Протокол HTTP/2

Зачем и как использовать в Perl

Владимир Леттиев

История протокола

- HTTP/0.9 (1991)

  HTTP-протокол был предложен Тимом Бернерсом-Ли как основа для Всемирной Сети WWW для передачи текстовых гипер-документов

История протокола

- HTTP/0.9 (1991)

  HTTP-протокол был предложен Тимом Бернерсом-Ли как основа для Всемирной Сети WWW для передачи текстовых гипер-документов

```
$ telnet server 80
GET /index.html

<html>...</html>
Connection closed by foreign host.
```

История протокола

- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)

  Информативный стандарт HTTP/1.0, принятый в 1996 году, оказался компиляцией де-факто используемых на практике расширений HTTP/0.9.

История протокола

- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)

  Информативный стандарт HTTP/1.0, принятый в 1996 году, оказался компиляцией де-факто используемых на практике расширений HTTP/0.9.

```
$ telnet server 80
GET / HTTP/1.0
User-Agent: Perl

HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 100500

<html>...</html>
Connection closed by foreign host.
```

История протокола

- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)
- HTTP/1.1 **RFC 2068** (1997), **RFC 2616** (1999),

  Официальный стандарт интернета через несколько месяцев после HTTP/1.0. Основные новшества: повторное использование соединений, поддержка кэширования, потоковая передача и другие возможности.

История протокола

- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)
- HTTP/1.1 **RFC 2068** (1997), **RFC 2616** (1999),

  Официальный стандарт интернета через несколько месяцев после HTTP/1.0. Основные новшества: повторное использование соединений, поддержка кэширования, потоковая передача и другие возможности.

```
$ telnet server 80
GET / HTTP/1.1
Host: server
User-Agent: Perl

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 100500
Connection: keep-alive

<html>...</html>

...
```

История протокола

- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)
- HTTP/1.1 **RFC 2068** (1997), **RFC 2616** (1999),
- HTTP/1.1 **RFC 7230-7235** (2014)

  Созданная в 2007 году рабочая группа HTTPbis, только к 2014 выработала финальную серию из 6 RFC, фиксирующая состояние HTTP/1.1

История протокола

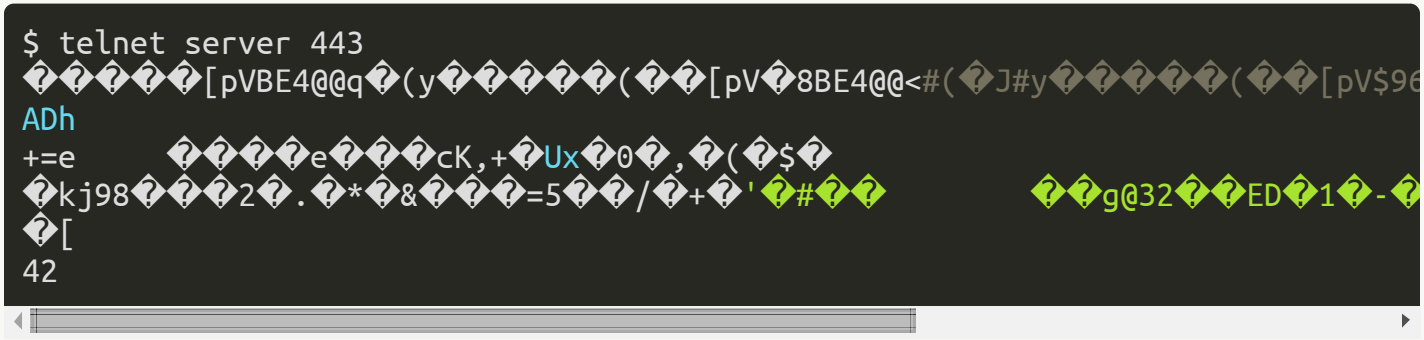
- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)
- HTTP/1.1 **RFC 2068** (1997), **RFC 2616** (1999),
- HTTP/1.1 **RFC 7230-7235** (2014)
- HTTP/2 **RFC 7540** (2015)

  Параллельно с фиксацией HTTP/1.1 группа HTTPbis предложила новую версию протокола HTTP/2 на основе протокола SPDY, разработанного в Google.

История протокола

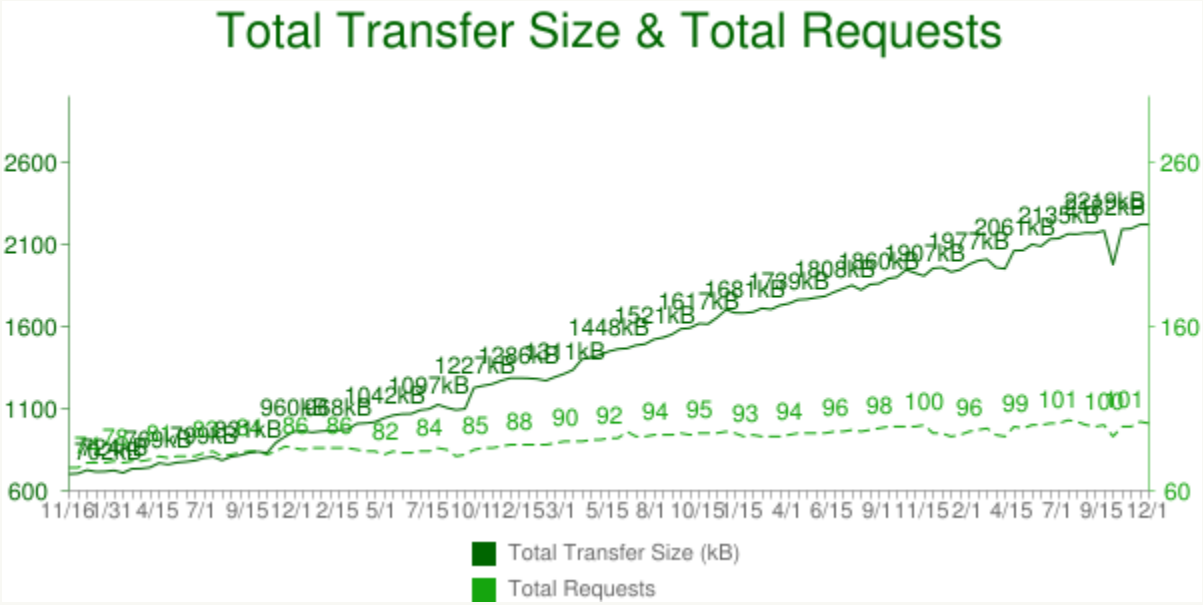
- HTTP/0.9 (1991)
- HTTP/1.0 **RFC 1945** (1996)
- HTTP/1.1 **RFC 2068** (1997), **RFC 2616** (1999),
- HTTP/1.1 **RFC 7230-7235** (2014)
- HTTP/2 **RFC 7540** (2015)

