# Сетевые технологии
# HTTP/2

Соловьев А. В.

ПетрГУ

(Rev. 2018 12 03)

# Предпосылки создания HTTP/2

- рост объёмов веб-страниц и количества элементов на них,
- задержка начала передачи объекта (lattency),
- блокировка начала очереди (неэффективность HTTP pipelining).

"Костыли" HTTP/1:

- спрайты,
- встраивание изображений,
- объединение скриптов в один файл,
- шардинг.

# История HTTP/2 (1)

Задачи разработки HTTP/2:

- Уменьшить чувствительность к RTT.
- Исправить конвейеризацию и проблему блокировки начала очереди, в т. ч. остановить необходимость и желание в увеличении числа соединений к каждому хосту.
- Сохранить существующие API, содержимое, формат/схемы URI.
- Сделать это внутри рабочей группы IETF.

Рабочая группа HTTPbis работала над обновлением HTTP/1.1 (RFC2616-1999) с 2007 г. В 2014 г. выпущены:

- RFC7230 – HTTP/1.1: Message Syntax and Routing
- RFC7231 – HTTP/1.1: Semantics and Content
- RFC7232 – HTTP/1.1: Conditional Requests
- RFC7233 – HTTP/1.1: Range Requests
- RFC7234 – HTTP/1.1: Caching
- RFC7235 – HTTP/1.1: Authentication

# История HTTP/2 (2)

В 2012 г. Google начала разработку протокола SPDY, предназначенного для манипулирования HTTP-трафиком:

- бинарный протокол,
- сжатие заголовков,
- мультиплексирование потоков,
- приоритезация.

Поддержка SPDY была реализована в Chrome и Firefox. SPDY работал только поверх TLS, для этого использовалось расширение NPN (Next Protocol Negotiation).

Третья версия SPDY взята рабочей группой HTTPbis за основу (черновик) HTTP/2. В 2015 г. разработка и поддержка SPDY прекращена.

Май 2015: RFC7540 – Hypertext Transfer Protocol Version 2 (HTTP/2)

# Концепция HTTP/2 (1)

Существующие URI не меняются.

При использовании схемы http:// браузер обнаруживает поддержку HTTP/2 через заголовок Upgrade:

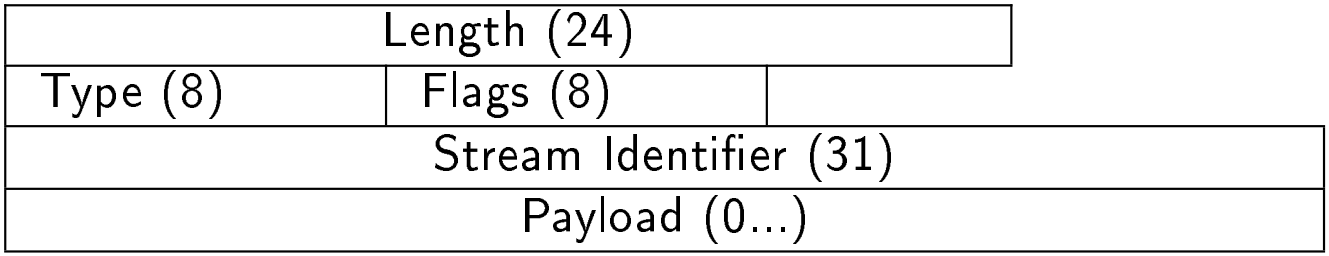
| | |
|---|---|
| GET / HTTP/1.1<br>Host: 172.20.175.60<br>Connection: Upgrade, HTTP2-Settings<br>Upgrade: **h2c**<br>HTTP2-Settings: AAMAAABkAARAAAAA<br><CRLF> | HTTP/1.1 101 Switching Protocols<br>Upgrade: h2c<br>Connection: Upgrade<br><CRLF><br>...<br>... |

При использовании схемы https:// браузер запрашивает поддержку HTTP/2 (фигурирует как "**h2**") через TLS-расширение ALPN (Application-Layer Protocol Negotiation), RFC7301.

Разработчики Firefox и Chrome, заявили, что не будут поддерживать h2c – только h2. В curl, IE поддержка h2c есть.

