OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование

Не забыть включить запись!

Меня хорошо видно && слышно?

Ставьте + , если все хорошо

Напишите в чат, если есть проблемы

Правила вебинара

Активно участвуем

Задаем вопрос в чат или голосом

Off-topic обсуждаем в Slack #канал группы или #general

Вопросы вижу в чате, могу ответить не сразу

Протокол HTTP (**часть** 2)

Викирюк Павел

Системный инженер

Маршрут вебинара

TCP handshake

↓

HTTP/2

↓

Websocket

Цели занятия | После занятия вы сможете

1. Понять влияние протокола TCP на протокол HTTP

2. Познакомиться с особенностями протокола HTTP/2 и узнать чем он отличается от HTTP/1.1

3. Познакомиться с протоколом Websocket

Смысл | Зачем вам это уметь

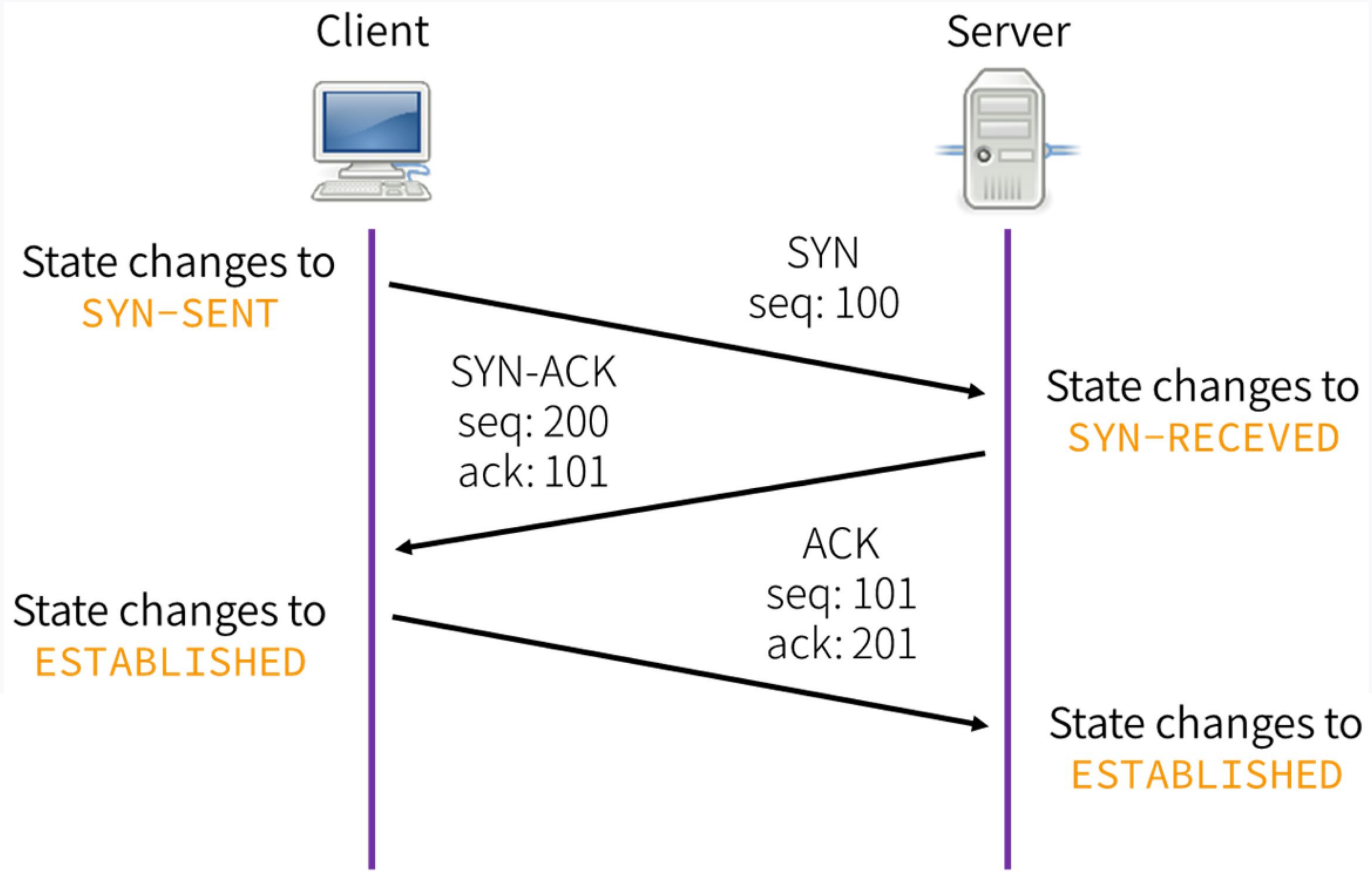
1. Чтобы понимать слабые места предыдущих версий протокола HTTP

2. Чтобы представлять каким образом увеличить производительность веб-проекта

3. Чтобы применять современные протоколы для решения своих задач

TCP handshake

TCP handshake



TCP handshake

TCP handshake (рукопожатие, трехэтапное рукопожатие) - процесс начала сеанса TCP, состояний из трех шагов:

1. **Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN**

   - сервер получает сегмент, запоминает номер последовательности и пытается создать сокет (буферы и управляющие структуры памяти) для обслуживания нового клиента
   - в случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED
   - в случае неудачи сервер посылает клиенту сегмент с флагом RST

TCP handshake

TCP handshake (рукопожатие, трехэтапное рукопожатие) - процесс начала сеанса TCP, состояний из трех шагов:

2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK

- если клиент одновременно получает и флаг ACK (что обычно и происходит), то он переходит в состояние ESTABLISHED
- если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться

TCP handshake

TCP handshake (рукопожатие, трехэтапное рукопожатие) - процесс начала сеанса TCP, состояний из трех шагов:

3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED

- В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED

TCP handshake

Завершение TCP соединения

Также происходит в три этапа:

1. Посылка серверу от клиента флага FIN на завершение соединения
2. Сервер посылает клиенту флаги ответа ACK , FIN, что соединение закрыто
3. После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто

HTTP Keep-alive

HTTP keep-alive

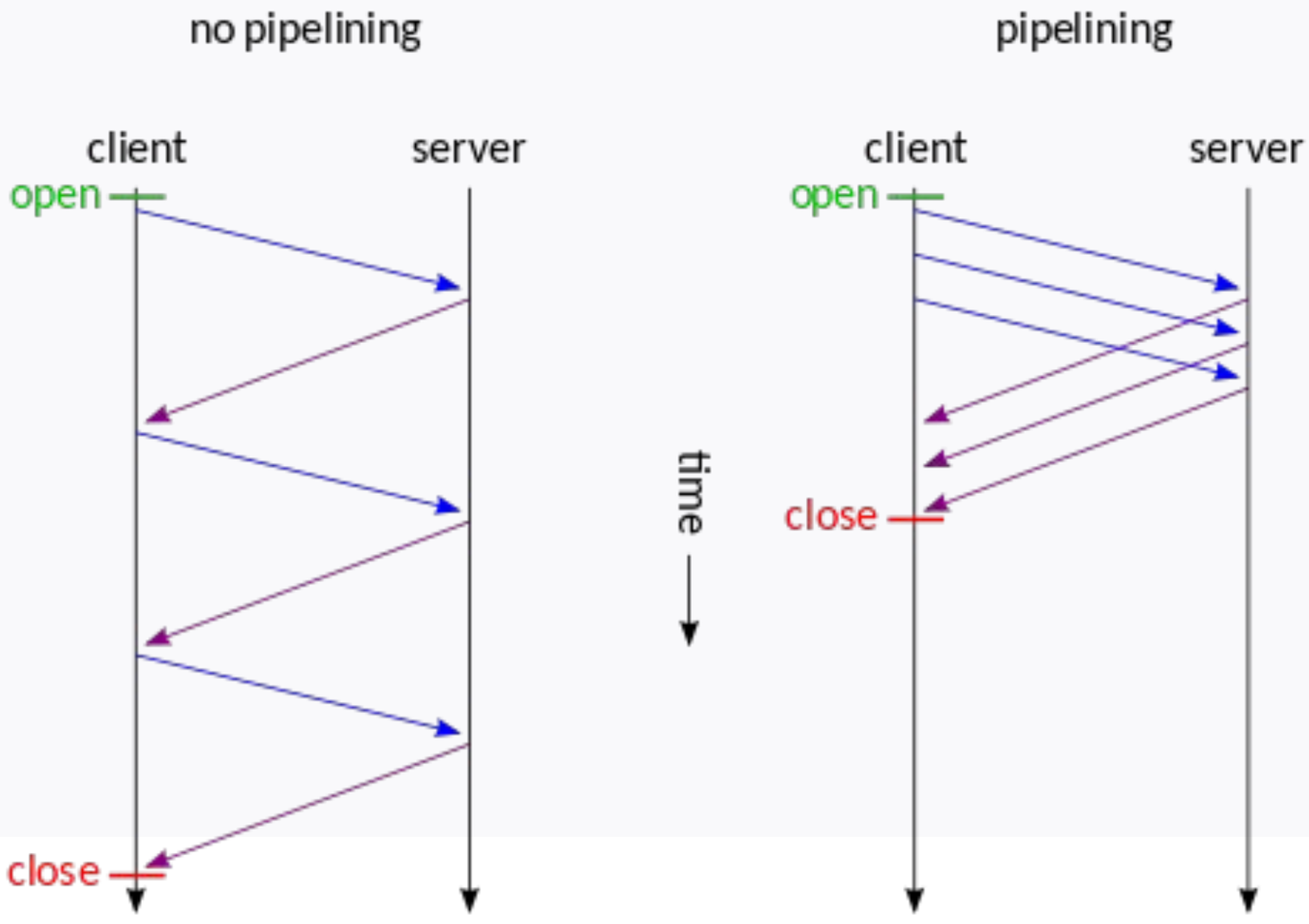
Постоянное HTTP-соединение (англ. HTTP persistent connection), также называемые **HTTP keep-alive** или повторное использование соединений HTTP (англ. HTTP connection reuse) — использование одного TCP-соединения для отправки и получения многократных HTTP-запросов и ответов вместо открытия нового соединения для каждой пары запрос-ответ

https://ru.wikipedia.org/wiki/Постоянное_HTTP-соединение

Особенности:

- в HTTP/1.0 используется заголовок **Connection: Keep-Alive** который позволяет использовать механизм keep-alive
- в HTTP/1.1 keep-alive включен если не указано иное
- снижает нагрузку на CPU и память
- позволяет использовать **HTTP pipelining**
- уменьшает нагрузку на сеть
- уменьшает задержку для следующих запросов

HTTP keep-alive



Ваши вопросы?

Маршрут вебинара

TCP handshake

HTTP/2

Websocket

HTTP/2

HTTP/2

HTTP/2 (изначально HTTP/2.0) — вторая крупная версия сетевого протокола HTTP

Особенности:

- протокол основан на SPDY от Google
- спецификация HTTP/2 была опубликована как **RFC 7540** в мае 2015
- является бинарным протоколом
- изменились способы разбиения данных на фрагменты и транспортирования их между сервером и клиентом
- сервер имеет право послать то содержимое, которое ещё не было запрошено клиентом
- сжатие заголовков
- явная приоритезация запросов
- в большинстве браузеров HTTP/2 работает только поверх TLS