  Параллельно с фиксацией HTTP/1.1 группа HTTPbis предложила новую версию протокола HTTP/2 на основе протокола SPDY, разработанного в Google.



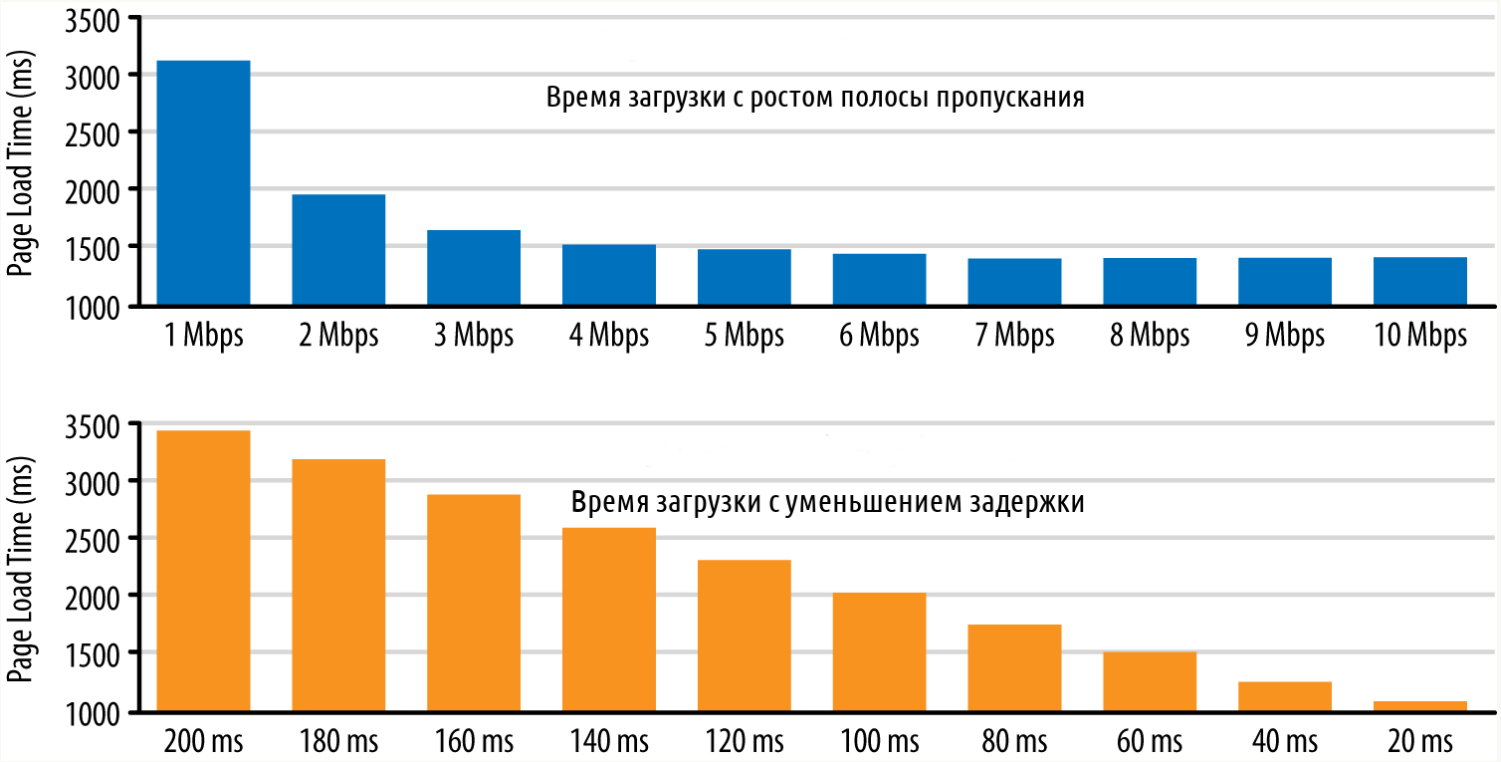
Тренды

Средний размер сайта и количество запросов



