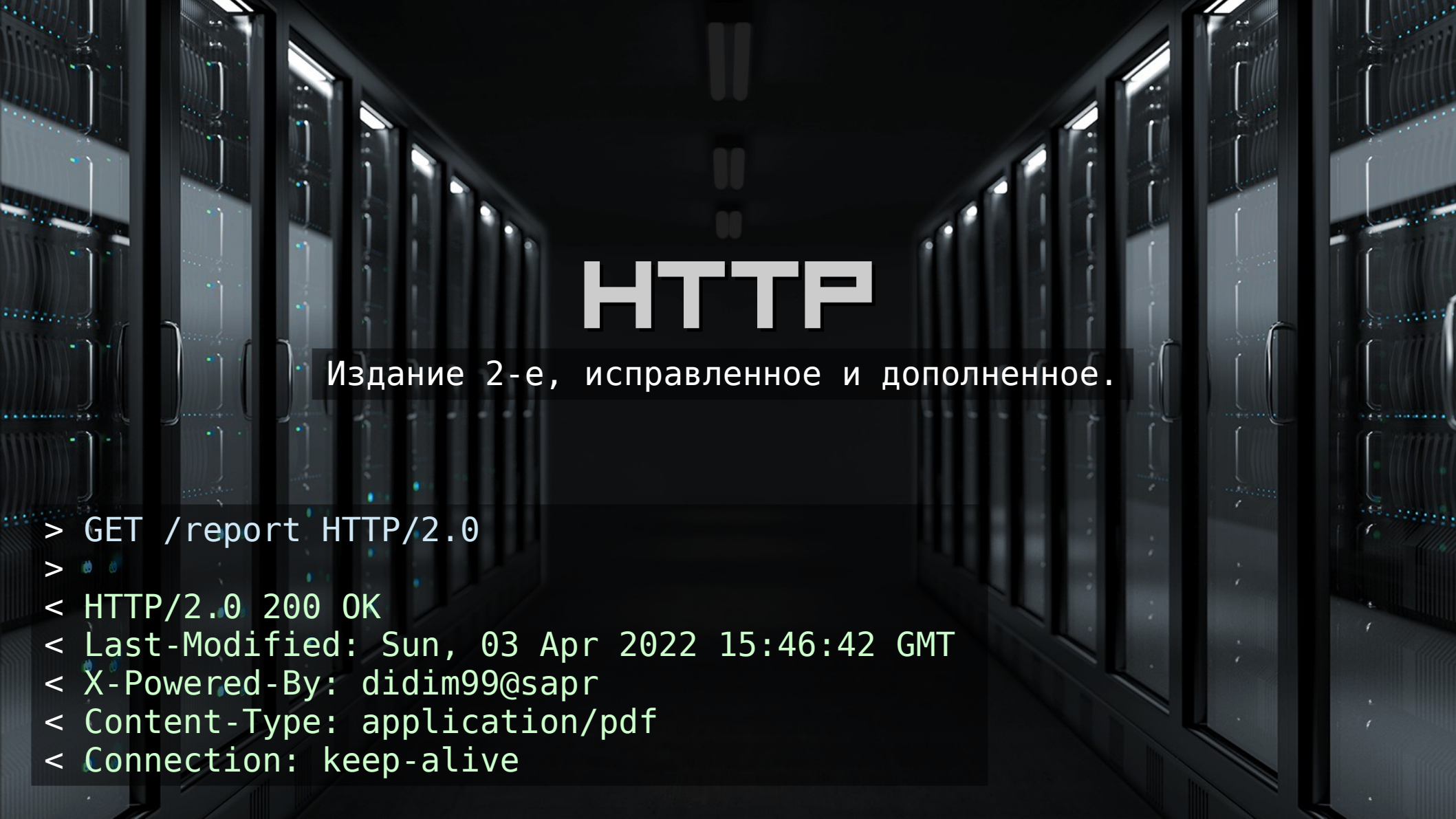


HTTP

Издание 2-е, исправленное и дополненное.

```
> GET /report HTTP/2.0
>
< HTTP/2.0 200 OK
< Last-Modified: Sun, 03 Apr 2022 15:46:42 GMT
< X-Powered-By: didim99@sapr
< Content-Type: application/pdf
< Connection: keep-alive
```

