HyperText Transfer Protocol

Кулаков Кирилл Александрович

История развития протокола HTTP

- Протокол HTTP предложен в 1991 г. рамках проекта WorldWideWeb, как протокол для доступа к гипертекстовым документам:
  - единственный вид запроса — GET, единственный формат ответа — HTML;
  - вырисовывается облик формата запроса и ответа современного HTTP;
  - первую задокументированную версию протокола называют HTTP V0.9, описание доступно в архиве W3C.

История развития протокола HTTP

- В 1996 г. опубликован RFC 1945 (информационный статус), описывающий HTTP/1.0:
  - расширенный набор операций;
  - формальное описание формата запроса и ответа;
  - появляется понятие заголовка;
  - рассмотрены вопросы безопасности;
  - кэш клиентской стороны;

История развития протокола HTTP

- В 1997 г. опубликован RFC 2068 (предполагаемый стандарт), описывающий HTTP/1.1 (в 1999г. заменен на RFC 2616):
  - улучшения, уточнения, применение текущих практик;
  - возможность отправки нескольких запросов в рамках одного TCP-соединения;
  - отправка заголовка "Host" в запросе становится обязательной.
- В 2009 г. компания Google анонсирует разработку протокола SPDY, основной задачей которого является снижение задержек при загрузке веб-страниц.
- В 2012 г. опубликован черновик стандарта на протокол HTTP/2, разработанный на основе SPDY.

История развития протокола HTTP

- В 2014 г. рабочей группой HTTPbis, созданной для ревизии и уточнения спецификации HTTP/1.1 выпущен набор RFC:
  - RFC 7230 - HTTP/1.1: Message Syntax and Routing
  - RFC 7231 - HTTP/1.1: Semantics and Content
  - RFC 7232 - HTTP/1.1: Conditional Requests
  - RFC 7233 - HTTP/1.1: Range Requests
  - RFC 7234 - HTTP/1.1: Caching
  - RFC 7235 - HTTP/1.1: Authentication
- В 2015 г. опубликован RFC 7540, описывающий HTTP/2. Google отказывается от дальнейшей работы над SPDY.

Архитектура HTTP

- Действующие стороны
  - Клиент (User Agent) — любое приложение, инициирующее HTTP-запрос (браузеры, web-роботы, системы обновления ПО, …).
  - Сервер-первоисточник (Origin server) — программа, которая может отдать достоверный ответ для запрашиваемого ресурса.
  - Посредники (Intermediaries)