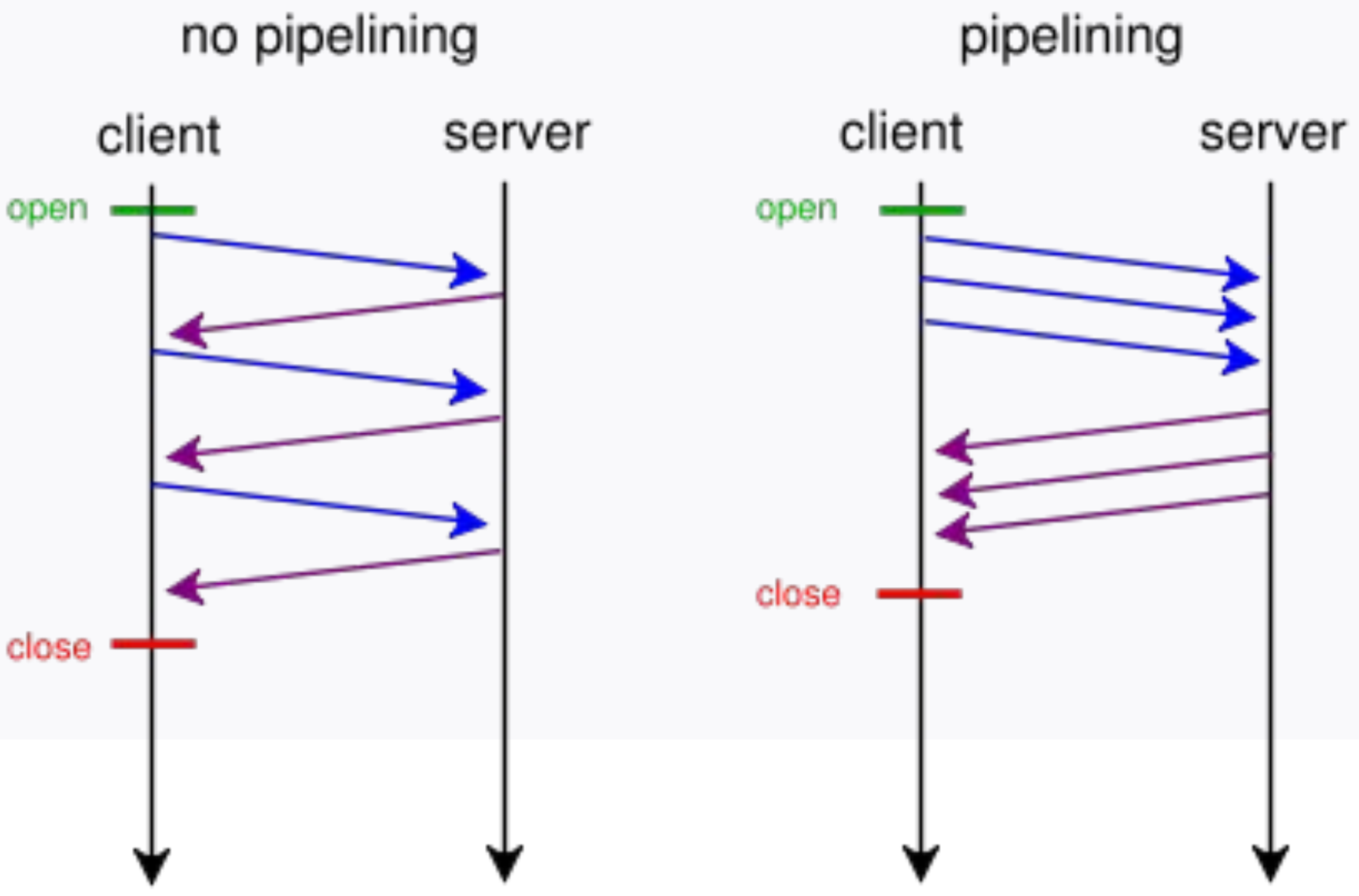
Двоичный уровень кадрирования

HTTP/2

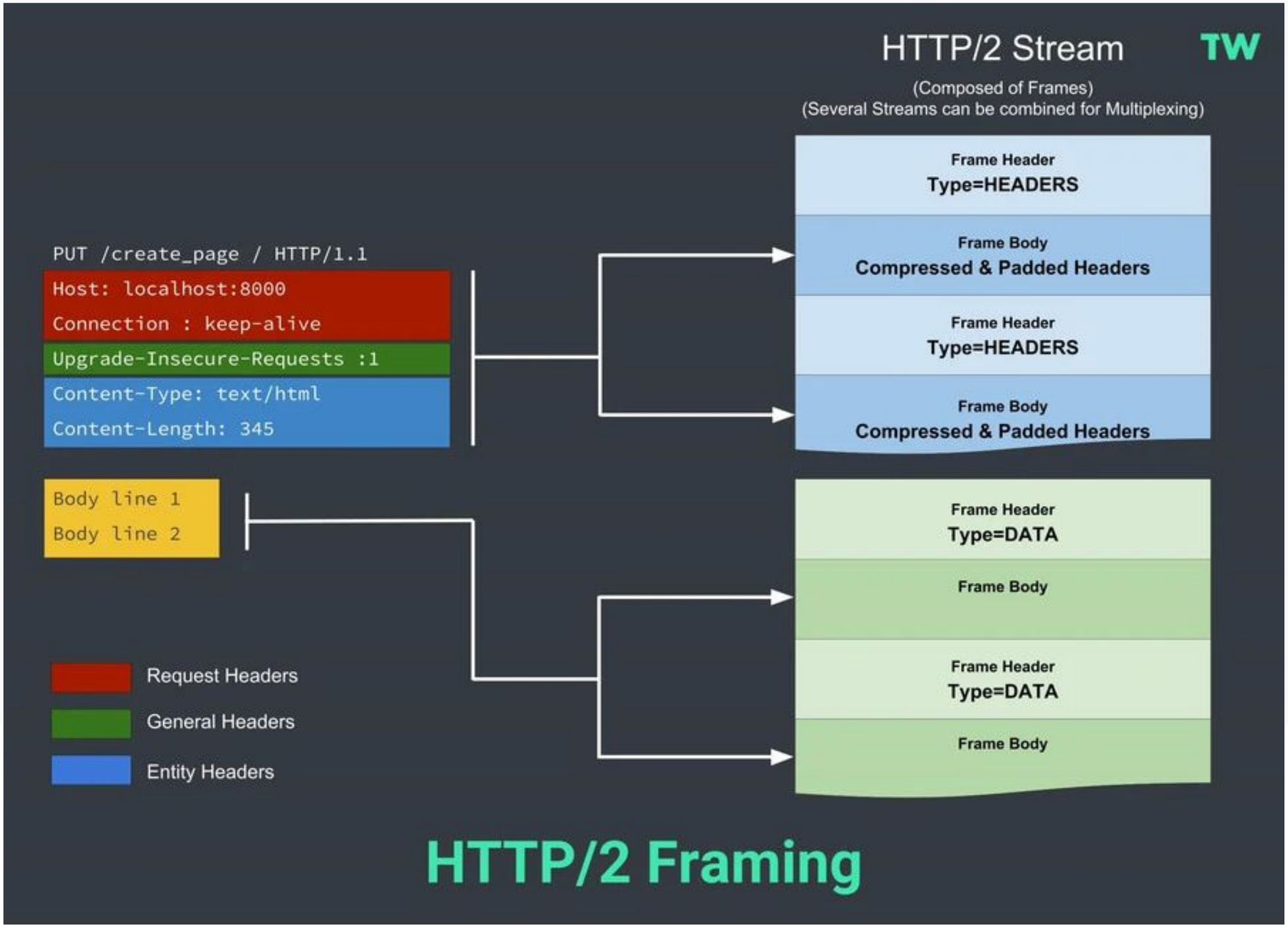
The number of simultaneous requests by browser:

Browser	Max Parallel Connections Per Host
IE 9	6
IE 10	8
Firefox 4+	6
Opera 11+	6
Chrome 4+	6
Safari	4