GET /report HTTP/2.0

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol — «протокол передачи гипертекста») — протокол передачи данных прикладного уровня

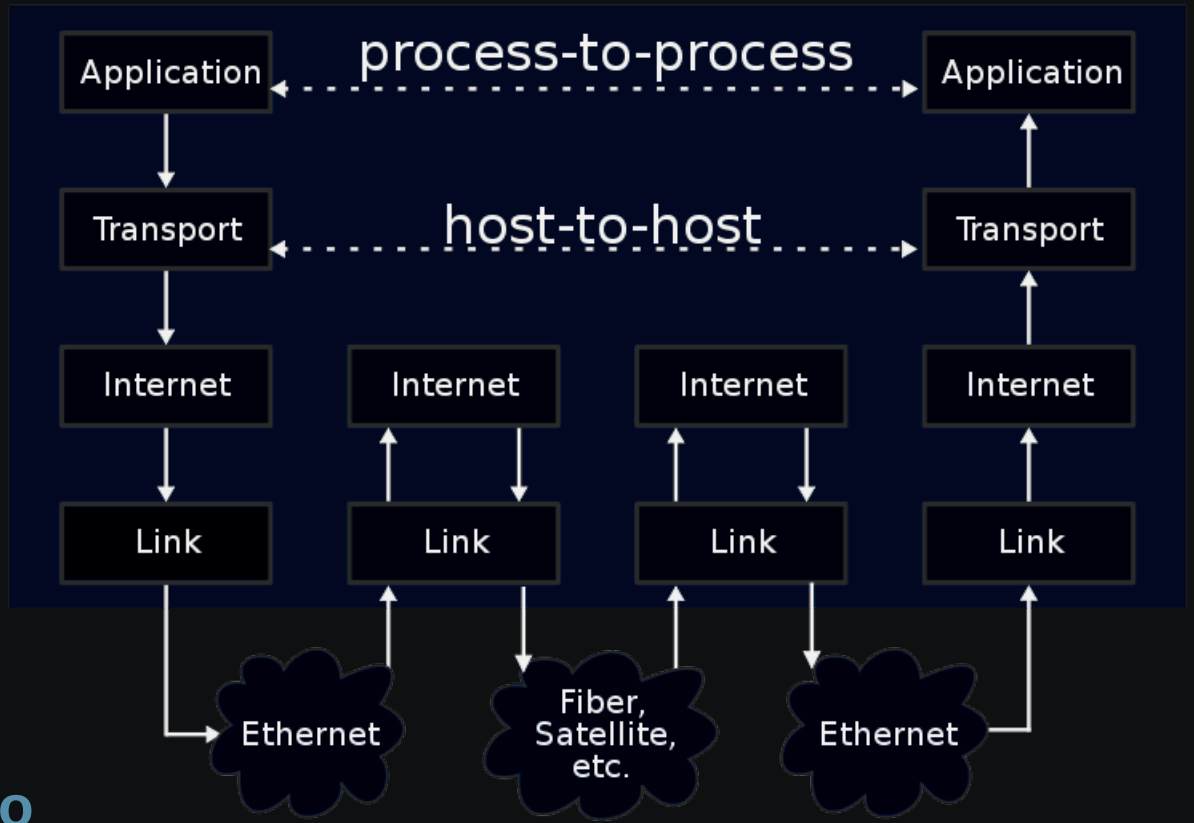
Прикладного уровня чего?