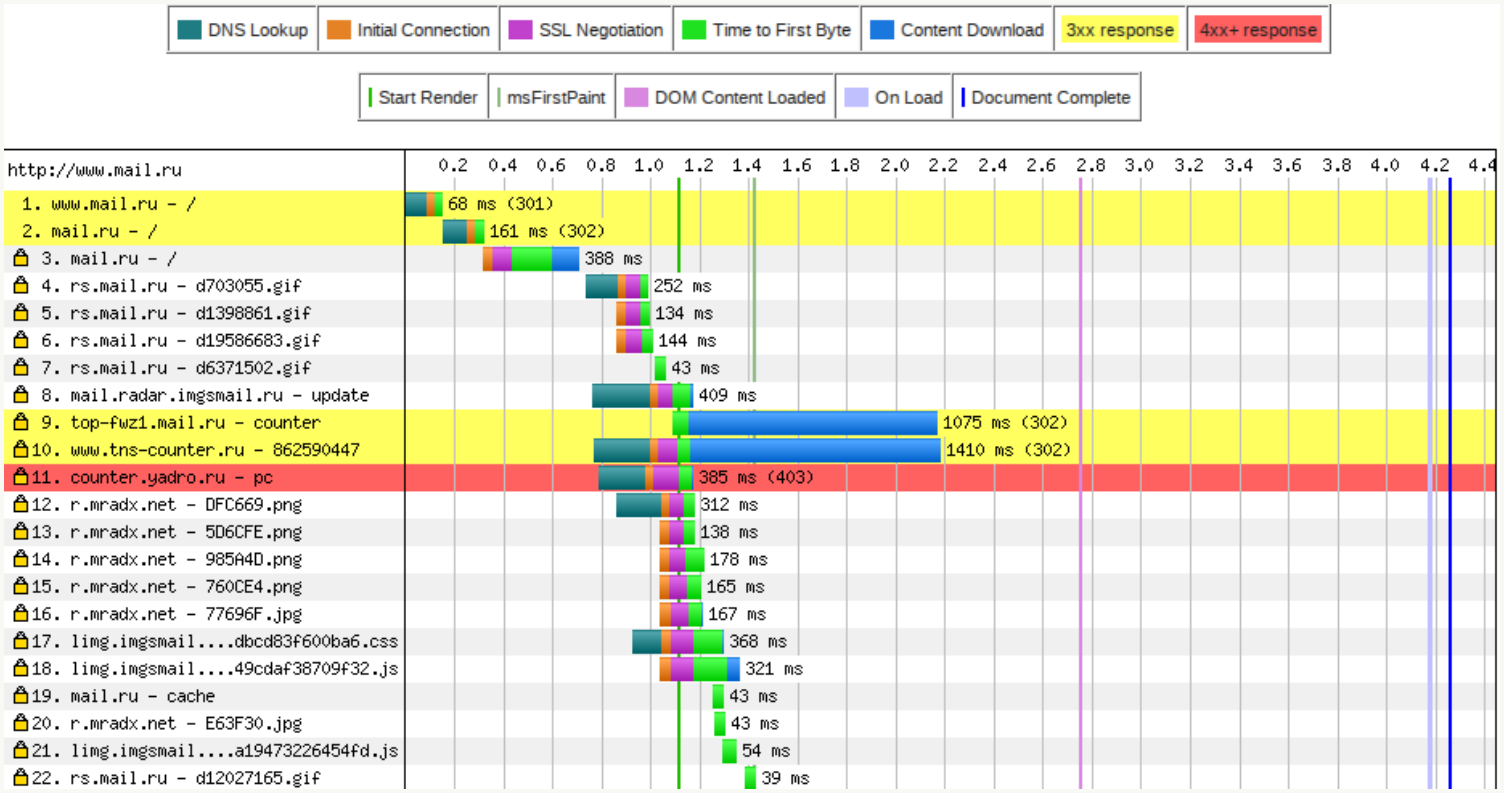
Данные **httparchive.org** за 2010-2015 гг.

RTT vs Bandwidth



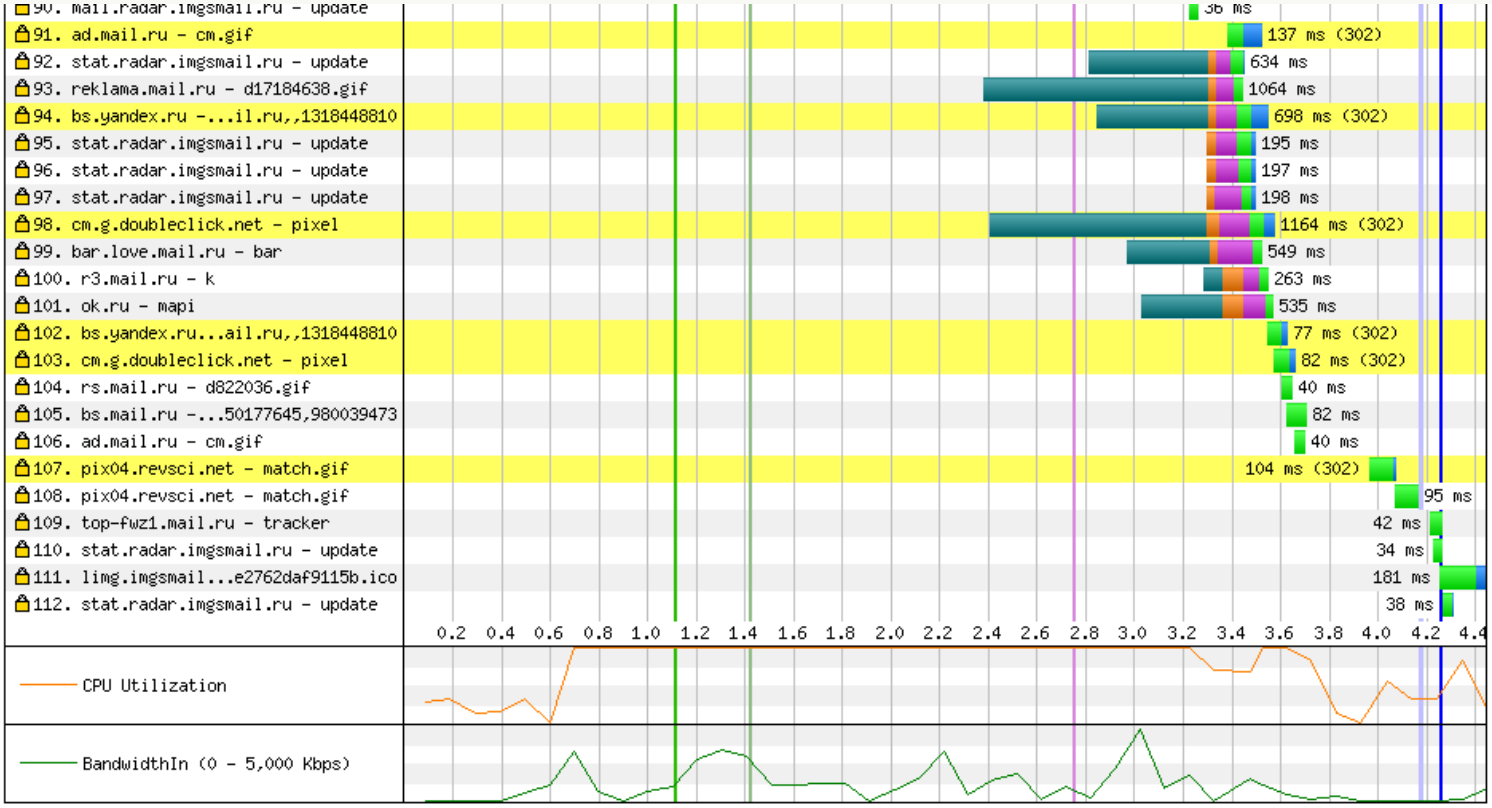
Сравнение влияния задержки и полосы пропускания на скорость загрузки