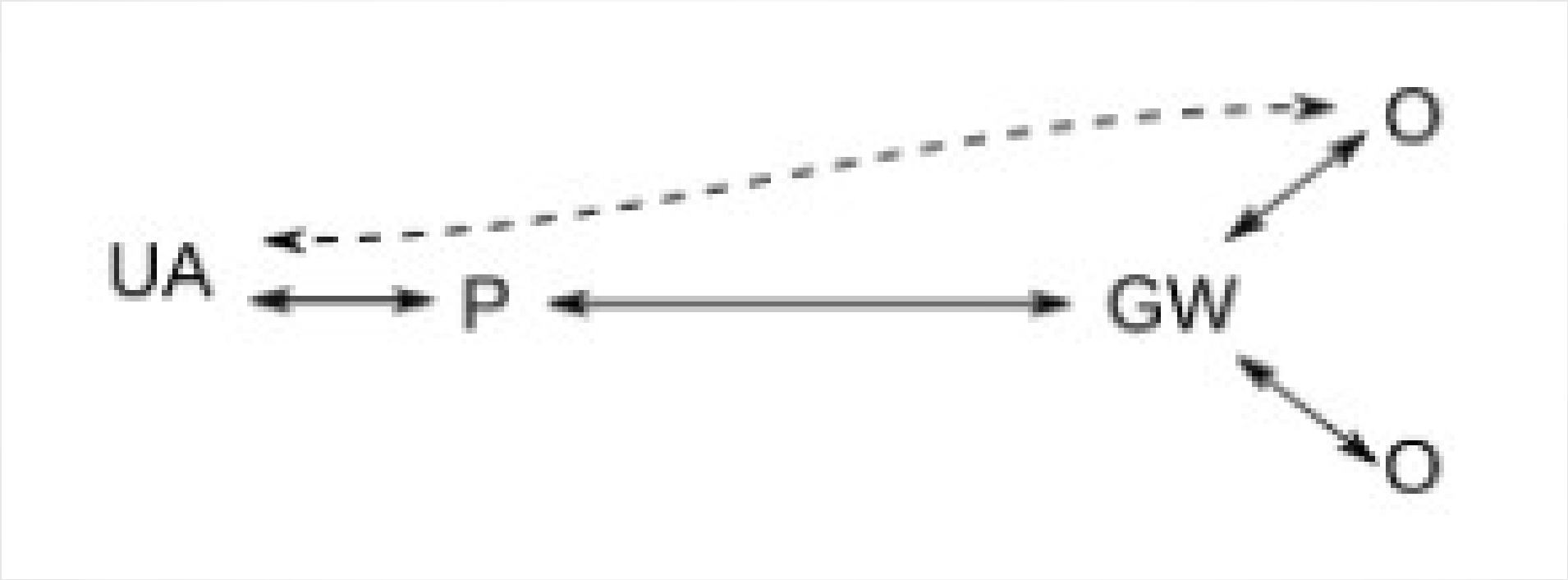
Архитектура HTTP

- Виды посредников
  - Прокси (Proxy) — выбранный клиентом агент пересылки сообщений.
  - Шлюз (Gateway) или обратный прокси (Reverse proxy) — перенаправляет пришедшие запросы на другой сервер или на группу серверов. Применение: кеширование, распределение нагрузки, трансляция запросов.
  - Тоннель (Tunnel) — транслирует сообщения между двумя соединениями, не внося в них никаких изменений.

Архитектура HTTP

- Другие сущности архитектуры
  - Кэш — локальное хранилище полученных ранее сообщений и система, управляющая этим хранилищем (наполнение, удаление).
  - Uniform Resource Identifiers (RFC 3986).
  - Тело сообщения форматируется в соответствии со стандартом на сообщения электронной почты (RFC 5322), а также его расширением — MIME (Multipurpose Internet Mail Extensions, RFC 2045).

Архитектура HTTP



Архитектура HTTP

- Ресурс и его представление
  - Предметом любого HTTP-запроса является некоторый ресурс, идентифицируемый с помощью URI.
  - Представление ресурса — информация, отражающая текущее или указанное в запросе состояние заданного ресурса.
  - Сервер-первоисточник может предоставлять различные представления одного и того же ресурса.
  - Выбор конкретного представления делается сервером на основании метода запроса, а также на основании заголовков согласования содержимого.

Архитектура HTTP

- Пример
  - Ресурс: термометр на метеостанции.
  - URI ресурса: http://meteo-example.com/russia/petrozavodsk/thermo.
  - Состояние ресурса: текущие сведения о температуре.
  - Представление ресурса: HTML-документ, XML-документ, изображение термометра…

Формат сообщений протокола HTTP

- Способы отладки HTTP-взаимодействия
  - Утилита curl, запущенная с ключом -v и URL в качестве аргумента выводит отправленный HTTP-запрос, HTTP-ответ и тело ответа.
    - $ curl -v http://webarch.cs.prv/welcome.html
  - Большинство современных браузеров имеют возможность отображения сетевой активности при загрузке страницы, в том числе детальное описание выполненных HTTP-запросов.
    - В Firefox/Chrome нажать F12, выбрать вкладку «Сеть».

Формат сообщений протокола HTTP

- Формат запроса

```
HTTP-request   = method SP request-target SP HTTP-version CRLF
                       *( header-field CRLF )
                       CRLF
                       [ message-body ]
```

```
$ curl -H "Accept-Language: en, ru" -v ya.ru
…
> GET / HTTP/1.1
> User-Agent: curl/7.32.0
> Host: ya.ru
> Accept: */*
> Accept-Language: en, ru
…
```