HTTP/2


no pipelining
client
server
open
close
pipelining
client
server
open
close
time

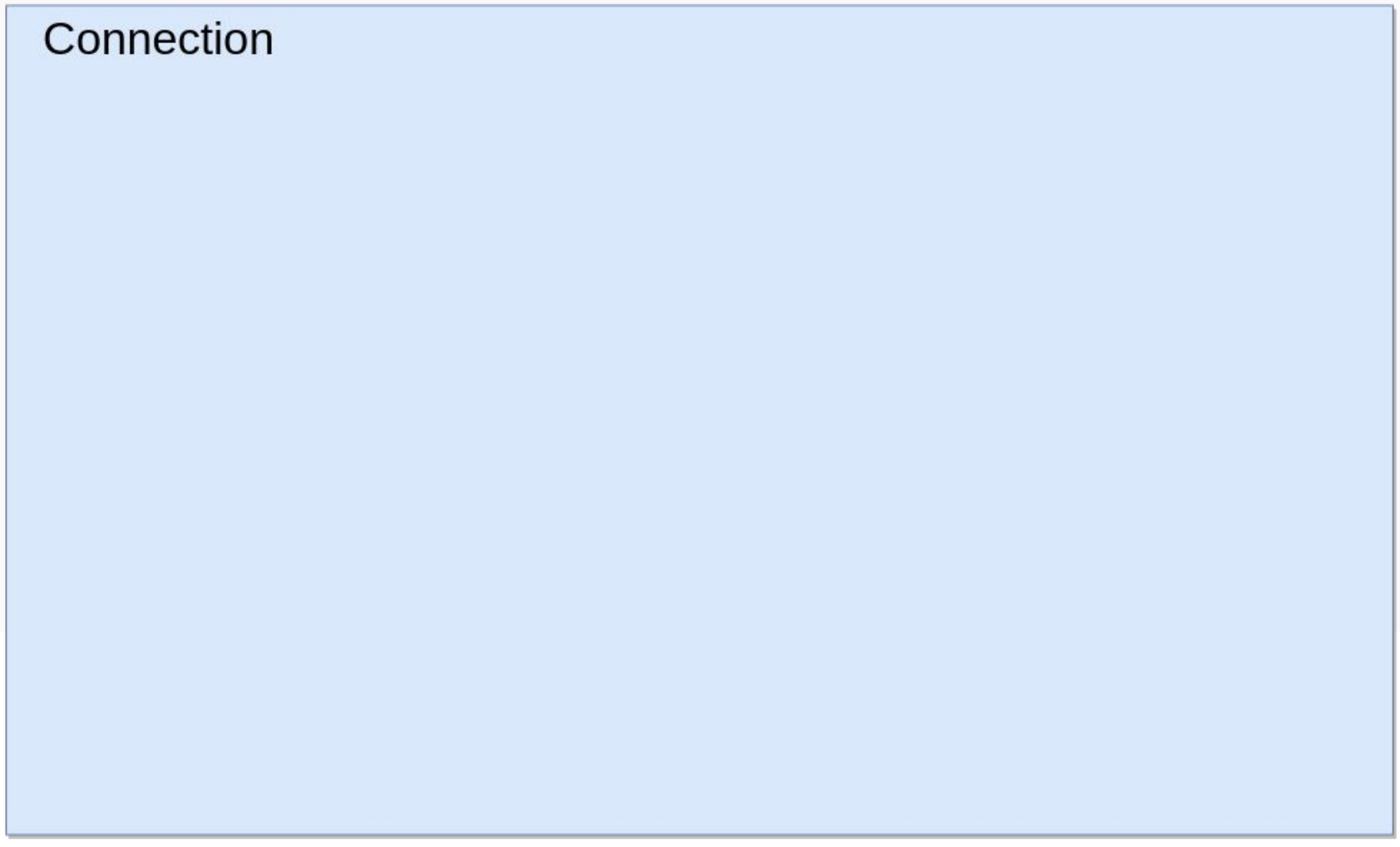
HTTP/2


HTTP/2 Stream
TW
(Composed of Frames)
(Several Streams can be combined for Multiplexing)
PUT /create_page / HTTP/1.1
Host: localhost:8000
Connection : keep-alive
Upgrade-Insecure-Requests :1
Content-Type: text/html
Content-Length: 345
Body line 1
Body line 2
Request Headers
General Headers
Entity Headers
Frame Header
Type=HEADERS
Frame Body
Compressed & Padded Headers
Frame Header
Type=HEADERS
Frame Body
Compressed & Padded Headers
Frame Header
Type=DATA
Frame Body
Frame Header
Type=DATA
Frame Body
HTTP/2 Framing