HTTP/2.0 200 OK


СЕТЕВЫЕ
ТЕХНОЛОГИИ
ЗА 5 МИНУТ

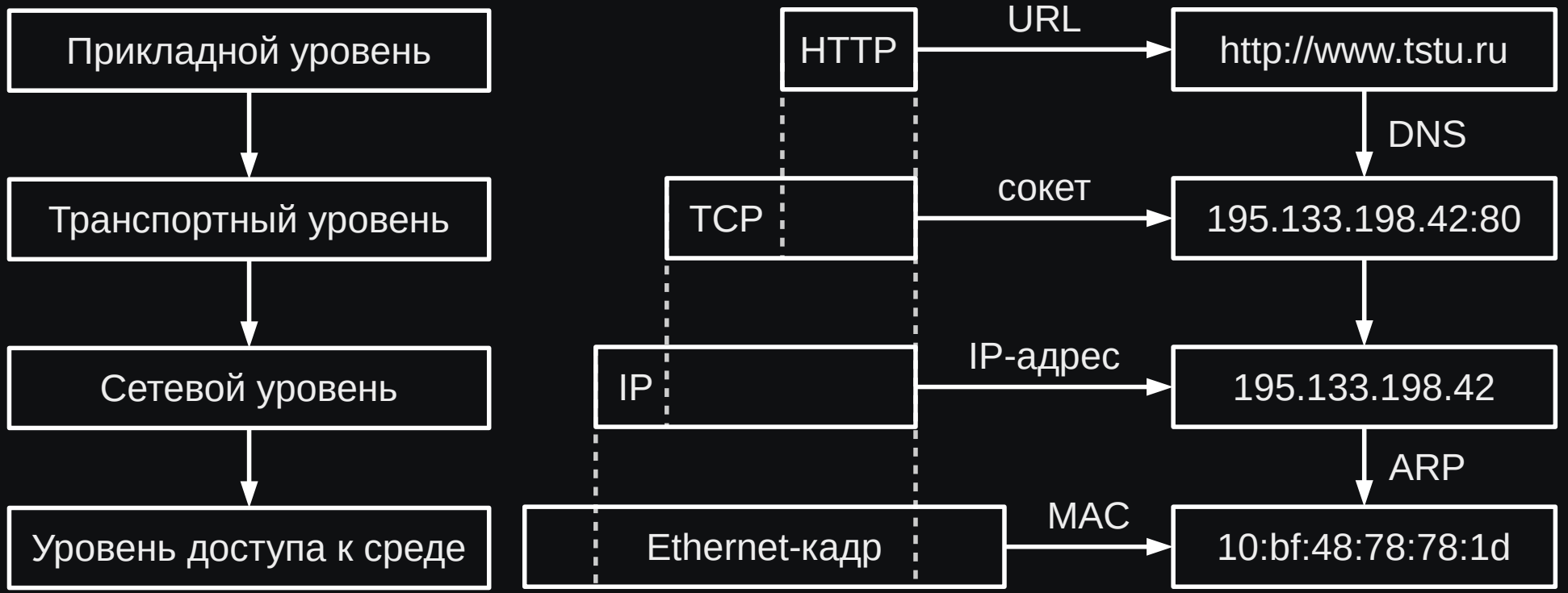
Как все это работает?

OSI? — Нет, TCP/IP!



OSI: Интернет, которого не было
статья на Хабре

Как все это работает?




GO BACK...

История

CERN

IETF

1992 — HTTP/0.9
1996 — HTTP/1.0
1999 — HTTP/1.1
2015 — HTTP/2.0
2018 — HTTP/3.0

HTTP

URI: что там в адресной строке?

Синтаксис:

```
protocol://host[:port]/path[?query=string][#hash]
```

`protocol` — наименование протокола
`host[:port]` — адерс узла и порт для подключения
`/path` — путь к файлу на сервере, всегда начинается со «/»
`?query=string` — дополнительные параметры запроса
`#hash` — якорь, используется только на стороне клиента

Структура Query String:

```
?key=value&[key2[]=value2][&key2[]=value3]...
```

```
http://des.tstu.ru:8103/izo/ways.html?s=090301#scroll3
```

Структура запроса

Стартовая строка:

```
Метод URI HTTP/Версия
```

- `OPTIONS` — получить список возможностей сервера
- `GET` — получить содержимое ресурса
- `HEAD` — получить заголовки ресурса
- `POST` — отправить данные на сервер

```
> GET / HTTP/1.1
> Host: tstu.ru
> User-Agent: curl/7.47.0
> Accept: */*
>
```