Формат сообщений протокола HTTP

- Формат ответа

```
HTTP-response   = HTTP-version SP status-code SP reason-phrase CRLF
                  *( header-field CRLF )
                  CRLF
                  [ message-body ]
```

```
$ curl -H "Accept-Language: en, ru" -v ya.ru
…
< HTTP/1.1 200 Ok
< Server: nginx
< Date: Tue, 30 Sep 2014 07:20:33 GMT
< Content-Type: text/html; charset=UTF-8
< Content-Length: 8128
< Connection: close
< Cache-Control: no-cache,no-store,max-age=0,must-revalidate
< Expires: Tue, 30 Sep 2014 07:20:34 GMT
< Last-Modified: Tue, 30 Sep 2014 07:20:34 GMT
< P3P: policyref="/w3c/p3p.xml", CP="NON DSP ADM DEV PSD IVDo OUR IND STP PHY PRE NAV UNI"
< Set-Cookie: yandexuid=3005000001412061634; Expires=Fri, 27-Sep-2024 07:20:33 GMT; Domain=.ya.ru; Path=/
< X-Frame-Options: DENY
< X-XRDS-Location: http://openid.yandex.ru/server_xrds/
…
```

Формат сообщений протокола HTTP

- Методы
  - HTTP метод — часть запроса, указывающая цель выполнения клиентом данного запроса и что ожидает получить клиент в случае успешного выполнения запроса. Подробнее см. RFC 7231, раздел 4.
  - Все реализации Веб-серверов должны поддерживать методы GET и HEAD, остальные методы опциональны.

Формат сообщений протокола HTTP

Method	Description
GET	Transfer a current representation of the target resource.
HEAD	Same as GET, but only transfer the status line and header section.
POST	Perform resource-specific processing on the request payload.
PUT	Replace all current representations of the target resource with the request payload.
DELETE	Remove all current representations of the target resource.

Формат сообщений протокола HTTP

Method	Description
CONNECT	Establish a tunnel to the server identified by the target resource.
OPTIONS	Describe the communication options for the target resource.
TRACE	Perform a message loop-back test along the path to the target resource.

Формат сообщений протокола HTTP

- Коды состояния
  - Код состояния — целое число, состоящее из трех цифр. Первая цифра указывает на класс кода состояния. Клиент не обязан «понимать» коды всех состояний сервера, но обязан «понимать» все классы кодов состояний. Подробнее см. RFC 7231, раздел 6.

- 1xx (Informational): The request was received, continuing process
  - 100 Continue
  - 101 Switching Protocols

Формат сообщений протокола HTTP

- 2xx (Successful): The request was successfully received, understood, and accepted
  - 200 OK
  - 201 Created
  - 204 No Content
- 3xx (Redirection): Further action needs to be taken in order to complete the request
  - 301 Moved Permanently
  - 302 Found
  - 304 Not Modified

Формат сообщений протокола HTTP

- 4xx (Client Error): The request contains bad syntax or cannot be fulfilled
  - 400 Bad Request
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed
  - 413 Payload Too Large

Формат сообщений протокола HTTP

- 5xx (Server Error): The server failed to fulfill an apparently valid request
- 500 Internal Server Error
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout

Заголовки запроса

- Управляющие заголовки (controls)

  Указывают специфику обработки запроса сервером.

  - Cache-Control — директивы, которые должны исполняться всеми кешами на пути следования запросами (Cache-Control: no-cache).
  - Expect — указывает серверу какой конкретно ответ ожидается (Expect: 100-continue).
  - Host — конкретное доменное имя сервера, к которому происходит обращение (Host: example.com).
  - Max-Forwards — максимальное количество раз, которое запрос может быть передан через прокси или шлюзы (Max-Forwards: 10).

Заголовки запроса

- Pragma — значение данного заголовка интерпретируется в зависимости от реализации сервера. Обычно используется как обратно-совместимый с HTTP/1.0 способ указания директив кеширования (Pragma: no-cache).
- Range — запросить только часть представления ресурса (Range: bytes=500-999).
- TE — список способов кодирования полезной нагрузки, которые клиент может обработать.

Заголовки запроса

- Условные заголовки (conditionals)

  Действие, соответствующее методу запроса не будет применено к запрашиваемому ресурсу если не будут выполнены условия, заданные заголовками. Подробнее см. RFC 7232.

  - If-Match
  - If-None-Match
  - If-Modified-Since
  - If-Unmodified-Since
  - If-Range
- Пример (проверка отсутствия изменений страницы по eTag): If-None-Match: "686897696a7c876b7e"

Заголовки запроса

- Заголовки согласования содержимого (content negotiation)

  Предпочтения клиента по способу представления содержимого. Подробнее см. RFC 7231, раздел 3.4.1.