HTTP/2

Двоичный уровень кадрирования

1. Создается один **объект соединения (connection)** между сервером и клиентом



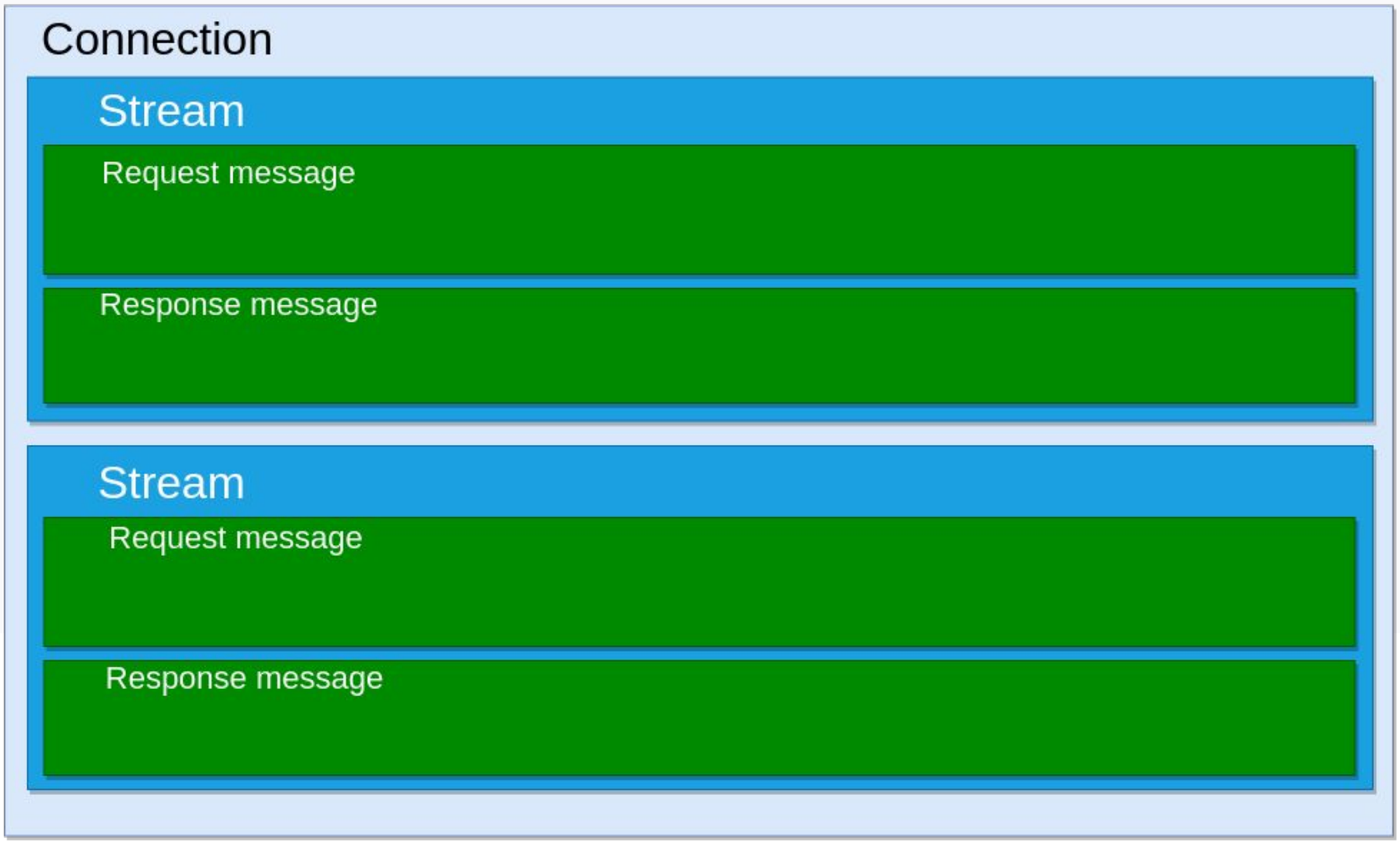
Двоичный уровень кадрирования

2. В объекте соединения есть несколько **потоков данных (streams)**

HTTP/2

Двоичный уровень кадрирования

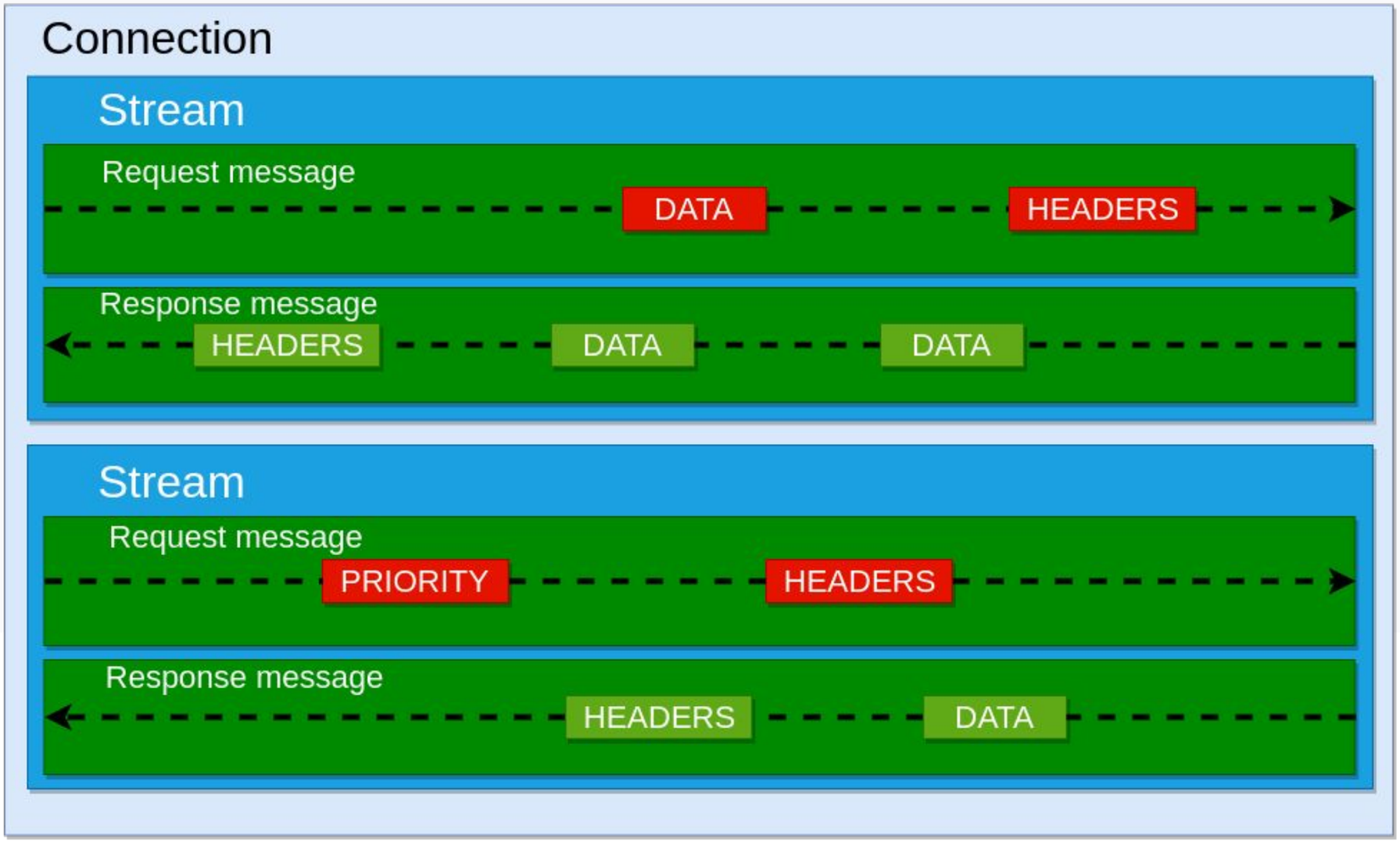
3. Потоки состоят из нескольких **сообщений (messages)** в формате “запрос-ответ” **(request или response)** каждый



HTTP/2

Двоичный уровень кадрирования

4. Каждое из сообщений разбивается на более мелкие блоки - **кадры (frames)**



HTTP/2

Двоичный уровень кадрирования

5. Кадры помечаются тэгами (**tags**)

6. Чередующиеся многочисленные запросы и ответы могут выполняться параллельно, не блокируя следующие сообщения в очереди

7. На другом конце соединения благодаря тэгам кадры успешно собираются обратно

Процесс разбивки и сборки называется **мультиплексированием**

Преимущества мультиплексирования:

- нет необходимости пересоздавать TCP соединение
- уменьшение необходимого объема памяти
- минимизирует необходимую полосу пропускания
- снижает общие накладные расходы на обработку запросов