Секция заголовков:

```
Имя-Заголовка: значение заголовка
```

Пустая строка

Тело запроса (опционально): произвольные данные

Структура ответа

Стартовая строка:

```
HTTP/Версия код статус
```

- 1xx Информационные
- 2xx Успех
- 3xx Перенаправление
- 4xx Ошибка клиента
- 5xx Ошибка сервера

```
< HTTP/1.1 301 Moved Permanently
< Date: Sun, 26 Apr 2020 21:56:08 GMT
< Server: Apache/2.4.38 (DilOS)
< Location: https://tstu.ru/
< Content-Length: 296
< Content-Type: text/html;
charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD
HTML 2.0//EN">
...
```

Секция заголовков:

```
Имя-Заголовка: значение заголовка
```

Пустая строка

Тело ответа: произвольные данные

Заголовки

Основные

Connection — сведения о проведении соединения.

Date — дата генерации отклика.

Transfer-Encoding — список способов кодирования, которые были применены к сообщению для передачи.

Content-Disposition — Способ распределения сущностей в сообщении при передаче нескольких фрагментов.

Заголовки

Заголовки запроса

Accept — список допустимых форматов ресурса.
Accept-Encoding — перечень поддерживаемых способов кодирования содержимого при передаче (gzip, deflate, ...).
Cookie — передача на сервер сохраненного значения Cookie.
Host — доменное имя и порт хоста запрашиваемого ресурса. Необходимо для поддержки виртуального хостинга на серверах.
Referer — URI ресурса, после которого клиент сделал текущий запрос.
User-Agent — список названий и версий клиента и его компонентов с комментариями.

Заголовки

Заголовки ответа

Accept-Ranges — перечень единиц измерения диапазонов.
Content-Length — размер содержимого сущности в байтах.
Content-Type — формат и способ представления сущности.
Last-Modified — дата последней модификации сущности.
Location — URI по которому клиенту следует перейти или URI созданного ресурса.
Server — список названий и версий веб-сервера и его компонентов с комментариями.
Set-Cookie — параметры записи Cookie для сохранения на стороне клиента

Коды ответа

200 OK: успешный запрос
301 Moved Permanently: запрошенный документ был окончательно перенесен на новый URI
400 Bad Request: сервер обнаружил в запросе клиента синтаксическую ошибку
401 Unauthorized: для доступа к запрашиваемому ресурсу требуется аутентификация
403 Forbidden: сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента
404 Not Found: сервер понял запрос, но не нашёл соответствующего ресурса по указанному URL
500 Internal Server Error: любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок

Передача данных методом POST

Тип данных определяется заголовком:
`Content-Type`

- `application/x-www-form-urlencoded` — данные кодируются аналогочно тому, как это делается для Query String
- `multipart/form-data` — каждое значение посылается как блок данных «body part», с заданными пользовательским клиентом разделителем
- `text/plain` — данные не подвергаются какому-либо кодированию, а передаются «как есть»

Передача данных методом POST

```
Content-Type: multipart/form-data;boundary="boundary"

--boundary
Content-Disposition: form-data; name="key"

value
--boundary
Content-Disposition: form-data; name="myfile";
filename="example.txt";
Content-Type: text/plain

A line of text in the file example.txt
--boundary--
```