# Концепция HTTP/2 (2)

Бинарный протокол:




- Мультиплексирование потоков.
- Каждый поток имеет приоритет (какие потоки отправлять в первую очередь при ограниченности ресурсов?). При помощи приоритетов также указывается, от какого другого потока зависит данный поток (дерево приоритетов).
- Сжатие HTTP-заголовков при помощи HPACK.
- Отмена потока (Reset).
- Посылка сервера (Server Push).
- Управление потоком (меняя размер окна, можно приостановить передачу данных в потоке).

# Типы кадров HTTP/2

- DATA (данные), type=0 (ES, P);
- HEADERS (сжатые заголовки), type=1 (ES, EH, P, PR);
- PRIORITY (зависимость, вес), type=2;
- RST_STREAM (код ошибки), type=3;
- SETTINGS, type=4 (A); id(16)=value(32): HEADER_TABLE_SIZE=1 (4096), ENABLE_PUSH=2 (1), MAX_CONCURRENT_STREAMS=3 (-), INITIAL_WINDOW_SIZE=4 (64K), MAX_FRAME_SIZE=5 (16K), MAX_HEADER_LIST_SIZE=6 (-);
- PUSH_PROMISE (stream id, заголовки), type=5 (EH, P);
- PING, type=6 (A);
- GOAWAY (last stream id, код ошибки), type=7;
- WINDOW_UPDATE (инкремент), type=8;
- CONTINUATION (сжатые заголовки), type=9 (EH).

Флаги: ES = end_stream, EH = end_headers, P = padded, PR = priority, A = ack

# Заголовки в HTTP/2

Идентификаторы заголовков всегда должны быть в нижнем регистре.

Сжатие HPACK (RFC 7541)

Куда делись строка запроса с методом и строка статуса? — Псевдозаголовки:

- :method (GET, POST, HEAD, ...)
- :scheme (http или https)
- :authority (вместо Host)
- :path (путь из URI)
- :status (число)

# Пример (1)

Запрос:

```
GET /resource HTTP/1.1
Host: example.org
Accept: image/jpeg
```

```
HEADERS (+ES, +EH)
  :method : GET
  :scheme : https
  :path : /resource
  host : example.org
  accept : image/jpeg
```

Ответ:

```
HTTP/1.1 304 Not Modified
ETag: "xyzzy"
Expires: Thu, 23 Jan ...
```

```
HEADERS (+ES, +EH)
  :status : 304
  etag : "xyzzy"
  expires : Thu, 23 Jan ...
```

# Пример (2)

Запрос:

```
POST /resource HTTP/1.1
Host: example.org
Content-Type: image/jpeg
Content-Length: 123

{binary data}
```

```
HEADERS (-ES, -EH)
  :method : POST
  :scheme : https
  :path : /resource
CONTINUATION (+EH)
  host : example.org
  content-type : image/jpeg
  content-length : 123
DATA (+ES)
  {binary data}
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: image/jpeg
Content-Length: 123

{binary data}
```

```
HEADERS (-ES, +EH)
  :status : 200
  content-type : image/jpeg
  content-length : 123
DATA (+ES)
  {binary data}
```

# Немного HPACK – RFC 7541

Заголовки индексируются и хранятся в таблице. Первые 61 элемент – статические (Приложение A к RFC, 61 элемент). Остальная часть – динамические (вычисляются и хранятся приложением по принципу FIFO). Размер таблицы – SETTINGS/HEADER_TABLE_SIZE (4096). Проиндексированный заголовок кодируется числом с 7-битным префиксом (бит 7 = 1).

Примеры статических элементов таблицы:

| Idx | Заголовок | Значение |
|---|---|---|
| 1 | :authority | |
| 2 | :method | GET |
| 3 | :method | POST |
| 4 | :path | / |
| 5 | :path | /index.html |
| 6 | :scheme | http |
| 7 | :scheme | https |
| 8 | :status | 200 |
| 9 | :status | 204 |
| ... | ... | ... |
| 15 | accept-charset | |
| 16 | accept-encoding | gzip, deflate |

# Что дальше? HTTP/3

HTTP-over-QUIC $\longrightarrow$ HTTP/3

QUIC (Quick UDP Internet Connections) – надстройка над UDP, обеспечивающая шифрование и мультиплексирование. Разработка начата Google в 2012. Сейчас разработка передана в IETF.