Приоритизация потоков

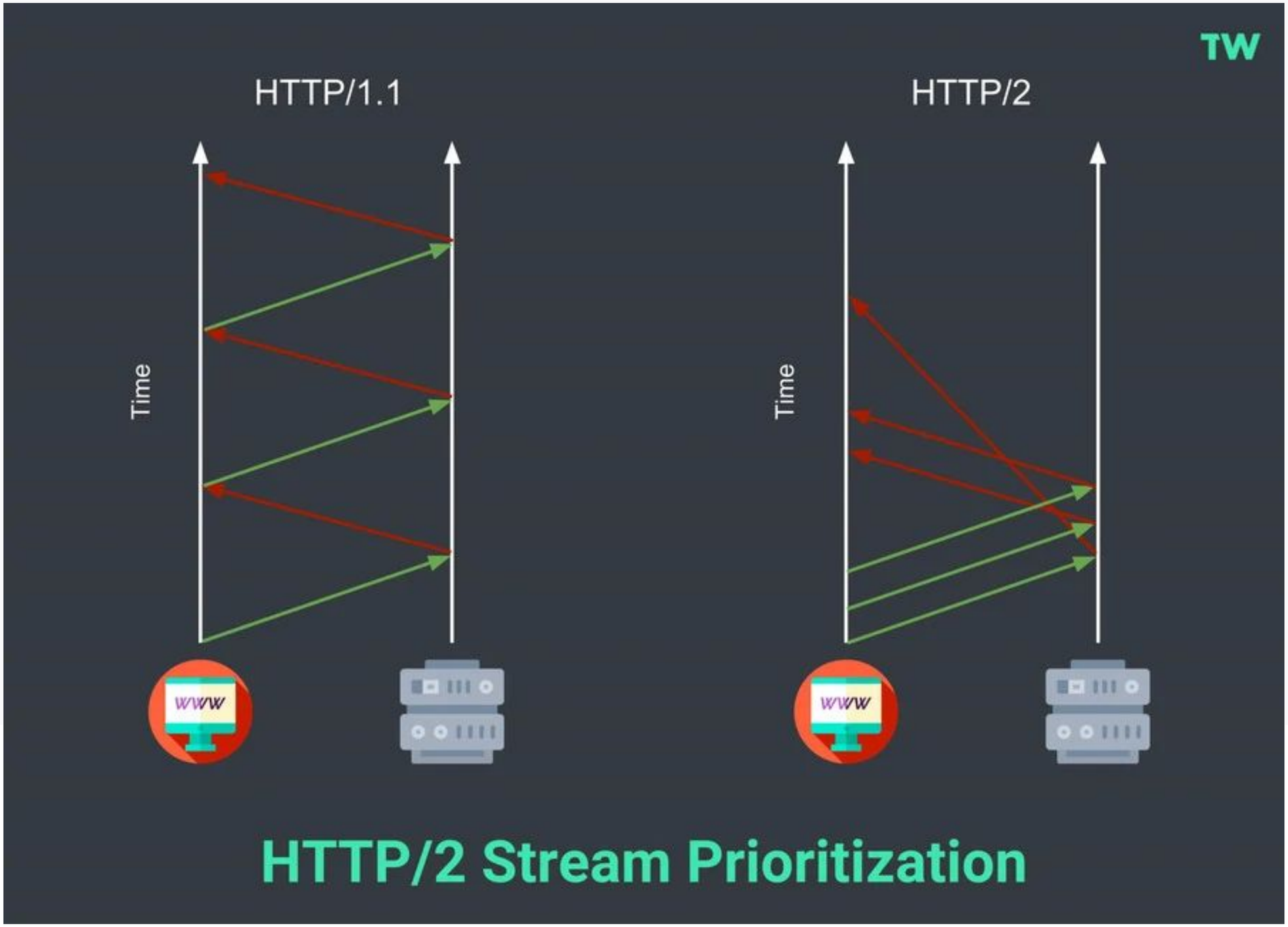
HTTP/2

Приоритизация потоков

Особенности:

- предотвращает проблемы с производительностью из-за ожидания ресурса несколькими потоками
- позволяет разработчикам настраивать относительный вес запросов для лучшей оптимизации производительности приложения

HTTP/2


TW
HTTP/1.1
Time
HTTP/2
Time
www
www
HTTP/2 Stream Prioritization

HTTP/2

Приоритизация потоков

Принцип действия:

1. Клиент при отправке запросов на сервер может расставить **приоритеты** запрашиваемых ответов, присвоив **вес** от 1 до 256 на каждый поток
2. Чем выше вес - тем выше приоритет
3. Клиент может определять зависимости между потоками
4. Сервер может изменить назначенные приоритеты самостоятельно, если определенный поток заблокирован от доступа к конкретному ресурсу

Переполнение буфера

HTTP/2

Переполнение буфера

Принцип действия:

- так как TCP соединение в случае с HTTP/2 одно - не получится настроить окно передачи в TCP (на L3 модели OSI)
- регулирование окна передачи происходит средствами HTTP/2 на прикладном уровне (L7 модели OSI)
- управление потоком может быть изменено или сохранено после первоначального подключения через кадр **WINDOW_UPDATE** на уровне мультиплексированных потоков
- достигается большая гибкость регулирования
- улучшается производительность веб-приложений

Прогнозирование запроса ресурсов

HTTP/2

Прогнозирование запросов ресурсов

Принцип действия:

- используется механизм **server push**
- чтобы отправить ресурс сервер посылает клиенту кадр **PUSH_PROMISE**, включающий в себя заголовок сообщения
- если предложенный ресурс уже есть у клиента - он отклоняет предложение кадром **RST_STREAM**

Сжатие

HTTP/2

Сжатие

Принцип действия:

- в HTTP/2 в отличие от версии 1.1 можно и нужно сжимать заголовки
- в HTTP/2 заголовки сжимаются с помощью **HPACK**

https://tools.ietf.org/html/draft-ietf-httpbis-header-compression-12

- **HPACK** также отслеживает и сжимает метаданные

Ваши вопросы?