Сохранение состояния: Cookie

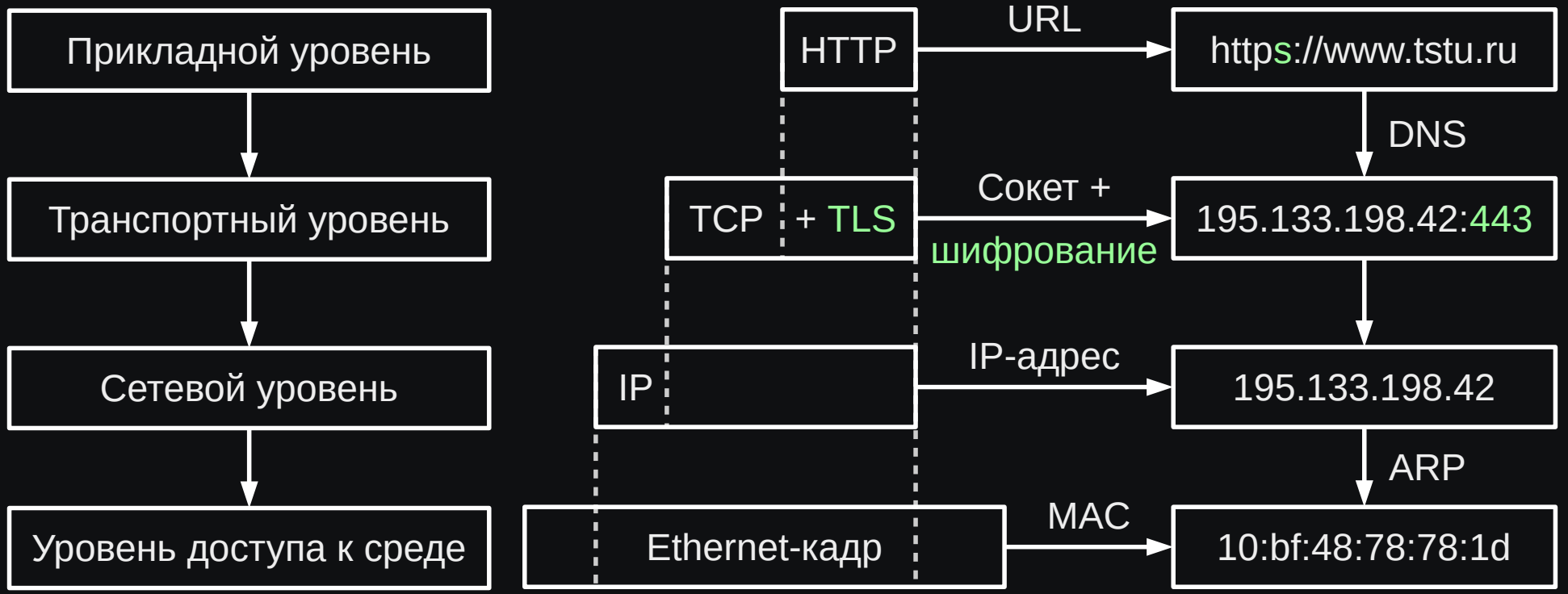
Синтаксис:

```
параметр=значение; [параметр=значение;]...
```

Параметры:
name — устанавливает имя cookie.
value — сохраняет значение cookie.
expires и **max-age** — определяют срок жизни cookie, по истечению которого она будет автоматически удалена.
path — указывает путь к директории на сервере, для которой будут доступны cookie.
domain — отмечает, какой домен или поддомен имеет доступ к этой cookie.


НЕМНОГО ОБ
HTTPS

Откуда берётся безопасность?



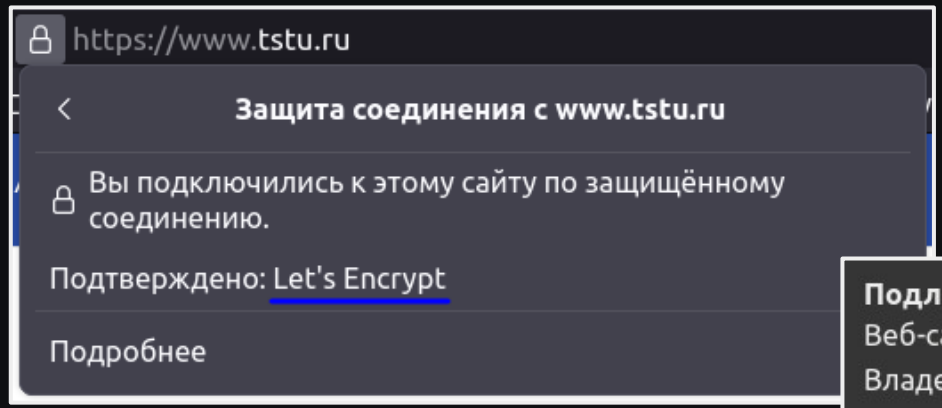
Как работает TLS?