  - Accept
  - Accept-Charset
  - Accept-Encoding
  - Accept-Language

Заголовки запроса

- Заголовки передачи аутентификационных данных (authentication credentials)
  - Authorization
  - Proxy-Authorization

Заголовки запроса

- Контекстные загловки (request context)

  Подробнее см. RFC 7231, раздел 5.5.

  - From. Электронный адрес создателя клиентского приложения.
  - Referer. URI ресурса, который ссылается на запрашиваемый ресурс.
  - User-Agent. Название клиентского приложения.

Заголовки ответа

- Управляющие заголовки (control data)

  Различные дополнения к коду состояния.

  - Age. Количество секунд, которые данное объект находится в кеше прокси-сервера.
  - Cache-Control. Сообщает о политике кеширования объекта, возвращаемого в даннмм ответе.
  - Expires. Содержит дату и время, когда возвращаемый объект становится просроченным.
  - Date. Время сервера на момент отправки ответа.
  - Location. URI перенаправления или созданного объекта.

Заголовки ответа

- Управляющие заголовки (control data)
  - Retry-After. Указание времени, когда запрошенный ресурс станет доступным.
  - Vary. Указывает какой заголовок запроса повлиял на выбор сервером представления запрошенного ресурса.
  - Warning. Дополнительные к коду состояния предупреждения от сервера.

Заголовки ответа

- Описание способа аутентификации

  WWW-Authenticate.

  - Сервер, генерирующий ответ 401 (неавторизованный), ДОЛЖЕН отправить поле заголовка WWW-Authenticate, содержащее хотя бы один запрос.
  - Поле может содержать несколько запросов или встречаться несколько раз
  - Пример:
    WWW-Authenticate: Newauth realm="apps", type=1, title="Login to \"apps\"", Basic realm="simple"

Заголовки ответа

- Метаданные выбранного представления запрошенного ресурса
  - ETag. Уникальный идентификатор версии ресурса. Используется при кешировании.
  - Last-Modified. Дата последней модификации.

Заголовки ответа

- Контекстные заголовки
  - Accept-Ranges. Поддерживаемые способы указания частичных запросов.
  - Allow. Список методов, которые могут быть применены к запрошенному ресурсу.
  - Server. Информация о сервере.

Заголовки ответа

- Метаданные представления ресурса
  - Content-Type. MIME Media Type представления ресурса, передаваемого в данном сообщении (text/html, application/javascript).
  - Content-Encoding. Способ кодирования представления ресурса (gzip, deflate, compress).
  - Content-Language. Язык документа.

Способы передачи полезной нагрузки HTTP-сообщений

- Полезная нагрузка (payload) HTTP-сообщений содержит представление запрашиваемого ресурса, если это предполагается методом запроса и кодом состояния ответа.
- Способы передачи полезной нагрузки внутри HTTP-сообщения:
  - указание способа кодирования при передаче (transfer encoding):
    - варианты: chunked, gzip, compress, deflate;
    - может изменяться по ходу передачи сообщения через сервера-посредники.
  - указание размера содержимого.

Способы передачи полезной нагрузки HTTP-сообщений

- Заголовки сообщения, указывающие на способ передачи полезной нагрузки:
  - Transfer-Encoding. Наименование способа кодирования полезной нагрузки.
  - Content-Length. Длина полезной нагрузки текущего сообщения.
  - Content-Range. Диапазон, отдаваемый сервером в данном сообщении в ответ на запрос с заголовком Range.
  - Trailer. Заголовки, которые будут переданы по окончанию передачи полезной нагрузки.

Способы передачи полезной нагрузки HTTP-сообщений

- Пример использования Trailer:

  HTTP/1.1 200 OK

  Content-Type: text/plain

  Transfer-Encoding: chunked

  Trailer: Expires

  7\r\n

  Mozilla\r\n

  9\r\n

  Developer\r\n

  7\r\n

  Network\r\n

  0\r\n

  Expires: Wed, 21 Oct 2015 07:28:00 GMT\r\n

  \r\n