Маршрут вебинара

TCP handshake

HTTP/2

Websocket

Websocket

Websocket

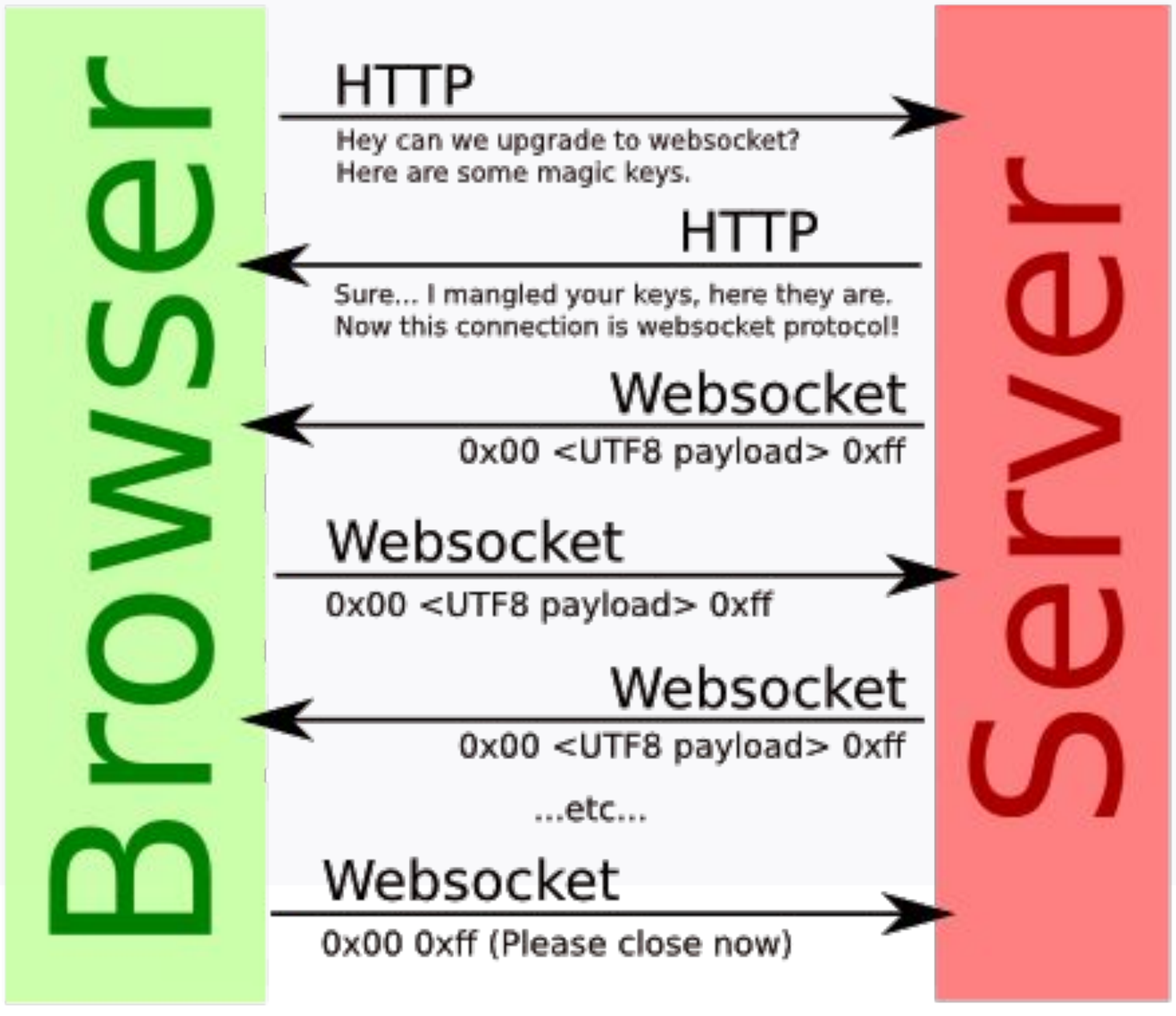
WebSocket — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени

https://ru.wikipedia.org/wiki/WebSocket

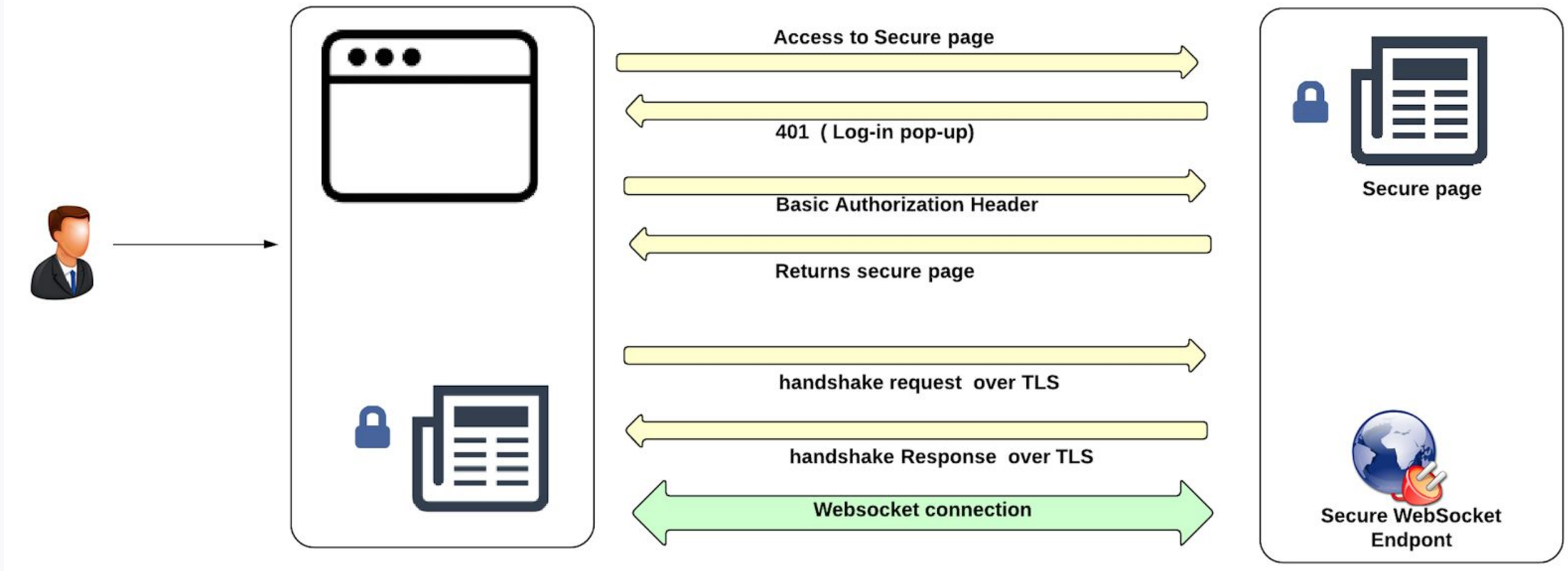
Особенности:

- независимый протокол, работающий поверх TCP
- работает по принципу “клиент-сервер”
- двунаправленный (сообщения идут в обоих направлениях)
- описан в **RFC 6455** **https://tools.ietf.org/html/rfc6455** (статус RFC получил в 2011 г.)
- схема URI: **ws:** для нешифрованных соединений и **wss:** для шифрованных
- сравнительно небольшой overhead (по 2 байта на сообщение)