Процесс загрузки сайта (начало)



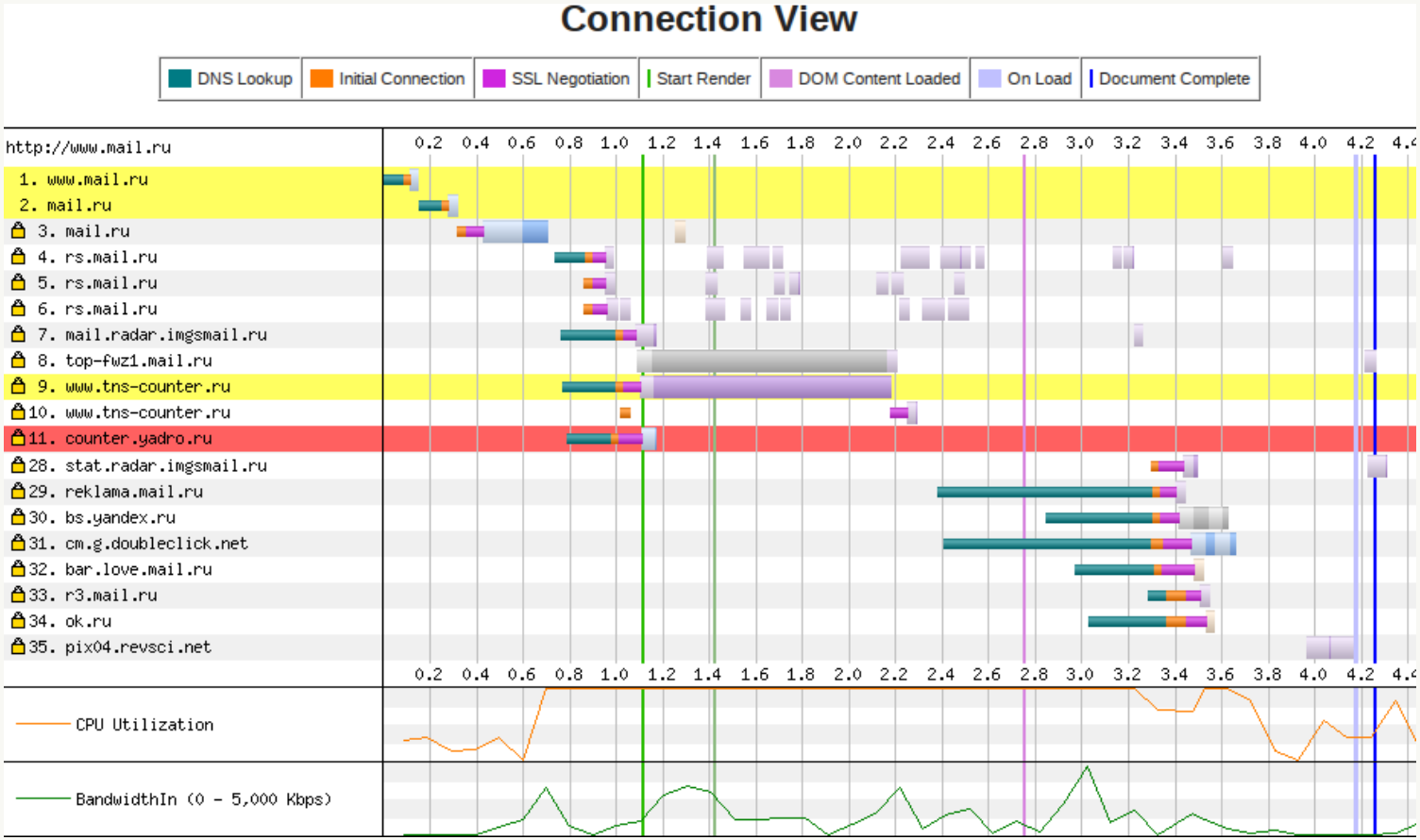
Данные **www.webpagetest.org** для сайта www.mail.ru