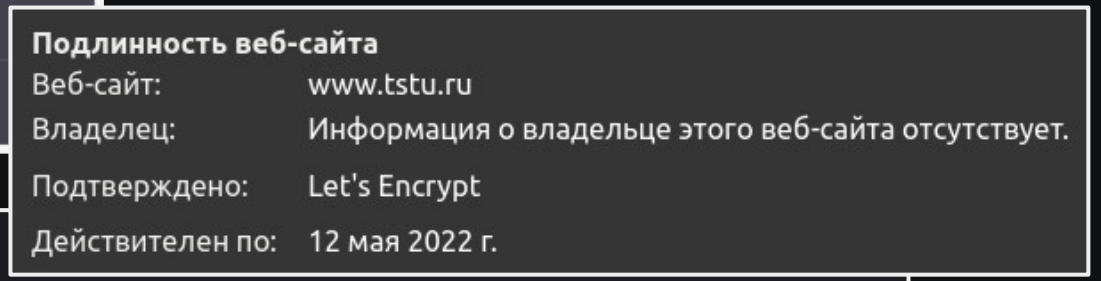
Transport **L**ayer **S**ecurity

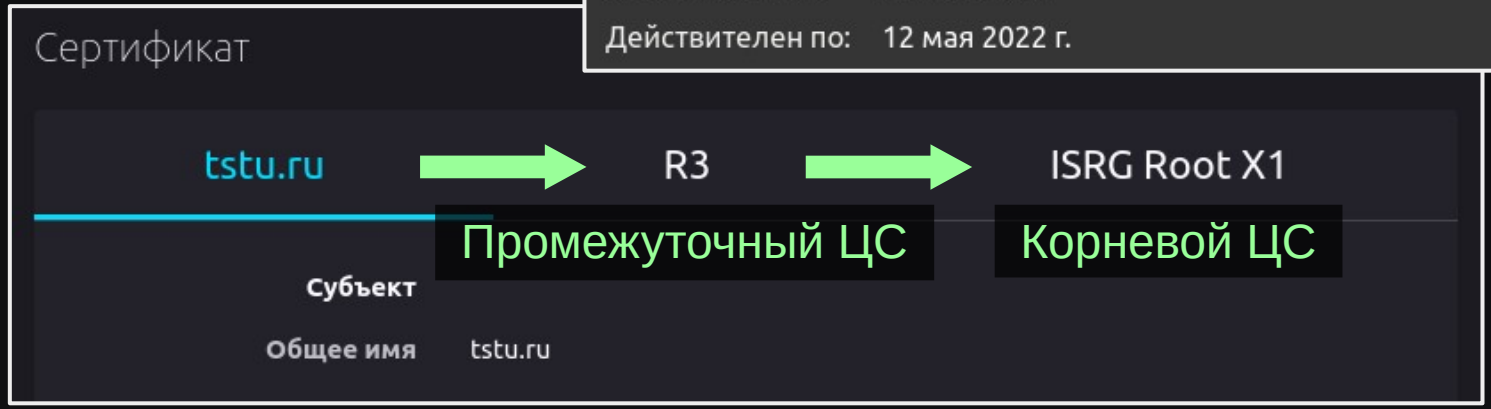
Асимметричное шифрование (RSA) для аутентификации;
Симметричное шифрование (AES) для конфиденциальности;
Коды аутентичности сообщений (HMAC) для сохранения целостности сообщений.