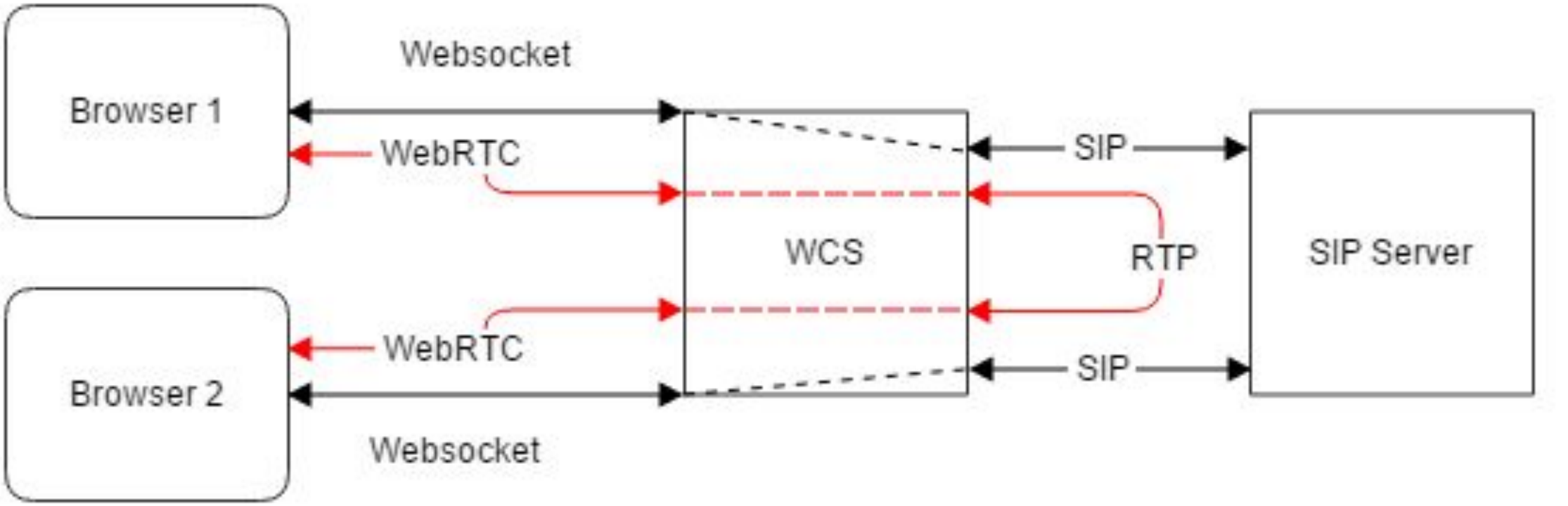
Websocket


Browser
Server
HTTP
Hey can we upgrade to websocket?
Here are some magic keys.
HTTP
Sure... I mangled your keys, here they are.
Now this connection is websocket protocol!
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
...etc...
Websocket
0x00 0xff (Please close now)

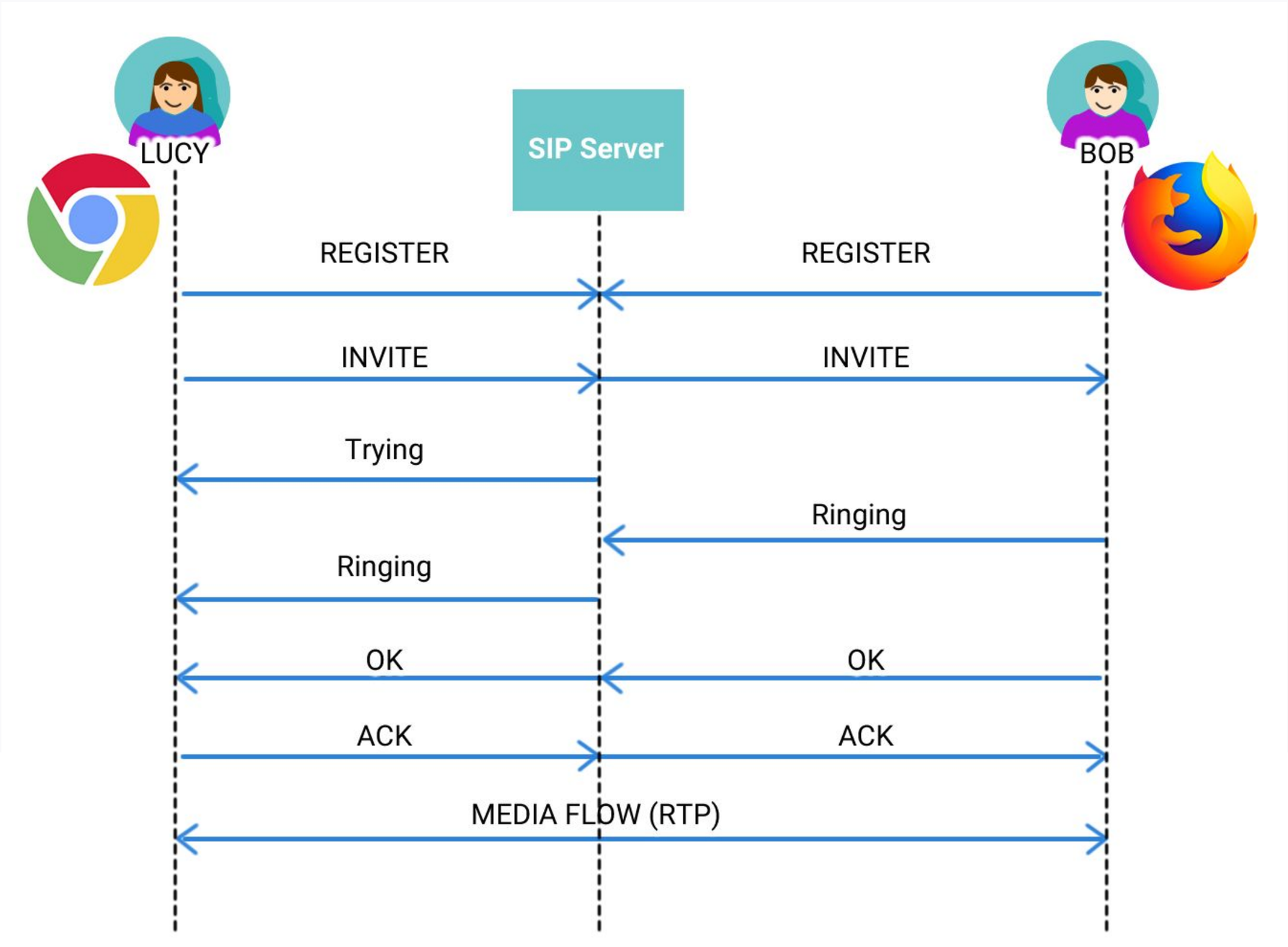
Websocket


Access to Secure page
401 (Log-in pop-up)
Basic Authorization Header
Returns secure page
Secure page
handshake request over TLS
handshake Response over TLS
Websocket connection
Secure WebSocket
Endpont

Websocket


Websocket
Browser 1
WebRTC
SIP
WCS
RTP
SIP Server
WebRTC
SIP
Browser 2
Websocket

Websocket


LUCY
SIP Server
BOB
REGISTER
REGISTER
INVITE
INVITE
Trying
Ringing
Ringing
OK
OK
ACK
ACK
MEDIA FLOW (RTP)

Ваши вопросы?

Рефлексия

Назовите 3 момента, которые вам запомнились в процессе занятия

Что вы будете применять в работе из сегодняшнего вебинара?

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Викирюк Павел

Системный инженер