Загрузка сайта (конец)



Данные **www.webpagetest.org** для сайта www.mail.ru

TCP-соединения



Данные **www.webpagetest.org** для сайта www.mail.ru

Недостатки HTTP/1.1

- Отсутствие возможности мультиплексирования запросов в рамках одного соединения
- Чувствительность к задержке: продолжительный запрос может заблокировать все последующие запросы.
- HTTP — протокол без сохранения состояния. Избыточная мета-информация при каждом новом запросе (cookie).
- Костыли для обхода ограничения на количество запросов: конкатенация (js, css), спрайты (images), встраивание — приводят к деградации кэширования и избыточнности загружаемых ресурсов.

HTTP/2

- Более быстрый и более простой протокол, лишённый недостатков HTTP/1.1
- Сохранена парадигма HTTP-протокола: запрос-ответ
- Неизменны схемы http:// и https://
- Сохранение состояния между запросами
- Сжатие http-заголовков
- Реализовано мультиплексированние данных и двусторонний обмен
- Управление приоритетами
- Бинарный протокол

Выбор протокола

- http:// используется стандартный механизм Upgrade

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: ...

HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c

...
```

- https:// применяет расширения TLS:
  - NPN (Next Protocol Negotiation) — наследие SPDY
  - ALPN (Application Layer Protocol Negotiation) — специально для HTTP/2

Пролог

Перед началом обмена, клиент должен отправить пролог — сообщение специального формата, сигнализирующее о начале передачи данных по протоколу HTTP/2:

```
"PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n"
```

Пролог

Перед началом обмена, клиент должен отправить пролог — сообщение специального формата, сигнализирующее о начале передачи данных по протоколу HTTP/2:

```
PRI * HTTP/2.0
SM
```

Пролог

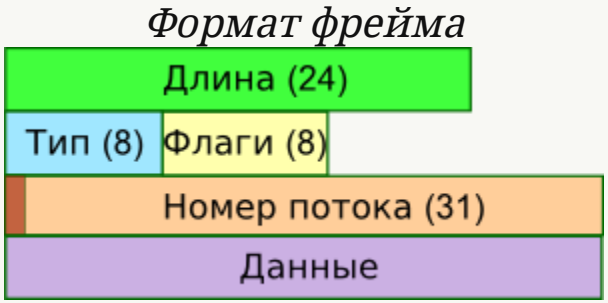
Перед началом обмена, клиент должен отправить пролог — сообщение специального формата, сигнализирующее о начале передачи данных по протоколу HTTP/2:

```
PRI * HTTP/2.0
SM
```

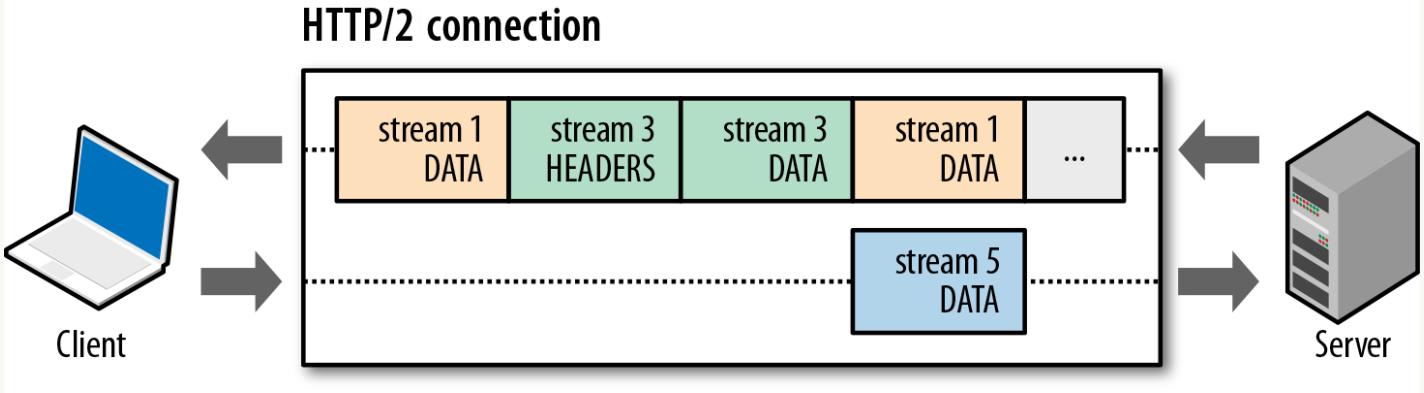
PRISM — программа слежения АНБ

Бинарный формат

- Данные передаются порциями — фреймами

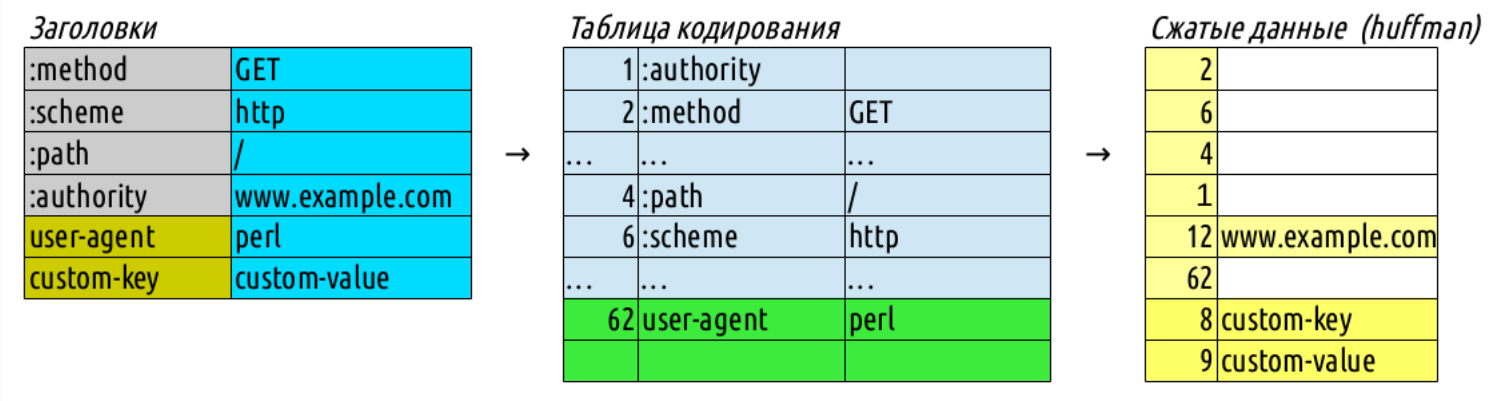


- Фреймы мультиплексируются внутри соединения



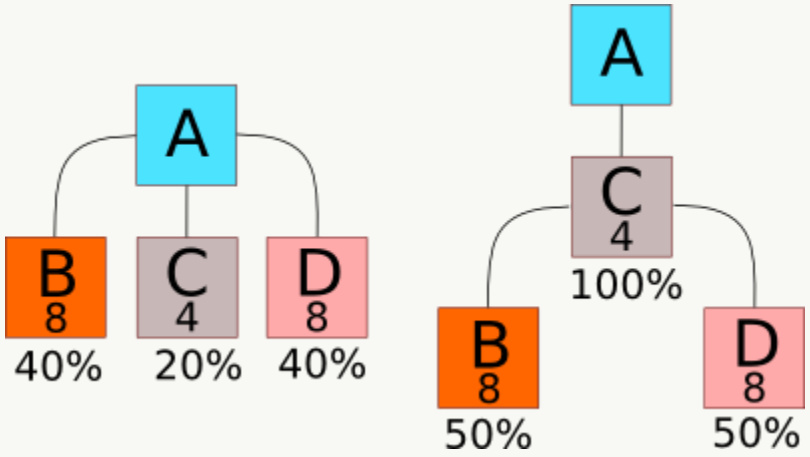
Сжатие заголовков

- Независимые контексты кодировщика и декодировщика
- Статическая и динамические таблица (де)кодирования



Приоритеты

- Управление приоритетами для различных потоков с помощью системы весов
- Эксклюзивные потоки



Server Push

- Сервер может инициировать передачу в сторону клиента: *server push* (посылка сервера)
- Клиент может отказаться от получения «запушенных» ресурсов

Реализации HTTP/2

- Wiki проекта спецификации HTTP/2 содержит ссылки на ~50 реализаций клиентов/серверов/прокси

  https://github.com/http2/http2-spec/wiki/Implementations

- Браузеры
  - Firefox 35+
  - Chrome 40+
  - Edge, IE11
  - Safari 9+
  - Opera 33+
- Сервера
  - Nginx 1.95+
  - Apache 2.4.17+
  - IIS 10
- Прочее
  - curl 7.33.0+
  - Wireshark 1.99.2

Perl

Популярные http-клиенты

- LWP (LWP::Simple, LWP::UserAgent)
- AnyEvent::HTTP
- Mojo::UserAgent
- Net::HTTP
- Net::HTTP::Tiny
- Net::Curl
- Furl
- HTTP::Tiny
- …

Вопрос: *где есть поддержка HTTP/2*?

Perl

Популярные http-клиенты

- LWP (LWP::Simple, LWP::UserAgent)
- AnyEvent::HTTP
- Mojo::UserAgent
- Net::HTTP
- Net::HTTP::Tiny
- **Net::Curl**
- Furl
- HTTP::Tiny
- …

Net::Curl

Пример запроса с использованием HTTP/2

```
use Net::Curl::Easy qw(:constants);
use Carp qw(croak);

my $curl = Net::Curl::Easy->new();
my $url = 'https://www.google.com';

$curl->setopt( CURLOPT_URL, $url );
$curl->setopt( CURLOPT_FOLLOWLOCATION, 1 );
$curl->setopt( CURLOPT_FILE, \my $body );
$curl->setopt( CURLOPT_HEADERDATA, \my $head );
$curl->setopt( CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_2_0);

my $ret = $curl->perform;

if ( ! defined $ret ) {
    my $res = $curl->getinfo(CURLINFO_HTTP_CODE);
    if ($res == 200 ) {
        print $head;
        print $body;
    }
} else {
    croak "error";
}
```

Net::Curl

```
$ perl curl_client.pl

HTTP/2.0 200
date:Wed, 16 Dec 2015 19:28:29 GMT
expires:-1
cache-control:private, max-age=0
content-type:text/html; charset=ISO-8859-1
p3p:CP="This is not a P3P policy! See https://www.google.com/support/accounts/answ
server:gws
...
```

Net::Curl

Multi-интерфейс Net::Curl::Multi

```
use Net::Curl::Easy qw(:constants);
use Net::Curl::Multi;

my $multi = Net::Curl::Multi->new;