Установка TLS-соединения
статья на Вики

Сертификаты в SSL/TLS







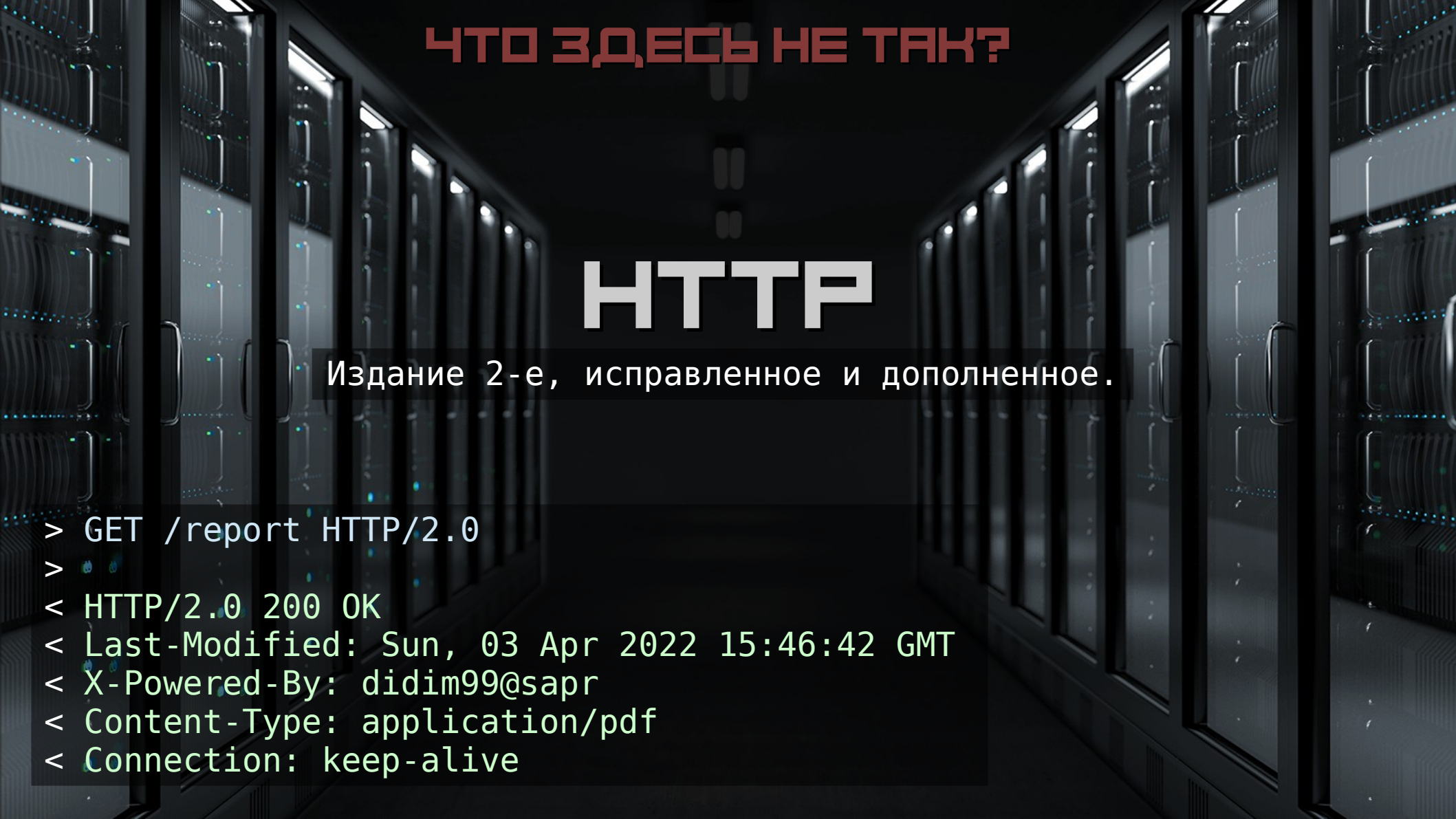

ОТ SPDY
К HTTP2

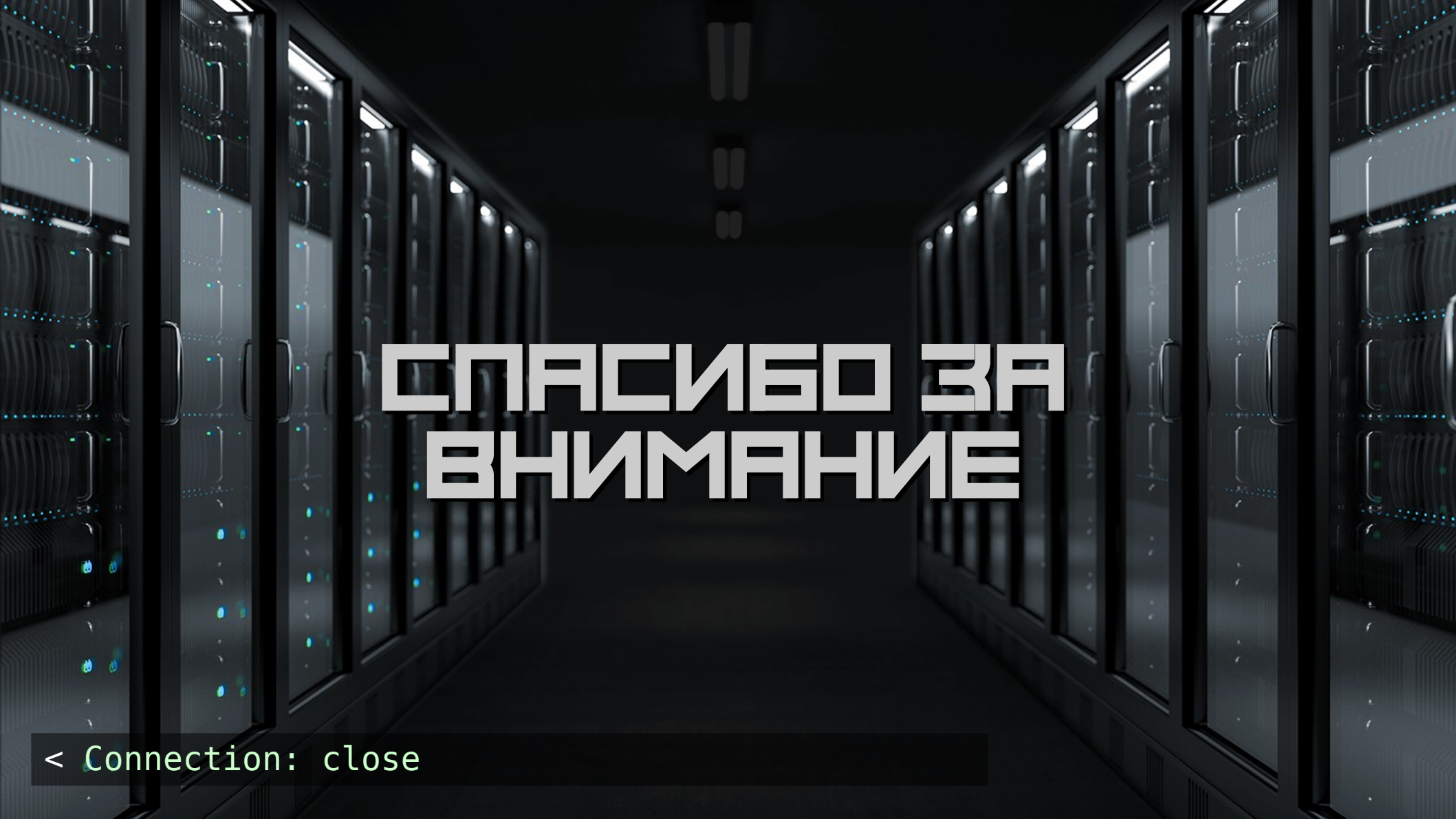
Новые возможности

- Протокол является бинарным
- Реализован механизм согласования протокола (может быть использован HTTP/1.1, HTTP/2 и возможно иные протоколы).
- Сервер имеет право послать то содержимое, которое ещё не было запрошено клиентом (push-отправка).
- Поддерживается сжатие данных в заголовках
- Поддерживается мультиплексирование нескольких транзакций в рамках одного TCP-соединения
- Поддерживается приоритезация запросов
- Расширенные возможности управления потоками

Критика

- Протокол спроектирован и сделан в Google
- Протокол полезен только для браузеров и больших сервисов
- Бинарный протокол — это неудобно
- Использование TLS делает его медленным


ЧТО ЗДЕСЬ НЕ ТАК?
HTTP
Издание 2-е, исправленное и дополненное.
> GET /report HTTP/2.0
>
< HTTP/2.0 200 OK
< Last-Modified: Sun, 03 Apr 2022 15:46:42 GMT
< X-Powered-By: didim99@sapr
< Content-Type: application/pdf
< Connection: keep-alive


СПАСИБО ЗА
ВНИМАНИЕ
< Connection: close