# Включаем мультиплексирование запросов
$multi->setopt(Net::Curl::Multi::CURLMOPT_PIPELINING, CURLPIPE_MULTIPLEX);

# Максимум 100 запросов в очереди
$multi->setopt(Net::Curl::Multi::CURLMOPT_MAX_PIPELINE_LENGTH, 100);

# Не более 1 соединения
$multi->setopt(Net::Curl::Multi::CURLMOPT_MAX_HOST_CONNECTIONS, 1);
```

Net::Curl

Multi-интерфейс Net::Curl::Multi

```
# Процедура запроса
sub get {
    my ($multi, $url) = @_;
    my $easy = Net::Curl::Easy->new() or die "cannot curl";
    $multi->add_handle($easy);

    $easy->setopt( CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_2_0);
    $easy->setopt( CURLOPT_PIPEWAIT, 1);
    $easy->setopt( CURLOPT_FOLLOWLOCATION, 1 );

    open(my $head, "+>", undef) or die $!;
    Net::Curl::Easy::setopt($easy, CURLOPT_WRITEHEADER, $head);
    open(my $body, "+>", undef) or die $!;
    Net::Curl::Easy::setopt($easy, CURLOPT_WRITEDATA, $body);

    $easy->setopt(CURLOPT_URL, $url);
}
```

Net::Curl

Multi-интерфейс Net::Curl::Multi

```
# 100 итераций
for (1..100) {
    # 100 параллельных запросов
    get ( $multi, 'https://google.com' ) for (1..100);

    my $running = 0;
    do {
        my ($r, $w, $e) = $multi->fdset();
        my $timeout = $multi->timeout();
        select $r, $w, $e, $timeout / 1000 if $timeout > 0;

        $running = $multi->perform;

        while (my (undef, $easy, $result) = $multi->info_read) {
            if ($result == 0 ) {
                print "ok\n";
            } else {
                warn "curl error: $result\n";
            }
            $multi->remove_handle($easy);
        }
    } while ($running);
}
```

Perl

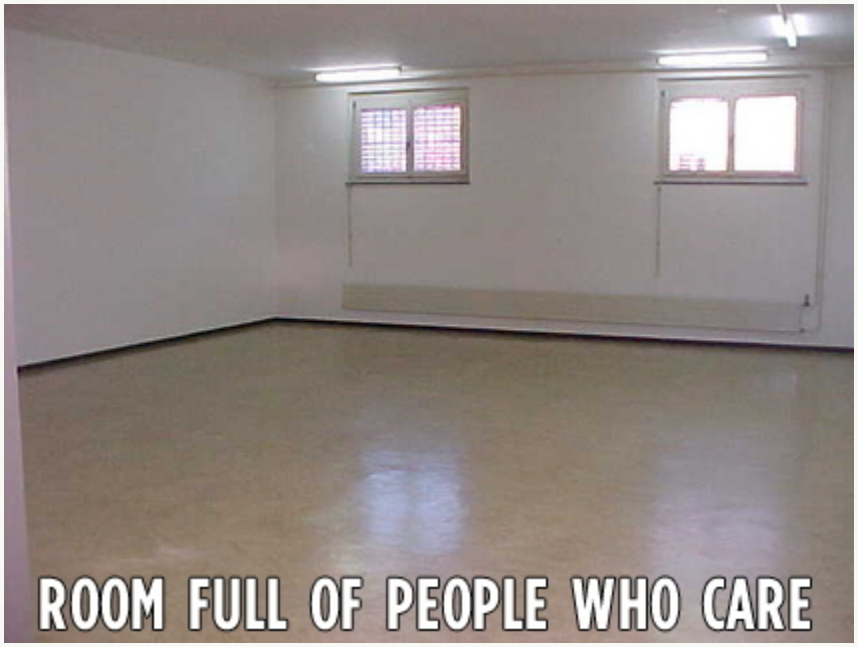
Популярные http-сервера

- Starman
- Starlet
- Hypnotoad, Morbo
- Catalyst::Engine::HTTP::Prefork
- HTTP::Server::PSGI
- HTTP::Server::Simple
- AnyEvent::HTTP::Server
- Twiggy
- Feersum
- ...

Вопрос: *кто реализовал HTTP/2*?

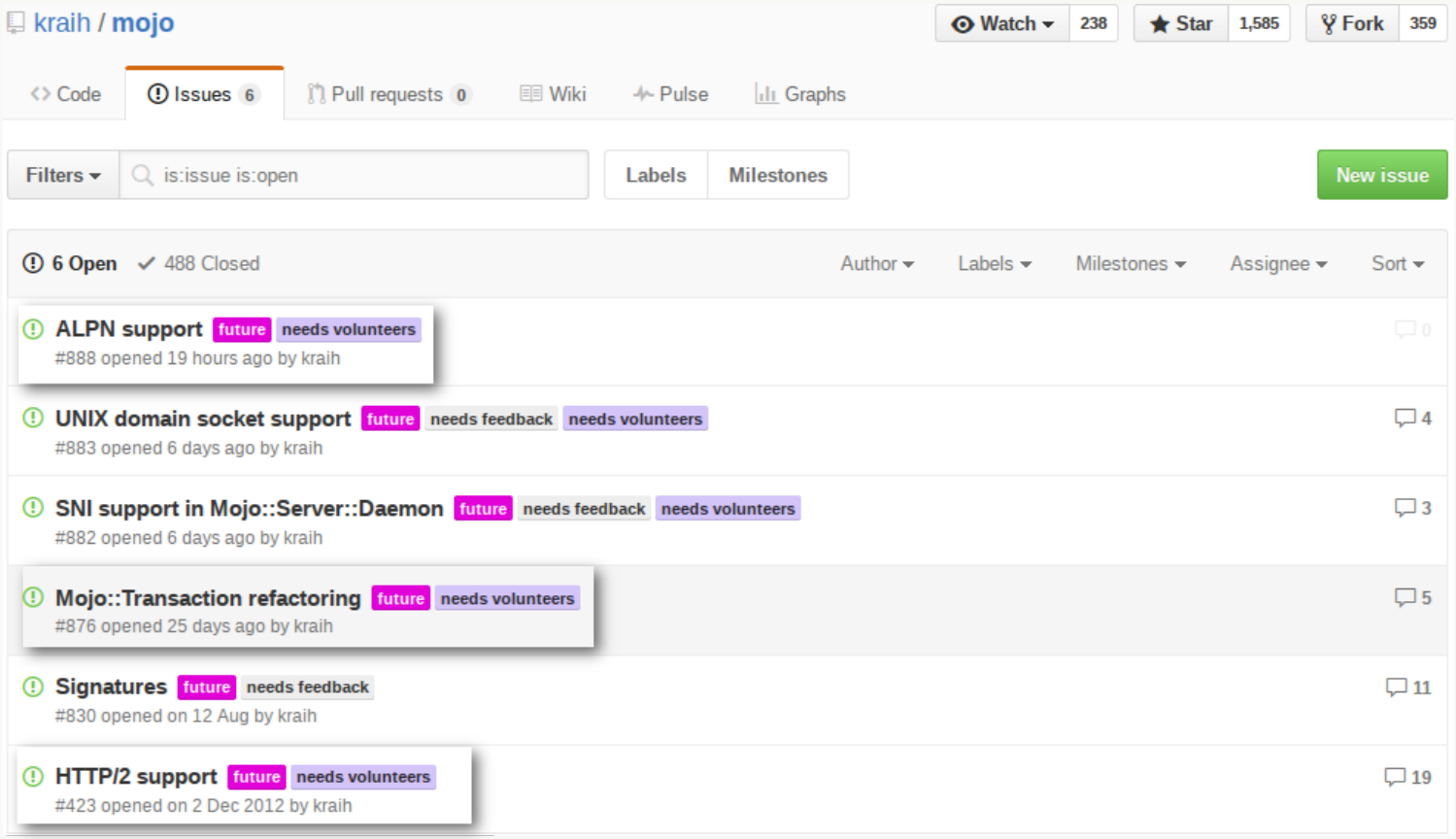
Perl

Популярные http-сервера



Ответ: *все, кого видите в этой комнате*

Mojolicious



Protocol::HTTP2

- `Protocol::HTTP2` — реализация протокола HTTP/2 на чистом Perl
  - `Protocol::HTTP2::Client` — реализация клиента протокола HTTP/2 на прикладном уровне
  - `Protocol::HTTP2::Server` — реализация сервера протокола HTTP/2 на прикладном уровне
  - `Protocol::HTTP2::HeaderCompression` — реализация сжатия заголовков HPACK (RFC 7541)

Клиент HTTP/2

```
use IO::Socket::SSL;
use IO::Select;
use Protocol::HTTP2::Client;

my $host = 'google.com';
my $port = 443;
# HTTP/2 клиент прикладного уровня
my $h2_client = Protocol::HTTP2::Client->new->request(
    # HTTP/2-заголовки
    ':method'    => 'GET',   ':path'      => '/',
    ':scheme'    => 'https', ':authority' => $host . ':' . $port,
    # HTTP-заголовки
    headers => [
        'user-agent' => 'Protocol::HTTP2'
    ],
    on_done => sub {
        my ( $headers, $data ) = @_;
    }
);
# Транспортный уровень TLS over TCP
my $client = IO::Socket::SSL->new(
    PeerHost => $host,  PeerPort => $port,
    SSL_npn_protocols => ['h2'],
) or die $!;
```

Клиент HTTP/2 (продолжение)

```
# Неблокирующая работа с сокетом
$client->blocking(0);

my $sel = IO::Select->new($client);

# Цикл работы с сокетом
while ( !$h2_client->shutdown ) {
    $sel->can_write;
    while (my $frame = $h2_client->next_frame ) {
        syswrite $client, $frame;
    }

    $sel->can_read;
    while ( sysread $client, my $data, 4096 ) {
        $h2_client->feed($data);
    }
}
```

Сервер HTTP/2

```
use IO::Select;
use IO::Socket::SSL;
use Protocol::HTTP2::Server;

# Транспортный уровень TLS over TCP
my $srv = IO::Socket::SSL->new(
    LocalAddr => '0.0.0.0:1234', Listen => 10,
    SSL_cert_file => 'test.crt', SSL_key_file  => 'test.key',
    SSL_npn_protocols => ['h2'],
) or die $!;

while ( my $client = $srv->accept ) {
    # HTTP/2 сервер прикладного уровня
    my $h2_srv; $h2_srv = Protocol::HTTP2::Server->new(
        on_request => sub {
            my ( $stream_id, $headers, $data ) = @_;
            $h2_srv->response(
                ':status' => 200,
                stream_id => $stream_id,
                headers   => [
                    'server' => 'Protocol::HTTP2::Server',
                ],
                data => "Hello, World!",
            );
        }
    );
```

Сервер HTTP/2 (продолжение)

```
    $client->blocking(0);
    my $sel = IO::Select->new($client);

    while ( !$h2_srv->shutdown ) {
        $sel->can_write;
        while (my $frame = $h2_srv->next_frame ) {
            syswrite $client, $frame;
        }

        $sel->can_read;
        my $len;
        while ( my $rd = sysread $client, my $data, 4096 ) {
            $h2_srv->feed($data);
            $len += $rd;
        }
        last unless $len;
    }
    undef $h2_srv;
}
```

Спасибо за внимание

Вопросы?

Владимир Леттиев

Twitter: @truecrux

CPAN: CRUX